CHAPTER 1

INTRODUCTION

1.1 Android

Android is an open source and Linux-based Operating System for mobile devices such as smart phones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development (Kit SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public.

Android Application

Android applications are usually developed in the Java language using the Android Software Development Kit. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play or the Amazon App store. Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide. This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

Service

Service is a background process that can run for a long time. There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

Content Providers are used to share data between the applications.

Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

It contains information's about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

1.2 Objective

Aim of the project is android based unlocking mobile waving pattern with Waving pattern for emergency support system for girl's safety.

1.3 Overview

Smartphones have become omnipresent platforms of personal computing for users to access the Internet and online services at anytime and anywhere. As more and more privacy information (e.g., text messages, emails, and contact list) and security information (e.g., passwords, CVS code of credit cards, and transaction information) are stored in smart phones, the risk of information leakage is becoming a major concern for the entire information society, especially with the consideration that the smart phones are much easier to get lost or stolen in comparison with conventional computing platforms, according to a recent survey on US state of Cybercrime.

The most common approach to address this problem is the use of authentication mechanisms, e.g., PIN-based and pattern-based pass codes, which have been integrated into smart phone systems like Android and iOS. Unfortunately, most smart phone users tend to choose simple and weak pass codes for the sake of convenience and memorability, and some recent studies have shown how simple an attacker can derive the PIN pass codes from the oily residues left on the screen or the pattern pass codes from the shoulder surfing attack. An attacker could even infer the pass codes from the accelerometer and gyroscope readings.

Therefore, it is highly desirable to enhance smart phone authentication with a passive and transparent authentication mechanism without active user involvement, to further detect whether the logged in user is the true owner of a smart phone. An on-going research project, the Active Authentication and Monitoring program initialized by DARPA (Defence Advanced Research Project Agency), aims to develop computational behavioural traits for validating the identity of the users in a meaningful and continual manner (without requiring the deployment of additional hardware sensors), through how users interact with the computing systems.

Of various potential solutions to this problem, a particularly promising technique is the use of touch-interaction behaviour. Compared with other biometric features on smart phones such as face and fingerprint, touch-interaction behaviour does not require specialized sensors to collect data, and the detection process can be integrated seamlessly into users' routine computing activities. Thus it can provide a non-intrusive and implicit solution for active authentication after entry-point based authentication by PIN-based or pattern-based pass codes, or could even substitute entry-point based authentication when reaching an acceptable level of performance.

Although there is a growing body of literature about touch-interaction behaviour for entry-point based authentication, there is little work on the use of this behaviour for active smart phone authentication. The major reasons may be the lack of in-depth analysis for various types of touch operations in terms of stability, discriminability, and usability for active authentication, and examination for its applicability across different application tasks and different application scenarios. Others might be the difficulty of extracting effective features or building reliable models from touch-interaction behaviour. We attempt to explore the reliability and applicability of users' touch-interaction behaviour for active smart phone.

CHAPTER 2

SYSTEM DESIGN AND IMPLEMENTATION

2.1 LITERATURE SURVEY

[1]Touch me once and i know it's you! Implicit authentication based on touch screen patterns

Alexander De Luca, Alana Hang, Frederick Brady, Christian Lindner, Heinrich Hussmann

Password patterns, as used on current Android phones, and other shape-based authentication schemes are highly usable and memorable. In terms of security, they are rather weak since the shapes are easy to steal and reproduce. In this work, we introduce an implicit authentication approach that enhances password patterns with an additional security layer, transparent to the user. In short, users are not only authenticated by the shape they input but also by the way they per-form the input.

We conducted two consecutive studies, a lab and a long-term study, using Android applications to collect and log data from user input on a touch screen of standard commercial smart phones. Analyses using dynamic time warping (DTW) provided first proof that it is actually possible to distinguish different users and use this information to increase security of the input while keeping the convenience for the user high.

[2]Smudge attacks on smartphone touch screens

Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith

Touch screens are an increasingly common feature on personal computing devices, especially smart phones, where size and user interface advantages accrue from consolidating multiple hardware components (keyboard, number pad, etc.)

into a single software definable user hardware components (keyboard, number pad, etc.) into a single software definable user interface. Oily residues, or smudges, on the touch screen surface, are one side effect of touches from which frequently used patterns such as a graphical password might be inferred.

Feasibility of such smudge attacks on touch screens for smart phones, and focus our analysis on the Android password pattern is examined. We first investigate the conditions (e.g., lighting and camera orientation) under which smudges are easily extracted. Finally, we provide a preliminary analysis of applying the information learned in a smudge attack to guessing an Android password pattern.

[3]Touchalytics: On the applicability of touchscreen input as a behavioural biometric for continuous authentication

Mario frank], raff biederty, eugene ma], Ivan martinovic, [dawn song]
Berkeley, German research Centre for artificial intelligence

We investigate whether a classifier can continuously authenticate users based on the Way they interact with the touch screen of a smart phone. We propose a set of 30 behavioural touch features that can be extracted from raw touch screen logs and demonstrate that different users populate distinct subspaces of this feature space. In a systematic experiment designed to test how this behavioural pattern exhibits consistency over time, we collected touch data from users interacting with a smart phone using basic navigation manoeuvres, i.e., up-down and left-right scrolling.

A classification framework that learns the touch behaviour of a user during an enrolment phase and is able to accept or reject the current user by monitoring interaction with the touch screen. The classifier achieves a median equal error rate of 0% for intra-session authentication, 2%-3% for inter-session authentication and below 4% when the authentication test was carried out one week after the enrolment phase. While our experimental findings disqualify this method as a standalone authentication mechanism for long-term authentication, it could be implemented as a means to extend screen-lock time or as a part of a multi-modal biometric authentication system.

[4]Performance Analysis of Touch-Interaction Behavior for Active Smartphone Authentication

Chao Shen, Member, IEEE, Yong Zhang, Xiaohong Guan, Fellow, IEEE, and Roy A. Maxion, Fellow, IEEE

The increasing use of touchscreen smartphones to access sensitive and privacy data has given rise to the need of secure and usable authentication technique. Smartphone users have their own unique behavioural characteristics when performing touch operations. These personal characteristics are reflected on different rhythm, strength, and angle preferences of touch interaction behaviour. This paper investigates the reliability and applicability on the usage of users' touch-interaction behaviour for active authentication on smartphones. For each common type of touch operations, both static and dynamic features are extracted and analysed for fine-grained characterization of users' touch behaviour. Classification techniques (nearest neighbour, neural network, support vector machine, and random forest) are applied to the feature space for performing the task of active authentication.

The extensive experimental results are included to show that touch-interaction behaviour exhibits sufficient discriminability and stability among smartphone users for active authentication, and achieves equal-error rates between 1.72% and 9.01% for different types of touch operations with the operation length of 11; the authentication accuracies improve when having long observation or small timespan between the training and testing phases, and express more reliably and stably in a specific task than in the free task. We also discuss a number of avenues for additional research that we believe is necessary to advance the state-of-the-art in this area.

[5]The science of guessing: analysing an anonmyzed corpus of 70 million passwords
Joseph Bonneau Computer Laboratory University of Cambridge

Report on the largest corpus of user-chosen passwords ever studied, consisting of anonym zed password histograms representing almost 70 million Yahoo! users, mitigating privacy concerns while enabling analysis of dozens of subpopulations based on demographic factors and site usage characteristics. This large data set motivates a thorough statistical treatment of estimating guessing difficulty by sampling from a secret distribution. In place of previously used metrics such as Shannon entropy and guessing entropy, which cannot be estimated with any realistically sized sample, we develop partial guessing metrics including a new variant of guesswork parameterized by an attacker's desired success rate engineering. Paring password distributions with a uniform distribution which would provide equivalent security against different forms of guessing attack, we estimate that passwords provide fewer than 10 bits of security against an online, trawling attack, and only about 20 bits of security against an optimal offline dictionary attack. We find surprisingly little variation in guessing difficulty.

2.2 SYSTEM ANALYSIS

2.2.1 Existing System

Unfortunately, most smart phone users tend to choose simple weak pass codes for the sake of convenience and memorability. Smart phone users have their own unique behavioural characteristics when performing touch operations .These personal characteristics are reflected on different rhythm, strength and angle preferences of touch interaction behaviour.

2.2.1.1 Disadvantages

- There is no security and alert system
- There is no pattern recognition for emergency system

2.2.2 Proposed System

Investigates the reliability and applicability on the usage of users' touch-interaction behaviour for active authentication on smart phones. Smartphone users have their own unique behavioural characteristics when performing touch operations. Android Application is developed in which user's Hand Waving Pattern is recorded & Stored as User's Pattern. We are using SVM Algorithm for User Identification.

2.2.2.1 Advantages

- High reliability
- Pattern recognition for emergency
- Emergency automatic alert to police and guardian

2.3 SYSTEM SPECIFICATION

This chapter describes the requirement analysis in accordance with the input and the resources and it also describes the implementation of the project with the technology used.

2.3.1 Hardware Requirements

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented.

- Processor: Core i3/i5/i7
- RAM : 2-4GB
- HDD : 500 GB

2.3.2 Software Requirements

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

- Platform : Windows XP/7/8
- Front End : Java-JDK1.7,Android-sdk and Eclipse, Apache TC
- Back End : MYSQL

2.4 SOFTWARE DESCRIPTION

2.4.1 Java

The Java platform is the ideal platform for network computing Running across all platforms -- from servers to cell phones to smart cards -- Java technology unifies business infrastructure to create a seamless, secure, networked platform for your business. The Java platform benefits from a massive community of developers and supporters that actively work on delivering Java technology-based products and services as well as evolving the platform through an open, community-based, standards organization known as the Java Community Process program.

2.4.2 Apache Tomcat

Tomcat, developed by Apache (www.apache.org), is a standard reference implementation for Java servlets and JSP. It can be used standalone as a Web server or be plugged into a Webserver like Apache, Netscape Enterprise Server, or Microsoft Internet Information Server. There are many versions of Tomcat.

The Server

The first container element referenced in this snippet is the <Server> element. It represents the entire Catalina servlet engine and is used as a top-level element for a single Tomcat instance. The <Server> element may contain one or more <Service> containers.

The Service

The next container element is the <Service> element, which holds a collection of one or more <Connector> elements that share a single <Engine>

element. N-number of <Service> elements may be nested inside a single <Server> element.

The Connector

The next type of element is the <Connector> element, which defines the class that does the actual Handling requests and responses to and from a calling client application.

The Engine

The third container element is the <Engine> element. Each defined <Service> can have only one <Engine> element and this single <Engine> component handles all requests received by all of the defined <Connector> components defined by a parent service.

The Host

The <Host> element defines the virtual hosts that are contained in each instance of a Catalina <Engine>. Each <Host> can be a parent to one or more web applications, with each being represented by a <Context> component.

The Context

The <Context> element is the most commonly used container in a Tomcat instance. Each <Context> element represents an individual web application that is running within a defined <Host>. There is no limit to the number of contexts that can be defined within a <Host>.

Java Web Applications

The main function of the Tomcat server is to act as a container for Java web applications. The concept of a web application was introduced with the release of the Java servlets specification 2.2. According to this specification, “a web application is a collection of servlets, html pages, classes, and other resources that can be bundled and run on multiple containers from multiple vendors.” What this really means is that a web application is a container that can hold any combination of the following list of objects:

- Servlets
- Java Server Pages (JSPs)
- Utility classes
- Static documents, including HTML, images, JavaScript libraries,
- Client-side classes
- Meta-information describing the web application

One of the main characteristics of a web application is its relationship to the Servlets Context. Each web application has one and only one Servlets Context. This relationship is controlled by the servlets container and guarantees that no two web applications will clash when accessing objects in the Servlets Context.

2.4.3 MYSQL DATABASE MANAGEMENT SYSTEM

My SQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The My SQL Web site (http://www.mysql.com/) provides the latest information about My SQL software.

- My SQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by My SQL AB. My SQL AB is a commercial company, founded in 1995 by the My SQL developers. It is a second generation Open Source company that unites Open Source values and methodology with a successful business model.

- The My SQL® software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. My SQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. My SQL is a registered trademark of My SQL AB.
- The My SQL software is Dual Licensed. Users can choose to use the My SQL software as an Open Source product under the terms of the GNU General Public License or can purchase a standard commercial license from My SQL AB.

The company name was given after co-founder Monty Widenius's daughter, My, the SQL is “Structured Query Language.”, AB is Sweedish for limited partnership.

MySQL Architecture

Three-layer model:

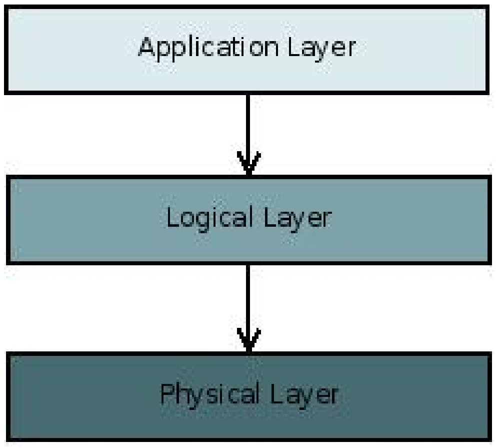


Fig 2.4.3.1: MYSQL Architecture

- The application layer contains common network services for connection handling, authentication and security. This layer is where different clients interact with MySQL these clients can write in different API's:.NET, Java, C, C++, PHP, Python, Ruby, Tcl, Eiffel, etc...

- The Logical Layer is where the MySQL intelligence resides, it includes functionality for query parsing, analysis, caching and all built-in functions (math, date.). This layer also provides functionality common across the storage engines.

The Physical Layer is responsible for storing and retrieving all data stored in “MySQL”. Associated with this layer are the storage engines, which MySQL interacts with very basic standard API's. Each storage engine has it strengths and weakness, some of this engines are MyISAM, InnoDB, CSV, NDB Cluster, Falcon, etc.

SOME INTERNAL COMPONENTS:

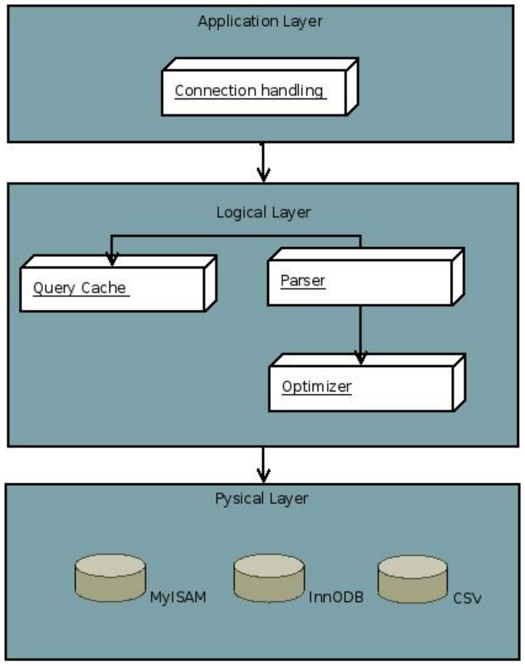


Fig 2.4.3.2: Internal Components

Each client connection gets its own thread within the server process. When clients (applications) connect to the MySQL server, the server needs to authenticate them.

Before even parsing the query, though, the server consults the query cache, which only stores SELECT statements, along with their result sets.

The storage engine does affect how the server optimizes query.

Storage Engines:

- MySQL supports several storage engines that act as handlers for different table types. MySQL storage engines include both those that handle transaction-safe tables and those that handle non-transaction-safe tables.

2.5 SYSTEM ARCHITECTURE

2.5.1Architecture Diagram

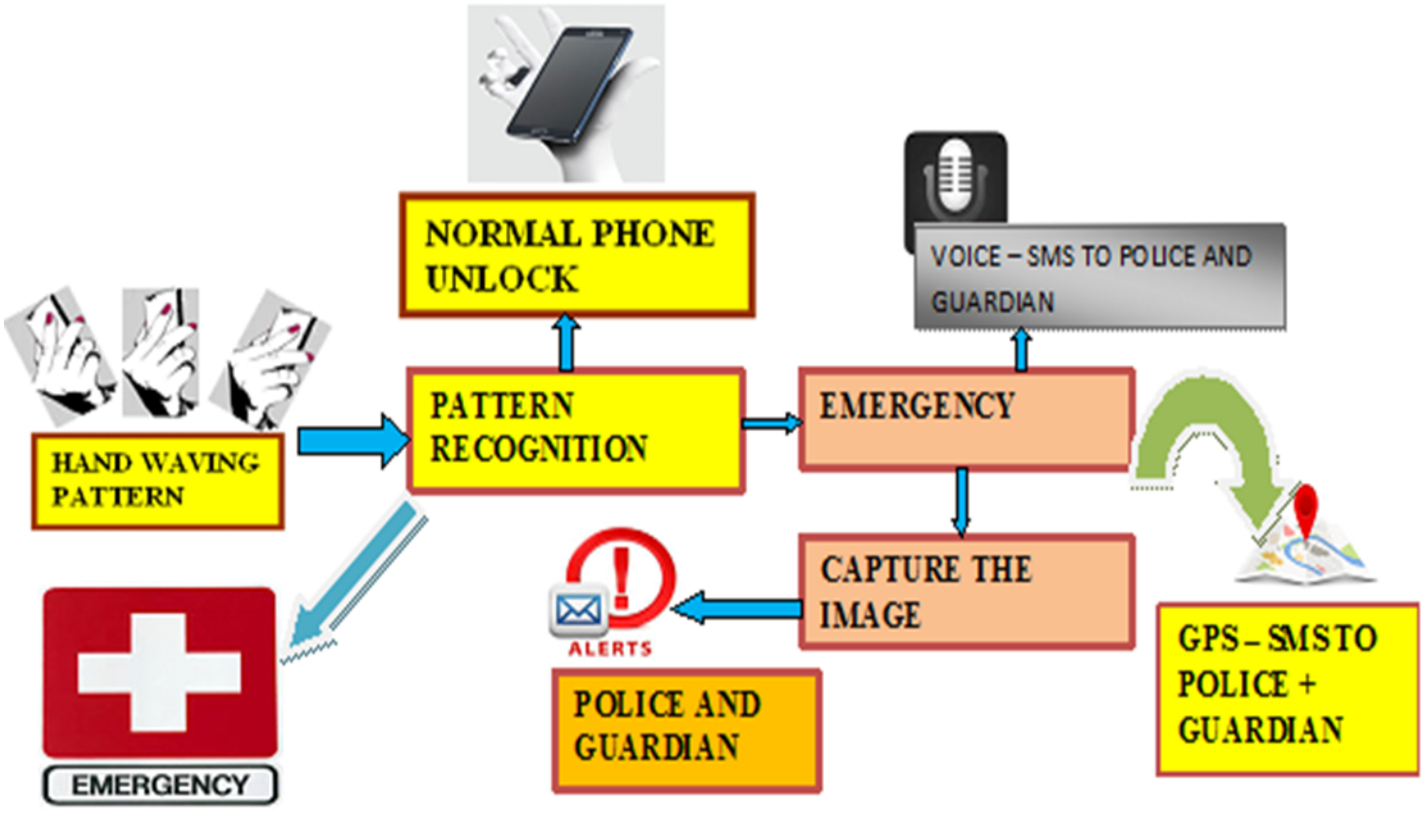


Fig 2.5.1: *Architecture Diagram*

2.5.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system. DFDs can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.

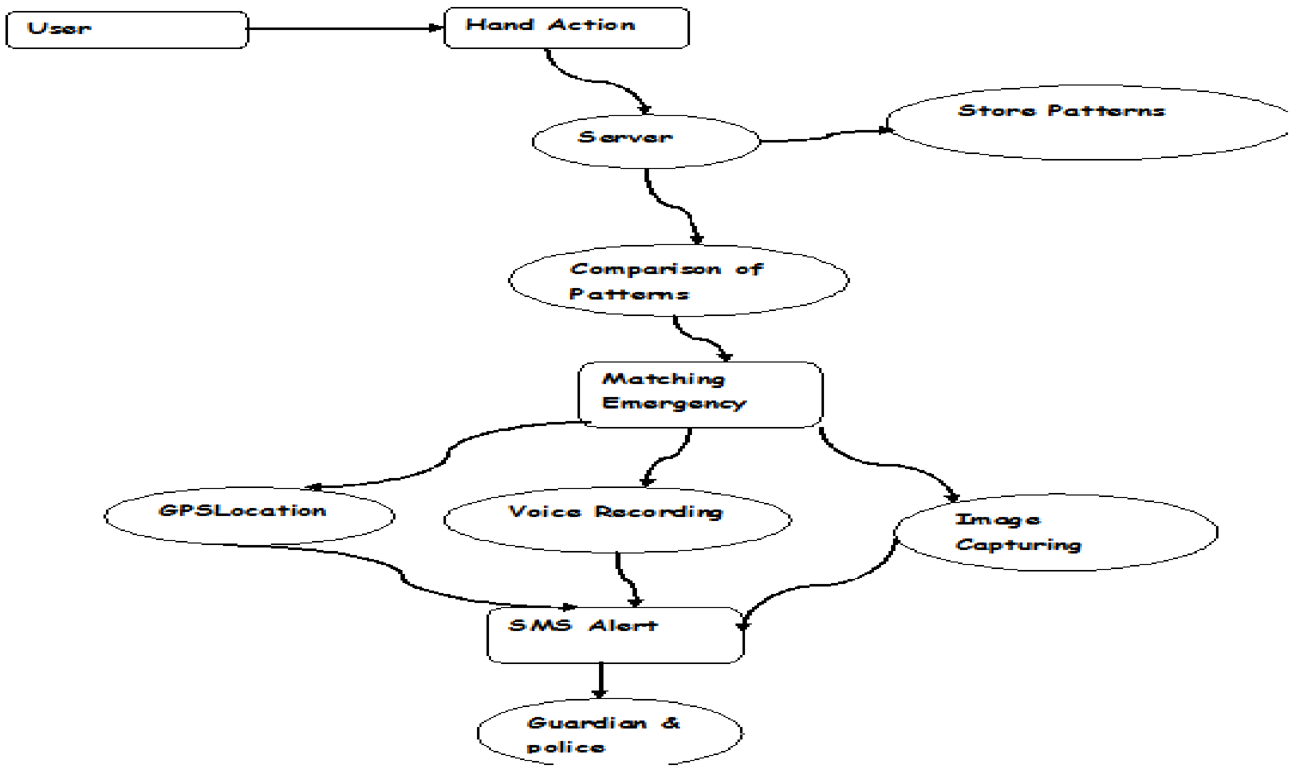


Fig.2.5.2: Data Flow Diagram

2.5.3 Use Case Diagram

Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provides something of measurable value to an action and is drawn as a horizontal ellipse. Use case diagrams are behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

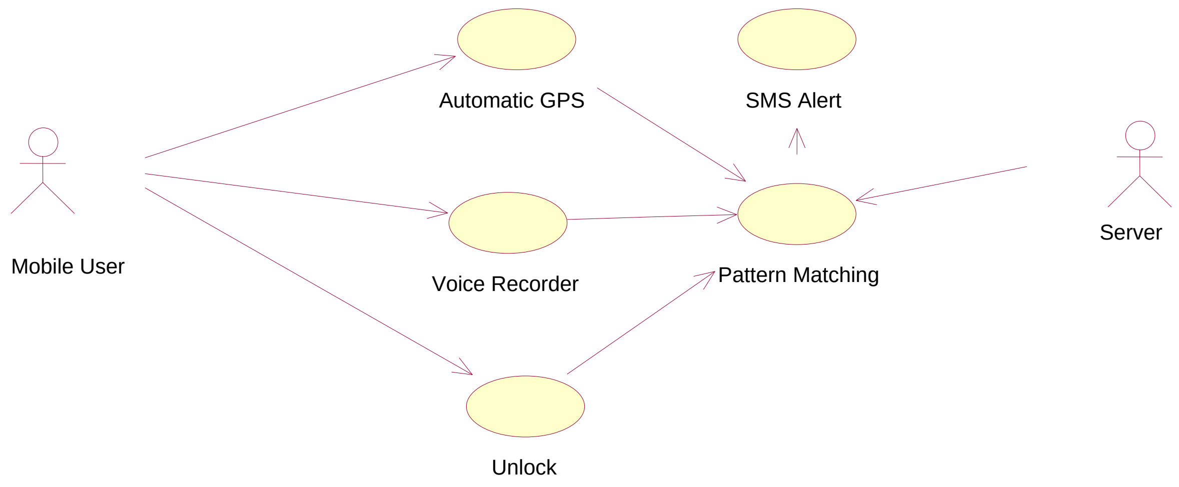


Fig.2.5.3: Use Case Diagram

2.5.4 Sequence Diagram

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system.

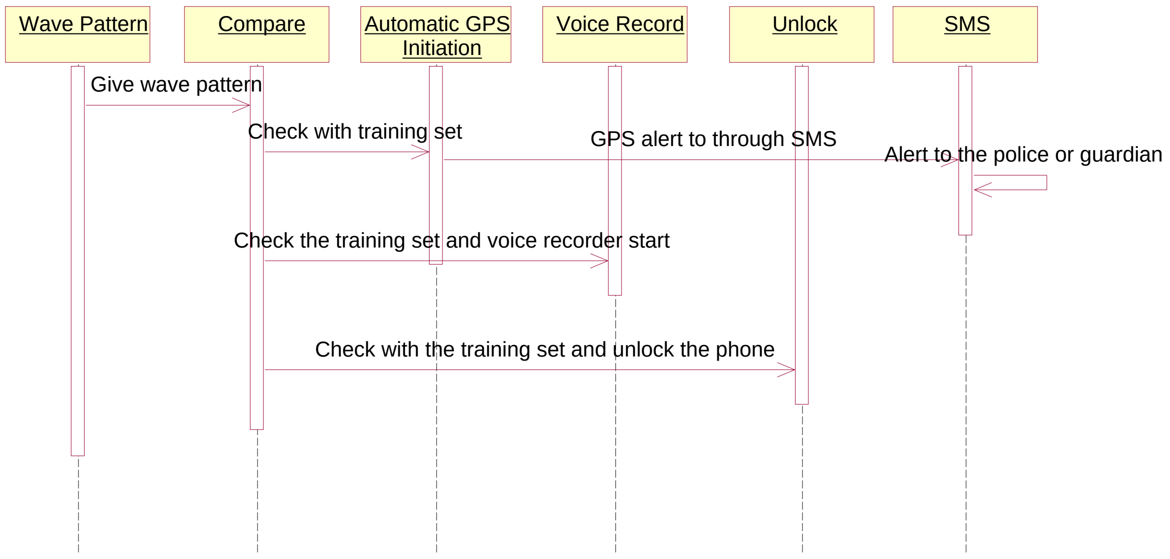


Fig.2.5.4: Sequence Diagram

2.5.5 Collaboration Diagram

Another type of interaction diagram is the collaboration diagram. A collaboration diagram represents a collaboration, which is a set of objects related in a particular context, and interaction, which is a set of messages exchange among the objects within the collaboration to achieve a desired outcome.

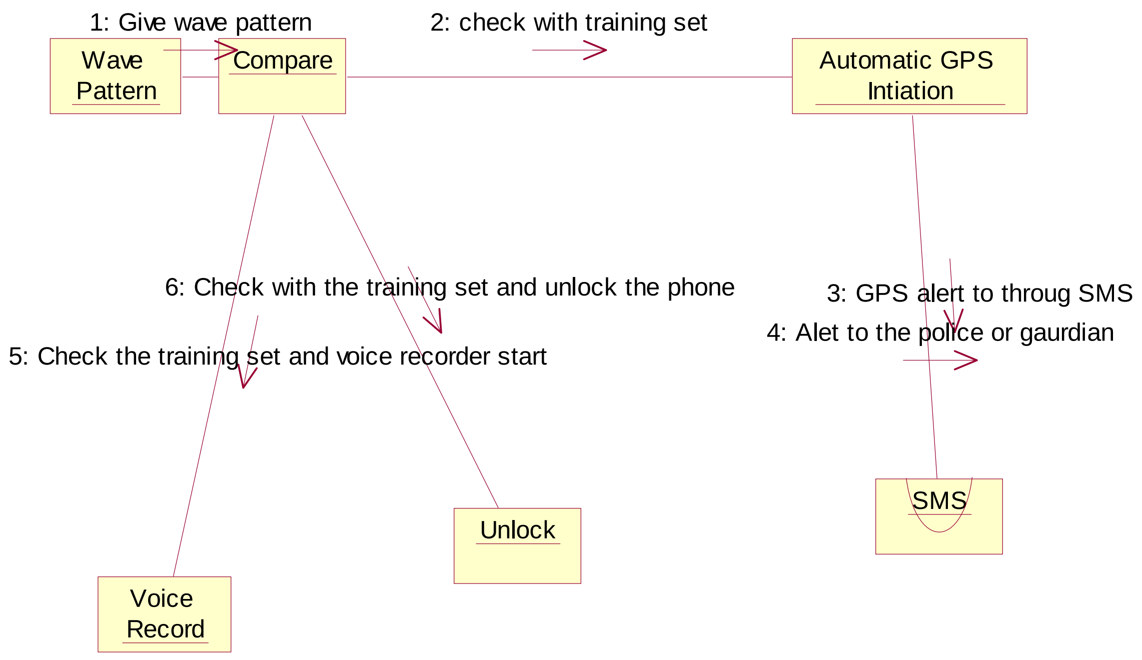

1: Give wave pattern
2: check with training set
Wave Pattern
Compare
Automatic GPS Intiation
6: Check with the training set and unlock the phone
3: GPS alert to throug SMS
4: Alet to the police or gaurdian
5: Check the training set and voice recorder start
SMS
Unlock
Voice Record

Fig.2.5.5: Collaboration Diagram

2.5.6 Activity Diagram

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

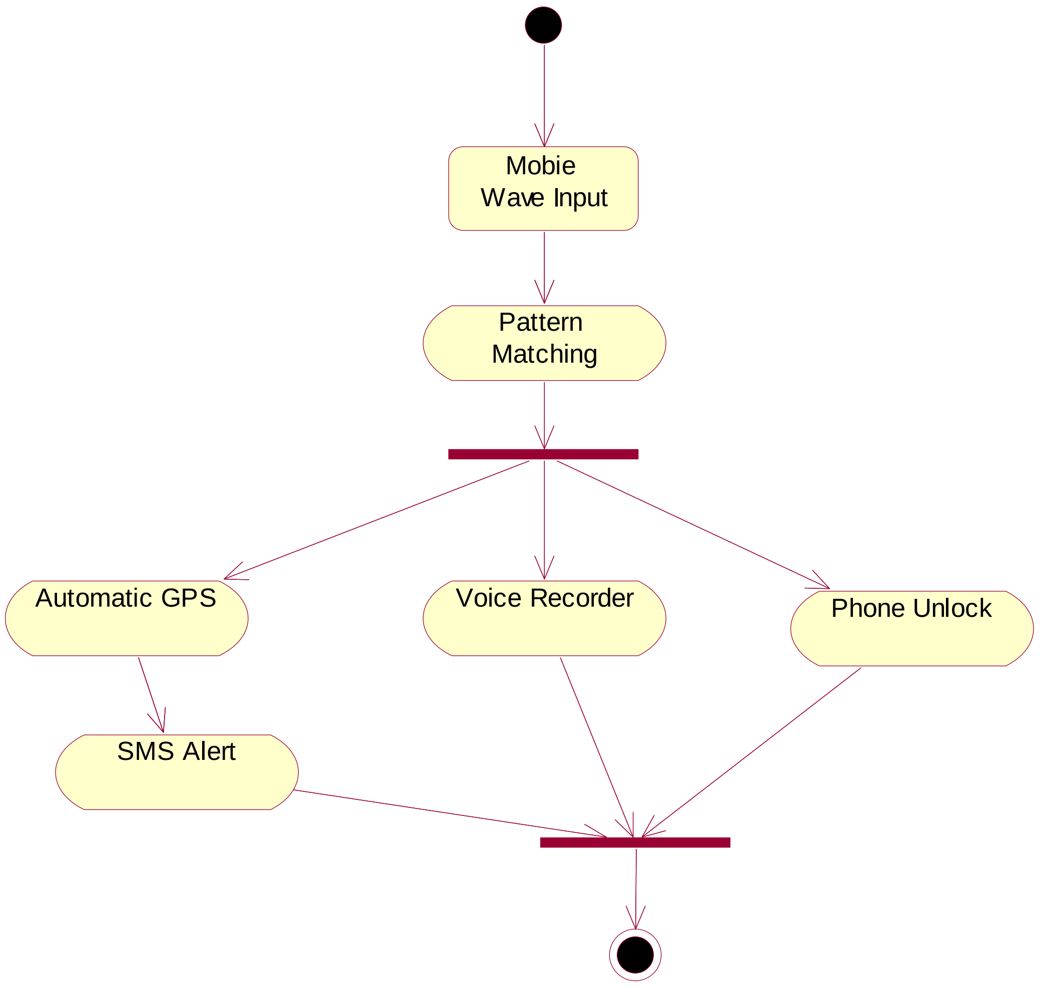


Fig.2.5.6: Activity Diagram

2.6 MODULES

2.6.1 List Of Modules

- ❖ Android User
- ❖ Server Deployment
- ❖ Pattern Registration
- ❖ Pattern Emergency Matching
- ❖ GPS Based Location identification

2.6.2 Modules Description

2.6.2.1Android User

Develop an android application. Mobile Client is an Android application which created and installed in the User's Android Mobile Phone. So that we can perform the activities. Application's First Page Consists of the User registration Process. We'll create the User Login Page by Button and Text Field Class in the Android. While creating the Android Application, we have to design the page by dragging the tools like Button, Text field, and Radio Button. Once we designed the page we have to write the codes for each. Once we create the full mobile application, it will generate as Android Platform Kit (APK) file. This APK file will be installed in the User's Mobile Phone as an Application.

2.6.2.2 Server Deployment

The Server will monitor the entire User's information in their database and verify them if required. Also the Server will store the entire User's information in their database and the Server has to establish the connection to communicate with the Users. The Server will update the each User's activities in its database.

The Server will authenticate each user before they access the Application. So that the Server will prevent the Unauthorized User from accessing the Application.

2.6.2.3 Pattern Registration

In this module we create an emergency matching system i.e. the user will be registering their ordinary unlock pattern and their emergency unlock pattern. Ordinary pattern is given as BAC i.e. user should wave the mobile in BAC direction to unlock their mobile. Emergency pattern is given as CAB i.e. user should wave their mobile in CAB, when the user is in the emergency situation.

2.6.2.4 Pattern Emergency Matching

In this module we create an emergency matching system i.e. when the user is in the emergency condition she will be waving the mobile. Then the pattern will be matched with the pattern which is already registered by the user. If the pattern is matched with the ordinary pattern the mobile will be unlocked or else if it matches with the emergency pattern the user can be able to rescue them from emergency situation.

2.6.2.5 GPS Based Location Identification

In this module we design an emergency support system by using GPS, when the user is in the bad circumstance they can wave their mobile if it is an emergency pattern then automatically front camera will be initiated and the voice will be recorded and also GPS value is triggered and the location with the latitudinal and longitudinal value will be send as SMS, so that person can be saved.

2.7 TESTING

2.7.1 Software Testing

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

2.7.2 Testing Techniques

There are two basics of software testing: Black box testing and white box testing.

2.7.2.1 White Box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

2.7.2.2 Black Box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

2.7.3 Software Testing Strategies

There are many types of testing like:

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing0
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

2.7.3.1 Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

2.7.3.2 Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

2.7.3.3 Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

2.7.3.4 System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment.

CHAPTER 3

CONCLUSION AND FUTURE ENHANCEMENT

3.1 Conclusion

This paper explores touch-interaction behaviour based active authentication under various application scenarios. The results showed that touch-interaction behaviour, under the scenarios which have long observation in the model-training phase or small time span between the model-training phase and detection phase would produce good and robust authentication performance. But these conditions may constrain the flexibility of this mechanism in some real-world application scenarios. One possible way is to employ effective online incremental learning strategy to continuously learn the new-coming touch operations, and to increasingly enhance the validity and flexibility of the authentication model.

3.2 Future Enhancement

In the future system Android Application is developed in which user's Hand Waving Pattern is recorded and when the user is in the emergency situation GPS value will be automatically triggered without the help of user.

Alert system can be improved by pressing any other button in the mobile before the time ends.

APPENDIX 1
SOURCE CODE

Girl’s safety:

```
package com.android.src.handwaving;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.telephony.SmsManager;

import android.view.KeyEvent;

import android.widget.Toast;

public class Girlssafety extends Activity {

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

Toast.makeText(this, "Girls safety started",

Toast.LENGTH_SHORT).show();

}

public boolean onKeyDown(int keyCode, KeyEvent event) {

switch(keyCode){

case KeyEvent.KEYCODE_VOLUME_UP:

Toast.makeText(this, "Volume Up pressed",

Toast.LENGTH_SHORT).show();

SmsManager smsManager = SmsManager.getDefault();

                    smsManager.sendTextMessage(HandWavingEntity.getGuardianNo(),null,"I am
In Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/IMG.jpg",null,
null);

                    smsManager.sendTextMessage(HandWavingEntity.getPoliceNo(),null,"I am In
Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/IMG.jpg",null,
null);

 Intent intent = new Intent(Girlssafety.this,PreviewActivity.class);

 startActivity(intent);

 event.startTracking();

 return true;

 case KeyEvent.KEYCODE_VOLUME_DOWN:

 Toast.makeText(this,"Volumen Down pressed",

 Toast.LENGTH_SHORT).show();

 SmsManager smsManager1 = SmsManager.getDefault();

 smsManager1.sendTextMessage(HandWavingEntity.getGuardianNo(),null,"I am   In Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/IMG.jpg",null,
null);

smsManager1.sendTextMessage(HandWavingEntity.getPoliceNo(),null,"I am In Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/IMG.jpg",null,
null);

  Intent intent1 = new Intent(Girlssafety.this,PreviewActivity.class);

 startActivity(intent1)

return true;}

return super.onKeyDown(keyCode, event);}

public boolean onKeyUp(int keyCode, KeyEvent event) {

switch(keyCode){

case KeyEvent.KEYCODE_MENU:

Toast.makeText(this, "Menu key released", Toast.LENGTH_SHORT).show();

return true;

case KeyEvent.KEYCODE_SEARCH:

Toast.makeText(this, "Search key released", Toast.LENGTH_SHORT).show();

return true;

case KeyEvent.KEYCODE_VOLUME_UP:
 if(event.isTracking() && !event.isCanceled())
Toast.makeText(this, "Volume Up released", Toast.LENGTH_SHORT).show();
 return true;
 case KeyEvent.KEYCODE_VOLUME_DOWN:
 Toast.makeText(this, "Volume Down released", Toast.LENGTH_SHORT).show();
 return true;
 }
 return super.onKeyDown(keyCode, event);
 }
  public boolean onKeyLongPress(int keyCode, KeyEvent event) {
  Toast.makeText(this, "Pressed for a long time =) ",
  Toast.LENGTH_SHORT).show();
  return true;
  }
  public void onBackPressed() {
  Toast.makeText(this, "Back key pressed =)", Toast.LENGTH_SHORT).show();
  super.onBackPressed();
  return false;}
  public void onBackPressed() {
  Toast.makeText(this, "Back key pressed =)", Toast.LENGTH_SHORT).show();
   return;
    }
    }
```

Audio record:

```
package com.android.src.handwaving;
import java.io.IOException;
import android.app.Activity;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.Socket;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class AudioRecord extends Activity {
private MediaRecorder myAudioRecorder
private String outputFile = null;
private Button start,stop,play;
private Socket client;
private FileInputStream fileInputStream;
private OutputStream outputStream = null;
private InputStream inStream = null;
protected void onCreate(Bundle savedInstanceState)
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
outputFile = Environment.getExternalStorageDirectory().
getAbsolutePath() + "/AudioRecording.3gp";;
myAudioRecorder = new MediaRecorder();
myAudioRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
myAudioRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
myAudioRecorder.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB);
myAudioRecorder.setOutputFile(outputFile);
try {myAudioRecorder.prepare();
myAudioRecorder.start();
```

```
Thread.sleep(12000);
myAudioRecorder.stop();
myAudioRecorder.release();
myAudioRecorder  = null;
try{
 String reply = getServerCommunication("AUDIO",outputFile);
 if(reply!=null && reply.startsWith("STORED")){
 SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(HandWavingEntity.getGuardianNo(),null,"I am In
Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/
AudioRecording.3gp",null,null);

smsManager.sendTextMessage(HandWavingEntity.getPoliceNo(),null,"I am In
Trouble...
"+"http://"+HandWavingEntity.getIpaddress()+":8080/Handwaving/StoredImage/
AudioRecording.3gp",null,null);
finish();
Intent intent = new Intent(AudioRecord.this,GPSLocation.class);
startActivity(intent);
}else{
Toast.makeText(getApplicationContext(), "Image not stored in
server",Toast.LENGTH_LONG).show();}\
}catch(Exception ex
ex.printStackTrace();
System.out.println(ex)
}} catch (IllegalStateException e) {e.printStackTrace
```

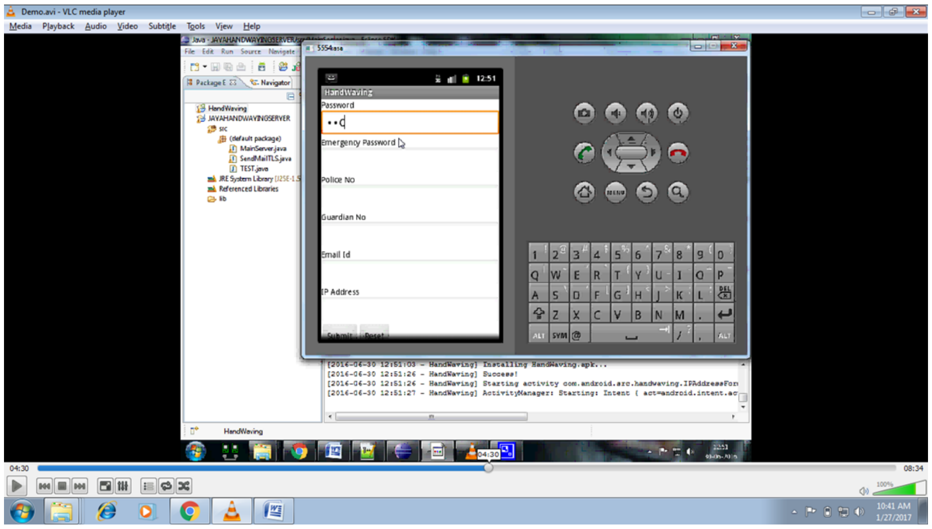
```
} catch (IOException e) {
e.printStackTrace();
}catch(Exception ex){
ex.printStackTrace();
}finally{
 Intent intent = new Intent(VoiceRecording.this,GPSLocation.class);
startActivity(intent);
}} public void start(View view){
try {
myAudioRecorder.prepare()
 myAudioRecorder.start
} catch (IllegalStateException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();}
start.setEnabled(false);
stop.setEnabled(true);
Toast.makeText(getApplicationContext(),"Recordingstarted",
Toast.LENGTH_LONG).show();}
 public void stop(View view){
 myAudioRecorder.stop();
 myAudioRecorder.release();
 myAudioRecorder  = null;
 stop.setEnabled(false);
 play.setEnabled(true);
 Toast.makeText(getApplicationContext(), "Audio recorded successfully",
 Toast.LENGTH_LONG).show();
 }public void play(View view) throws IllegalArgumentException,
 SecurityException, IllegalStateException, IOException{
 MediaPlayer m = new MediaPlayer();
 m.setDataSource(outputFile);
 m.prepare();
 m.start();
Toast.makeText(getApplicationContext(),                    "Playing                    audio",
Toast.LENGTH_LONG).show();}
public String getServerCommunication(String action,String imagePath){
String replyString = null;
File file = new File(imagePath);
try {client = new Socket( HandWavingEntity.getIpaddress(),4444);
ObjectOutputStream out = new ObjectOutputStream(client.getOutputStream());
fileInputStream = new FileInputStream(file);
byte[] mybytearray = new byte[(int) file.length()];
fileInputStream.read(mybytearray);
fileInputStream.close();
if(action.equals("AUDIO")){
out.writeObject(("AUDIO").getBytes());
out.writeObject(mybytearray);
}ObjectInputStream oin = new ObjectInputStream(client.getInputStream());
replyString = new String((byte [])oin.readObject());
```

```
 Toast.makeText(getApplicationContext(), "Reply String..."+replyString,
Toast.LENGTH_LONG).show();

 }catch(Exception ex){

 ex.printStackTrace();

 System.out.println(ex);

 } return replyString;
```

APPENDIX 2

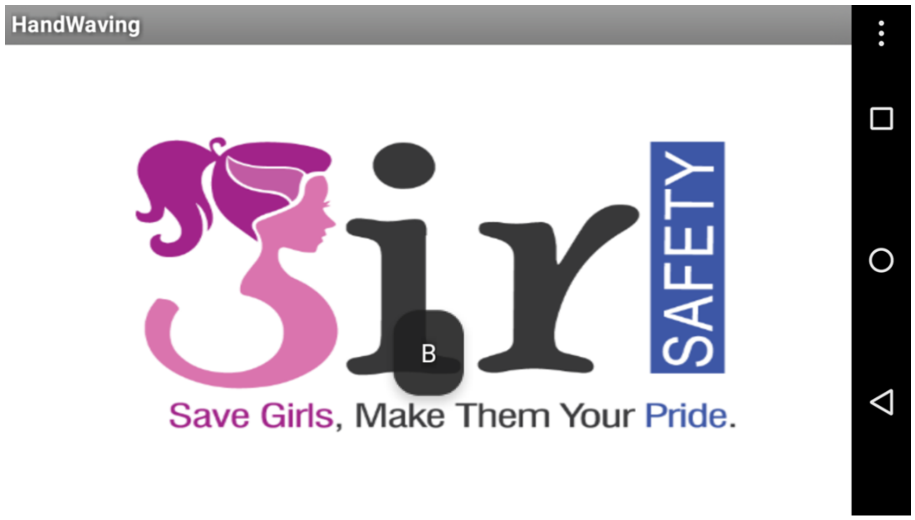
SCREENSHOTS

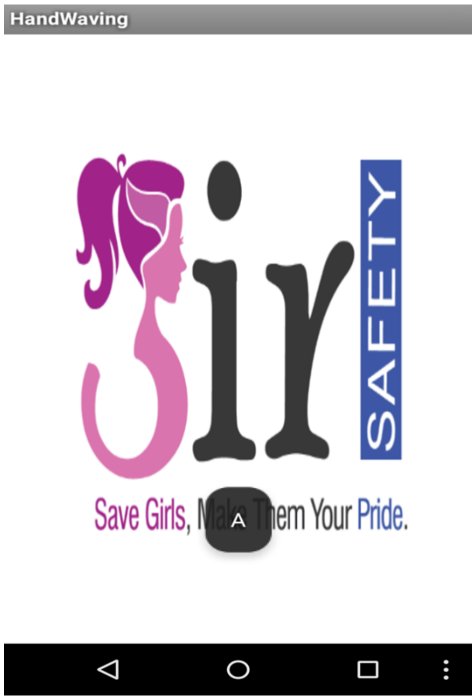
REGISTRATION

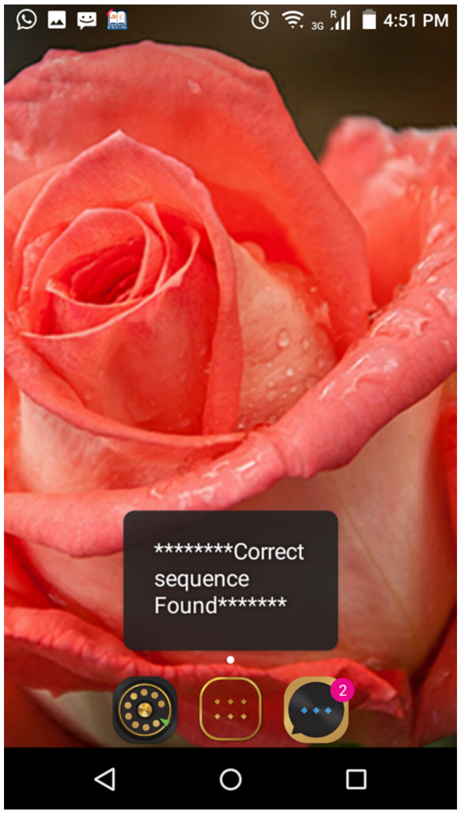


```
[2016-06-30 12:51:03 - HandWaving] Installing HandWaving.apk...
[2016-06-30 12:51:26 - HandWaving] Success!
[2016-06-30 12:51:26 - HandWaving] Starting activity com.android.src.handwaving.IPAddressFor
[2016-06-30 12:51:27 - HandWaving] ActivityManager: Starting: Intent { act=android.intent.ac
```

HANDWAVING ACT UNLOCKING

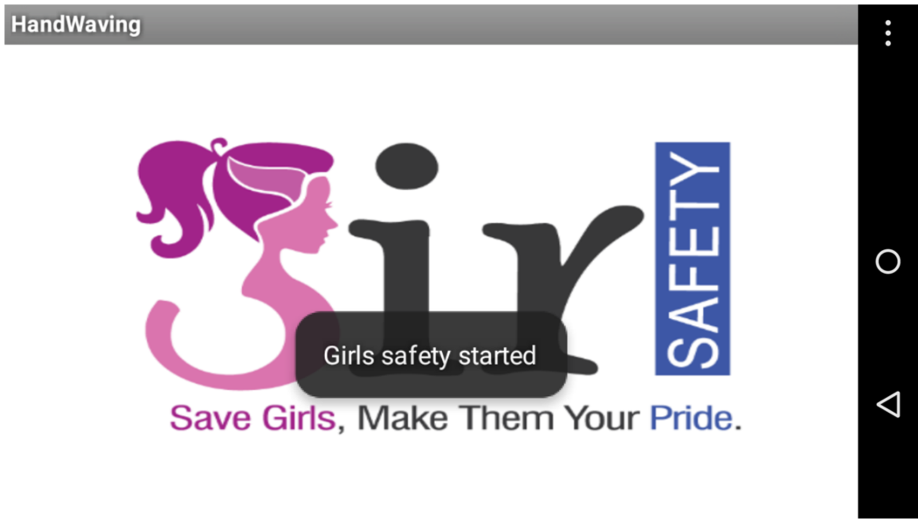


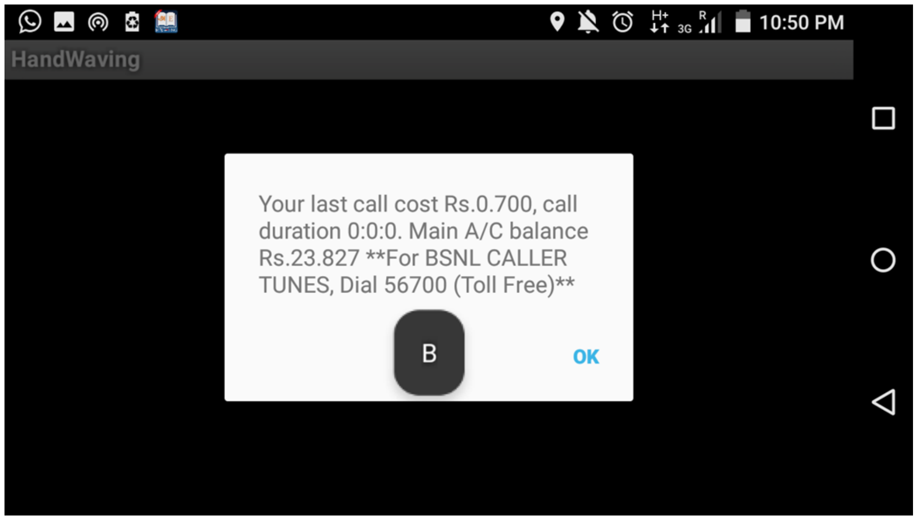




********Correct sequence Found*******

HANDWAVING ACT EMERGENCY





Your last call cost Rs.0.700, call duration 0:0:0. Main A/C balance Rs.23.827 **For BSNL CALLER TUNES, Dial 56700 (Toll Free)**

REFERENCES

1. A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, 'Smudge attacks on smartphone touch screens', in Proc. 4th USENIX Conf. Offensive Technol., Washington, DC, USA, (2010)

2. 'Active Authentication', document DARPA-BAA-12-06,Defence Advanced Research Projects Agency, Arlington, VA, USA, (2012).

3. J. Bonneau, 'The science of guessing: Analysing an anonym zed corpus of 70 million passwords,' in Proc. IEEE Symp. Secure, Privacy, May (2012)

4. Chao Shen, Member, IEEE, Yong Zhang, Xiaohong Guan, Fellow, IEEE and Roy A.Maxion, Fellow, IEEE.'Performance Analysis of Touch-Interaction Behavior for Active Smartphone Authentication,'Mar(2016)

5. M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, 'Touchalytics: On the applicability of touchscreen input as a behavioural biometric for continuous authentication,' IEEE Trans. Inf. Forensics Security, vol. 8, no. 1, pp. 136–148, Jan(2013).

6. E.Owusu, J. Han, S. Das, A.Perrig, and J. Zhang, 'Accessory: Password inference using accelerometers on smartphones,' in Proc. 12th Workshop Mobile Compute. Syst. Appl., (2012)