Table of Contents

[1.1 History 2](#_Toc473034979)

[1.2 Definitions 3](#_Toc473034980)

[1.3 Secure Data base API 4](#_Toc473034981)

[1.3.1 SQLiteDatabase.loadLibs 5](#_Toc473034982)

[1.3.2 getWritableDatabase(password) 7](#_Toc473034983)

## History

1.0

* Integrated ORMLite with SQLCipher.

## Definitions

|  |  |
| --- | --- |
| SQLite | Android provides several ways to store user and app data. SQLite is one way of storing user data. SQLite is a very light weight database which comes with Android OS, using this to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. |
| ORMLite | ORMLite is lightweight Java ORM supports for android Sqlite Database.The full form of ORMLite is Object Relational Mapping Lite(ORMLite).and it provides some light weight functionality to store and retrieve Java Objects.And it avoiding the complexity and more standard Object Relational Mapping. |
| SQLCipher | **SQLCipher is used for a full database encryption**. What *full* means is that the DB file is entirely encrypted, not only separate rows or tables. Zetetic, the company behind SQLCipher, provides Community and Commercial edition. |
|  |  |
|  |  |
|  |  |

## Secure Data base API

Following are the list of individual api to implement Secure Data base.

### SQLiteDatabase.loadLibs

SQLiteDatabase.*loadLibs*(**this**);

The call to SQLiteDatabase.loadLibs(this) must occur before any other database operation else will get a runtime exception - java.lang.UnsatisfiedLinkError: No implementation found for void net.sqlcipher.database.SQLiteDatabase.dbopen(java.lang.String, int) (tried Java\_net\_sqlcipher\_database\_SQLiteDatabase\_dbopen and Java\_net\_sqlcipher\_database\_SQLiteDatabase\_dbopen\_\_Ljava\_lang\_String\_2I) and it's because we need to load a few native libraries first. So, add this line to the onCreate() method: SQLiteDatabase.loadLibs(this)

Note: The import of net.sqlcipher.database.SQLiteDatabase  instead of android.database.sqlite.SQLiteDatabase.

To Support 64bit device have to add below lines in build.gradle file else it will try to load all types of native .so file and failed.

ndk {  
 abiFilters **"armeabi"**, **"armeabi-v7a"**, **"x86"**, **"mips"** }

App exclude [arm64-v8a](https://developer.android.com/ndk/guides/abis.html#arm64-v8a) libraries from the APK in order to prevent runtime crashes due to library architecture conflicts (https://developer.android.com/studio/build/configure-apk-splits.html#configure-abi-split ?)

**Sample Implementation:**

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);

SQLiteDatabase.loadLibs(this);

}

}

**getPassword()**

Calls to open or create the database need read or write permission , in SQLCipher for security reason we have to passing the password as an argument.

For example:

getWritableDatabase('my\_secure\_key')

After this we can start database operations.

**getPassword()**

Calls to open or create the database need read or write permission , in SQLCipher for security reason we have to passing the password as an argument.

For example:

getWritableDatabase('my\_secure\_key')

After this we can start database operations.

**getPassword()**

### getPassword()

In SQLCipher for security reason we have to passing the password as an argument when open or create database. To using helpher class instance with we can call this API.

Before use this API’s , first we have to use App Infra’s Secure Storage module password generation api called createKey. It will generate password, using AES Key Generator along with KeyStore. This method comes under App Infra’s Secure storage library so we need Add below lines in build.gradle

compile(group: 'com.philips.cdp', name: 'AppInfra', version: '1.4.0', ext: 'aar')

After generate password via Secure Storage createKey(KeyType.Key, “keyName”, ssError), we have to use same key to retrieve generate password so we have to pass KeyName via constructor of helper class to call super class SecureDbOrmLiteSqliteOpenHelper constructor.

To Retrieve this password we to call getPassword() it will return password as a String.

For Example:

AppInfraInterface appInfra = **new** AppInfra.Builder().build(getApplicationContext());  
SecureStorageInterface

*mSecureStorage* = appInfra.getSecureStorage();  
SecureStorageInterface.SecureStorageError sse = **new** SecureStorageInterface.SecureStorageError(); *// to get error code if any  
mSecureStorage*.createKey(SecureStorageInterface.KeyTypes.***AES***, *DATABASE\_PASSWORD\_KEY*, sse);

Pass same key (*DATABASE\_PASSWORD\_KEY*)to SecureDbOrmLiteSqliteOpenHelper constructor.

**public class** SecureDataBaseHelper<T> **extends** SecureDbOrmLiteSqliteOpenHelper {  
   
  
 **public** SecureDataBaseHelper(Context context, String dataBaseName, **int** databaseVersion, String databaseKey) {  
 **super**(context, dataBaseName, **null**, databaseVersion, databaseKey);  
   
  
 }

}

Create instance of above class and call getPassword.

SecureDataBaseHelper **secureDataBaseHelper** = **new** SecureDataBaseHelper<>(**this**, ***DATABASE\_NAME***, *DATABASE\_VERSION*, *DATABASE\_PASSWORD\_KEY*);

**secureDataBaseHelper.getPassword().**

### getWritableDatabase(password)

Calls to open or create the database need read or write permission , in SQLCipher for security reason we have to passing the password as an argument. We have to call this method with the help of helper class instance.

For example:

getWritableDatabase('my\_secure\_key')