Android Based Encryption SMS System

Sandip Nair - 106116078
Shubhendra Singhal - 106116088

April 19, 2019

Contents

0.1 Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to Prof. R. Leela Velusamy and Prof. Divya for their guidance and constant supervision as well as for providing necessary information regarding the project also for their support in completing the project.

Our thanks and appreciations also go to the team members in developing the project and people who have willingly helped us out with their abilities.

0.2 Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shubhendra Pal Singhal
National Institute of Technology, Tiruchirappalli

Sandip Dilip Nair
National Institute of Technology, Tiruchirappalli

0.3 Abstract

The project focuses on the basic sms transfer from one sender to another safely. The "safe" word describes that any intruder should not be able to retrieve the message from the known attacks in cryptography, hence making it trustable source of communication. The application consists of a instant and secure login. The application, then asks the user whether he/she wants to craft a message or see his inbox to figure out what's new? The crafting of message includes the feature of accessing the contacts. The user can instantly see his message and the corresponding cipher text. There is a tutorial accompanied to understand how the encryption and decryption is being done. The receiver finally will receive the sms which this application will decrypt and store in its inbox. The app also supports group messaging, wherein a common message can be sent to a group of receivers.

0.4 Why Encryption?

Encryption is the process to transfer information securely in a secretive way. It protects your information over the communication link. It helps to guard your privacy or anonymity and conversations, whether video, voice, or text. Encryption is needed when you don't want anyone else to have access to.[2]

A5/1 is a stream cipher used to provide over-the-air communication privacy in the GSM cellular telephone standard. It is one of seven algorithms which were specified for GSM use.[1] It was initially kept secret, but became public knowledge through leaks and reverse engineering. A number of serious weaknesses in the cipher have been identified.[1] This is generally used for the encryption but a number of attacks on A5/1 have been published, and the American National Security Agency is able to routinely decrypt A5/1 messages according to released internal documents.[5]

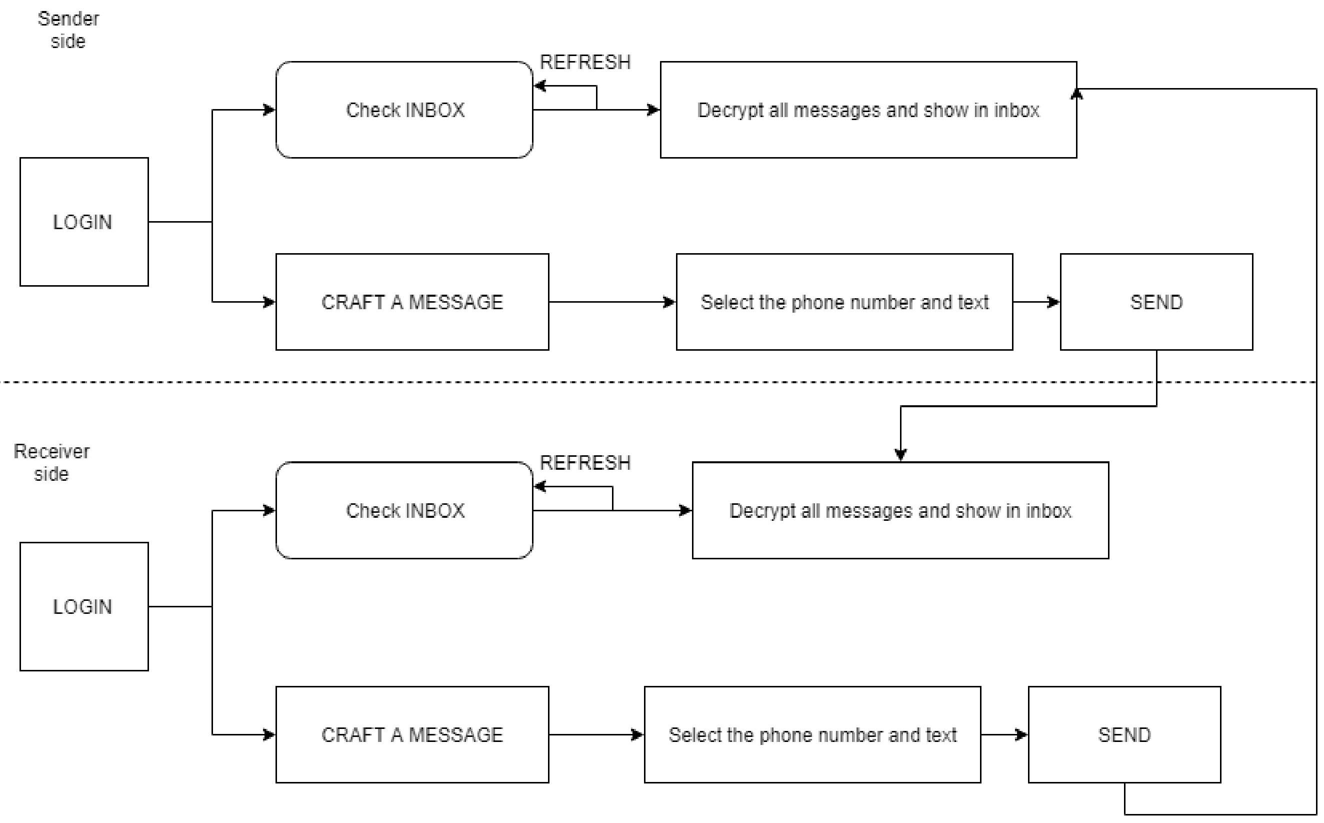
Some attacks require an expensive preprocessing stage after which the cipher can be broken in minutes or seconds. So, that's why the application does not depend on the sms provider service to encrypt rather the application encrypts the message itself to be safe.

0.5 Introduction

The application consists of a instant and secure login. The database for the following will be stored with the trusted third party, suggestions are Google cloud which can securely store huge amounts of data. The next step in the application, then asks the user to craft the message. The crafting of message includes the feature of accessing the contacts and this is possible with the help of contacts adapter which allows the user to access the contact and direct to the textbox in the edit text of the android application. The recipients can be listed as comma separated values. The user can instantly see his message and the corresponding cipher text along with the phone number he has send to. The decryption similarly happens at the receiver side and before accessing the inbox, the particular cipher text is decrypted with the same key that the receiver can access from the database stored at the cloud storage.

0.6 Workflow of the system

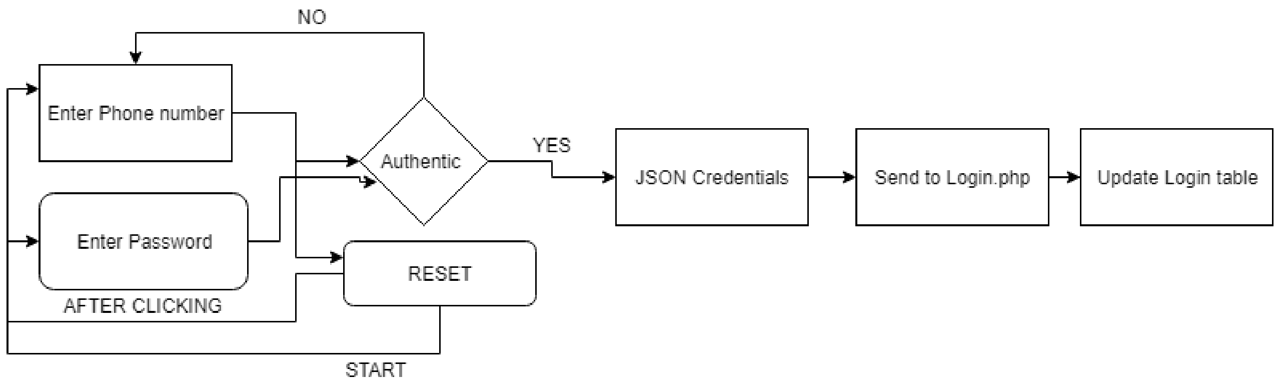
The following diagram illustrates the overall flow of the data in the application and what all options a user have.


Sender side
LOGIN
Check INBOX
REFRESH
Decrypt all messages and show in inbox
CRAFT A MESSAGE
Select the phone number and text
SEND
Receiver side
LOGIN
Check INBOX
REFRESH
Decrypt all messages and show in inbox
CRAFT A MESSAGE
Select the phone number and text
SEND

0.6.1 How does Login works?

The login contains the phone number to be registered. The phone number can be registered only when the user enters both password and phone number. As soon as the user presses the login button, a separate thread is created using AsyncTask which helps in reducing the load of the Main UI thread. This thread executes the following onPre, InBackground and onPost where the registering of data happens in background and the subsequent helps to go nack to the main function again for carrying out other intent functions.
The respective credentials are then directed as json objects from android application as phoneno : , password : to the php file which hashes the password using md5() function, an inbuilt function in the php language. The table Login conatins the credentials stored. The following database is connected to the emulator using the localhost from wamp server using 10.0.2.2 as the host that emulator understands as 127.0.0.1 or localhost for that matter.
The disadvantage of this particular system is that anybody can register with the other person's phone number thus disallowing him to use the application, so we have introduced the concept of reseting the password if you feel that the you have'nt registered or you have forgot the password. Reseting ensures that all your messages are securely deleted without allowing anybody to know

about your previous messages. The OTP concept is also very famous nowadays but the only problem that usually customers faces is that there are times when OTP does not come on time and takes one day or even two just for one authentication. This login is a secure one in terms of data, as nobody can steal your previous data, if they dont have the access to which the customer has to take care of.



0.6.2 How to make the access contact list adapter?

Contacts is one of the most important feature in every mobile phone whether its android device or not because its help us to store our related persons mobile phone number into device. By default android device has its own contact showing application but some times when android app developer needs so specific functionality in his project then you can easily display the contact list into app. The recipients of the encrypted message, if greater than one can be specified as a list of contacts separated by commas for a group message. The contact list is accessed by the user by following steps :
STEP 1 :
Manifest.permission.READCONTACTS, is the permission that needs to be given to the app for accessing the contact list.
STEP 2:
ActivityCompat.requestPermissions() is used for granting the permission if the permission is not set, thus allowing you to access the list without you taking care of the permissions by going in the your phone setiings.
STEP 3:
Intent.ACTIONPICK, ContactsContract.Contacts.CONTENTURI, have to be specified to see where is the contact list present that is available in the form of URI and the following action that needs to be performed once you access the list.
STEP 4 :
The cursor and uri is intialised and the following content resolver is provided the access to list.
STEP 5 :
cur.getString(cur.getColumnIndexOrThrow(ContactsContract.Contacts.ID)), helps in getting the phone numbers accessed via ids stored in the content provider and the data is returned in the form of cursor objects.
STEP 6:
The phonecursor specific to the contact that we want is declared using the id as reference to find the details for the action user performed on the touchscreen.
STEP 7 :
Finally, the phone number is received and is placed on the edit text which we wanted.

0.6.3 SMS adapter for sending the message

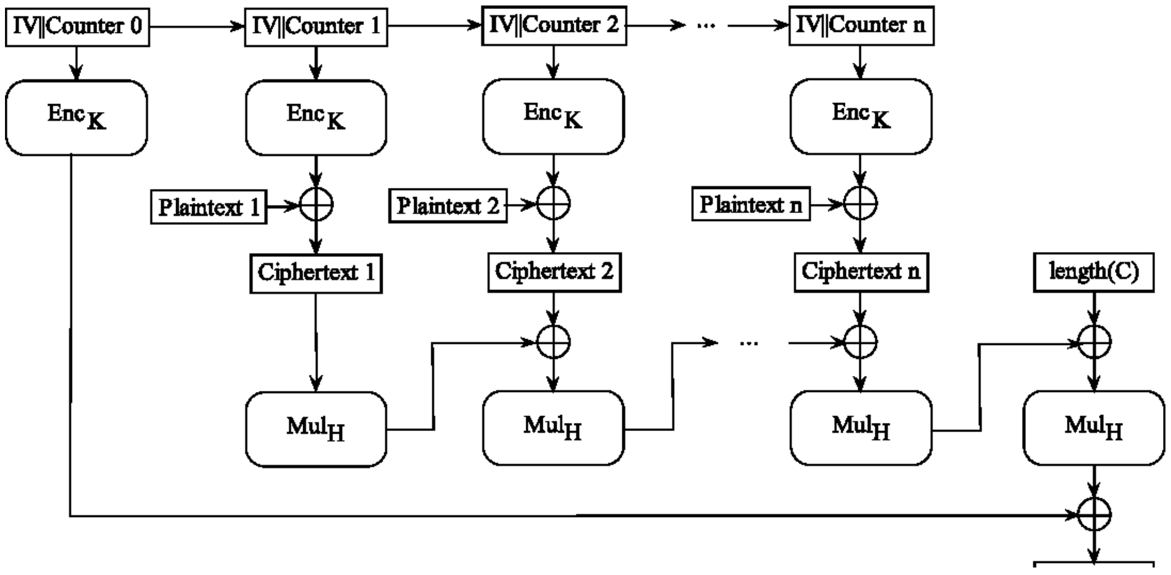
STEP 1 :
Android uses a permission-based policy where all the permissions needed by an application need to be specified in the AndroidManifest.xml file. By doing so, when the application is installed it will be clear to the user what specific access permissions are required by the application. For example, as sending SMS messages will potentially incur additional cost on the user's end, indicating the SMS permissions in the AndroidManifest.xml file will let the user decide whether to allow the application to install or not. In the AndroidManifest.xml file, add the two permissions – SENDSMS and RECEIVESMS.
STEP 2 :
In the SMS activity, we wire up the Button view so that when the user clicks on it, we will check to see that the phone number of the recipient and the message is entered before we send the message using the sendSMS() function.
STEP 3 :
To send an SMS message, you use the SmsManager class. Unlike other classes, you do not directly instantiate this class; instead you will call the getDefault() static method to obtain an SmsManager object. The sendTextMessage() method sends the SMS message with a PendingIntent. The PendingIntent object is used to identify a target to invoke at a later time. For example, after sending the message, you can use a PendingIntent object to display another activity.
STEP 4 :
When an SMS message is sent, the BroadcastReceiver‘s onReceive event will fire. This is where you check the status of the sending process. The PendingIntent object monitors the delivery process. The BroadcastReceiver‘s onReceive event will fire when an SMS is successfully delivered and the toast will be generated saying "SMS SENT".

0.6.4 SMS Receiver side and Inbox configuration

We have used a BroadcastReceiver to receive the messages and parse the SMS message out of it and update the UI.
STEP 1 :
In manifest file, add permission to receive, read and write SMS to the application. Then add the broadcast receiver with an intent-filter to receiving SMS.
STEP 2 :
SmsBroadcastReceiver.java This is a BroadcastReceiver which receives the intent-filtered SMS messages. onReceive, we extract the SMS message bundle and show a toast message and also update the UI by adding the SMS message to the SMS inbox list. When a SMS message arrives, the inbox is automatically refreshed.
STEP 3 :
SmsActivity.java This is the main Android activity of the SMS application. It acts as the SMS inbox. It has a ListView to show the SMS messages. onCreate we read all the messages present in the internal SMS inbox from its Uri and show them using the ListView known as inbox.
STEP 4 :
The Android emulator provides the facility to send test SMS messages to the deployed app via the DDMS Emulator Control.

0.6.5 Encryption and Decryption Algorithm

The encryption scheme used in the application is the AES-GCM cipher, or the AES encryption scheme under the Galois Counter mode of operation.
In AES, each block is 128 bits wide. Hence, of we have a message, we pad it with bits such that the number of bits in the result is an integral multiple of 128 bits.
In the CTR mode, a nonce value IV is used to generate an initial value for a counter, which is XORed with the first block to produce the first block of the ciphertext. Then, the counter is incremented, encrypted and then XORed with the next block of the plaintext to produce the next block of the ciphertext and so on.



The decryption algorithm is executed as follows. Each block of the ciphertext is XORed with the corresponding encrypted counter value, which acts as the key to give the corresponding message block.

Let $AES(k,m)$ be the AES encryption function, $Counter[0] = IV$ be the initial value of the counter. If the message m is divided into 128 bit blocks m[0], m[1], ...m[t], and the key stream is $k = (k[0], k[1], ..., k[t])$, then,

$c[0] = AES(k[0], Counter[0]) \oplus m[0]$

$c[1] = AES(k[1], Counter[1]) \oplus m[1]$

and so on.

Similarly, decryption can be carried out as

$m[0] = AES(k[0], Counter[0]) \oplus c[0]$

$m[1] = AES(k[1], Counter[1]) \oplus c[1]$

and so on.

0.6.6 Dealing with Coding

There are some of the suggestions that we should in general follow while making an application.

Try and Catches

This is the condition which many a times android will tell you by itself. Try and catch is implemented for a reason that even if your app is not working properly, it will not crash. The crashing of app will lead to intense debugging which might not be easy to do. Secondly never leave the catches blank because they are even worse than not stating them because the app is mislead by the catch. So app doesnt know whether to crash or execute catch. Safely, android has been built in such a way that it will handle such situations and ensure that app crashes but to be on a safer side and not detroy your phone, declrae them properly.

Logcat for Verbose and Error

Once the error occurs, tracking the error becomes difficult as we cannot just read code and see what's wrong. So Logcat error region will show the errors in blue region specifying the exact error.

Log.d()

Errors sometimes may not be listed because its correct according to the application, but the data flow might have raised an exception. To detect such instances, we can list the particular Log.d("string" , "string") statement which will get printed in the debugger if the following was executed thus allowing to debug the program and exactly finding whether the data is flowing or memory leak is happening.

0.7 References

[1] https://en.wikipedia.org/wiki/A5/1
[2] https://encryption.bsa.org/
[3]https://stackoverflow.com/questions/7771461/android-contact-list-adapter
[4]https://developer.android.com/training/contacts-provider/retrieve-names
[5] https://javapapers.com/android/android-receive-sms-tutorial/
[6] https://developer.android.com/reference/android/telephony/gsm/SmsManager.html
[7] https://en.wikipedia.org/wiki/Galois/CounterMode