Secure Data Access from Android Smartphone without Internet

Mahesh Mookanagoudar
GearUp Technology,
Dharwad

S P Sajjan
Lecturer
Karnatak University,
Dharwad

Shashidhar Halligerimath
Lecturer
KHKIE Dharwad

ABSTRACT
In recent year adoption of Android as a platform for mobiles has increased. To perform remote accessing of mobile contents there are many ways which make use of internet. This technique mainly deals with secure access of the textual contents of mobiles like contacts, SMS, Call log, location, as well as secure deletion of the same data from the android mobiles remotely using any mobile(need not be android phone) through SMS using different keywords for performing different actions and doesn't need Internet for accessing of data.

Keywords
Android, SMS, Phonebook / Contacts, Call log, location, file

1. INTRODUCTION
Recently with the introduction of android in smartphone, there exists many applications for different purposes which meet wide variety of user requirements. Among those, this paper tries to provide a solution in a situation where in somebody wants to access or delete certain specific data from the mobile which is geographically away from the user's place by just sending an SMS with appropriate keywords to the android phone wherein this solution is implemented. The keywords include password and the action that the user wants to perform on their mobile. The concept involved is like this, the app(solution) implemented will be triggered for every SMS that the mobile receives. On receiving the SMS, the app will fetch the SMS and verify for the password. On successful authentication, it checks for the remaining keywords. Considering that the keywords match with the predefined keywords then based on the category of keywords whether they are meant for accessing the data like SMS/Call log/Contacts/Location or they are meant for deletion of SMS/Call log/Contacts, the actions will be performed and for access operations on successful fetching of the requested data, the data will be sent to the user mobile which had made the request via SMS whereas for deletion operations no result is sent. On unsuccessful match or if no data exists then nothing is performed. The major advantages of this concept are that it need not be running continuously in the background and it doesn't need internet and the client can have any mobile to perform operations on the Android mobile wherein the solution is implemented.

2. PROPOSED SYSTEM:

The proposed system makes use of built in features like SMS, Phonebook, Call log, GPS and files to serve the requested action. This system makes use of SharedPreferences for setting the password which can be used to authenticate the user when a request is made to access specific data through SMS. The technique of performing actions on a android mobile needs a pattern which will make the mobile to understand and process the action/request. Considering that the password is set, keywords need to be defined to perform various requests. Let the keywords be, GET_CONTACT for accessing the desired contact number, GET_SMS for accessing the SMS, GET_CALL_LOG for accessing the call log, GET_LOC to access the present location of the mobile, GET_FILE to access the contents of a file. Similarly let the keywords be, DEL_SMS to delete the contents of the SMS, DEL_CONTACTS to delete the contents of PhoneBook, DEL_CALLLOG to delete the call log. Once the keyword is defined, the user can make a request to access specific data by sending an SMS like the following.

Password,keyword

The above mentioned pattern remains same for all actions apart from accessing contacts.

The following is the pattern for accessing contacts

Password,keyword,desired_contact_name

Considering the user has made a request for accessing a contact number of a particular contact. Then the system behaves in the following manner, the first stage of operation deals with monitoring for the new SMS. Once the SMS enters the mobile, the BroadcastReceiver class informs our system about the arrival and the system fetches the newly arrived SMS to verify whether or not the application is meant for the system. Once the system fetches the SMS, it retrieves the content of it to verify that the first word in the SMS is the password. If the password matches then it tries to verify what action is being requested by the user by verifying the second word of the SMS with the predefined keyword. On successful match, it tries to perform requested action. The only

Difference over here would be if the second keyword is GET_CONTACT then the system needs to fetch the third word from the SMS which reveals the name of the desired contact number. Based on the second keyword the actions will be performed. For accessing the contact, the third word from the SMS will be considered. The so extracted word will be used with a query to check whether the requested contact exists in the phonebook, if it exists then it will be extracted and an SMS will be composed with the body part containing the extracted contact number and the to part containing the number from which the request was made and will be sent.

3. SAMPLE CODE

1. Accessing contact Code

```
ContentResolver contentResolver =
context.getContentResolver();
Cursor cursor =
cr.query(ContactsContract.Contacts.CONTENT_U
RI,null, "DISPLAY_NAME    = '" + keyword3 +
"'", null, null);
if (cursor.moveToFirst())
{
        String
contactId=cursor.getString(cursor.getColumnIndex(
ContactsContract.
        Contacts._ID));
        Cursor phones =
cr.query(Phone.CONTENT_URI,
null,Phone.CONTACT_ID + "
      = " + contactId, null, null);

        while (phones.moveToNext())
        {
              number =
        phones.getString(phones.getColumnIndex
        (Phone.NUMBER));
        }

}
```

2. In the same way when the user requests to delete the contacts from phonebook then

```
if(keyword2.equals(delContactKeyword))
{

Cursor cursor =
contentResolver.query(ContactsContract.Contacts.C
ONTENT_URI, null, null, null, null);
while (cursor.moveToNext())
{
        String lookupKey
=cursor.getString(cursor.getColumnIndex

        (ContactsContract.Contacts.LOOKUP_K
EY));
          Uri uri = CONTENT_LOOKUP_URI,
lookupKey);

Uri.withAppendedPath(ContactsContract.Contacts.
contentResolver.delete
      (uri, null, null);
}

}
```

To delete the call log

```
if(keyword2.equals(delCallLog))

      contentResolver.delete(CallLog.Calls.CONTENT_
URI, null, null)
```

To delete SMS

```
for (int i = 0; i < vector_id.size(); i++)
{
      str_id = vector_id.get(i);
      where = str_column_name + "=" + str_id;
      delRow = cr.delete(uri_sms, where, null);
}
```

To access call log

```
while (managedCursor.moveToNext())
{

      String
cntName=managedCursor.getString(callName1);
      String phNumber =
managedCursor.getString(number1);
      String callType1 = managedCursor.getString(type);
     String callDate1 = managedCursor.getString(date);
      Date callDayTime = new
Date(Long.valueOf(callDate1));
     String callDuration =
managedCursor.getString(duration1);
     String dir = null;
     int dircode = Integer.parseInt(callType1);
     switch (dircode)
     {
            case CallLog.Calls.OUTGOING_TYPE:
dir = "OUTGOING";
            break;
          case CallLog.Calls.INCOMING_TYPE: dir =
"INCOMING";
           break;
           case CallLog.Calls.MISSED_TYPE:dir =
"MISSED";
            break;

      }
      result=" Name= "+cntName+" Phone Number
      "+phNumber+" Date "+callDayTime+" Call Type
      "+dir+" Duration in sec "+callDuration;
      }
      managedCursor.close();

}
```

To access SMS

```
Uri uriSMSURI = Uri.parse("content://sms/inbox");
Cursor cur = context.getContentResolver().query(uriSMSURI,
null, null,null, null);
String smscontent = "";
while (cur.moveToNext())
{
       if(cur.getPosition()<=2 && (null !=
cur.getString(2) || "" != cur.getString(2)) && (null
!= cur.getString(11) || "" != cur.getString(11)))
     {
```

```
                System.out.println( cur.getPosition());
            smscontent += "From :" + cur.getString(2) + " : " +
cur.getString(11)+"\n";
        }
}
```

In the same way user present location can be accessed by using LocationManager class with the function lastknownlocation()

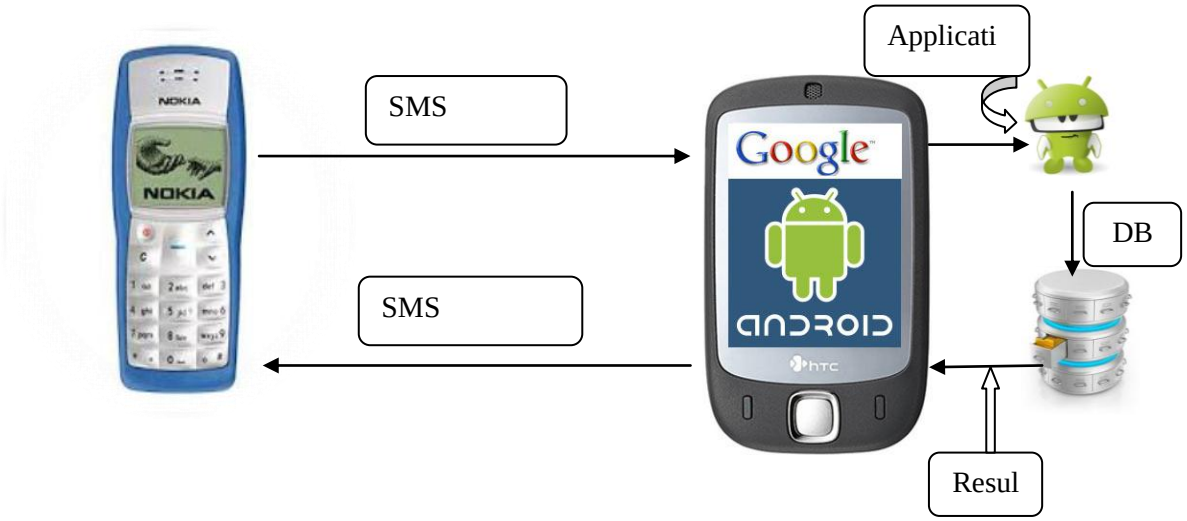

Fig 1: Architecture Design

4. FLOW DIAGRAM

1.**DELETE SMS/Call log/Contacts**

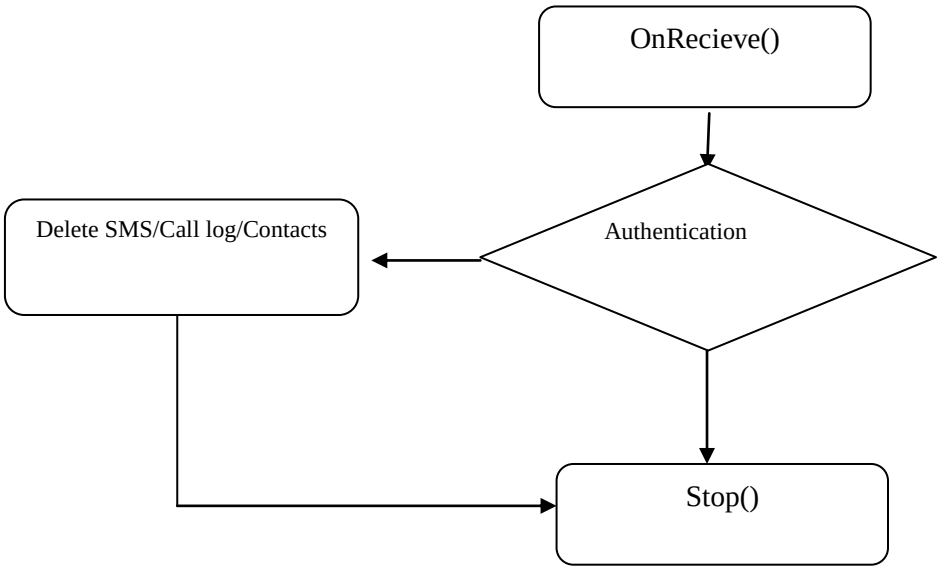

Fig 2: Delete flow Diagram

2. Data Accessing

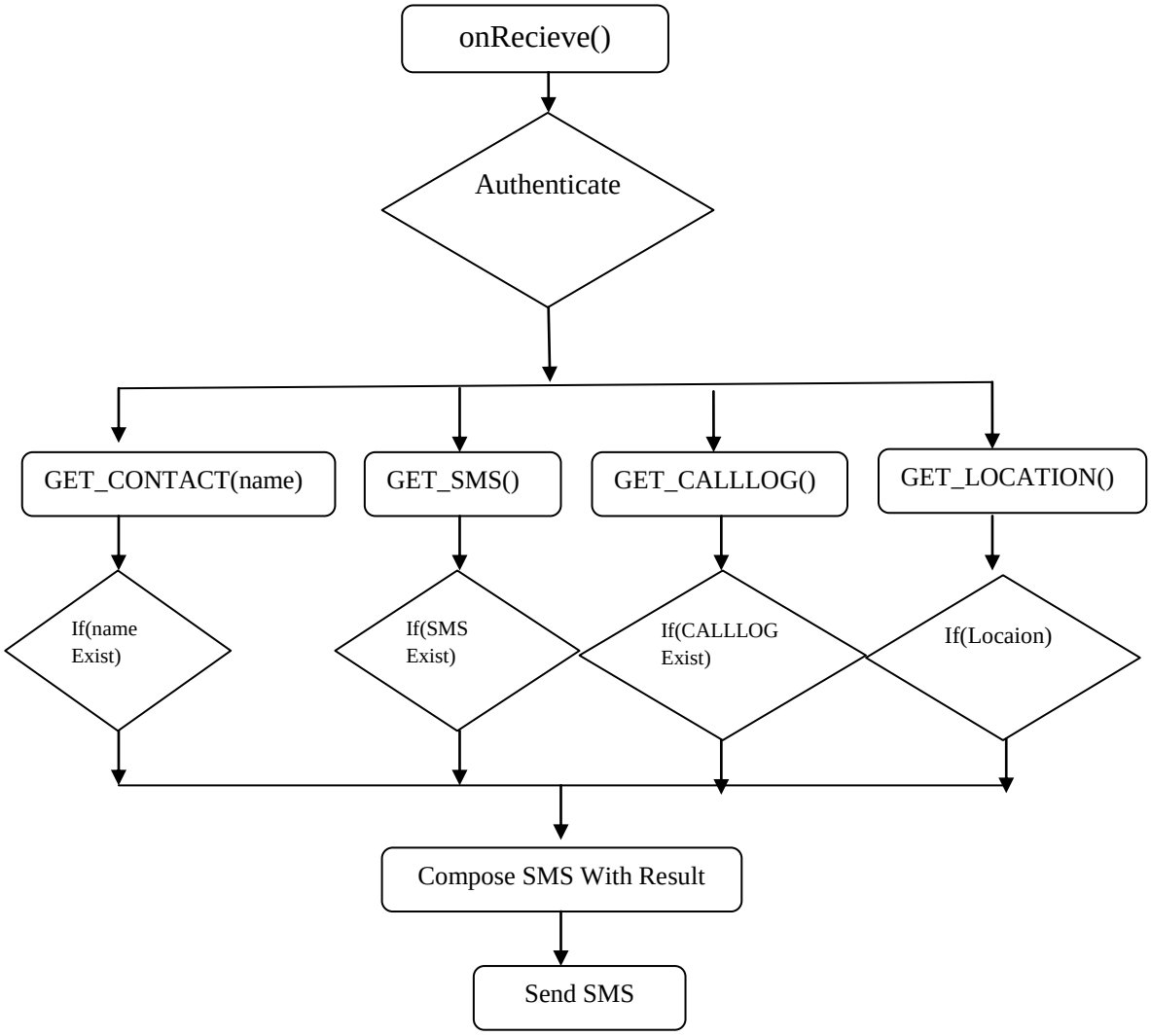

Fig 3: Data accessing flow Diagram

5. CONCLUSION:

This paper is based on the concept of SMS and many Android telephony API's. As Android is open source, there exists loads of resources to understand and it becomes easy to implement and deploy the solution. Finally we would like to conclude that this paper provides a secure solution for majority of the users who tend to forget their mobile and need to access their mobiles from any geographic corner of the world using any mobile.

6. REFERENCES:

[1] Beginning Android 2 (Paperback) by Mark Murphy

[2] Sams Teach Yourself Android Application Development in 24 Hours.

[3] Professional, Android Application Development (Paperback) by Reto Meier.

[4] http://android-developers.blogspot.in/

[5] http://developer.android.com/reference/android/content/BroadcastReceiver.html

[6] http, ://stackoverflow.com