Unobservable Re-authentication for Smartphones

Lingjun Li, Xinxin Zhao, and Guoliang Xue

Arizona State University, Tempe, USA

{*li.lingjun, zhao.xinxin, xue*}*@asu.edu*

Abstract

The widespread usage of smartphones gives rise to new security and privacy concerns. Smartphones are becoming a personal entrance to networks, and may store private information. Due to its small size, a smartphone could be easily taken away and used by an attacker. Using a victim's smartphone, the attacker can launch an impersonation attack, which threatens the security of current networks, especially online social networks. Therefore, it is necessary to design a mechanism for smartphones to re-authenticate the current user's identity and alert the owner when necessary. Such a mechanism can help to inhibit smartphone theft and safeguard the information stored in smartphones. In this paper, we propose a novel biometric-based system to achieve continuous and unobservable re-authentication for smartphones. The system uses a classifier to learn the owner's finger movement patterns and checks the current user's finger movement patterns against the owner's. The system continuously re-authenticates the current user without interrupting user-smartphone interactions. Experiments show that our system is efficient on smartphones and achieves high accuracy.

1 Introduction

The past few years have witnessed an exponential growth of smartphones, in both technology and market shares. According to a research done by canalys [5], smartphones were sold about 73 million more than personal computers (PCs) in 2011. Compared with PCs, smartphones are more privately owned. People may share a desktop, but few are willing to share their smartphones.

At the same time, smartphones are becoming an important personal entrance to various networks, such as the Internet or online social networks. Many apps and websites now allow people to store their accounts, profiles, passwords, etc., in smartphones for automatic re-access. Besides, people also use smartphones to keep contact with friends and families, take pictures of special moments, and arrange schedules. No one would like to disclose such information to an untrusted person. However, due to its small size, a smartphone could be easily taken away by an attacker. The attacker can acquire a good profit from reselling stolen smartphones. It is reported by lookout.com that $2.5 billion worth of devices were lost or stolen in 2011 [19]. Besides, having a victim's private information, an attacker can steal the victim's identity and launch impersonation attacks in networks. Such attacks substantially threaten the security of the networks, especially online social networks. Impersonation attacks also threaten most current trust and reputation systems for networks. Therefore, protecting smartphones against unauthorized usage has significant meaning to safeguarding users' privacy and network security. A smartphone can alert the owner and lock itself when unauthorized usage is detected, which will inhibit most smartphone thefts.

To prevent unauthorized usage of smartphones, a *re-authentication* system is more suitable than an authentication system. An authentication system authenticates a user for one time when he logs in, such as inputting a password to unlock a smartphone. The purpose of a re-authentication system is to continuously authenticate the current user during the whole system execution. In the absence of re-authentication, it is easy for an attacker to access a smartphone if the owner forgets to lock it and leaves it in a public place. Even if the smartphone is locked, an attacker can use operating system (OS) flaws to bypass the lock screen, which is reported to exist in Android [17] and iOS [16] systems. The continuous protection provided by re-authentication is necessary for smartphones.

A straightforward re-authentication approach is to periodically invoke an authentication system, such as asking the user to enter a password. This approach interrupts user-smartphone interactions and leads to bad user experiences. For smartphones, it is preferable that the re-authentication takes place in a way that users do not "observe" its existence.

Current short unlock passwords, such as 6-digit num-

bers, cannot protect smartphones against a powerful attacker. However, long and complicated passwords are difficult to memorize. Hence, a re-authentication system should rely on certain "password" that is easy to memorize but difficult to forge. A good candidate for such passwords is the owner's biological data. Many works have studied biometric-based authentication, such as fingerprint recognition [8], face recognition [12], and iris recognition [27]. However, these methods are not suitable for smartphone re-authentication because they either rely on special equipments, which are not available for smartphones, or need the users to stop interactions to assist the re-authentication. In addition, continuous face recognition requires keeping the camera on all the time, which dramatically reduces a smartphone's battery life.

In this paper, we propose a re-authentication system for smartphones using users' finger movements. The system first learns the owner's finger movement patterns, keeps running in the background, continuously monitors the current user's finger movement, and compares the current user's movement patterns against the owner's patterns. Our system does not need user assistance in re-authentication and users are not aware of its execution.

The main contributions of our work are as follows:

- We propose and study the unobservable smartphone re-authentication problem. We design a novel system architecture especially for smartphones to reduce the computational overhead on smartphones.
- We propose to use the finger movement as a biometric characteristic to authenticate a user. When users use smartphones, the smartphones sense the users' finger movements and interpret the sensed data as different *gestures*. Since users have to use gestures to interact with smartphones in most cases, our proposed approach can enforce re-authentication to every user. In addition, our approach can continuously re-authenticate the current user without being noticed by the user.
- We propose an efficient biometric-based re-authentication system for smartphones using the classification method. We design the biometric features for the classification algorithm and discuss their performance. We implemented our system on a Motorola Droid smartphone to demonstrate its efficiency. Extensive experiments were performed to evaluate the effectiveness of the features and the performance of our system.

The rest of this paper is organized as follows. We introduce the background and related work in Section 2. The attack model is introduced in Section 3. We discuss the design goals for a smartphone re-authentication system in Section 4. We present our re-authentication system in Section 5. We discuss the feature design and selection in Section 6. We evaluate our re-authentication system in Section 7, and conclude our work in Section 8.

2 Background and Related Work

Compared with traditional authentications, biometric-based authentications are easy to carry out, natural to use, and invulnerable to forgery. Traditional approaches are based on possessions of secret information, such as passwords. Biometric based approaches make use of distinct personal features, such as fingerprint or iris.

A biometric-based re-authentication system involves an enrollment phase and a re-authentication phase. A user is enrolled by providing his biological data. The system learns patterns from the provided data and stores the learned patterns for future reference. During the re-authentication phase, the system compares the observed biological data against the stored data to re-authenticate a user.

Previous studies on biometric-based re-authentication concentrated on either physiological or behavioral features [29]. Physiological biometrics study static physical features of humans. Currently, there are many different physiological biometrics for re-authentication, such as fingerprint [8], face patterns [12], and iris [27]. However, physiological biometric-based re-authentication approaches usually rely on specific devices, which are unavailable on most smartphones. In addition, most approaches need human assistance in the re-authentication. For example, most face recognition systems need the users to stay still at a specific angle to the camera during re-authentication. Hence, these physiological biometric-based approaches cannot achieve continuous unobservable re-authentication.

Behavioral biometrics assume that people have distinct stable patterns on a certain behavior, such as keystrokes on a keyboard. Behavioral biometric-based re-authentication uses the behavior patterns to authenticate a user's identity. For personal computers, most previous studies concentrated on two operations: keystrokes [3, 22, 23] and mouse movements[18, 31]. Typing actions happen much less frequently on smartphones than on personal computers, because people hardly use smartphones to do heavy text input. Therefore, it is difficult to collect a sufficient number of keystrokes on smartphones for re-authentication.

Although smartphones do not have mouse input devices, previous studies [2, 24] on mouse movements help us to understand finger movements on smartphones. Hence, we give more detailed review of prior works on mouse movements.

2.1 Mouse Movements

When a mouse moves, the hardware captures the movement and sends the mouse events to the OS, including raw movement coordinates, button up, and button down events. The OS interprets these mouse events to a series of point data, which form a mouse movement. In the approach proposed by Ahmed and Traore, point data are aggregated as *point-and-click* or *drag-and-drop* actions [1, 2]. A point-and-click action contains a click event and a mouse movement following the click. A drag-and-drop action is a mouse movement with one button pressed down. The reason to study the two actions is that they are both performed intentionally by users. Ahmed and Traore characterized each action using action type, moving distance, duration, and direction [2]. They computed 39 dynamics related features and used a neural network to classify new observed actions.

Recently, Zheng *et al.* [31] proposed to use only the point-and-click action and three features: *direction, angle of curvature*, and *curvature distance*, to authenticate a user. The classifier they used is SVM. They aggregated the features of 20 point-and-click actions as a feature vector. Their work required 20 mouse movements, compared with 2000 mouse movements required in Ahmed and Traore's work [2]. This reduction decreases the data collection time and hence increases the re-authentication frequency. In their work, they collected 81218 point-and-click actions from 30 users in a controllable environment and one hour raw mouse events from 1074 anonymous users from an online forum. The average false rejection rate and the average false acceptance rate were both 1.3% in their tests.

In another approach, Pusara and Brodley utilized the connections between each pair of points within a window of a configurable size [26]. The features, such as angle, distance, and speed, were extracted from the points rather than the actions. They used C5.0 decision tree as the classifier in their system, which achieved an average false acceptance rate of 0.43% and an average false rejection rate of 1.75% in the experiments on an eleven-user data set. Gamboa and Fred [13] aggregated the points between two clicks. Each click is represented by 63 features. For each user, they proposed a greedy approach to reduce the feature set to a best fit subset.

2.2 Smartphone Features

One of the biggest differences between personal computers and smartphones is that smartphones are equipped with many sensors, such as multi-touch screen, accelerometer, and gyroscope. Although different smartphones may have different sensors, multi-touch screen, accelerometer, and compass are provided by most smartphones.

Multi-touch screen is a basic equipment on a smartphone. A multi-touch screen is able to respond to more than one finger touch. The number of supported touch points varies from device to device. Some basic screens can only support two touch points while some advanced ones are able to support up to ten touch points. The multi-touch screen records the touch position, touch area, and touch pressure, packs them as a single touch event, and sends it to the OS. A series of touch events are connected together and recognized as different gestures, such as sliding, tap, double tap, or spread.

Accelerometer measures the phone's acceleration on three axis, x, y, and z [15]. This captures a smartphone position in a three-dimensional space.

Orientation indicates whether a smartphone is held in portrait mode or landscape mode.

Compass measures the position of magnetic north in relation to the X, Y, and Z axies of the phone.

Various sensors in smartphones provide a lot of biological data of a user, which can be used in biometric-based authentication. Some previous works have studied using smartphone sensors for security purpose. Some works used accelerometer to sense a person's shake motion data to securely pair two devices [6, 21]. Mäntyjärvi *et al.* first considered using sensors to record users' behavioral patterns and to continuously re-authenticate a user [20]. They suggested to use accelerometer and orientation sensors to monitor a user's walking patterns. Their approach can successfully recognize a user at rates between 60% and 85%. Okumura *et al.* proposed to authenticate a user using the accelerometer data sensed when the user is swinging his arm [25]. Instead of using the false acceptance rate or the false rejection rate, they claimed their system's equal error rate – the error rate when the false acceptance rate is equal to the false rejection rate – was able to achieve as low as 5%. Recently, Conti *et al.* proposed to re-authenticate a user using the arm movement patterns, sensed by the accelerometer and orientation sensors, while the user is making a phone call [9]. They achieved a false acceptance rate of 4.44% and a false rejection rate of 9.33% in their tests.

Recently, Biometric Signature ID company has proposed to use gesture based signature to re-authenticate a user in the log-in phase [4]. This approach records a user's signature during the enrollment phase and compares an input signature against the recorded one during re-authentication. Luca *et al.* [11] proposed to use gesture information, such as touching area or duration, as an additional re-authentication approach on top of the current password pattern approach. The two methods are both one time re-authentication and will interrupt user-smartphone interactions if they want to achieve continuous re-authentication. Different from these

works, our system aims to provide a continuous unobservable re-authentication.

Existing continuous re-authentication approaches have paid extensive attention to the accelerometer and orientation sensors and used behaviors that may not happen during an attack in our scenario. For example, an impostor may not swing arms when he uses a victim's smartphone. Therefore, we need an approach that can continuously re-authenticate a user as long as he is using the smartphone. We propose to use and monitor the gesture on smartphone touch screen, which is the most important and necessary interface between users and the smartphone OS.

2.3 Smartphone Gesture Patterns

Here we first give several observations on smartphone gestures, which differentiate the finger movement on smartphones from the mouse movement on computers.

Usage Intermittence: people may not always use smartphones for a long time. Typical usages are to wake up a smartphone, click an email app, check if there is any new email, and then turn off the screen. The collected gestures are thus not temporarily continuous.

Spacial Disconnection: In the study on mouse movement patterns, each movement can be captured by hardwares and used to formulate patterns, such as the point-and-click patterns. On smartphones, not every finger movement can be captured by a touch screen. For example, a user lifts up his finger, moves in the air, and clicks a link on a webpage. In these cases, the screen cannot capture the finger movement in the air, which corresponds to the point action in mouse movements.

Orientation Dependent: Users may use smartphones in either portrait or landscape orientations. Users' gestures have different patterns in different orientations. For example, a sliding up distance becomes shorter in the landscape mode.

3 Attack Model

We consider an attacker who has physical access to the smartphone and wants to use the resources in it, such as applications or music. For example, an attacker may steal a victim's smartphone and enjoy the music in it without paying any money. The attacker may also steal the network account information and the personal information stored in the smartphone. For example, the attacker can post a fake message in a social network using the victim's account. The purpose of our work is to design a continuous re-authentication system running in the background. The system keeps authenticating the current user in an unobservable way, i.e., it does not interrupt the user's interactions with the smartphone. In this paper, we only consider the authentication of a user against the smartphone owner, because a smartphone is usually privately owned and not shared by multiple users. If the user is found to be a stranger, the re-authentication system alerts the OS, which performs corresponding actions.

4 Design Goals

We summarize the goals that a smartphone re-authentication system should achieve in the following.

- **Continuity**: A smartphone re-authentication system should keep authenticating the current user as long as the smartphone is being used.
- **Unobservability**: A smartphone re-authentication system should neither interrupt user-smartphone interactions nor need human assistance during re-authentication.
- **Light-weight**: A smartphone re-authentication system should not need intensive computations on smartphones.

5 Approach

We are ready to present our smartphone re-authentication system, which achieves the design goals discussed in the above section.

Our idea stems from the observation that users' finger movements on smartphone screens are different from person to person when they use smartphones. For example, some people like to use the index finger to slide up the content displayed on the screen while some people prefer to use the thumb. Following customs in smartphone development, we call a continuous finger movement on the screen a *gesture*. We assume that a user's gestures contain his distinct behavioral characteristics.

Our work uses such characteristics to re-authenticate users. We illustrate our system architecture in Figure 1. Considering the limited computational and storage resources in a smartphone, our system is divided into two modules, the re-authentication module and the training module. The re-authentication module is deployed in a smartphone and the training module is executed on a PC. To provide a better security and performance guarantee, we suggest to implement the re-authentication module as part of the smartphone OS services in practice.

The re-authentication module keeps running in the background of smartphones. It monitors a user's raw touch event data and sends it to the preprocessing component, which assembles every single raw data into different gestures and then sends them to the feature extraction component. The

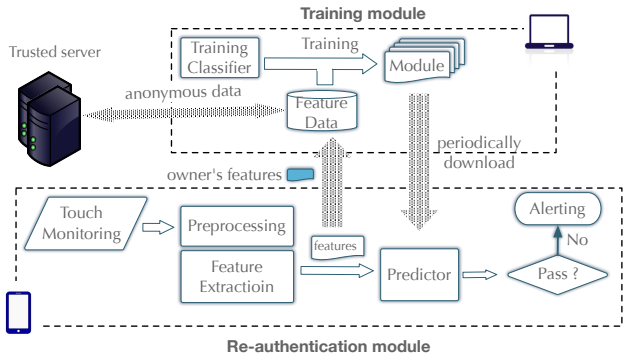


Figure 1. System architecture

latter component extracts features from the gesture data, forms a feature vector, and feeds it into the predictor component. Finally, the predictor component makes a prediction. If the feature vector is predicted to be from the smartphone owner, this re-authentication is passed. Otherwise, an alert message will be sent to the OS. Different OSs may take different actions in response to the alert. One possible action is to lock the system and ask the user to input an administrator password. Another possible action is to send a message, with the current GPS information in it, to the owner's e-mail box.

The predictor component consists of a classifier and multiple classification modules, as shown in Figure2. A classifier is a classification algorithm that uses an object's feature vector to identify which class the object belongs to. The classification algorithm used in our work is the support vector machine (SVM) algorithm. Each classification module is in charge of a main gesture type or a combination of a main gesture type and an auxiliary type. A classification module is a file containing parameters for the classification algorithm and determines the classifier's functionality. The basic classification algorithm is embedded in the classifier. Using different classification modules, the classifier can make predictions on feature vectors of different gesture types. When a feature vector is fed in, the classifier chooses a corresponding classification module and makes a prediction.

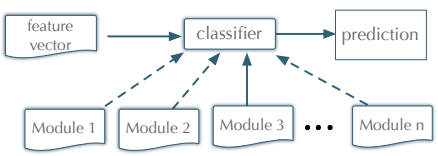


Figure 2. Predictor component

The training module is executed on the owner's PC, because it requires significant computations. When a smartphone owner first enrolls in the system, the system collects the owner's gesture features by using the touch monitoring, preprocessing, and feature extraction components of the re-authentication module. Our system deploys a trusted data server to collect feature data from smartphone owners and downloads them to the training modules when necessary. To protect an owner's privacy, the data collection is done anonymously. This can be achieved by using anonymous group messaging [10, 30]. A fixed number of a user's feature data and a time stamp form a ready-to-submit feature message. Every user in our system is a group member and the server is the data collector in the anonymous group messaging system. The users collaboratively shuffle their messages before the messages are handed in to the server. Eventually, the server does not know the connection between a message and its owner. In this way, a user's training module can use other users' feature data but has no way to know the user identities. We note that a user only wants to download other users' feature data. Therefore, a user compares every downloaded feature message against his own submissions and drops the one that is the same with one of his submissions. The comparison can be based on the hash value of the messages to reduce the time and storage overhead.

The training module uses the owner's features and other people's features in the training algorithm to obtain classification modules. After training, the classification modules are downloaded onto the smartphone. The training module anonymously uploads the owner's data to the trusted server and obtains anonymous features from it. We note that this trusted data server does not participate in the re-authentication and is only needed when an owner wants to re-train his classification modules, which is done offline and on-demand. Therefore, our system does not pose a high requirement on the communication delay between smartphones and the server.

An owner's usage pattern usually stays stable. But sometimes, the owner may change his usage pattern over weeks or months, which may cause more false alarms. When this happens, the classification modules need to be re-trained. To keep the modules up to date, our system also allows an on-demand re-training. When the owner requests a re-training, the re-authentication module captures the owner's gestures, calculates and uploads the owner's feature vectors to the training module. The training module then downloads anonymous feature messages from the server, filters out his own submissions, and runs the classifier training algorithm again to obtain new classification modules.

We note that the access to the system enrollment and re-training process should be restricted to the smartphone owner only. This can be achieved, for example, by using traditional password based protection.

6 Characterizing Gestures

Our system monitors five types of gestures: sliding up, sliding down, sliding left, sliding right, and tap, as shown in Figure 3. Usually, slidings are used to move contents

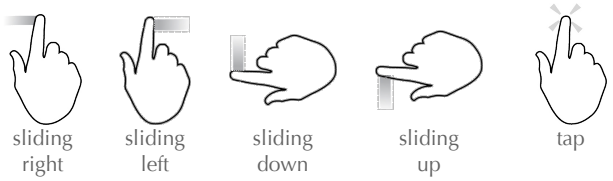


Figure 3. Five essential gestures

displayed on the screen and tap is used to click a link or a button within the contents. Although there are some other gestures, such as double tap, spread, and pinch, the five gesture types are the most often used types when users interact with smartphones. We collected 75 users' gestures when they used smartphones. We show the proportions of different gesture types in a pie chart in Figure 4. It shows that the above five types of gestures take a dominant part of all the gestures. In other words, most users inevitably used at least one of the above gesture types when they used smartphones. As shown in Figure 4, slidings and taps occupy 88.8% of all

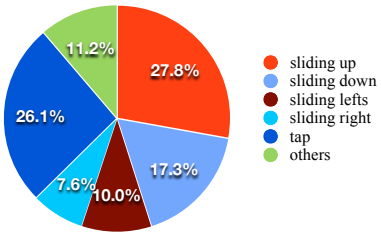


Figure 4. Pie chart for collected gestures

the gestures. We remark that we do not consider virtual keyboard strokes here, because they are not suitable for smartphone re-authentications. Keystroke based authentications usually need a number of continuous keystrokes, but most users do not continuously input many texts on smartphones. In addition, an attacker can use a victim's smartphone without many continuous keystrokes.

Users can hold smartphones in either portrait mode or landscape mode. As pointed out in Section 2.3, the orientation mode affects a user's gesture patterns. Hence, our system has two classification modules for every gesture to deal with each orientation mode.

6.1 Data Collection

The open-source Android system is selected as our implementation platform. Specifically, all of our experiments and data collections were carried out on Android 2.2.2. The first thing we need is a program that can monitor a user's finger movements in the background. However, for security reasons, Android requires that only the topmost apps can obtain touch events, dispatched from the Android system management service. In other words, we cannot enjoy the convenience that Android API provides to developers and have to work around this problem. We found that Android relies on Linux kernel for core system services, including the maintenance of hardware drivers [14]. When some touch event happens, a screen reads in raw data and sends it to the Linux kernel. The kernel then packages the raw data and sends it to the upper layer Android library. Since we cannot get input data from Android API, our idea is to read input data directly from lower layer Linux kernel.

Linux kernel uses device files to manage devices, located under the directory `/dev/`. Same as other devices, a multi-touch screen also has a corresponding device file, say `/dev/event3`. When the multi-touch screen reads inputs, the data are put in the device file by the kernel. The data orgnization follows the Linux multi-touch event protocol [28]. In the protocol, touch details, such as position, touch area, pressure, etc., are sent sequentially as Linux ABS event packets [28]. Each packet contains an ABS event indicating a specific touch data. Packets are separated by a `SYN_MT_REPORT` event (type `0002`). When all touch packets in a multi-touch action arrive, a `SYN_REPORT` event (type `0000`) is generated. A typical multi-touch ABS packet is as follows:

```
0003   0030   00000005
0003   0032   00000018
0003   0035   000002b6
0003   0036   00000296
0000   0002   00000000
0003   0030   00000003
0003   0032   00000012
0003   0035   0000024d
0003   0036   000001e4
0000   0002   00000000
0000   0000   00000000
```

The first byte indicates the event type: `0003` is an ABS event and `0000` is an `SYN` event. The second byte indicates the data type: `0030` is `ABS_MT_TOUCH_MAJOR` major axis of touch ellipse; `0032` is `ABS_MT_WIDTH_MAJOR` major axis of approaching ellipse; `0035` and `0036` are `ABS_MT_POSITION_X` and `ABS_MT_POSITION_Y`, respectively, giving the central position of an ellipse. These four basic data types are supported by all Android smartphones. Other data types include `ABS_MT_TOUCH_MINOR`, `ABS_MT_WIDTH_MINOR`, `ABS_MT_ORIENTATION`, `ABS_MT_TOOL_TYPE`, `ABS_MT_BLOB_ID`, and `ABS_MT_TRACKING_ID`.

The multi-touch protocol recognizes a finger touch area as an ellipse and describes it using its major and minor axises. Some low-end devices, such as the Motorola Droid smartphone we used, recognize a touch area as a circle and omit the minor axis value. `TOUCH` type data describes the area that a finger directly contacts the screen and `WIDTH` type data describes the shape of a finger itself. The ration of `ABS_MT_TOUCH_MAJOR`/`ABS_MT_WIDTH_MAJOR` gives the touch pressure. The last two bytes in each line represent the data value. The above packet contains two finger data details, separated by `0000 0002 00000000`.

Our monitoring program needs the root privilege to hack into the lower layer of an Android system. Such a re-authentication system is usually integrated into the OS and can be granted the root privilege by the OS.

We carried out our data collection and all the experiments on two Motorola Droid phones, with 550MHz A8 processor, 256MB memory, 16GB sdcard, and Android 2.2.2 OS. In order to collect gesture data, 75 users were invited to take away our smartphones for days and use them freely. We did not specify any requirement on the usage and let the users use the smartphone in any way they feel comfortable. The users can browse web pages, including news, online forums, social network websites, etc., or use the installed apps, such as twitter, facebook, google reader, etc.. Users were not required to continuously use the smartphone. They could lock the smartphone and resume using it later. In summary, we want the participants to use smartphones in the same way that they use their personal smartphones in their daily life.

6.2 Metric Design

Good features are critical to a supervised machine learning approach, which is used in our system. In this section, we design the metrics to characterize the five types of gestures. In Section 6.3, we test whether a metric is good and drop the bad ones. A feature is the average metric value over a *block* of gesture data.

6.2.1 Metrics of Sliding

First, we inspect what happens during a sliding gesture. Figure 5 shows the sensed data of a sliding gesture, a sliding up, recorded by our touch monitoring component. We note that the coordinate on the smartphone platform puts the origin at the top left corner of a screen. Each circle represents a finger touch, because Motorola Droid phone views a finger touch as a circle. The size of a circle shows the size of the touch area and the brightness of a circle shows the strength of the touch pressure. The movement starts at point F and ends at point C. The time between every pair of circles is the same. Apparently, the finger moves slowly at first, then faster, because the circles become sparser as the finger moves.

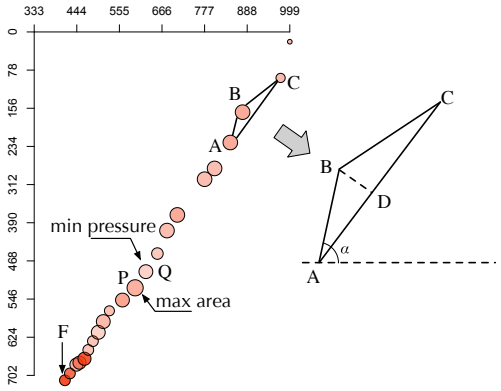


Figure 5. A sliding up gesture

Our first interesting observation – which is different from our intuition – is that the maximum touch area may not happen at the first touch and the minimum pressure may not happen at the last touch. As can be seen from the figure, the touch point P has the largest touch area and point Q has the smallest touch pressure, both of which are neither the first touch nor the last touch. Another observation is that strong pressures happen at the beginning of a sliding. Despite the first 5 points, the variations of touch pressures are not as big as that of touch areas.

We propose the following metrics for a sliding gesture:

- **First touch position:** the coordinates, x and y, of the starting point in a sliding.
- **First touch pressure:** the pressure of the first touch.
- **First touch area:** the touch area of the first touch.
- **First moving direction:** the moving direction of a touch point is the angle between the horizontal line and the line crossing the point and its succeeding point. The angle α in Figure 5 is the moving direction of point A. First moving direction is the moving direction of the starting point.
- **Moving distance:** the total distance of the sliding gesture. Particularly, it is the summation of the distances between every two continuous touch points.
- **Duration:** the time duration of the whole sliding gesture.
- **Average moving direction:** the average value of all the point's moving directions. We note that the last point is not counted, because it does not have a moving direction.
- **Average moving curvature:** given any three temporally continuous touch points, such as A, B, and C in Figure 5, the corresponding moving curvature is angle $\angle ABC$. The average value of the moving curvatures in the sliding gesture is selected as a metric.

- **Average curvature distance:** given any three consecutive points, such as A, B, and C in Figure 5, the corresponding curvature distance is the distance from point B to line AC. We take the average of all the curvature distances as a metric.
- **Average Pressure:** the average of all the touch pressures in the sliding gesture.
- **Average touch area:** average of all the touch areas.
- **Max-area portion:** we index all the points according to the time order, starting from 1. The max-area proportion of the sliding gesture is the index of the max area touch point divided by the total number of the points in the sliding. This metric reflects which portion of the sliding contains the maximum touch point.
- **Min-pressure portion:** Similar to max-area portion, the min-pressure portion is the index of the minimum pressure touch point divided by the total number of the points.

The final feature vector is calculated over a *block* of sliding gestures. The block size is denoted by n_s. Each feature value in the feature vector corresponds to an average metric value over the block of sliding gestures.

6.2.2 Metrics of Tap

Tap is a simple gesture and does not provide much information about a user's finger movement patterns. In contrast to our intuition, many tap gestures contain more than one touch points. It is due to the screen's high sample frequency and the slight tremble of a user's fingertip when he is touching above the screen. The metrics for a given tap gesture are as follows:

- **Average touch area:** the average of all the touch areas.
- **Duration:** time duration of the tap gesture.
- **Average pressure:** the average of all the touch pressures.

Similar to the calculation of a sliding feature vector, a tap feature vector is also the average metric values over a block of tap gestures. The block size is denoted by n_t.

6.3 Metric Selection

According to our observations about users' behaviors of using smartphones, we proposed different metrics in Section 6.2, trying to characterize a user's gestures. Selecting good metrics is essential for a supervised machine learning method, such as SVM used in this work. In this section, we test the performance of each metric and drop the bad metrics. If a metric can be used to easily distinguish two users, we say the metric is a good metric. We view a metric value calculated from a person's gesture as a data sampled from an underlying distribution of the metric. For a metric to distinguish two different persons, it is necessary to require the two underlying distributions to be different. Therefore, for a metric, we construct a metric data set for each invited user in the data collection by calculating the metric value from each of his sliding gestures. Then, we tested whether two metric data sets are from the same distribution. If most pairs of the data sets are from the same distribution, the metric is bad in distinguishing two persons and we need to drop it.

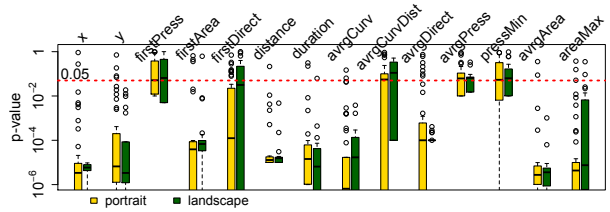
We use two-sample Kolmogorov-Smirnov test (K-S test) to test if two metric data sets are significantly different. Two-sample K-S test is a nonparametric statistical hypothesis testing based on maximum distance between the *empirical cumulative distribution functions* of the two data sets. The two hypotheses of K-S test are:

H_0 : *the two data sets are from the same distribution;*
H_1 : *the two data sets are from different distributions.*

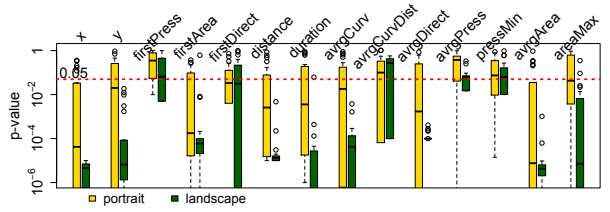
A K-S test reports a p-value, i.e. the probability that obtaining the maximum distance is at least as large as the observed one when H_0 is assumed to be true. If this p-value is smaller than a significant level α, usually set to 0.05, we will reject H_0 hypothesis because events with small probabilities happen rarely. For each metric, we calculated the p-value for each pair of the metric data sets and drop the metric if most of its p-values are smaller than α.

6.3.1 Sliding Gesture

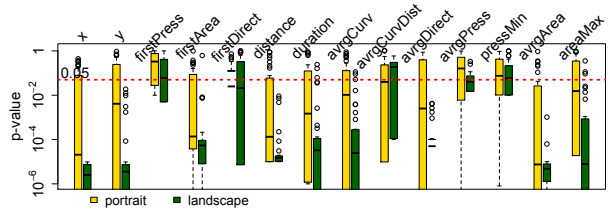
Figure 6 shows the testing results for the metrics of the four sliding gestures in both portrait and landscape modes. Due to space limitation, we abbreviate some metric names in the figures. firstPress is "first touch pressure", firstArea "first touch area", firstDirect "first moving direction", distance "moving distance", avgCurv "average moving curvature", avrgCurvDist "average curvature distance", avrgDirect "average moving direction", avrgPress "average pressure", pressMin "min-pressure portion", avrgArea "average touch area", and areaMax "max-area portion". For each metric, the resulting p-values are drawn in a box plot. The bottom and the top of the box denote the lower quartile $Q1$ and the upper quartile $Q2$, defined as the 25th and the 75th percentiles of the p-values. The middle bar denotes the median of the p-values. The lowest and the highest bars outside the box denote the lower and the upper outer fences, defined as $4Q1 - 3Q2$ and $4Q2 - 3Q1$, respectively. The results from portrait orientation are represented by yellow boxes and those from landscape orientation are represented by green boxes. The y-axes in Figure 6 are drawn in logarithmic scale. The red dashed line in each subfigure represents the significance level α. Hence, the better a metric is,



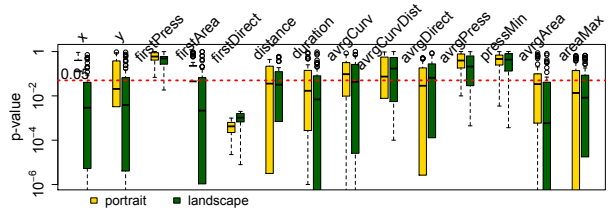
(a) Sliding up gestures



(b) Sliding down gestures



(c) Sliding left gestures

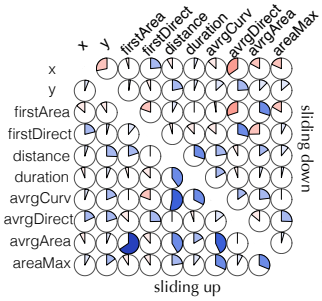


(d) Sliding right gestures

Figure 6. K-S test on sliding metrics

the more portion of its box plot is below the red line. It denotes that more pairs are significantly different. We initially thought touch pressures should be a good metric to distinguish different people. However, from Figure 6, we can see that none of the three pressure related metrics, first touch pressure, average pressure, and min pressure portion, is a good metric because at least half of their p-values are above the red line in all of the four subfigures. This means that the pressure data is bad in distinguishing two different persons. Besides, Figure 6 also shows that average curvature distance is a bad metric. The remaining metrics have most of their p-values below the red line, indicating that most data sets are significantly different to one another in the statistical sense. Therefore, we finally select first touch position, first touch area, first moving direction, moving distance, duration, average moving direction, average moving curvature, average touch area, and max-area portion as the metrics for sliding features.

Next, we tested the correlation between each pair of metrics. A strong correlation between a pair of metrics indicates that they are similar in describing a person's gesture pattern. In other words, a weak correlation implies that the selected metrics reflect the different characters of the desired gestures. For each user's gesture data set in one orientation, we calculated Pearson's correlation coefficient between each two metrics. Then, for each two metrics, we took the average of all resulting correlation coefficients between the two metrics. The average is taken over different users and different orientations. Figure 7 shows the resulting average coefficients. Each subfigure can be viewed as a



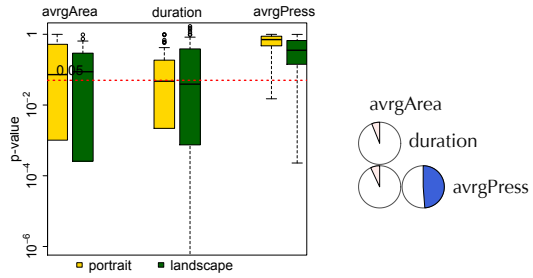
(a) Sliding up and sliding down gestures (b) Sliding left and sliding right gestures

Figure 7. Correlations between each pair of metrics of sliding gestures

10 by 10 matrix and shows two sliding types using an upper triangle and a lower triangle, respectively. A pie chart in a triangle denotes the average correlation coefficient between the two metrics. The names of the metrics are listed on the top and the left sides. For a pie chart, blue represents a positive correlation and red represents a negative correlation. A larger shaded area in a pie chart indicates a stronger correlation. From the figure, we can see that most correlations are weak correlations and there are more positive correlations than negative correlations. We note that the correlation between the average touch area and the first touch area is remarkably positive in sliding up and sliding right. This is because people's first touch usually affects the remaining touches in a sliding gesture. If a person touches hard at first, it is quite possible that he will continue touching hard the rest of the sliding. However, we are not going to delete any of the two metrics because the correlation is not strong enough in sliding down and sliding left.

6.3.2 Tap Gesture

Tap gesture is a simple gesture. Hence, we do not design many metrics for it. Figure 8(a) shows the K-S test results on each tap metric. It is obvious that the average touch area



(a) K-S test results (b) Correlations

Figure 8. K-S test on tap metrics and the correlations between the metrics

Table 1. Classification modules and the corresponding gesture types

PORTRAIT		LANDSCAPE	
No.	**gestures**	**No.**	**gestures**
1.	sliding up	2.	sliding up
3.	sliding down	4.	sliding down
5.	sliding left	6.	sliding left
7.	sliding right	8.	sliding right
9.	sliding up + tap	10.	sliding up + tap
11.	sliding down + tap	12.	sliding down + tap
13.	sliding left + tap	14.	sliding left + tap
15.	sliding right + tap	16.	sliding right + tap

metric and the average touch pressure metric are not good in distinguishing users, because their medians are above the significance level. The median of p-values of the duration metric is just a little below the significance level. In summary, the tap metrics are not as good as the sliding metrics. The reason is that a tap gesture is usually so quick and simple that it provides few distinct features. The average correlation coefficients between every two metrics are shown in Figure 8(b). We can see that the correlations between each pair of metrics are not strong, i.e. every coefficient is smaller than 0.5. Therefore, using tap gesture as a single gesture to re-authenticate a user is not reliable and may cause high error rates. Therefore, we propose to use tap as an auxiliary gesture. If the desired number of taps are captured, our system combines the tap feature vector together with a sliding feature vector to enhance the authentication accuracy.

6.4 Designing Modules

As illustrated in Figure 2, the predictor component contains several classification modules, each of which can independently re-authenticate a user. In Table 1, we list the gesture or gestures used in each module. In total, we have 16 classification modules in the predictor component – 8 modules for portrait mode and 8 modules for landscape mode. In each orientation mode, we use 4 modules to classify 4 sliding types. Another 4 modules are used to classify the combination of a sliding gesture and a tap gesture. Each module sends an alert to the OS if it finds the feature vector to be abnormal.

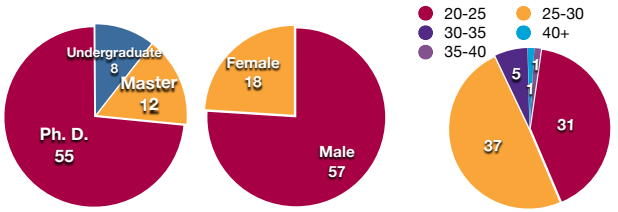
As pointed out in Section 6.2, a feature vector consists of the average metric values taken over a block of gestures. The block sizes are different for slidings and taps, denoted by n_s and n_t, respectively. For example, when a sliding up gesture is captured in portrait mode by the gesture monitor component, 10 metric values will be extracted and stored as a group. If there are already n_s such groups, average values are calculated by metric and fed into the classifier as a feature vector. The classifier uses module 1 to classify the feature vector. If it is an abnormal vector, an alert message will be sent out to the OS. If there is a new portrait tap feature vector ready as well, it will be combined with the sliding feature vector and the classifier will use module 9 instead. We emphasize that our system does not require the block of gestures to be temporally close to each other. In other words, any n_s sliding up gestures can be used to calculate a feature vector. This property of our system is important to smartphone usage because most people use smartphones intermittently. For example, a user may turn on his smartphone, check emails, and turn it off. There may be only a few sliding gestures in this operation cycle. Therefore, gestures are usually collected group by group and there may be a long time interval between two groups.

7 Evaluation

7.1 Setup

We used the SVM algorithm as the classification algorithm in the system and selected LIBSVM [7] as our implementation. For two-class classification tasks, the SVM finds a hyperplane in training inputs to separate two different data point sets such that the margins are maximized. A margin is the distance from the hyperplane to a boundary data point. The boundary point is called a support vector and there may be many support vectors. Sometimes, we need to map the original data points to a higher dimensional space by using a kernel function so as to make training inputs easier to separate. The kernel function used in our SVM is the Gaussian radial basis function $K(\mathbf{x}_a,\ \mathbf{x}_b) = e^{-\gamma||\mathbf{x}_a - \mathbf{x}_b||^2}$, where γ is equal to the reciprocal of the feature number. In our classification modules, we label the owner's data as a positive class and all other users' data as a negative class.

As described in Section 6.1, 75 people participated in our data collection. The participants are either students in or visitors to our lab building. We recorded the demorgraphics — education, gender, and age range — of the participants and show them in Figure 9. All our participants are older than 20 years old. Here, education is the highest degree that a participant has or is pursuing. The numbers in the pie chart are the numbers of the participants in the corresponding category.



(a) Education (b) Gender (c) Age

Figure 9. Demographics of the 75 participants

Among them, 28 people are *target users* who were asked to use the smartphones till at least 150 sliding up gestures, 150 sliding down gestures, 150 sliding right gestures, 150 sliding left gestures, and 300 tap gestures were collected. Other people, called *non-target users*, were asked to use the phone till the total using time hit 15 minutes. The target users are CSE graduate students at Arizona State University. The demographics of the target users are listed in Table. 2.

Table 2. Demorgraphics of the target users

	Category	# of users
Education	Master students	2
	Ph. D. students	26
Gender	Female	9
	Male	19
Age	20-25	10
	25-30	15
	30-35	3

In our experiments, we generated training and testing data sets for each target user. For a specific gesture type and a target user, the user's corresponding gesture data set was divided into two halves. In each half, we randomly selected a block of gesture data of necessary size, such as n_s for sliding up gestures, and calculated the feature vector. The vector was labeled as a positive feature vector. We generated training positive feature vectors using the first half gesture data set and testing positive feature vectors using the other half gesture data set. In order to generate negative feature vectors, we divided the remaining target users and the non-target users into two halves, respectively. The first half of the target users and the first half of the non-target users consisted of the training user pool. The remaining users consisted of the testing user pool. To generate a training (testing) negative class feature vector, we first randomly chose a user from the training (testing) user pool, then randomly selected a block of gestures from the user's gesture set, and finally calculated a feature vector, labeled as negative. We dropped a feature vector if the selected gesture block was previously used. Training feature vectors, including positive and negative ones, were put together in a training data set and testing feature vectors were in a testing data set. We remark that positive training and testing feature vectors were generated from two disjoint sets of gestures. For negative feature vectors, the users used to generate testing vectors are totally different from those used to generate training vectors. Hence, in our experiments, the feature vectors used to test a classification module are never met by the module in its training phase.

7.2 Experiment Results

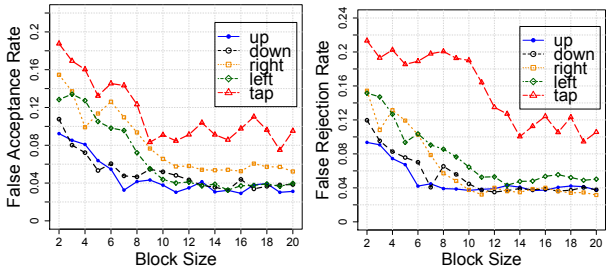
In this section, we test the classification performance of our system under different system parameters. We also implemented the re-authentication module on a Motorola Droid smartphone to test its computation overhead.

During data collection, we did not put any usage restrictions on the participants. Users were free to walk, sit, travel by vehicle, or perform other common activities, while they were using our smartphones.

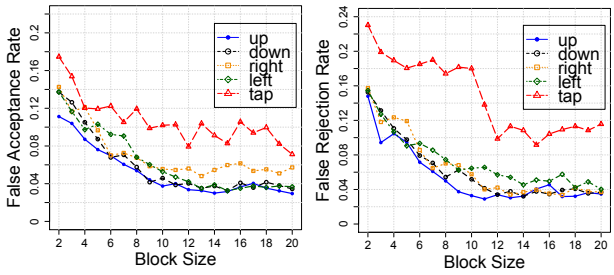
7.2.1 Gesture Block Size

A feature vector is calculated over a block of gestures. The block size is thus an important system parameter, which determines the number of gestures that our system needs to collect to perform a re-authentication. Hence, the size determines our system's re-authentication frequency. For each gesture type, we changed the necessary block size from 2 to 20. Given a block size and a gesture, for each target user, we generated 400 positive feature vectors and 400 negative ones in the training data set and the testing data set, respectively. We trained the classifier using a training data set, obtained a classification module, tested it using the testing set, and recorded false acceptance rates and false rejection rates. A false acceptance rate is the fraction of the testing data that is negative but classified as positive. A false rejection rate is the fraction of the testing data that is positive but classified as negative. In the sense of protecting smartphones, a false acceptance is more harmful than a false rejection. For each gesture type, we take the average false acceptance rates and the average false rejection rates over all target users' results. The results are shown in Figure 7.2.1, which contains

the two smartphones orientation modes, portrait mode and landscape mode. We note that, in this experiment, we took tap gesture as a single re-authentication gesture type and used its feature vectors to obtain a classification module in order to show its classifying accuracy. For each mode, we show the change of the average false acceptance rates and the average false rejection rates of each gesture type with increment of the block size. From the figure, we can see that,



(a) Portrait mode



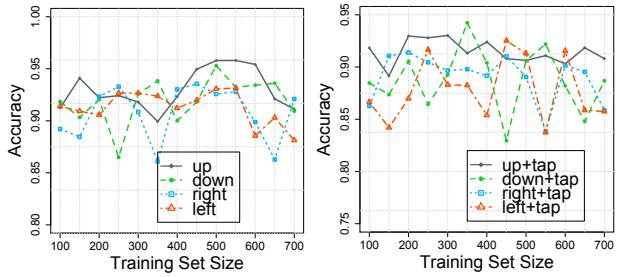
(b) Landscape mode

Figure 10. False acceptance rate/ false rejection rate vs. gesture block size

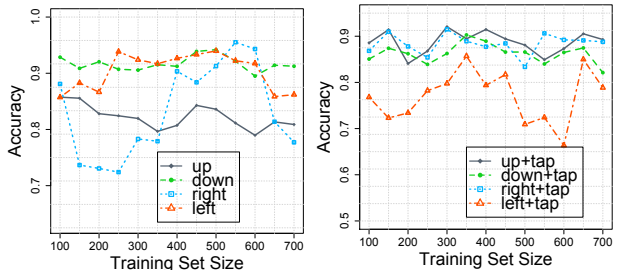
for a sliding gesture, its false acceptance and false rejection rates get stable when the block size is greater than 14. In both modes, the two rates of tap gesture are approaching stable when block size is getting close to 20 although they are not as stable as sliding gestures. Among the five types, tap has the worst performance, having the highest false acceptance rates and false rejection rates in both modes. This also confirms our previous analysis of the tap metrics in Section 6.3.2.

7.2.2 Training Size

The size of a training data set affects a module's classification accuracy, because a larger training data set gives the classification algorithm more information. We tested the performance of each classification module under different training set sizes, from 100 to 700 at intervals of 100. In the feature generation, we selected block size $n_s = 14$ and $n_t = 20$ to generate feature vectors. Our system monitors both portrait mode and landscape mode in the background, using 8 classification modules for each mode (Section 6.4). Given a training set size and a classification module, for each target user, we used the approach introduced in Section 7.1 to generate a training data set and a testing data set. Each testing data set was of the same size as its corresponding training data set. For each training set size, we obtained 16 classification modules for each user. We tested each classification module and recorded the classification accuracy. Then for each module and each training set size, we took the average of all user's classification accuracies. The results are shown in Figure 7.2.2. From Figure 11(a), we can



(a) Portrait mode



(b) Landscape mode

Figure 11. Training set size vs. classification accuracy

see that when the training set size increases, the accuracy of a classification module first increases, approaches to a maximum point, and then decreases. We observe that the maximum point is around 500 for single gesture type modules and around 300 for combinatory gesture type modules. The same trend is observed on the results under landscape mode in Figure 11(b). The observations indicate that tap gestures provided extra useful information to a combinatory gesture type module and the module thus did not need more training data to learn a user's patterns. The accuracy decreases after the training set size passes the maximum point because a large training data set makes the module specific to the training data so that it makes more errors in prediction.

Besides, we list the average classification accuracy for each classification module in Table 3 when the training size is 500 for single gesture type modules and 300 for combinational gesture type modules.

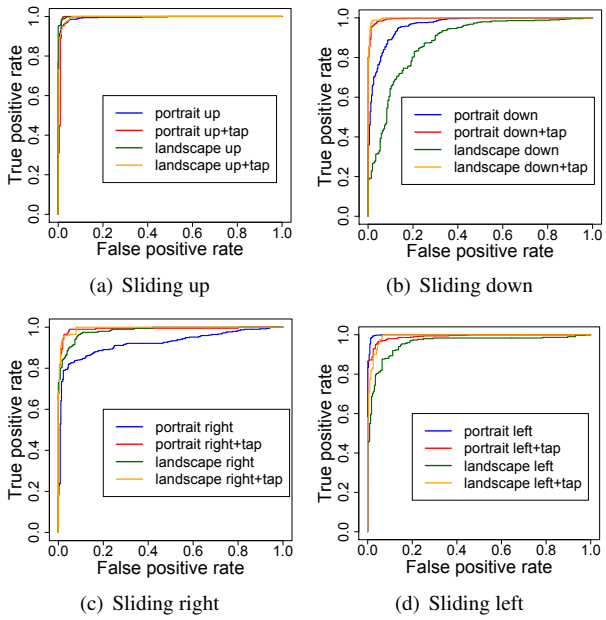
Table 3. Classification accuracy

PORTRAIT		LANDSCAPE	
gestures	**Accuracy**	**gestures**	**Accuracy**
sliding up	95.78%	sliding up	83.60%
sliding down	95.30%	sliding down	94.20%
sliding left	93.06%	sliding left	93.97%
sliding right	92.56%	sliding right	91.27%
up + tap	93.02%	up + tap	92.05%
down + tap	89.25%	down + tap	86.25%
left + tap	88.28%	left + tap	79.74%
right + tap	89.66%	right + tap	91.50%

7.2.3 Using Tap

In the study of classification, the receiver operating characteristic (ROC) curve is a good way to graphically reflect the performance of a binary classifier. It is a plot of true positive rate $TP/(TP + FN)$ versus false positive rate $FP/(FP + TN)$. Here, TP, FN, FP, and TN are the number of true positive predictions, false negative predictions, false positive predictions, and true negative predictions, respectively. A true positive prediction is a correct prediction on a positive class. False positive, true negative, and false negative predictions are defined in the similar fashion. Generally, if a ROC curve is close to the top-left corner, it indicates the corresponding classifier can obtain a high true positive rate with a low false positive rate. Therefore, such a classifier is considered to be a good classifier.

We fixed a target person and drew 16 ROC curves for his 16 classification modules. We set $n_s = 14$ and $n_t = 20$. The training data set size and the testing data set size were both 400 for all the 16 modules. The results are shown in Figure 7.2.3. The purpose is to test the improvement of using tap gesture as an auxiliary as well as the performance of each classification module. In all the plots, we can see that most modules having tap as an auxiliary gesture perform better than the ones having only sliding gesture types. For example, in Figure 12(a), two modules having tap gestures (red and yellow lines) are closer to the top-left corner than the other two lines. At the same time, we notice that sliding left performs better than "left+tap" combination in the landscape mode. Our single sliding gesture type modules also perform well in the two modes, since most of them are close to the top-left corner. Among the 16 modules, the classification modules for portrait and landscape sliding down gestures have the worst performance while the modules for sliding left gestures have the best performance. A possible explanation to this result is that people usually slide left for more contents while they are reading and they usually hold the contents, sliding slowly. In most cases, people slide down to get back to the top without reading. So they slide quick and slight. For a slow and "holding" sliding,



(a) Sliding up (b) Sliding down

(c) Sliding right (d) Sliding left

Figure 12. ROC curves for 16 classification modules

the screen can sense more personally specific information, which leads to a better classification performance.

7.2.4 System Overhead

As shown previously, our system needs as less as 14 same type slidings to construct a feature vector. The following table (Tab. 4) shows the average time interval for each gesture type according to our collected user data. For example, on average, a sliding up gesture happens every 8.24 seconds in portrait mode. Therefore, our system can collect 14 portrait sliding up gestures in 115.6 seconds, which means the system can usually re-authenticate a user in 2 minutes using sliding up gestures.

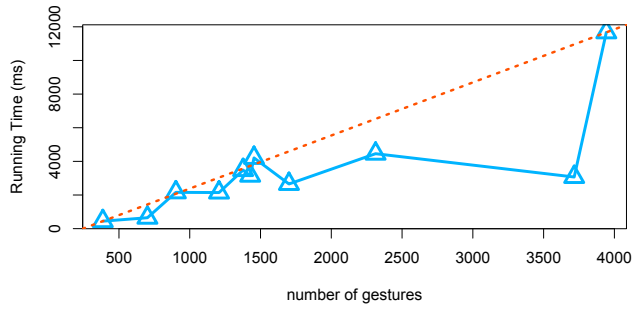
Table 4. Average time interval for gesture types (second)

	up	down	left	right	tap
portrait	8.24	14.25	37.13	22.47	14.12
landscape	12.14	19.23	50.74	34.27	18.73

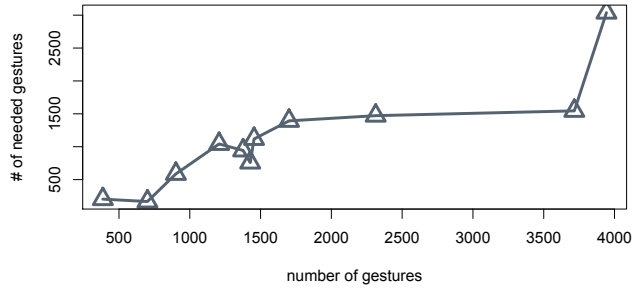
Learning an owner's patterns is deployed on a PC in our architecture and performed offline. Feature extraction and classification is performed online by a smartphone, which directly affects the user experience of our system. Given a feature vector, the classification can be done in a short time because our feature vector is of small dimension. Particu-

larly, given a combinatory feature with 13 feature values in it, our implementation only needed 17 milliseconds to give a prediction. The implementation used LIBSVM [7] on a Motorola Droid phone.

Feature extraction contains filtering necessary gestures and calculating each feature values, and thus takes more time. We tested our feature extraction scheme on a Motorola Droid smartphone with a single 550MHz A8 CPU, 256MB memory, 16GB sdcard, and Android 2.2 OS. We fed 386, 701, 902, 1207, 1377, 1427, 1454, 1701, 2313, 3716, and 3943 gestures to the preprocessing and the feature extraction modules on the smartphone. The running time and the number of filtered features are shown in Figure 13.



(a) Running time vs. total gesture number



(b) Filtered gesture number vs. total gesture number

Figure 13. Running time of feature extraction

In practice, gestures will be fed to our monitoring component immediately after they are captured by the screen. In some cases, the OS may buffer the gestures and suspend our system for a while to run another high priority process. Since security is important in many cases, we assume that our system is not suspended for a long time so that the number of gestures it deals with at one time is within several hundreds. Looking at the second point in Figure 13, we can see that filtering 427 needed gestures from 902 gestures and extracting features from them takes only 648 milliseconds. We note that this process was carried out on a low-end smartphone and we can expect a dramatic performance enhancement on current main-stream smartphones.

7.2.5 Discussion

We carried out all our data collections and experiments on Motorola Droid phones, which are equipped with low-end touch screens. Therefore, some metrics may be dropped due to the smartphone's hardware limitations. For example, we left out the pressure related metrics because the touch screen did not provide accurate pressure measurements. The metrics may need to be carefully tested or even re-designed before deploying our system on another smartphone platform. For example, pressure may become useful and provide more user information on some smartphone platforms. However, our work provides a guideline for the metric design on other platforms and our methodology can still be adopted. Our work shows that using gestures to construct an unobservable continuous re-authentication on smartphones is practical and promising.

8 Conclusions

In order to prevent unauthorized usage, we have proposed a re-authentication system using user finger movement. The system performs continuous re-authentication and does not need human assistance during re-authentication. We have discussed biometric feature design and selection for finger movement. We have demonstrated the effectiveness and efficiency of our system in extensive experiments.

Acknowledgement

This research was supported in part by ARO grant W911NF-09-1-0467 and NSF grants 1218038 and 0901451. The information reported here does not reflect the position or the policy of the federal government.

We would like to thank the anonymous reviewers for their comments on an earlier version of this paper. We would also like to thank Dr. Glenn Wurster for his guidance and valuable comments while shepherding the revision process of this paper. These comments have helped to improve our paper significantly both in content and in presentation.

References

[1] A. Ahmed and I. Traore. Anomaly intrusion detection based on biometrics. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 452–453. IEEE, 2005.

[2] A. Ahmed and I. Traore. A new biometric technology based on mouse dynamics. *IEEE Transactions on Dependable and Secure Computing*, 4(3):165–179, 2007.

[3] F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[4] Biometric Signature ID. BSI recommends all multi factor authentication include gesture biometrics to prevent data breaches to existing secure systems for highest cyber security. http://www.biosig-id.com/bsi-recomm ends-all-multi-factor-authentication-include-gesture-biometrics-to-prevent-data-breaches-to-existing-secure-systems-for-highest-cyber-security/, 2012.

[5] Canalys Inc. Smart phones overtake client PCs in 2011, http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011, 2012.

[6] C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *MobiSys*, pages 51–64, 2005.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] T. Clancy, N. Kiyavash, and D. Lin. Secure smartcard-based fingerprint authentication. In *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pages 45–52. ACM, 2003.

[9] M. Conti, I. Zachia-Zlatea, and B. Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of ASIACCS2011*, pages 249–259, Hong Kong, China, 2011. ACM.

[10] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and Communications Security*, pages 340–350. ACM, 2010.

[11] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.

[12] B. Duc, S. Fischer, and J. Bigun. Face authentication with gabor information on deformable graphs. *IEEE Transactions on Image Processing*, 8(4):504–516, 1999.

[13] H. Gamboa and A. Fred. A behavioral biometric system based on human-computer interaction. In *Proceedings of SPIE*, volume 5404, pages 381–392, 2004.

[14] Google Inc. Android developer guide. http://developer.android.com/guide/basics/what-is-android.html, Feb 2012.

[15] Google Inc. Android sensor manager, http://developer .android.com/reference/android/hardware/sensorma nager.html, Feb. 2012.

[16] Jaden. iOS 5 iphone lockscreen glitch allows you to bypass password protection and explore the phone.app! http://www.ijailbreak.com/iphone/ios-5-iphone-lockscreen-glitch/, Feb. 2012.

[17] John A. How to reset your android lock screen password. http://droidlessons.com/how-to-reset-your-android-lock-screen-password/, Mar. 2012.

[18] Z. Jorgensen and T. Yu. On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the ASIACCS2011*, pages 476–482, Hong Kong, China, 2011.

[19] Lookout Inc. Mobile phone lost and found. https://www.lookout.com/resources/reports/mobile-lost-and-found/billion-dollar-phone-bill, 2012.

[20] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto. Identifying users of portable devices from gait pattern with accelerometers. In *Proceedings of ICASSP'05*, volume 2, pages ii–973. IEEE, 2005.

[21] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Trans. Mob. Comput.*, 8(6):792–806, 2009.

[22] F. Monrose, M. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–83, 2002.

[23] F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56. ACM, 1997.

[24] Y. Nakkabi, I. Traoré, and A. Ahmed. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(6):1345–1353, 2010.

[25] F. Okumura, A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto, and A. Koike. A study on biometric

authentication based on arm sweep action with acceleration sensor. In *Proceedings of ISPACS'06*, pages 219–222. IEEE, 2006.

[26] M. Pusara and C. Brodley. User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8. ACM, 2004.

[27] M. Qi, Y. Lu, J. Li, X. Li, and J. Kong. User-specific iris authentication based on feature selection. In *Proceedings of ICCSSE 2008*, volume 1, pages 1040–1043. IEEE, 2008.

[28] H. Rydberg. Multi-touch protocol. . http://www.mjmwired.net/kernel/Documentation/input/multi-touch-protocol.txt, 2009.

[29] R. Yampolskiy and V. Govindaraju. Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 1(1):81–113, 2008.

[30] X. Zhao, L. Li, G. Xue, and G. Silva. Efficient anonymous message submission. In *INFOCOM, 2012 Proceedings IEEE*, pages 2228–2236, Orlando, FL, USA, March 2012. IEEE.

[31] N. Zheng, A. Paloski, and H. Wang. An efficient user verification system via mouse movements. In *Proceedings of ACM CCS2012*, pages 139–150. ACM, 2011.