College of William & Mary
Department of Computer Science

WM-CS-2012-06

You Are How You Touch: User Verification on Smartphones via Tapping Behaviors

Nan Zheng, Kun Bai, Hai Huang, and Haining Wang

December 06, 2012

You Are How You Touch: User Verification on Smartphones via Tapping Behaviors

Nan Zheng*, Kun Bai†, Hai Huang†, and Haining Wang*

*College of William and Mary
Williamsburg, VA, USA
{nzheng,hnw}@cs.wm.edu

†IBM T.J. Watson Research Center
Hawthorne, NY, USA
{kunbai,haih}@us.ibm.com

ABSTRACT

Smartphone users have their own unique behavioral patterns when tapping on the touch screens. These personal patterns are reflected on the different rhythm, strength, and angle preferences of the applied force. Since smartphones are equipped with various sensors like accelerometer, gyroscope, and touch screen sensors, capturing a user's tapping behaviors can be done seamlessly. Exploiting the combination of four features (acceleration, pressure, size, and time) extracted from smartphone sensors, we propose a non-intrusive user verification mechanism to substantiate whether an authenticating user is the true owner of the smartphone or an impostor who happens to know the passcode. Based on the tapping data collected from over 80 users, we conduct a series of experiments to validate the efficacy of our proposed system. Our experimental results show that our verification system achieves high accuracy with averaged equal error rates of down to 3.65%. As our verification system can be seamlessly integrated with the existing user authentication mechanisms on smartphones, its deployment and usage are transparent to users and do not require any extra hardware support.

Keywords

Smartphone, tapping behavior, user verification

1. INTRODUCTION

Smartphones have become ubiquitous computing platforms allowing users anytime access to the Internet and many online services. On one hand, as a personal device, a smartphone contains important private information, such as text messages, always-logged-in emails, and contact list. On the other hand, as a portable device, a smartphone is much easier to get lost or stolen than conventional computing platforms. Therefore, to prevent the private information stored in smartphones from falling into the hands of adversaries, user authentication mechanisms have been integrated into mobile OSes like Android and iOS.

Due to having a much smaller screen and keyboard on a smartphone than the traditional user input/output devices, PIN-based and pattern-based passcode systems have been widely used in smartphones for user authentication. However, many people tend to choose weak passcodes for ease of memorization. A 2011 survey on iPhone 4-digit passcode reveals that the ten most popular passcodes represent 15% of all 204,508 passcodes and the top three are 1234, 0000, and 2580 [1]. Moreover, recent studies show that an attacker can detect the location of screen taps on smartphones based on accelerometer and gyroscope readings and then derive the letters or numbers on the screen [6, 23, 27, 32]. An attacker could even exploit the oily residues left on the screen of a smartphone to derive the passcode [2]. Therefore, it is highly desirable to enhance the smartphone's user authentication with a non-intrusive user verification mechanism, which is user-transparent and is able to further verify if the successfully logged-in user is the true owner of a smartphone.

In this paper, we explore the feasibility of utilizing user tapping behaviors for user verification in a passcode-enabled smartphone. The rationale behind our work is that individual human users have their own unique behavioral patterns while tapping on the touch screen of a smartphone. In other words, you are how you touch on the screen, just like you are how you walk on the street. The rich variety of sensors equipped with a smartphone including accelerometer, gyroscope, and touch screen sensors, make it possible to accurately characterize an individual user's tapping behaviors in a fine-grained fashion. With over 80 smartphone users participated in our study, we quantify the user tapping behaviors in four different aspects: acceleration, pressure, size, and time. Based on the behavioral metrics extracted from these four features, we apply the one-class learning technique for building an accurate classifier, which is the core of our user verification system.

We evaluate the effectiveness of our system through a series of experiments using the empirical data of both 4-digit and 8-digit PINs. In terms of accuracy, our approach is able to classify the legitimate user and impostors with averaged equal error rates of down to 3.65%. Overall, our verification system can significantly enhance the security of a smartphone by accurately identifying impostors. Especially for practical use, our tapping-behavior-based approach is user-transparent and the usability of traditional passcodes on a smartphone remains intact. As our approach is non-intrusive and does not need additional hardware support from smartphones, it can be seamlessly integrated with the existing passcode-based authentication systems.

The remainder of the paper is structured as follows. Sections 2 and 3 review the background and related work in the area of smartphone user authentication, respectively. Section 4 describes our data collection and measurement, including our choice of metrics. Section 5 details the proposed classifier for user verification. Section 6 presents our experimental design and results. Section 7 discusses issues which arise from the details of our approach, and finally Section 8 concludes.

2. BACKGROUND

The tapping behaviors of individual users on touchscreen vary from person to person due to differences in hand geometry and finger agility. Each user has a unique personal tapping pattern, reflected on the different rhythm, strength, and angle preferences of the applied force. As our tapping-behavior-based approach verifies the owner of a smartphone based on "who you are" – your physical and behavioral traits, instead of "what you know", it belongs to biometrics-based user authentication. In general, a biometrics authentication system authenticates users either by their physiological traits like faces and voices [5, 21] or behavioral patterns like finger typing and hand movements [24, 34].

While physiological traits can achieve high accuracy in the process of user authentication, they have not been widely used in mobile devices. Recent studies have also shown that the physiology-based mechanisms deployed in mobile devices are sensitive to certain environmental factors, which could significantly diminish their accuracy and reliability. For example, face recognition may fail due to a different viewing angle and poor illumination [28], and voice recognition degrades due to background noise [5]. However, given the same mobile device, behavioral biometrics tend to be less sensitive to the surrounding environmental factors like darkness or noise.

Exploiting the behavioral information captured by multiple sensors on a smartphone, we can exclusively create a detailed user profile for verifying the owner of the smartphone. Since our approach works seamlessly with the existing passcode-based user authentication mechanisms in mobile devices, it plays a role of *implicit authentication*. In other words, our approach can act as a second factor authentication method and supplement the passcode systems for stronger authentication in a cost-effective and user-transparent manner. More recently, seminal works have been proposed to explore the feasibility of user verification employing the behaviors of pattern-based passwords [12]. However, the false reject rate (FRR) of their work is rather high, which means there is a high chance that the owner of a mobile device would be mistakenly regarded as an impostor and be blocked from accessing the device.

3. RELATED WORKS

A) Keystroke Dynamics and Graphical Passwords. Keystroke dynamics has been extensively studied in distinguishing users by the way they type their personal identification number (PIN) based passwords [19]. Research done on the analysis of keystroke dynamics for identifying users as they type on a mobile phone can be found in [3, 9, 10, 18, 25, 33]. Clarke *et al.* [10] considered the dynamics of typing 4-digit PIN codes, in which the researchers achieve an average Equal Error Rate (ERR) of 8.5% on physical keyboard on a Nokia 5110 handset. Zahid *et al.* [33] examined this approach on touchscreen keyboards and achieve, in one best scenario, a low Equal Error Rate of approximately 2% with training set required a minimum of 250 keystrokes.

Researchers have also suggested the use of graphical passwords as an easier alternative to text-based passwords [7, 17], based on the idea that people have a better ability to recall images than texts. A good overview of popular graphical password schemes has been reported in [4]. Chang *et al.* [7] proposed a graphics-based password KDA system for touchscreen handheld mobile devices. The experiment results show that EER is 12.2% in the graphics-based password KDA proposed system, and EER is reduced to 6.9% when the pressure feature is used in the proposed system. Different usability studies have outlined the advantages of graphical passwords, such as their reasonable login and creation times, acceptable error rates, good general perception and reduced interference compared to text passwords, but also their vulnerabilities [31].

B) Inferring Tapped Information from On-board Motion Sensors. Several independent researches have found that simply by using data acquired by smartphone motion sensors, it is sufficient to infer which part of the screen users tap on [6, 23, 27, 32]. The first effort was done by Cai *et al.* in 2011 [6]. They utilized features from device orientation data on an HTC Evo 4G smartphone, and correctly inferred more than 70% of the keys typed on a number-only soft keyboard. Very soon, Xu *et al.* further exploited more sensor capabilities on smartphones, including accelerometer, gyroscope, and orientation sensors [32]. Evaluation shows higher accuracies of greater than 90% for inferring an 8-digit password within 3 trials. Miluzzo *et al.* demonstrated another key inference method on soft keyboard of both smartphones and tablets [23]. 90% or higher accuracy is shown in identifying English letters on smartphones, and 80% on tablets. Owusu *et al.* [27] infers taps of keys and areas arranged in a 60-region grid, solely based on accelerometer readings on smartphones. Result showed that they are able to extract 6-character passwords in as few as 4.5 trials.

C) User Authentication by Their Behavior on Touch Screens. Research has been done in exploring different biometric approaches for providing an extra level of security for authenticating users into their mobile devices. Guerra-Casanova *et al.* [15] proposed a biometric technique based on the idea of authenticating a person on a mobile device by gesture recognition, and achieve Equal Error Rate (EER) between 2.01% and 4.82% on a 100-users base. Unobtrusive methods for authentication on mobile smart phones have emerged as an alternative to typed passwords, such as gait biometrics (achieving an EER of 20.1%) [13, 26], or the unique movement users perform when answering or placing a phone call (EER being between 4.5% and 9.5%) [11].

Very recently De Luca *et al.* [12] introduced an implicit authentication approach that enhances password patterns on android phones, with an additional security layer, which is transparent to user. The application recorded all data available from the touchscreen: pressure (how hard the finger presses), size (area of the finger touching the screen), x and y coordinates, and time. Evaluation is based on 26 participants, with an average accuracy of 77%.

A latest work conducted by Sae-Bae *et al.* [29] makes use of multi-touch screen sensor on iPad (not phone) to capture the palm movement. They achieved a classification accuracy of over 90%. However, palm movements is not suitable for smartphone screens, since the screen is typically too small for palm movements. Citty *et al.* [8] presented an alternative approach to inputting PINs on small touchscreen devices. It uses a sequence of 4 partitions of a selection of 16 images, instead of 4-digits PINs, to increase the possible combination of authentication sequences. However, inputting the sequence needs extra efforts in memorizing the images se-

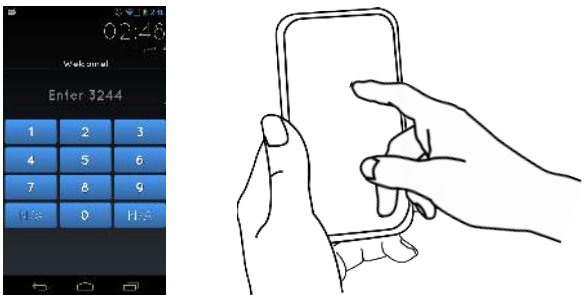
Table 1: Collected Data

PIN	Users	Actions	Average Actions Per User	Filtered-Out
3-2-4-4	53	1,751	33	0.80%
1-1-1-1	41	2,577	63	2.64%
5-5-5-5	42	2,756	66	3.70%
1-2-5-9-7-3-8-4	27	1,939	72	7.37%
1-2-5-9-8-4-1-6	25	2,039	82	4.76%

quences. Kim *et al.* [20] introduced and evaluated a number of novel tabletop authentication schemes that exploit the features of multi-touch interaction.

4. MEASUREMENT AND CHARACTERIZATION

Over 80 participants are involved in our data collection. Five different PINs are tested, in which three of them are 4-digit, and two are 8-digit. Here we choose PINs 3-2-4-4, 1-2-5-9-7-3-8-4, and 1-2-5-9-8-4-1-6 to represent these normal cases, but PINs 1-1-1-1 and 5-5-5-5 to represent the two extreme cases, one at the corner and the other at the center, respectively. Each participant is asked to enter an error-free PIN for at least 25 times and we collect a total of 11,062 error-free actions. The user's timing and motion data are recorded during the process. In this paper, we refer to an *action* (or user input action) as the process of tapping one PIN, instead of individual digits. The detailed information of the collected data is listed in Table 1.



(a) Application Layout (b) Two-Hand Typing

Figure 1: Screen layout of our data collection application, and the two-hand typing action.

The timing information is in resolution of milliseconds. Occasionally, some participants fail to make a smooth tapping intentionally or unintentionally. Therefore, we employ a simple outlier removal process to all the collected raw data. An outlier tapping action is often signaled by a markedly longer-than-usual time interval, especially for a user who is very familiar with its own PIN. In our data set, a *smooth* PIN tapping action takes at most 600 milliseconds between subsequent keys for all participants. As a result, an inter-key time of greater than one second always signals such an outlier behavior. By this standard, a small amount of raw data is filtered out, as listed in the right-most column of Table 1.

Table 2: Features of Touchscreen Tapping Behaviors

		# of Dimensions	
Feature Set	**Description**	**4-digit**	**8-digit**
Acceleration (linear & angular)	At `TouchDown`	8	16
	At `TouchUp`	8	16
	Min in key-hold	8	16
	Max in key-hold	8	16
	Mean in key-hold	8	16
Pressure	At `TouchDown`	4	8
	At `TouchUp`	4	8
Touched Size	At `TouchDown`	4	8
	At `TouchUp`	4	8
Time	Key hold time	4	8
	Inter-key time	3	7
Total	All features	63	127

All the data are collected on a Samsung Galaxy Nexus. Its fastest sampling rate on motion sensor readings is about 100Hz. Figure 1(a) shows the layout of our Android application for the data collection. In the experiments, all the participants are asked to hold the phone with their left hands, and tap with their right hand index fingers, as shown in Figure 1(b).

We make use of the Android APIs to detect the touch event, including both key-press and key-release. Between each key-press and key-release, we record raw data of timestamps, acceleration, angular acceleration, touched-size, and pressure. Acceleration and angular acceleration are from API `SensorEvent`, while touched-size and pressure are from API `MotionEvent`.

4.1 Feature Extraction

Based on the raw data, we compute four sets of features for each PIN typing action: acceleration, pressure, size, and time. We describe each of them in the following:

- **Acceleration**: For each digit d in a PIN action, we calculate the five acceleration values:
  - $A_{d,1}$: the magnitude of acceleration when the digit d is pressed down;
  - $A_{d,2}$: the magnitude of acceleration when the digit d is released;
  - $A_{d,3}$: the maximum value of magnitude of acceleration during digit d key-press to key-release;

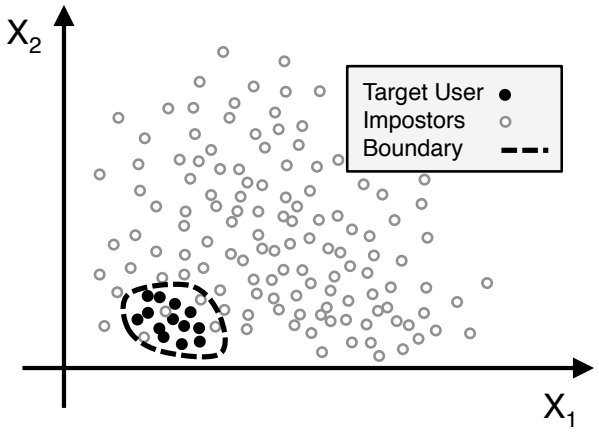


Figure 2: An illustration of two-feature space of a target user and many others. X_1 and X_2 are the two features. The dashed lines define the boundary of the target user's behavior. Because the target user's behavior is limited to a concentrated area, the boundary blocks the majority of potential impostors.

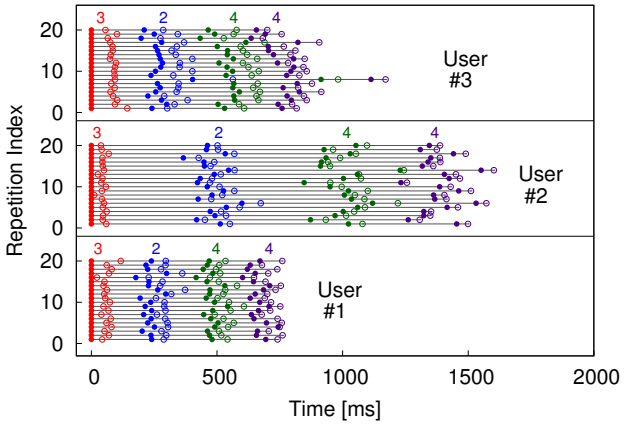


Figure 3: Timing of tapping on the smartphone from three different users, shown in three vertical panels. Each user typed 20 times of the number string "3244". The solid dots represent key-press time, and the open dots are key-release time. Different colors represent the timestamps of different digits.

  - $A_{d,4}$: the minimum value of magnitude of acceleration during digit d key-press to key-release;
  - $A_{d,5}$: the average value of magnitude of acceleration during digit d key-press to key-release.

  All above values are the magnitude of acceleration $\|\vec{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. We choose not to use individual components, because the phone coordinate system is sensitive to location change. A similar procedure is applied to calculate the features from angular accelerations. Combining both acceleration- and angular-acceleration-related features, there are total of 40 in a 4-digit PIN action and 80 in an 8-digit PIN action.

- **Pressure**: We obtain the pressure readings through Android API `MotionEvent.getpressure()`. The returned pressure measurements are of an abstract unit, ranging from 0 (no pressure at all) to 1 (normal pressure), however the values higher than 1 could occur depending on the calibration of the input device (according to Android API documents). In the feature set, we include pressure readings at both key-press and key-release. There are 8 pressure-related features for a 4-digit PIN, and 16 for an 8-digit PIN.

- **Size**: Similar to pressure readings, another Android API call `MotionEvent.getsize()` measures the touched size, associated with each touch event. According to Android document, it returns a scaled value of the approximate size for the given pointer index. This represents the approximation of the screen area being pressed. The actual value in pixels corresponding to the touch is normalized with the device's specific range and is scaled to a value between 0 and 1. For each key-press and key-release, we record the size readings and include in the feature set. A 4-digit PIN contains 8 size-related features, and an 8-digit PIN contains 16.

- **Time**: key-hold times and inter-key time intervals between two nearby keys. They are measured from the `TouchEvent` timestamps, of both `TouchUp`s and `TouchDown`s. Overall, a 4-digit PIN action contains 7 time-related features, while 8-digit PIN contains 15.

For a 4-digit PIN, each action results in a total of 63 features; for an 8-digit PIN, the number of features for one action is 127. Table 2 summarizes the description of the above four feature sets.

4.2 Touchscreen Tapping Characterization

Our underlying assumption is that a user's feature distribution should be clustered within a reliably small range compared with many others. As a result, those metrics can be exploited to block the majority of impostors, as illustrated in Figure 2.

4.2.1 Uniqueness of User Pattern

As described above, we define four sets of features in order to characterize a user's tapping behaviors on smartphones: acceleration (both linear and angular), pressure, size, and time. All these features can be easily obtained from a smartphone's on-board sensors, and can accurately characterize a user's unique tapping behaviors. Based on the feature data, we observe that each user demonstrates consistent and unique tapping behaviors, which can be utilized for differentiating itself from other users.

Figure 3 shows the timestamps of entering the same PIN 3-2-4-4 from three different users, including the moments of each key-press and key-release. Each individual's timing patterns clearly differ, but are very consistent within themselves. This is similar to the observations on a regular computer keyboard [22].

In addition to timing information, motion data such as pressure, touched size, and acceleration also reveal user-specific patterns. Generally speaking, acceleration is proportional to the tapping force applied to the touchscreen, while angular acceleration represents the moment of force.

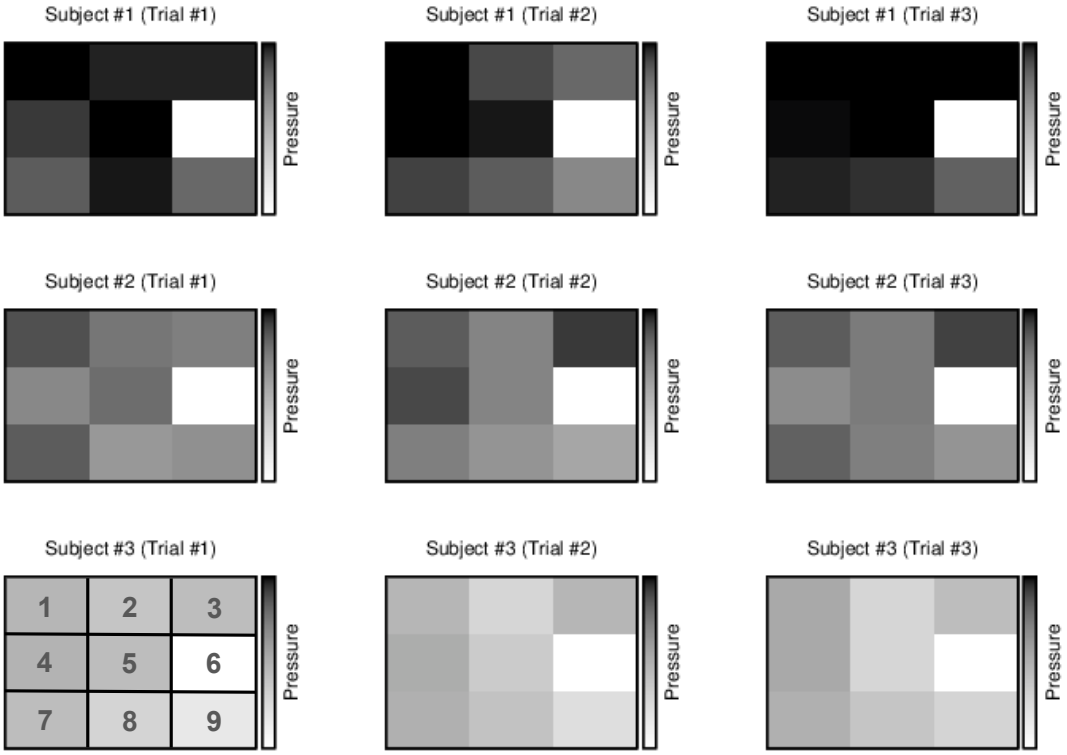


Figure 4: Users' tapping pressure on smartphone touchscreen, while entering an 8-digit PIN 1-2-5-9-7-3-8-4. Each figure shows pressure readings on a 3×3 smartphone number pad. Darker color indicates a larger tapping pressure. Note that number "6" has no pressure because it is not in the PIN. Figures in the same row are from a same user while typing the PIN for three times.

Touched size is related to both user finger size and tapping force. Figure 4 shows the tapping pressure from three different users. We can see that three different users' tapping pressure form distinguishable individual patterns, with Subject #1 taps the hardest, Subject #2 taps much more gently, and Subject #3 is gentlest. Meanwhile, the level of tapping pressure is relatively consistent within one subject.

4.2.2 Dissimilarity Measures

We represent each user action as n-dimensional feature vectors, where n is the number of feature dimensions. Using the *dissimilarity score* between two feature vectors, we further verify if our extracted features of a user remain relatively stable over multiple repetitions, in comparison with those of the other participants.

As the first step, we compute a target user's *template* as an average feature vector over its N PIN tapping actions, where $N = 150$ in our case. At the same time, each feature's standard deviation is computed based on these N actions.

In our approach, given a new biometric data sample, we evaluate its *dissimilarity score* from the target user's template as follows. Suppose the new data sample's feature vector is $\mathbf{X} = \{X_1, X_2, ..., X_i, ..., X_n\}$, where X_i represents the ith feature dimension; and the target user's template is represented similarly as $\overline{\mathbf{T}} = \{\overline{T}_1, \overline{T}_2, ..., \overline{T}_n\}$. The dissimilarity score is the accumulated deviation from the two vectors over all *normalized* features:

$$D(\mathbf{X}, \overline{\mathbf{T}}) = \sum_i \left\| \frac{X_i - \overline{T}_i}{\sigma_i} \right\|, \quad (1)$$

where σ_i denotes the standard deviation of the ith feature over the N trials in obtaining the target user's template. By dividing σ_i, we give higher weights to those features that have smaller variation within the target user, because they more reliably reflect the target user's specific pattern. This is a standard procedure mostly seen in outlier removal (also known as standard score or z-score in statistics [30]).

Figures 5, 6, and 7 show the distributions of dissimilarity scores, calculated from a target user's template entering three different PINs, respectively, to both the target user itself and the rest of other users. It is clear that in all three PINs, the dissimilarity scores to the target user itself is highly concentrated on the lower end, indicating a high similarity to its own behavioral template. Meanwhile, the dissimilarity scores of *other* users are dispersed and located on the higher end. For the 4-digit PIN 3-2-4-4 (Figure 5), there is a small overlap of the target user itself with others. It implies that only few members among the other 52 users behave similarly to the target user, and may be misclassified. For the two 8-digit PINs (Figures 6 and 7), the target user's and others' distribution curves are *completely* separated with a clear gap in between. Likely this is because an 8-digit PIN action contains more cognitive information that is user-specific than a 4-digit PIN action.

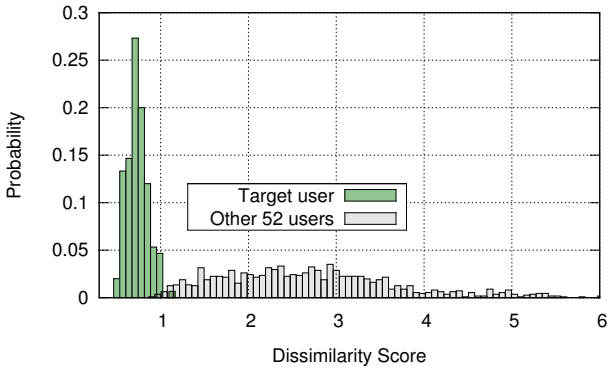


Figure 5: Distribution of dissimilarity score of typing 3-2-4-4 from a target user's template, to both the target user itself and other 52 users.

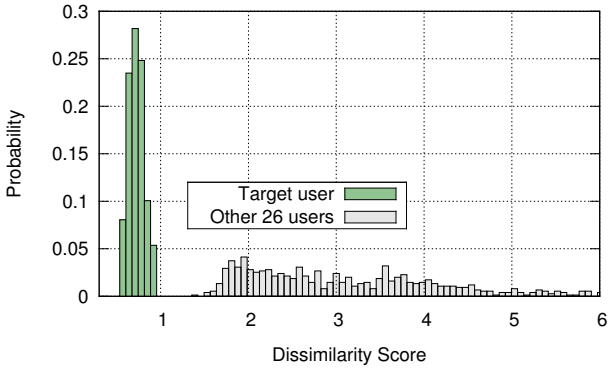


Figure 6: Distribution of dissimilarity score of typing the 8-digit PIN 1-2-5-9-7-3-8-4 from a target user's template, to both the target user itself and other users.

5. CLASSIFICATION

The system architecture of our approach consists of a feature module, a classifier module, and a decision maker module as shown in Figure 8. Firstly, raw data are recorded during user's tapping actions. Then, four sets of features are calculated and fed into the classifier, which derives a decision score featuring its similarity to the target user's template. The decision score is used by the decision maker to make a final decision, with respect to a predefined threshold value. The final decision is to label whether an user tapping action is originated from the target user or an impostor.

User behavioral pattern can be derived from either one-class or two-class learning. In one-class learning, only the target user's data is needed in training phase; but the learned model can be applied to classify both the target user or an unknown impostor. Additionally, if other users' data are available, together with the target user's own data, we can conduct a two-class learning. One-class learning is straightforward and more practical because it does not involve other users' data, but with slightly lower verification accuracy. For a two-class classifier, device manufacturers could pre-load some anonymized user data into smartphones before shipping them to their customers. With the pre-load anonymized user data, two-class classification is also feasible to perform

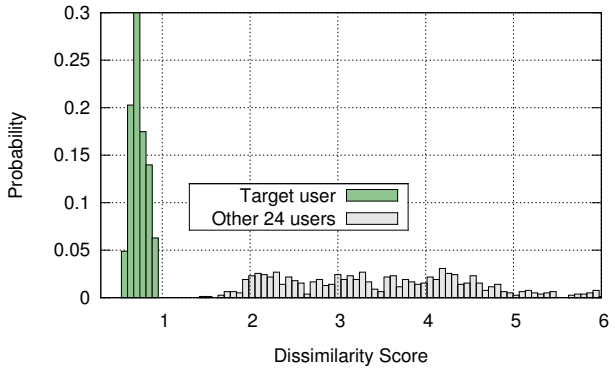


Figure 7: Distribution of dissimilarity score of typing another 8-digit PIN 1-2-5-9-8-4-1-6 from a target user's template, to both the target user itself and other users.

in practice and can achieve higher verification accuracy. Due to page limit, here we only present one-class learning and its evaluation results in Section 6.

Our one-class learning process consists of the enrollment and testing phases. In the enrollment of a target user I, taking its N input actions, we calculate the standard deviations of every feature as σ_j for the jth feature. In the testing phase, given an unknown sample as n-dimensional feature vector X_Q, its distance from each of the N feature vectors in the enrollment phase is calculated as:

$$d(X_Q, X_i) = \sum_{j=1}^{n} \frac{\|X_{Q,j} - X_{i,j}\|}{\sigma_j}, \quad i = 1, ..., N, \qquad (2)$$

where $X_{Q,j}$ is the jth feature of feature vector X_Q, and $X_{i,j}$ is the jth feature of the ith feature vector in the enrollment phase. Following this, the distance of X_Q's nearest neighbor $d_{min}(X_Q, I)$ will be chosen as the dissimilarity measurement to the target user's template. The underlying assumption is that if X_Q belongs to the target user, it should have a short distance to its nearest neighbor in the target user's data. And if $d_{min}(X_Q, I)$ is below a pre-defined threshold value, it is labeled as from the target user; otherwise, it is labeled as from impostors. Implementation wise, setting a large threshold value means a higher probability of recognizing the target user, but allowing more impostors to slip through. A small threshold value strictly blocks out impostors, but may falsely reject the target user.

6. EXPERIMENTAL EVALUATION

Generally, the accuracy of a biometrics-based authentication is evaluated by the following error rates:

- False Reject Rate (FRR) — the probability that a user is wrongly identified as an impostor;
- False Accept Rate (FAR) — the probability that an impostor is incorrectly identified as a legitimate user.

The point at which both FAR and FFR are equal is denoted as the Equal Error Rate (EER). The value of EER can be obtained by tuning a certain threshold until FAR and FAR are equal.

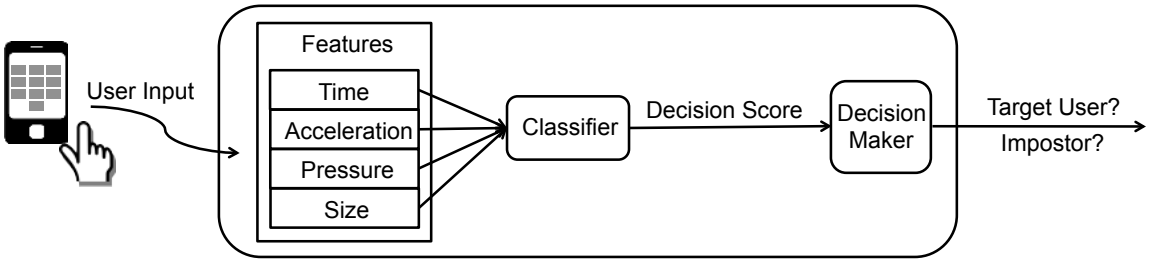


Figure 8: System Architecture

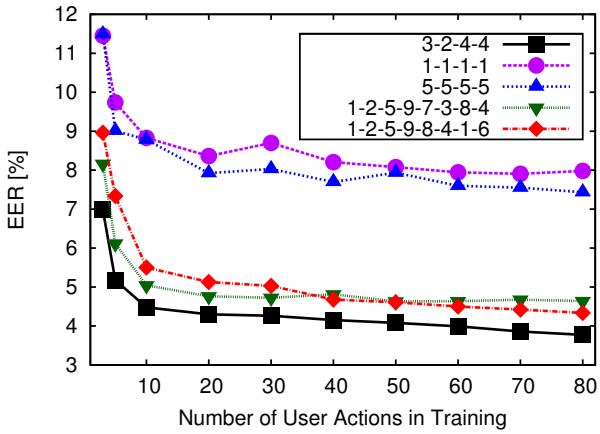


Figure 9: Variation of average EER with number of user actions in one-class training for five PIN combinations.

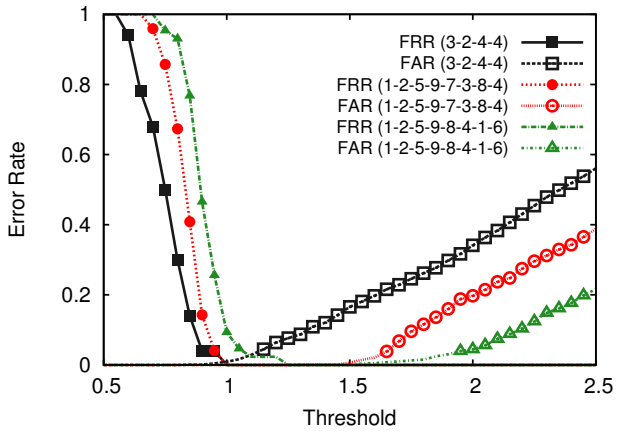


Figure 10: Variation of FRR and FAR with the value of threshold for 3-2-4-4 and the two 8-digit PINs for one target user.

A formal description of a biometric-based verification system is summarized as [16]: given an unknown sample to be verified towards a target user I, its feature vector X_Q is compared with the target user's template X_I. A dissimilarity score $D(X_Q, X_I)$ is calculated, where D is a function that evaluates the dissimilarity between two feature vectors. The dissimilarity function D varies with different methods of classification. Finally, a threshold value t is set to determine if X_Q is from the target user or an impostor:

$$(I, X_Q) \in \begin{cases} \text{target user}, & \text{if } D(X_Q, X_I) \leq t \\ \text{impostor}, & \text{otherwise} \end{cases}$$

Tuning the threshold would give the classifier a preference towards either the target user or the impostors, thus reducing one error rate while increasing the other. Because our approach acts as a second factor authentication, which supplements the passcode-based mechanisms for higher assurance authentication in a cost-effective fashion, we focus more on being user-transparent and user-friendly while enhancing the security of PIN-based authentication.

In the following, we present the evaluation results of the one-class based verification system, along with the effect of threshold and number of actions in training, the comparison with different combination of PINs, and the associated system overhead.

6.1 Verification Accuracy

There are two parameters that affect the accuracy in one-class learning: the number of actions in training, and the threshold.

By increasing the number of actions in training, user behavioral patterns become more precise since more actions yield a higher statistical significance. Figure 9 shows that the averaged equal error rate (EER) decreases as more user actions are included in training. All five PIN combinations present similarly shaped curves, while the results of 1-1-1-1 and 5-5-5-5 are less accurate than those of 3-2-4-4 and the two 8-digit PINs. From lower accuracy of 1-1-1-1 and 5-5-5-5, it seems that a PIN number with higher repetition of digits reduces the difference in individual users' tapping behaviors, leading to a less accurate verification result. Moreover, for all five PINs, the accuracy remains on a similar level after 20 user actions. This implies, as more user actions are added in training, there is a diminishing gain in accuracy. For example, increasing user actions from 20 to 40 requires twice the time waiting for user input, but only limited accuracy increase is seen.

Table 3 further lists the exact values of averaged EER, with its standard deviation in parenthesis. In computing EERs, there are 85 user actions included in the training process. As shown in Figure 10, in all three PIN combinations, there is a trade-off between FRR and FAR.

As mentioned earlier, four sets of features are included: acceleration, pressure, size, and time. To measure how the four sets of features contribute to the final accuracy, we make four additional rounds of classification, solely based on each

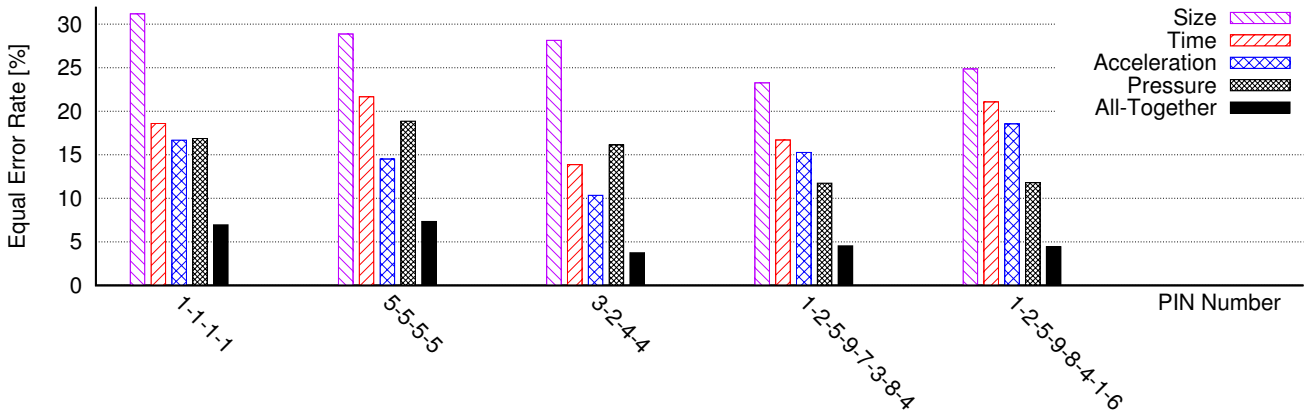


Figure 11: Comparison of performance from each feature set in one-class learning, as well as when they are combined together.

Table 3: User Verification Accuracies

PIN	Equal Error Rate (EER)[a]
3-2-4-4	3.65% (3.58%)
1-1-1-1	6.96% (6.01%)
5-5-5-5	7.34% (5.38%)
1-2-5-9-7-3-8-4	4.55% (6.23%)
1-2-5-9-8-4-1-6	4.45% (4.15%)

[a] with standard deviation in parenthesis

feature set. Figure 11 shows the accuracy results for the four individual feature sets, as well as those of combining them all together.

It can be seen from Figure 11 that, the combination of all four feature sets always outperforms individual feature set, as it is always with the smallest EER in all different scenarios. This is because the four feature sets capture the different aspects of user tapping behaviors, and having them all together should most precisely represent who the target user is. Meanwhile, among the four individual feature sets, acceleration, pressure, and time perform similarly well and achieve more accurate results than size.

6.2 System Overhead

In our implementation, the verification system is entirely built on a smartphone. As a stand-alone system, there is only a single user present for verification at any given time. There is no communication overhead associated with our user verification.

We first estimate the memory overhead of the verification process. The verification process is profiled using the Android SDK tool DDMS, and we find out that it only consumes 11.195 MBytes of heap memory during a one-class testing process and this memory consumption is a one-time cost. The computational overhead is the sum of CPU costs in raw data processing (calculating features) and detecting (including classifying and decision making). The preprocessing on one user input action of a 4-digit PIN takes only 0.022 seconds. The detecting process takes another 0.474 seconds, where the major part lies in finding the nearest neighbor from all 85 reference feature vectors. The CPU cost is measured on a Samsung Galaxy Nexus, using two `Date.getTime()` utility call at the beginning and end of the running time. Overall, the induced computational overhead is minor on the smartphone. In terms of disk space for storing user template, the signature of a single user profile generated by the training process consumes only 150.67KBytes. It is very affordable on an entry-level smartphone, let alone high-end models.

7. ADDITIONAL ISSUES IN REALITY

7.1 Multiple Positions

So far we only measure the user tapping behaviors in a given position. However, it is quite possible that a user types in its passcode under different positions (e.g., single handed using the thumb). To handle different input positions, we can measure and store multiple behavioral patterns for different positions during the training period. The rich sensors equipped with smartphones allow us to easily detect the physical position of the device and choose the appropriate behavioral pattern for verification. For example, accelerometer readings straightforwardly signal if a user is in a moving or non-moving status. And gyroscope readings can even infer the user's hand position (one-handed vs. two-handed) when tapping on the smartphone, as shown in a recent study [14].

To explore on multiple tapping positions, we further conduct two more sets of empirical measurements. First, we collect data with only one-handed tapping, which is in parallel with the two-handed case in Section 6. In one-handed tapping, the phone is held in one hand, and tapped with the thumb finger of the same hand. Due to the use of a different finger, the one-handed tapping behavior is different from that of two-handed. However, with the one-handed data set as the training data, we perform the same evaluation process as in Section 6 and achieve an average EER of 3.37% over all PINs in the on-handed case, indicating the effectiveness of our approach just like in two-handed tapping.

In addition to different hand positions, there are also various body positions a user would switch from time to time. While tapping its passcode, a user can be sitting, standing,

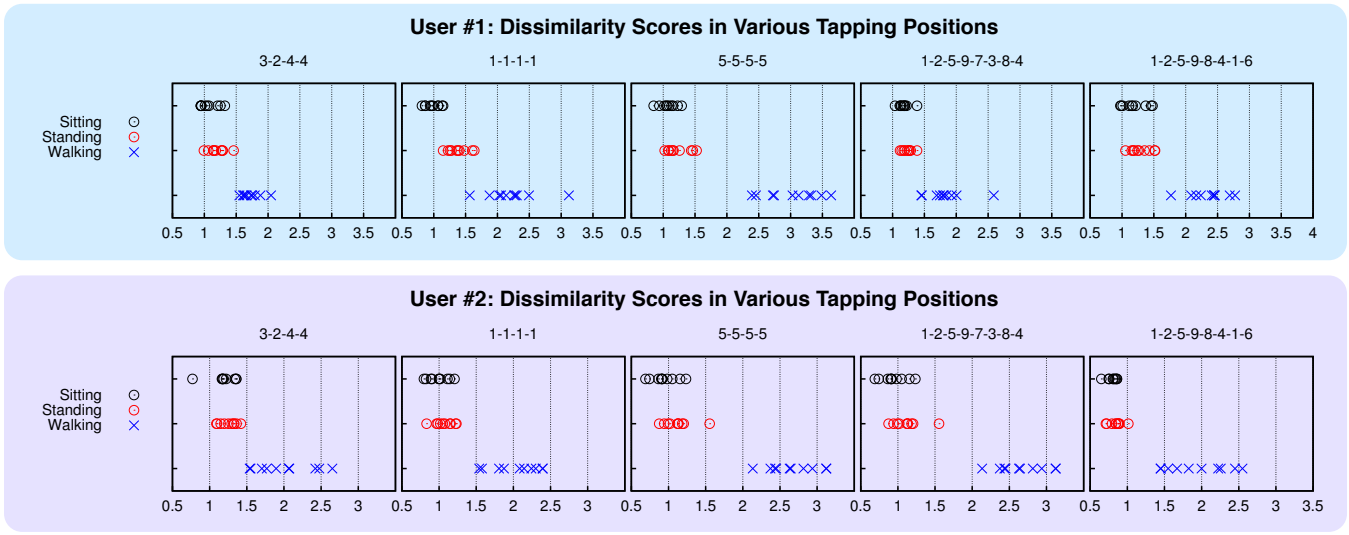


Figure 12: Dissimilarity scores from one tapping position (i.e., one-handed while sitting) to the other two tapping positions of a same user. Upper and lower panels are for the two different users, respectively.

or even slow walking. It is desirable to see how different body positions affect a user's tapping behavior. To answer this question, we carry out an additional experiment with two users, who tap in PINs with three body positions: sitting, standing, and walking. Using the trained model with one-handed tapping while sitting as the baseline, Figure 12 shows the dissimilarity scores to three different tapping positions, sitting, standing, and walking. Our major observation is that: as long as the user remains static, its tapping behaviors are similar under different body positions. Note that we do *not* intend to cover all possible tapping positions in different environments (which is almost impossible), but instead to draw some insights based on the common scenarios.

Our approach will work well for different input positions: sitting or walking, single-handed or two-handed. The challenge is merely to increase the training period and cover different input positions with more feature sets, which will impose a larger memory and CPU overhead in verification. However, we could further reduce the system overhead by optimizing the classifier implementation from different aspects of mobile devices. Note that the current one-class classifier has not been optimized. We will further explore this direction in our future work.

7.2 Mimic Attacks

Theoretically, our behavior-based verification system can be bypassed if an impostor can precisely mimic the tapping behaviors of the device's owner. However, this is extremely difficult if not impossible in practice. Intuitively, even if the impostor has overseen how the device's owner previously entered the passcode, it might be able to mimic the timing aspect. But the other features, such as pressure, acceleration, and size, are much more difficult to observe and reproduce.

In order to quantitatively measure the effect of mimic by observation, we set up an experiment involving three users. One of them is the target user who is observed closely by the other two "impostors" who try to mimic the target user's tapping behaviors. Impostor #1 has the same gender and similar hand/finger size as the target user, while impostor #2 has different gender and larger hands/fingers than the target user. The goal is to see how physiological differences can impact the outcome of mimic attacks. Before mimicking, the two impostors closely observe (over the shoulder) how the target user tapped in the PIN 3-2-4-4 for 10 times each. They are also guided to especially pay attention to the four features in our approach, i.e., tapping rhythm, acceleration, pressure, and touched size.

Figure 13 plots the dissimilarity scores from the target user's model to the two impostors before and after their mimic trials. The leftmost subfigure corresponds to all features included, and the rest four correspond to individual contribution from the four different features, respectively (acceleration, pressure, size, and time). Our experimental results clearly show that there is no significant improvement in mimicking given the behavior observation. Taking all four features into account, it is evident that a mimic attack is very hard to succeed. For each individual feature (acceleration, pressure, size, and time) shown in Figure 13, we can see that only the dissimilarity scores of acceleration are consistently reduced (i.e., its score range shifts towards that of the target user after observation). However, for the other three features (including pressure, size, and time), out of the 10 mimic attempts, just one or two trials may be slightly closer to the target's model, but their score ranges spread even wider. Thus, the behavior mimicking does not increase the chance of evasion with respect to these three features. This somewhat contradicts our intuition that timing would be easier to mimic than the other features.

With the experimental results shown in Figure 13, we believe that the robustness of our approach against a mimic attack mainly lies in the following three aspects.

- There are multiple dimensions in the features we used and most of them are independent from each other. Although an impostor may mimic one dimension without much difficulty, mimicking multiple dimensions simultaneously is extremely difficult as small physical movements like tapping are hard to observe and precisely reproduce. For example, acceleration directly

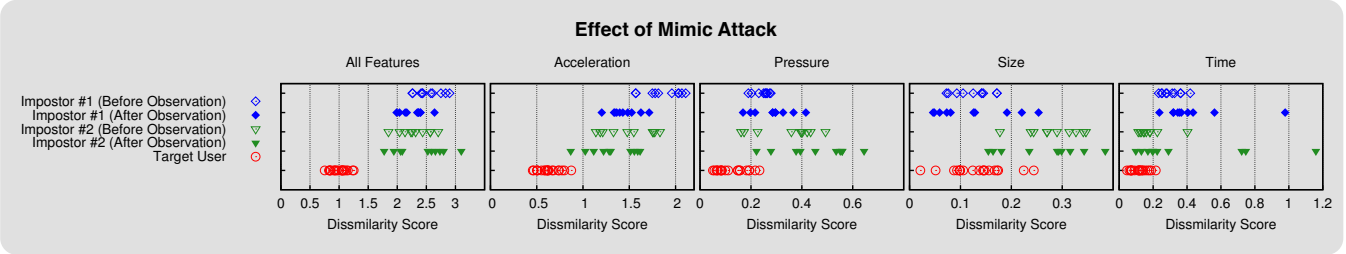


Figure 13: Effect of mimic attack shown in degree of dissimilarity from the target user. Subfigures, from left to right, correspond to all features considered, and each individual feature set (acceleration, pressure, size, and time). There are two mimic "impostors", and 10 trials before and after their observations on the target user.

relates to tapping force ($F = m \cdot a$), so if the impostor intentionally manages to tap in a gentler or harder fashion, its behavior can get closer to that of the target user. However, pressure is harder to mimic because it equals to tapping force divided by touched area. These two independent factors must be adjusted at the same time, which is more challenging. Timing (or tapping rhythm) is also hard to mimic, because timing contains multiple dimensions in our approach: 7 in a 4-digit PIN, and 15 in a 8-digit PIN. Those individual time intervals (especially key-to-key intervals) are relatively independent. An impostor may mimic the target user with a roughly fast or slow rhythm, but it is hard to reproduce the specific key-to-key dynamics.

- The fine-grained measurement resolution makes our features hard to mimic. For example, in our experiment, timing is measured in order of millisecond. This time resolution is much higher than human perception, and hence it is very hard for an impostor to accurately and consistently mimic tapping rhythm at such a low-level resolution.

- The physiological differences from the target user set up another barrier for mimic impostors. In our feature set, the touched size is heavily affected by the finger size, and the tapping rhythms also depend on hand agility and geometric shape. In general, it is very difficult for a person with bigger hand/fingers to mimic someone with smaller hand/fingers, and vice versa.

As more sensors have been available on mobile devices, more features will be included for more accurate user verification, and hence mimic attacks will just become less likely to succeed.

7.3 User Behavior Changes

This work builds on the assumption that a user's behavior is consistent and no abrupt change happens over a short period of time, but the assumption might not always be true, e.g., due to a physical injury. In such scenarios, the behavioral-based verification mechanism should stay minimally intrusive to the user. One feasible solution is to contact with the service providers to disable the verification function remotely and start the re-training. The purpose of our user verification is to provide additional security in common day-to-day usage while still allowing the user to disable it in rare cases. As we have shown previously, the sensitivity to false positives and negatives are controlled by various threshold values. Whether or not exposing the sensitivity control, e.g., setting it to Low, Medium, and High, can improve user experience is debatable. On one hand, it allows users to make a conscience choice to trade off between security and convenience. On the other hand, it is no longer user-transparent.

7.4 Passcode Changes

In our approach, only the tapping features of the currently active passcode are measured and recorded in a user's smartphone. One might ask what happens when the user need to change its passcode? Although people do not frequently change their passcodes, updating passcode in a quarterly or yearly basis is recommended or required by most passcode-based systems. When this happens, our verification system could automatically remain inactive for a while and start another training session to build a new set of tapping features based on the newly created passcode. The characterization of tapping features are conducted in background till a stable pattern has been successfully compiled after multiple trials. Note that the methodology of our scheme is not bounded to certain passcodes. In other words, our approach can be applied to any passcode a user chosen in practice.

8. CONCLUSION

As mobile devices are getting widely adopted, ensuring their physical and data security has become a major challenge. A simple peek over the shoulders of the device owner while the passcode is being entered and a few minutes of hiatus would allow an attacker to access sensitive information stored on the device. Using more complex passcodes and/or secondary passcodes can reduce the chance of such attacks, but it brings significant inconvenience to the users. We have found that a user's tapping signatures if used in conjunction with the passcode itself can also achieve the same goal, and moreover, the added security can be obtained in a completely user-transparent fashion. Previous works have shown the feasibility of this approach, but their high error rate makes these mechanisms impractical to use as too many false positives will defeat the purpose of being user-transparent. Having collected data of over 80 different users, explored the one-class machine learning technique, and utilized additional motion sensors on newest generation of mobile devices, we are able to demonstrate accuracies with equal error rates of down to 3.65%.

9. REFERENCES

[1] D. Amitay. Most Common iPhone Passcodes. `http://amitay.us/blog/files/most_common_iphone_passcodes.php`. [Accessed: Nov. 2012].

[2] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, pages 1–7, 2010.

[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] R. Biddle, S. Chiasson, and P. Van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR)*, 44(4):19, 2012.

[5] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, and D. A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP J. Appl. Signal Process.*, 2004:430–451, Jan. 2004.

[6] L. Cai and H. Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *USENIX Workshop on Hot Topics in Security (HotSec)*, 2011.

[7] T.-Y. Chang, C.-J. Tsai, and J.-H. Lin. A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. *J. Syst. Softw.*, 85(5):1157–1165, May 2012.

[8] J. Citty and D. R. Hutchings. TAPI: Touch-screen authentication using partitioned images. In *Elon University Technical Report 2010-1*, Tech Report, 2010.

[9] N. Clarke, S. Karatzouni, and S. Furnell. Flexible and transparent user authentication for mobile devices. *Emerging Challenges for Security, Privacy and Trust*, pages 1–12, 2009.

[10] N. L. Clarke and S. M. Furnell. Authenticating mobile phone users using keystroke analysis. *Int. J. Inf. Secur.*, 6(1):1–14, Dec. 2006.

[11] 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 the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS'11, pages 249–259, 2011.

[12] 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 *ACM CHI'12*, pages 987–996, 2012.

[13] M. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 306–311. IEEE, 2010.

[14] M. Goel, J. Wobbrock, and S. Patel. Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *UIST'12*, pages 545–554, 2012.

[15] J. Guerra-Casanova, C. Sanchez-Avila, G. Bailador, and A. de Santos Sierra. Authentication in mobile devices through hand gesture recognition. *International Journal of Information Security*, 11(2):65–83, Apr. 2012.

[16] A. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4 – 20, jan. 2004.

[17] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin. The design and analysis of graphical passwords. In *USENIX Security Symposium*, pages 1–14, 1999.

[18] S. Karatzouni and N. Clarke. Keystroke analysis for thumb-based keyboards on mobile devices. *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 253–263, 2007.

[19] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE DSN'09*, pages 125–134, 2009.

[20] D. Kim, P. Dunphy, P. Briggs, J. Hook, J. W. Nicholson, J. Nicholson, and P. Olivier. Multi-touch authentication on tabletops. In *ACM CHI'10*, pages 1093–1102, 2010.

[21] C. Mallauran, J.-L. Dugelay, F. Perronnin, and C. Garcia. Online face detection and user authentication. In *Proceedings of the 13th Annual ACM International Conference on Multimedia (MM)*, pages 219–220, 2005.

[22] R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *IEEE DSN'10*, pages 201–210, 2010.

[23] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: your finger taps have fingerprints. In *ACM MobiSys'12*, 2012.

[24] F. Monrose and A. D. Rubin. Authentication via keystroke dynamics. In *ACM CCS'97*, pages 48–56, 1997.

[25] M. Nauman and T. Ali. Token: Trustable keystroke-based authentication for web-based applications on smartphones. *Information security and assurance*, pages 286–297, 2010.

[26] C. Nickel, M. Derawi, P. Bours, and C. Busch. Scenario test for accelerometer-based biometric gait recognition. In *3rd International Workshop on Security and Communication Networks (IWSCN 2011)*, 2011.

[27] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: password inference using accelerometers on smartphones. In *HotMobile'12*, pages 9:1–9:6, 2012.

[28] P. Phillips, J. Beveridge, B. Draper, G. Givens, A. O'Toole, D. Bolme, J. Dunlop, Y. M. Lui, H. Sahibzada, and S. Weimer. An introduction to the good, the bad, & the ugly face recognition challenge problem. In *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 346 –353, march 2011.

[29] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. Biometric-rich gestures: a novel approach to authentication on multi-touch devices. In *ACM CHI'12*, pages 977–986, 2012.

[30] R. C. Sprinthall. *Basic Statistial Analysis*. Prentice Hall, 2011.

[31] P. C. Van Oorschot, A. Salehi-Abari, and J. Thorpe. Purely automated attacks on passpoints-style graphical passwords. *Trans. Info. For. Sec.*, 5(3):393–405, Sept. 2010.

[32] Z. Xu, K. Bai, and S. Zhu. Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In *WISEC'12*, pages 113–124, 2012.

[33] S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq. Keystroke-based user identification on smart phones. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID'09, pages 224–243, 2009.

[34] N. Zheng, A. Paloski, and H. Wang. An efficient user verification system via mouse movements. In *ACM CCS'11*, pages 139–150, 2011.