Biometric System Design for Handheld Devices

Naif Alotaibi, Richard Barilla, Francisco Betances, Aditya Chohan, Alexander Gazarov, Mantie Reid, Alexandra Scolaro, and Vinnie Monaco

Seidenberg School of CSIS, Pace University, White Plains New York

Abstract—The technical devices people use, and in particular such devices as smartphones and tablets, give us new ways to work with information, but at the same time make people vulnerable to unauthorized access. Biometrics based on device related input is a relatively new area which helps to respond to and limit those threats. The purpose of this study is to devise a novel biometric modality system to identify and authenticate users. The biometric system consists of unique components of biometric data based on common user gestures like scrolling, pinch to zoom, and clicking. This paper includes the study of additional data from sensors which are becoming increasingly ubiquitous in mobile devices. It provides a description of the architecture of the biometric input capturing system, the feature vector, and evaluates the performance of the system.

Index Terms—biometrics, pattern recognition, touch biometric, device motion, device orientation, sensors, user authentication, user identification, mobile devices

I. INTRODUCTION

Mobile phones, tablets, and other handheld devices have become increasingly important in our society for both recreational use and business use. According to the Pew Research Internet Project Mobile act Sheet, 90% of American adults have a cell phone and 42% of American adults own a tablet computer. 67% of cell phone owners find themselves checking their phones for messages, alerts, or calls — even when they did not notice their cell phones ringing or vibrating. As of May 2013, 63% of adult mobile phone users used their phones to go on line while 34% of mobile internet users go online mostly using their mobile phones and not using some other devices such as a desktop or laptop computer [12]. The increasing dependency on mobile devices creates vulnerability in unauthorized access and therefore creates a need for more than a one-factor unit password authentication in the form of password, pin, or finger touch swipe gesture pattern to keep vital information protected from unauthorized users. Mobile phones and tablets capabilities allow individuals to work away from their desks. This includes telecommuting, working from client locations, planes, trains, or just moving freely around with the use of a wireless network. In a recent survey of small business owners, 94% of them believe that their mobile devices make them more efficient. IT managers reported that 89% of their employees use mobile devices for work, but just 51% of the managers say that their company has an effective strategy for managing the security of the devices [10]. The key to making mobile units more secure is to design concrete protection against the security challenges that will be introduced.

Most mobile devices have some type of biometric security measure, either an alphanumeric pin or a touch screen gesture. Biometric methods are used all the time; whether it’s a user entering a pin number on an ATM or a person showing an ID card to prove their identity. It is a way to determine the identity of a person based on some measurable trait or characteristic. Since the inception of mobile handheld devices, many people have been trying to come up with different methods to authenticate users and prevent security violations. Governments and businesses use mobile devices thus the need for extra level of authentication is great.

A simple pin or pattern gesture can be memorized by anyone and hacked, which means that a user would need to be on the alert at all times in order to protect their mobile device entry which is the only line of defense. What is needed is a personalized approach — a mobile handheld authentication biometric system that takes into consideration an individual’s special feature measurements and is able to determine in reasonable real-time if the is person trying to access the device or if the user is authorized. Just as keyboard and mouse biometrics take a person’s virtual keyboard touches, mouse clicks, and drag speed to authenticate a user, a viable alternative could be to develop a gesture plus motion orientation biometric authentication system. This system could be supported by touch, long press, swipe, or drag and various finger pinches.

With this novel biometric design idea in mind, researchers might have found a new way to authenticate the identity of a user; by using gestures, motions, extrapolating patterns and traits from these data, a person may be able to be identified just as they would by using keystroke biometrics.

II. RELATED WORK

The use of touch gestures for mobile authentication is a relatively new research area. The use of touch gestures is becoming an increasingly promising biometric field just like its sister biometric field of keyboard authentication.

In 2012, researchers from the City University of Hong Kong analyzed touch dynamics of 20 Android smartphone users [11]. The touch dynamics analyzed were: Single-Touch consisting of a touch press down followed by a touch press up, Multi-Touch consisting of simultaneous touches from multiple fingers, and Touch-Movement consisting of a touch press down, a
movement (also called drag) and a touch press up with a neural network classifier. An average error rate of 7.8% was achieved which was further improved to 3% by incorporating Particle Swarm Optimization which handled variations in the user’s usage pattern.

At the 2012 ACM SIGCHI Conference on Human Factors in Computing Systems, researchers showed that using five finger touch gestures gave good accuracy results [13]. A 90% accuracy was achieved when using single gestures and a noticeable improvement was observed when multiple gestures were performed in a sequence.

Researchers at the University of Houston investigated touch based continuous mobile authentication by using a unique Graphic Touch Gesture Feature (GTGF) [14]. The six touch gestures used were flick up/down, flick right/left, and zoom in/out. From each gesture, the length of the trace, direction, pressure and the touch duration were gathered. This data was then analyzed by converting the touch traces to images. A remarkably low Equal Error Rate (EER) of 2.62% was achieved.

Another novel authentication technique was explored by researchers from the University of Houston wherein smartphone sensor data was collected while the user performs the mobile device picking up (MDP) motion [15]. The research takes into consideration the trajectory of the phone as it is raised, as well as the angle in which the person holds it as they are looking at the screen or holding it up to their ear. An EER of 6.13% and 7.09% were achieved, although the accuracy of the tests declined during inter-session tests and user movement such as walking.

Our research looks to improve on the existing biometric features by collecting smartphone touch gestures data based on scrolling and device motion/orientation (combined with the sensor-based data) using the Pace University Keystroke Biometric System for user authentication. The present paper focuses on collecting data using the latest Google Nexus 5 phones which have the most complete set of sensors [5].

III. BACKGROUND

In this section, we will cover some background information on three major components of handheld devices that will help in developing the gesture plus device motion/orientation biometric authentication system.

The first component is the touchscreen, which is considered the most commonly adopted interacting panel for handheld devices nowadays [3]. The touchscreen technology was first implemented in 1968, however, this technology was not introduced into the phones market until 1993 when IBM and BellSouth launched "The Simon Personal Communicator" — the first touchscreen phone in the market [2]. With the vast development of smartphones in the current decade, the touchscreen technology started to play a major role in the smartphone market. In 2007, Apple introduced the first multi-touch smartphone for the masses when it presented the first generation of the iPhone. As of 2014, the touchscreen capability of the device is considered a very important factor for the device functionality. Given the current touchscreen technologies as they are implemented in current smartphones, we are able to extract valuable information to aid in the development of our touch based biometric system.

For our system, we decided to implement it on an Android platform due to the reasons explained in detail in the methodology section. The Android touchscreen allows us to extract a set of axis values that include the X and Y coordinates of the touch, which describes the position of the moment. Also we are able to get the pressure and size of the contact area in addition to the action code which specifies the state change that occurred such as going up or down or moving between the two. Certain multi-touch screens (such as the Nexus 5 device which we used to run the experiment) track the movement traces of individual fingers using pointers. Each pointer is assigned a unique number to identify the associated data related to that pointer.

The second component that will help in developing our system is the gesture recognition capability of the device. For Android-based devices, the following are the core gestures supported as illustrated in the Android developers documentation [6].

- **Touch**: it triggers the default functionality of the item. Gesture is performed using: press, lift;
- **Long press**: enters data selection mode, and allows for multiple item selection in a view. Gesture is performed using: press, wait, lift;
- **Swipe or drag**: scrolls overflowing content, or navigate between views. Gesture is performed using: press, move, lift;
- **Long press drag**: reorganizes data within a view, or move data into container. Gesture is performed using: long press, move, lift;
- **Double touch**: scales up a standard amount around the target. Gesture is performed using: two touches in quick succession;
- **Double touch drag**: scales content by pushing away or pulling closer around gesture. Gesture is performed using: Single touch followed in quick succession by a drag;
- **Pinch open**: zooms into content. Gesture is performed using: 2-finger press, move outward, lift;
- **Pinch close**: zooms out of content. Gesture is performed using: 2-finger press, move inwards, lift;

Other mobile operating systems have similar sets of gestures. The third critical component for developing our Android biometric system is the device sensors. The sensors can be broadly grouped into three categories: motion sensors, position sensors, and environmental sensors. Motion sensors are used to measure acceleration and rotational forces along the axes [9]. These sensors are helpful for monitoring any movement in the device such as tilt, shake, rotation, or swing. Position sensors are used for capturing data about the physical position of the device [9]. Environmental sensors are used to measure various environmental parameters [9]. In our system, we will be focusing on the data captured from the motion and position sensors.
sensors since they will provide us with the required biometric data that will help in authenticating users. The most commonly used sensors in Android devices are the following [7][8]:

- **GPS**: global position sensors are used to provide location and time information for the device;
- **Gyroscope**: used to maintain orientation of the device. It measures the rate or rotation around the device’s axes;
- **Accelerometer**: used to measure the acceleration applied to the device, including the gravity force;
- **Linear Accelerometer**: provides a three-dimensional vector representing acceleration along each device axis, excluding gravity;
- **Rotation Vector**: provides the orientation of the device as a combination of an angle and an axis;
- **Orientation**: allows to monitor the position of a device relative to the earth’s frame of reference.

The Android platform provides the ability to capture this data using the sensors, which adds more information that could be used for developing our biometric authentication system.

**IV. METHODOLOGY**

**A. OS Choice**

The first step in implementing a project is selecting the mobile platform which the system is going to be built on. The most popular platforms are currently Android, iOS, BlackBerry, and Windows Phone with the first two accounting for more than 90% of the mobile market [4].

We are looking for the platform which allows us to capture non-text input in the easiest way. In order to do this, desirable characteristics for the platform are as follows:

1) it allows to run applications as services (so that we can collect the data in background);
2) there is access to screen motion data in other applications;
3) devices have multiple sensors, and the OS allows applications to access them.

1) **iOS**: Modern iOS devices have all common sensors found in mobile phones: accelerometer, gyroscope, and compass. However, on this platform, there are certain complications with running applications in background. If an application needs to perform some actions in background, it must fall into one of the predefined categories based on the performed actions, and collecting sensor data or screen touches is not among them [1]. This rules out the possibility of using this platform for background data collection.

2) **Android**: Android devices have different sets of sensors, but most high-end devices have at least the same sensors as iOS devices. Most importantly, the platform allows to run services which allow for the code to actively execute in background. Therefore, Android platform is preferable for building the system.

**B. Raw Data Capture**

In general, all the data that can be captured from the handheld device input system can be divided into three large groups, as shown in the table I.

<table>
<thead>
<tr>
<th>GROUPS OF DATA</th>
<th>Touch Screen Data</th>
<th>Configuration-based Data</th>
<th>Sensor-based Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch Screen Data</td>
<td>Touch action</td>
<td>Screen size</td>
<td>Device acceleration</td>
</tr>
<tr>
<td></td>
<td>Touch coordinates</td>
<td>Screen orientation</td>
<td>Device position</td>
</tr>
<tr>
<td></td>
<td>Finger pressure</td>
<td>Screen density</td>
<td>Device rotation</td>
</tr>
<tr>
<td></td>
<td>Touched area size and shape</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) **Touch Screen Data**: The touch screen is one of the primary sources of data as touch screen gestures are the main method of interaction with modern smartphones, as it was mentioned in the previous sections. Scrolling and the pinch-to-zoom gesture are ubiquitous, and usually have the same meaning for the users regardless of the platform. However, other types of gestures may have different context. For example, in a browser dragging to right or left is much less common than dragging the map around in a map application. The meaning of such gestures as long clicks and double taps is even less ubiquitous. On the latest versions of Android long clicks are used to select items in a list; in earlier versions of Android long clicks were used to bring up a context menu. Double taps are frequently used in browsers to zoom in on webpage text. These differences have to be taken into account when designing a scenario for capturing the data.

2) **Configuration-based Data**: Mobile devices are very different in terms of screen sizes and pixel densities. While this alone does not directly affect the authentication of a user, it must still be taken into account while collecting touchscreen based data. However, the same device can be used in two modes: portrait orientation and landscape orientation, and, consequently, the user holds and uses the device differently. Orientation has a big impact on the applications, many even have different layouts for different screen orientations. Also orientation may help to distinguish between users.

3) **Sensor-based Data**: Many mobile devices are equipped with various sensors. The most widespread sensor is accelerometer, which allow us to track device position relative to the ground and its acceleration. Among those which are useful to the present study there is the gyroscope, which allows to detect rotation along the axes. Another useful sensor is the magnetic field sensor, which allows to track rotations parallel to the ground. In Android some of these sensors are combined via software means into fusion sensors, which are single sensors in the context of applications, but use several hardware sensors at the same time to generate the data.

**C. Features**

Some of the collected raw events can be used in the raw form. This applies to the following types of data:
1) pointer action act, which can be up, down, or movement;
2) pointer id ptr;
3) pressure p;
4) the lengths of the major and minor axes of an ellipse that describes the touched area — tcmaj and tcmin;
5) the length of the major and minor axes of an ellipse that describes the size of the finger — tlmaj and tlmin;
6) gravity force over the axes gs, gz, and gz;
7) angular speed over the axes rx, ry, and rz.

Other data types require some feature extraction to be performed on them. Each pointer event happens at a certain time measured in ms since the time the system was booted. This allows us to extract the time difference Δt between the events.

Touch coordinates require correction as smartphones can have different screen configurations. To do that, we calculate normalized coordinates as

\[ x_{\text{norm}} = \frac{x}{w} \]
\[ y_{\text{norm}} = \frac{y}{h} \]

where w and h are the width and height of the screen in pixels, respectively.

Values received from the accelerometer have to be adjusted so that they include only acceleration and not the force of the gravity. In order to achieve that, we subtract the values reported by the gravity sensor from the accelerometer values and get pure acceleration over axes aclx, acly, and aclz.

The resulting vector of attributes v associated with the event is as follows:

\[ v = (\Delta t, \text{act}, \text{ptr}, x_{\text{norm}}, y_{\text{norm}}, p, t_{\text{cmaj}}, t_{\text{cmaj}}, t_{\text{cmin}}, t_{\text{lmaj}}, t_{\text{lmaj}}, t_{\text{tlmaj}}, t_{\text{tlmin}}, a_{\text{clx}}, a_{\text{cly}}, a_{\text{clz}}, g_{s}, g_{z}, g_{z}, r_{x}, r_{y}, r_{z}) \]

A sample S is a sequence of N events which are associated with the corresponding feature vector.

\[ S = (v_i)_{i \in 1...N} \]

D. System Design

In the first version of the system a decision was made to focus on a specialized application without attempting to track other applications. This means that for the purposes of data collection users will be interacting with the content present in the application itself.

We had three ways to build the application:

1) building a native application;
2) building a native application with an embedded WebView inside it;
3) building a service to listen to a standalone browser (possibly system-provided) through a full-screen overlay.

The last way proved impossible as the events are passed down to only one application at a time. Each application has an associated set of windows, and motion events can only end up in one certain window. Once an application gets a motion event, it can’t be passed to another application, so there is no way to have the same event in two different applications at the same time without patching the system.

The first variant would involve building a custom application. This will allow to test basically any gestures that can be performed on the device, but this will not reflect a typical usage scenario. Therefore, a decision was made to choose the second variant. The Web page represents a natural interaction with the phone. The only downsides of this approach is that some kinds of gestures, such as double taps and flicks to right and left, are not tested with this scenario as they are not present in the Android’s WebView.

Fig.1 demonstrates the basic overview of the system architecture. The main component is the main activity, which is responsible for showing content to the user and capturing the input. This class is responsible for reading all the touch screen and sensor events and feeding them to the buffer.

Besides the events, there is a session data object which contains data specific to the current device configuration. The session object is recreated each time the device orientation changes (this behavior adheres to the standard Android development practice). The buffer’s main task is to accumulate the events up to a certain threshold and then asynchronously feed them to all registered consumers. It is possible for a consumer to be simply a network data transmitter. However, if the data can’t be transmitted from the buffer due to network unavailability, it will be lost, which outlines a need for
persistent storage. The ideal choice for such storage is SQLite databases as the necessary APIs are built into Android. The data can retrieved later: it may be saved into a local file, or transmitted over the network.

V. DATA COLLECTION

As mentioned before, a Web page was deemed appropriate to capture a large amount of finger gestures from the user. The main objective in designing the page is to collect as many types of gestures as possible.

The Web page is composed of three sections to elicit various types of finger movements, as shown on the Fig.2.

1) Image section: causes the user to zoom in and out and drag right and left in order to find hidden objects within a small size image. The image on the left shows the application in progress, displaying a picture that asks the user to find the hidden pretzel. As the person zooms in, it is possible to see better and find the item;  
2) Text section: text in the form of quotes is used for scrolling up and down (shown on the picture in the center);  
3) Question section: shown on the picture on the right, is used for touching/pressing on the screen and also for fast scrolling (flicks) as some of the questions are based on the quotes section.

The application was used to collect samples from 10 participants in two different occasions which rendered about 15,000 gestures per person each time, which is suitable for classification and for determining feature vectors.

VI. RESULTS

Data gathering was conducted in two separate sessions, with the same procedure as described in the data collection section. A dataset was collected from the first data collection session (dataset A), and another dataset was collected in the second data collection session two weeks following the first session (dataset B). In order to classify the data, we used C4.5 algorithm to generate the classification decision tree using a data-mining tool (Weka). To estimate the performance of our decision tree, we used 10-fold cross-validation in addition to splitting the dataset into 66% for training and the remainder for testing. Based on this classification algorithm, we built our training model based on the combined dataset AB and we were able to get 98.4% of the instances correctly classified based on the 10 fold cross-validation. With the percentage split test model, we were able to get 97.7% correctly classified instances rate. When the individual datasets A and B were used as the training model for the algorithm, we noticed a drop in the correctly classified rate by 2% from the rate of the model that was built on the combined AB dataset.

To test how effective this classification algorithm is on a new dataset, we built the training model on one dataset and used the other one for testing. The algorithm showed a huge drop in the correct classification percentage when tested on a new dataset that was not included in the training model. The accuracy percentage for this model was 25%. Therefore, it is clear that the more datasets we use in building our training model, the better accuracy we get when trying to classify a new dataset. However, due to the fact that we were not able to gather more than two datasets so far, we were not able to test this hypothesis.

VII. DISCUSSION

The results for the classification of the whole set were unexpectedly high. However, the results of training on one set and testing on the different set are much lower, which can have several explanations. In order to explain such results, we

Fig. 2. Application screenshots.
created an application to visualize gestures performed on the screen. The application is able to play the gestures over the time, or it can show all the traces at once, optionally with some gesture type filtering.

Fig. 3 shows traces of two users. The first two screenshots on the left depict the traces left during the second data collection, and the two images on the right show the traces of the same users during the first data collection. It can be clearly seen that the pattern remains consistent across the data collection. The first user has two centers where the traces are located, the lines are short and curved, while the second user has straight lines which span a huge part of the screen.

Based on the findings with this application, we can suggest the following possible improvements to the classification algorithm.

- Some of the gestures clearly must be filtered out as some types of gestures are much less suitable for biometric identification. For example, dragging the picture may leave a chaotic trace which can identify a single user in the whole data collection, but it won’t be performed in exactly the same way during the next data collection, which renders it completely useless. An example of such gesture is shown on the Fig. 4 (the green circle in the center).

- Time difference, which is included in the feature vector, could be more hardware dependent. This means that instead of measuring the time between events, it is better to measure the speed of finger movement during the gesture.

- Fig. 3 shows that two users have different speeds of performing the gestures. One of them has dotted lines which means that the finger is moving very fast, and the other one doesn’t have such lines at all.

- Some types of gestures must be treated differently than others. For example, taps usually occur on the control elements, which means that unlike such gestures as scrolling, their position will be useless for the purposes of classification as it is dictated by the interface of the application and not by the user’s biometric parameters. Fig. 4 shows these gestures in the yellow circle on the left.

- Some useful biometric parameters may be specific to certain gestures. For example, pinch-zoom involves placing two fingers on the screen with a certain distance between them. This is a unique feature of this type of gestures which cannot be calculated for other gesture types.

- Users can have different ways of performing the same gesture. This means that one data collection is not enough to identify a single user with a low error rate. The way to mitigate this is to collect the data over several sessions, combine it, and only then the classifier will be able to correctly identify the users.
VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we have described the design of our gestures biometric system and illustrated its implementation. This system is currently capable of gathering data and conducting experiments on any handheld device running on Android operating system. The data gathered using this system were analyzed and the results were extremely promising for additional research opportunities to be conducted in order to improve the accuracy of the system. As described in the discussion section, we have suggested several possibilities for improving the performance of the classification algorithm when applied on a new dataset. These suggestions include filtering out specific gestures from data analysis, implementing additional gesture-specific features into the system, and other suggestions regarding the volume of the data collected and data collection process. In the future, we believe that the research should be focused on enhancing the performance of this system by taking the mentioned improvement possibilities and implementing them into the system.

REFERENCES


