# Oflameron SQLiteandRSA

Open source project - Android JAVA application for working with the SQLite database and RSA encryption of records.

Created by Valery Shmelev https://www.linkedin.com/in/valery-shmelev-479206227/

This project is a great **startup** for developing a commercial application: for logistics, sales, customer systems, reference and reporting databases, accounting systems, etc. The application contains many ready-made methods and detailed comments.

You or your programmers don't have to spend time looking for and debugging code. Use ready-made open source code.

The application template **SQLiteandRSA** has an activity of unlimited length - it can do scrolling activities.

SQLiteandRSA - https://github.com/vallshmeleff/sqliteandrsa

## Review

Class **RSACode.java** - class for encrypting/decoding, generating, saving, restoring, exporting/importing keys, creating, editing, viewing, exporting/importing a SQL database, obfuscating and processing text.

Methods

**public Key[] RSAKeyGen()** – method. Creates an array of KeyMass keys from two elements - public and private keys.

**return KeyMass;** - returns an array of two keys

**public Key[] RSAKeyReGenerate** – method. Recovering Public Key and Private Key RSA from Text Variable. This is necessary so that you can send the client a public key for encryption and save your private key for decoding.

**return KeyMass;** - returning an array of two keys Public and Private

**public byte[] RSATextEncode** - method. Encodes short text. For large texts, you need to use a different set of methods (listed below).

**return encodedBytes;** - return coded Text (bytes array)

**public byte[] RSATextDecode** – method. Decode encrypted short Text

**return decodedBytes;** - Return Decoded Text (bytes array)

**public String Load** – method. Read text file – converted keys, text. For read and import.

**return sSTR2;** - return data.

**public void Save** – method. Write data to the text file MODE_PRIVATE. For save keys, texts. For write and export

**public void ClipBrdWrite** – method. Write text to the Android Clipboard

**public String String2Code** – method. Convert String to CharCodeString. Convert TEXT (STRING) to UNICODE (STRING) to work with national alphabets

**return chcstr;** - return CharCodeString

**public String Code2String** – method. Convert CharCodeString to String. Convert UNICODE (STRING) to text string (STRING) to work with national alphabets

**return RestoreTXT;** - return CharCodeString

**public String ObfuscationD** – method. Simple digit data obfuscation. This is the simplest example for checking Java code. Real examples of obfuscation based on this code are created individually to preserve privacy and make it difficult to detect text.

**return lstr;** - Return Obfucated Text

**public String[] eFragment** – method. For large text fragmentation.

**return masString;** - return Text array - Fragments array

**public String[] eDEFragment** – method. Divides a large ciphertext by tags into fragments suitable for decoding.

Class **MainActivity.java** - This class provides the user interface (no design) and application functionality (constantly evolving). Changes are constantly made to the class - new methods, features, comments.

Th Currently, a small number of fields in the SQLite database records are used - name, e-mail, phone number, note. It's very easy to add the number of fields. Later, the number of fields will be increased to 10. This is enough for most projects. You can change the number of fields in which the data is RSA encrypted. To search for records, it is enough to leave the username field unencrypted (for example). e interface is the simplest. This makes it more convenient to upgrade the code for a specific order.

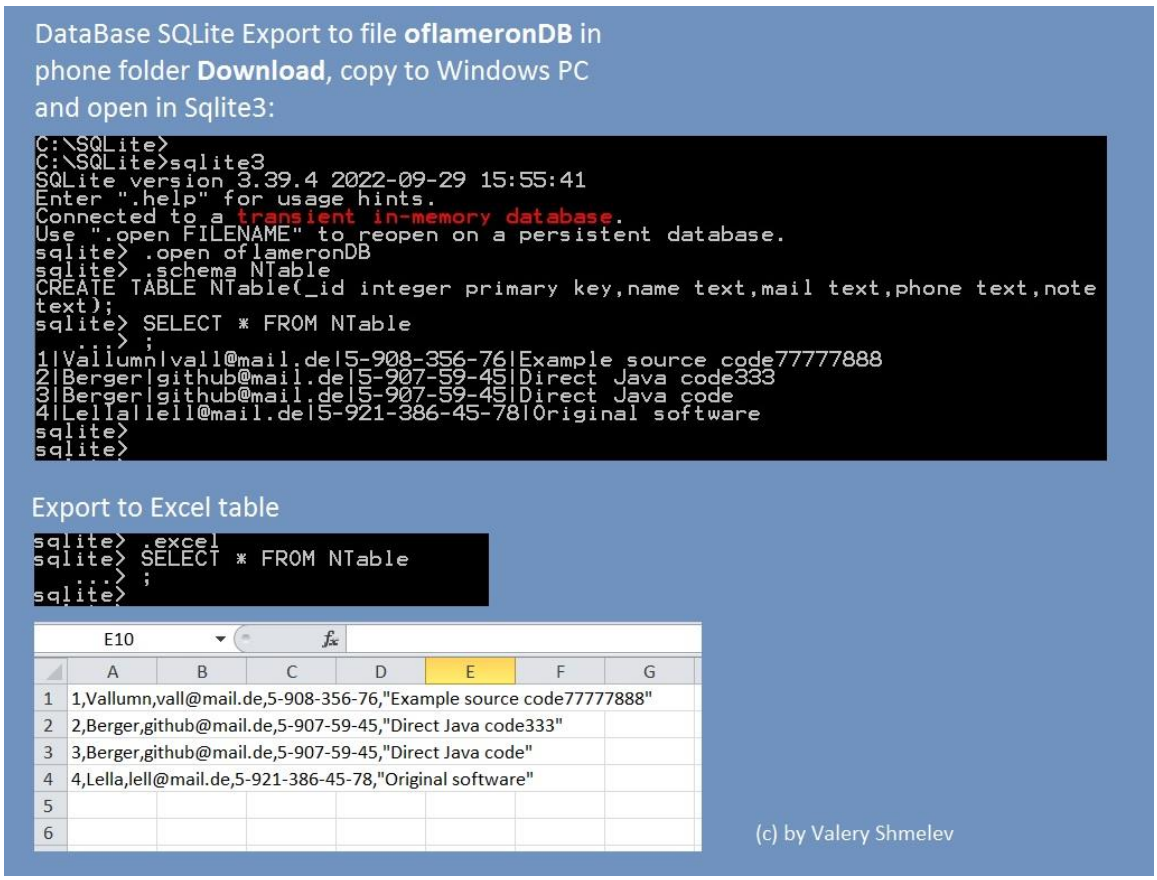
Handle pressing the button to export the SQLite database:

```
case R.id.btnExport: // Export SQLite DataBase
   ExportImport Export = new ExportImport(); // Class instance ExportImport.java
   Export.dbExport("oflameronDB"); // Export database file oflameronDB
   break;
```

The database file **oflameronDB** is copied to the standard **Download** folder of the Android smartphone. It is very comfortable. You can connect a USB cable and copy the database file to a Windows computer with the **sqlite3** application installed. Your data bank is easy to read and convert to MS Excel.

In MS Excel table, it is enough to use macros (Visual Basic Script- VBS) to separate records - parse the string by the separator. In this example, the separator is a comma.

The simplest example:




If your SQLite database contains encrypted record fields, you can send (export) it to a PHP server via FTP or POST and decrypt the data in PHP. An example of working with RSA encryption is in another repository.

If you use a special encryption key obfuscation mode, then even if your smartphone is lost or stolen, it will be almost impossible to read the data. Even serious people need many, many months for this.

Database file import

```
case R.id.btnImport: // Import SQLite DataBase
  ExportImport Import = new ExportImport(); // Class instance ExportImport.java
  Import.dbImport("oflameronDB"); // Import database file oflameronDB
  break;
```

Loading a database file into an Android smartphone. The database file **oflameronDB** must be placed in the same Download folder where the database was exported.

**public String NOTEncrypt()** – method. Encode NOTES field text with RSA public key

**return note;** - Return Text

**public String NOTEDecrypt()** – method. Decode NOTES text with RSA private key

*19.11.2022 Edition*

**MainActivity class** - this Java class is heavily modernized. Management may be a little behind.

To improve the efficiency of work on the Open Source project, partners are required - programmers, editors, native English speakers. Preferably from Europe, USA, Canada. This is due to the limited knowledge of foreign languages.

For letters - e-mail can be found in the https://github.com/vallshmeleff/feedback repository. Programmers will easily understand how to do this.

Sincerely OFLAMERON. 2022