ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3, Issue 11 November, 2014 Page No. 9044-9047

Digital Signature Authentication for Android Smart Phones

Spandana.M[1], Bhaskar.T[2]

[1]M.Tech in CSE Dept , Ganapathy Engineering College
Warangal, ANDHRA PRADESH, INDIA
spandhana.m123@gmail.com

[2]Assistant Professor in CSE Dept, Ganapathy Engineering College
Warangal, ANDHRA PRADESH, INDIA
bhalu7cs@gmail.com

Abstract: ***Due to the innovations in Mobile technologies, Gestures play an important role for the Authentication in the mobile devices. With increasing popularity of the Authentications, the security risks are also growing and that is evident in the incidents of UnAuthentication in terms of various kinds. Therefore it is essential to have a mobile based Authentication system for security. Gestures allow users to interact with your app by manipulating the screen objects you provide. Security systems are rapidly being developed, as well as solutions such as remote control systems. However, even with these solutions, major problems could still result after a mobile device is lost. In this thesis, we present our upgraded Lock Screen system, which is able to support authentication for the user's convenience and provide a good security system for smartphones. We also suggest an upgraded authentication system for Android smartphones.***

KeyWords: - Gestures, Intents, Shared Preferences, Intent

1. INTRODUCTION

The devices most often used for IT services are changing from PCs and laptops to smart phones and tablets. These devices need to be small for increased portability. These technologies are convenient, but as the devices start to contain increasing amounts of important personal information, better security is required. Security systems are rapidly being developed, as well as solutions such as remote control systems. However, even with these solutions, major problems could still result after a mobile device is lost. In this thesis, we present our upgraded Lock Screen system, which is able to support authentication for the user's convenience and provide a good security system for smart phones. We also suggest an upgraded authentication system for Android smart phones.

ADW Launcher is Home Screen launcher to change the screen. This paper is to review the various studies that have explored the technical and legal issues associated with change the home screen of particular user. Users can download Home Launchers from the Android Market; the most commonly used ones are ADW Launcher, Launcher Pro, and GO Launcher. These systems provide convenience but not good security.

Mobile technologies and learning is primarily considered personal portable technologies. This review advocates an activity-focused perspective on the use of mobile technologies authentication for user. Reference to both emerging trends in mobile technology and learning research, we speculate on the future of mobile technologies and learning the implications this will have.

2. RELATED WORK

The Gestures Authentication in Android page flow

1. Dialog -> Save Email -> Shared Preferences
2. Enter Gesture -> Add and Edit Gestures -> Save in Spells.xml

3. User Mode -> Keep Signature using gestures -> Limited Access to Apps
4. Admin Mode -> Keep Signature using gestures -> Can Access to all Apps ->Add permission to user
5. Email-> Signature and the Code

3. PROPOSED WORK

1. We will analyze the problems with the current Home Launchers, Lock Screens and suggest an upgraded authentication system for Android smart phones.

2. The recorded pattern is send for the user mail id for future assistance

3. We present our upgraded Lock Screen system, which is able to support authentication for the user's convenience and provide a good security system for smart phones.

Gestures are defined by binary resources which can be created with an example program from the Android SDK. In your activity you can load Gestures via GestureLib.fromRawResource (). If a gesture is detected then the method "onGesturePerformedListener" is called. For this the activity must implement the interface "OnGesturePerformedListener" and must register itself at the GestureOverlayView with the method "addOnGesturePerformedListener()".

Android shows the gestures in yellow for recognized gestures and a ligher yellow for not recognized gestures. You can turn this off, via setGestureColor(Color.TRANSPARENT) or setUncertainGestureColor(Color.TRANSPARENT) on the GestureOverlayView.

If you create the gesture in the Android simulator via the program "GestureBuilder". You can create several gestures with the same name. That may help you to determine the right one. If you create an Android Emulator for Android 1.6 this application will be preinstalled on your device. Make sure to create a device with sdcard otherwise you cannot save gestures. All gestures will be saved in a file called *gestures* on your emulator.

You can copy the gestures from the emulator via the adb onto your local machine via the command:

./adb pull /sdcard/gestures ~/test

The gesture file must be copied into your application under "res/raw". Afterwards it can be used in your GestureOverlayView.

4. PROBLEM DEFINITION AND ASSUMPTIONS

Authentication is essential in Mobiles. How can we provide the authentication for the mobile devices? Because the biometric is work properly with the mobile devices which use android. How can we manage the Employees in an organization in a compact way? The problem of Authentication is solved by using gestures instead of the biometric and the compact version of employees can be managed by using the Employee details. In Employee details look for employees by name, view their details, add them to your contacts, and see their manager and direct reports, as well as call, text, or email them.

5. GESTURES DETECTION ALGORITHM

5.1. Creating a View : Gesture View

An easy way to get a Canvas object to drawing on is by overriding the onDraw() method of a View object. Conveniently, this method has a single parameter: the Canvas object. Drawing a Bitmap graphic on a Canvas object is as easy as calling the drawBitmap() method of the Canvas object.

```
private class GestureView extends View {
   private Matrix translate;
   private Bitmap droid;
   protected void onDraw(Canvas canvas) {
      canvas.drawBitmap(droid, translate, null);
      Matrix m = canvas.getMatrix();
      Log.d(DEBUG_TAG, "Matrix: "+translate.toShortString());
      Log.d(DEBUG_TAG, "Canvas: "+m.toShortString());
   }
}
```

Figure 1: Shows How to create a Gesture View

5.2. Configuring the Gesture View

A GestureDetector is an Android class that can take motion events, do some mathematical magic to determine what they are, and then delegate calls to a GestureListener object as specific gesture or other motion callbacks. The GestureListener object, a class we implement, receives these calls for specific gestures that the GestureDetector recognizes and allows us to react to them as we see fit (in this case, to move a graphic around within our PlayAreaView). Although the GestureDetector handles the detection of certain motions, it doesn't do anything specific with them nor does it handle all types of gestures.

```
public GestureView(Context context) {
   super(context);
   translate = new Matrix();
   gestures = new GestureDetector(GestureFunActivity.this,
         new GestureListener(this));
   droid = BitmapFactory.decodeResource(getResources(),
         R.drawable.droid_g);
}
```

Figure 2: Shows GestureView Constructor

5.3. Connecting to a Gesture Detector

GestureDetector object receives the motion data it needs to do its gesture recognizing magic. GestureDetector object called gestures to receive events.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
   return gestures.onTouchEvent(event);
}
```

Figure 3: Shows Gesture Detection Object

5.4. Implementing the Gesture Listener

In order to react to the events recognized by the GestureDetector class, we need to implement the GestureListener class. The motion events we are most interested in are double taps and gestures of any kind. To listen for these types of motion events, our GestureListener class must implement both the OnGestureListener and OnDoubleTapListener interfaces.

```
private class GestureListener implements
GestureDetector.OnGestureListener,
      GestureDetector.OnDoubleTapListener {
   GestureView view;
   public GestureListener(GestureView view) {
      this.view = view;
   }
}
```

Figure 4: shows Gesture Listener

5.5. Handling Motion Events

A scroll event occurs when the user touches the screen and then moves their finger across it. This gesture is also known as a drag event. This event comes in through the onScroll() method of the OnGestureListener interface.

```
@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2,
      float distanceX, float distanceY) {
   Log.v(DEBUG_TAG, "onScroll");
   view.onMove(-distanceX, -distanceY);
   return true;
}
```

Figure 5: Shows onScroll()

5.6. Shared Preferences

The Shared Preferences class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use Shared Preferences to save any primitive data: Booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).

Shared preferences are not strictly for saving "user preferences," such as what ringtone a user has chosen. If you're interested in creating user preferences for your application, Preference Activity, which provides an Activity framework for you to create user preferences, which will be automatically persisted (using shared preferences).

To get a Shared Preferences object for your application, use one of two methods:

- getSharedPreferences() - Use this if you need multiple preferences files identified by name, which you specify with the first parameter.
- getPreferences() - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

To write values:

1. Call edit() to get a SharedPreferences.Editor.
2. Add values with methods such as putBoolean() andputString().
3. Commit the new values with commit() To read values, use SharedPreferences methods such as getBoolean() and getString()..

6. IMPLEMENTATION

6.1. Admin

6.1.1. Add Gesture

The admin has the permission to add the Gesture signature of all the Employees and can view all the details of the employee.

Figure 6: Adding the Signature

Each and every Employee will have a particular signature. Based on the signature the employee login. The Employee signatures are added by the Admin.

6.2.User Login

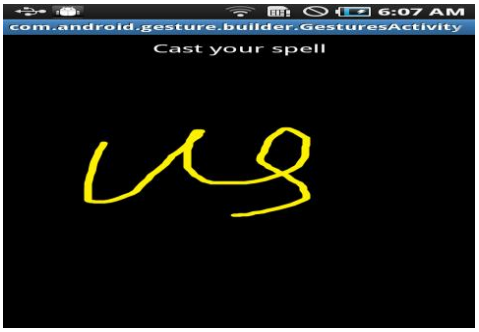

Figure 7: Login of the User

If the Employee Login Gesture Signature is correct then the Employee will be redirected to the next page where the employee can view the other employee details along with their details and their manager.

6.3. On/Off Broadcast receiver

The Lock Screen consists of activities, so it is included in the Home Launcher application package. The Screen receives the "On/Off Broadcast Receiver" so it is processed with Intent from Screen-On, activating the Lock Screen Activity.

6.4. Lock Screen Activity

The Lock Screen Activity consists of Password settings and, to unlock the screen, a password must first be entered. After setting the password, two options are available: a user can either enter the password or shake the mobile phone. The User mode can be entered by entering the password; the Guest mode can be entered by using the acceleration sensor (shaking).

6.5. User Mode and Guest Mode

Binding the Home key is different in Guest mode than it is in User mode. If the Home key button is entered in the Lock Screen, the Guest mode is automatically entered and the user can only use a few allowable applications.

6.6. Email pattern to mail

In the Home Launcher application, there are five slides and declared icons with widgets. The user can make widgets, icons, folders, and setting slides through various kinds of touches (touch, long-click, drag, home button, and menu button). If there are any changes or User/Guest processes, Home Launcher re-organizes the view.

7. CONCLUSION

In this thesis, we analyzed the problems in current HomeLaunchers' Lock Screens and suggested improving the authentication systems for Android smartphones. By dividing the mode of entry into the authentication system, we improved the user's convenience and security power. Android is being installed in tablets and many other IT devices that require good security systems. The use of our improved authentication system ensures protection of personal information.

8. REFERENCES

[1] ADW Launcher, AnderWeb, http://jbthemes.com/anderweb/
[2] LauncherPro, *http://www.launcherpro.com/*
[3] Go Launcher, Go Dev Team, *http://www.goforandroid.com/*
[4] DAI-Labor, *"Malicious Software for Smartphones," Technical Report, 2008.*
[5] Ken Dunham, *"Mobile Malware Attacks and Defense," SYNGRESS 2009, 2009.*
[6] Android Security Overview, Android open source project, *http://source.android.com/tech/security/index.html*
[7] Atrix, Motorola, *http://www.motorola.com*