Your Song Your Way: Rhythm-Based Two-Factor Authentication for Multi-Touch Mobile Devices

Yimin Chen*, Jingchao Sun*, Rui Zhang†, and Yanchao Zhang*
*School of Electrical, Computer and Energy Engineering (ECEE), Arizona State University
†Department of Electrical Engineering, University of Hawaii
{ymchen, jcsun, yczhang}@asu.edu, ruizhang@hawaii.edu

***Abstract*—Multi-touch mobile devices have penetrated into everyday life to support personal and business communications. Secure and usable authentication techniques are indispensable for preventing illegitimate access to mobile devices. This paper presents RhyAuth, a novel two-factor rhythm-based authentication scheme for multi-touch mobile devices. RhyAuth requires a user to perform a sequence of rhythmic taps/slides on a device screen to unlock the device. The user is authenticated and admitted only when the features extracted from her rhythmic taps/slides match those stored on the device. RhyAuth is a two-factor authentication scheme that depends on a user-chosen rhythm and also the behavioral metrics for inputting the rhythm. Through a 32-user experiment on Android devices, we show that RhyAuth is highly secure against various attacks and also very usable for both sighted and visually impaired people.**

I. Introduction

Mobile devices such as smartphones, tablets, and eReaders have penetrated into everyday life. According to a recent Cisco report [1], the number of mobile-connected devices would exceed the world population in 2014 and hit 10 billion in 2018. More and more mobile devices have a multi-touch screen that can simultaneously detect more than one point of contact. People are using mobile devices in every aspect of life, including voice/video communications, Internet browsing, web transactions, online banking, business operations, route planning and navigation, personal health and wellbeing, *etc*.

There is urgent need for mobile authentication techniques to prevent illegitimate access to mobile devices. On the one hand, people are storing increasingly more private information on multi-touch mobile devices. On the other hand, many users do not or often forget to log out of personal accounts such as web accounts, email accounts, and various on-device application accounts. Therefore, illegitimate access to a mobile device may seriously jeopardize the legitimate user's information and communication security. Mobile authentication techniques allow the legitimate user to unlock a mobile device and also deny illegitimate access. This is commonly accomplished by letting a user input a password only the legitimate user knows.

Sound mobile authentication techniques for multi-touch mobile devices should be both secure and usable. The security requirement demands strong resilience to notably three attacks. The first is the random-guessing attack in which an attacker tries to guess or emulate the password the legitimate user uses to unlock a mobile device; the second is the shoulder-surfing attack in which malicious bystanders try to observe the password of the legitimate user [2]; and the last is the smudge attack in which an attacker tries to infer the password based on the finger smudges the legitimate user left on the screen [3]. In contrast, the usability requirement has two implications. First, the authentication technique should be very easy to use by the legitimate user. Second, it should be highly accessible to visually impaired people with visual impairment. The second aspect is often neglected in the literature, despite that there are 285 million people worldwhile [4] and 21.5 million US adults aged 18 and older with visual impairment [5].

Existing authentication techniques for multi-touch mobile devices can be broadly classified into three categories.

Something-You-Know. This category of techniques require a user to input the correct password on the device screen to be admitted. The legitimate user presets the correct password, which can be an alphanumeric password or a gesture/picture password used in Android, iOS, and Windows 8. This category of techniques have some well-known drawbacks. Firstly, such techniques are quite vulnerable to shoulder-surfing attacks in public places. Secondly, these techniques require users to input at specific positions on a touch screen. This requirement may be a great frustration for people with fat fingers, and it may also open the door to smudge attacks. Finally, these techniques are not accessible to people with visual impairment.

Something-You-Have. This category of techniques require auxiliary hardware device only the legitimate user should possess. Examples include tMagkey/Mickey [6] and signet rings [7]. Although resilient to shoulder-surfing attacks, these techniques require additional hardware components to be specifically built. Also, such techniques authenticate a hardware component rather than a user to a mobile device.

Someone-You-Are. This category of techniques require physiological or behavioral biometrics of mobile users. Physiological biometrics relates to a person's physical features such as fingerprints, which are susceptible to well-known spoofing mechanisms. For example, the fingerprint-based Touch ID security system has been broken shortly after iPhone 5S was launched [8]. In contrast, behavioral biometrics relates to a user's behavioral patterns such as location traces [9], [10], gaits [11], [12], and touch dynamics [13]–[15]. These techniques are best suitable as secondary authentication mechanisms supplementing the primary password-based authentication mechanism, as they may be vulnerable to the adversary (*e.g.*, a close friend) familiar with the target's behavioral patterns.

This paper explores a new direction to authenticate a mobile user based on her rhythmic taps/slides on the device screen. The strong promise of this direction is firmly rooted in some observations in daily life. First, many people tend to

tap/slide on something nearby with a rhythm while singing a melody loudly or silently. Second, a user can easily repeat her rhythmic taps/slides over time for a familiar melody. Finally, different people are very likely to have different personal interpretations about the same melody and thus tap/slide in different ways; a user can even compose her own melody in mind instead of picking up a known melody. Therefore, rhythmic taps/slides are very difficult to emulate by an attacker with or without knowledge of the legitimate user's melody.

Our contributions in this paper are threefold.

- We propose RhyAuth, a novel two-factor rhythm-based authentication scheme for multi-touch mobile devices. RhyAuth requires a user to perform a sequence of rhythmic taps/slides on the device screen. The user is authenticated and admitted only when the features extracted from her rhythmic taps/slides match those stored on the device. RhyAuth is a two-factor authentication scheme because it requires both the correct rhythm (something-you-know) and the right way of performing the rhythm (someone-you-are).
- We theoretically analyze the security of RhyAuth. We show that RhyAuth is much more secure than the commonly used 4-digit PIN method, complex alphanumeric passwords, and Android Pattern Lock.
- We report comprehensive experimental evaluations of RhyAuth on Google Nexus 7 tablets, involving 22 legitimate users and 10 attackers. Our results show that RhyAuth is highly secure with false-positive and false-negative rates up to 0.7% and 4.2%, respectively. RhyAuth is also very efficient and can authenticate a user in less than 500 ms.

RhyAuth has many desirable features over existing techniques. Firstly, RhyAuth is highly resilient to brute-forth guessing attacks due to its two-factor nature. Secondly, RhyAuth is robust to shoulder-surfing attacks because it is very difficult for the attacker to figure out the exact rhythm by pure observations. Thirdly, RhyAuth is immune to smudge attacks, as the user can tap/slide on anywhere on the touch screen such that finger smudges can be more randomly distributed. Lastly, RhyAuth does not require the user to look at the screen while performing rhythmic taps/slides. The last feature indicates the high usability of RhyAuth to visually impaired people. It also means that a discrete user can conduct authentication with her device put under some cover (*e.g.*, a jacket or table) to eliminate shoulder-surfing attacks.

II. Basics of Multi-Touch Screens

We introduce some background on multi-touch screens to help illustrate the RhyAuth design. Since we implement RhyAuth as an application on Google Nexus 7 tablets powered by Android 4.2, our illustrations here focus on Android and are applicable to iOS with small modifications. A multi-touch screen can recognize two or more simultaneous contacts with the screen. When the screen is touched, a *touch event* is generated. The individual fingers or other objects, *e.g.*, a pen, that generate such events are referred to as *pointers*. Hereafter we assume that touch events are generated by fingers for simplicity.

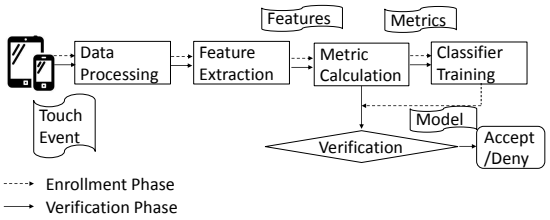


Fig. 1: A system overview of RhyAuth, in which the dash and solid arrows represent the data flows in enrollment and verification phases, respectively.

RhyAuth collects the information about a touch event as a vector $\mathsf{info} = [t, \mathsf{fID}, x, y, P, S]$, where t is the time of the event, fID is the ID of the finger, x and y are the x and y coordinates of the touch point on the screen, respectively, P is the pressure generated by the finger, and S is the size of the touch point on the screen. We refer to (x, y) as a *sampled point*, or more simply, a *point*. Both P and S are normalized values in the range of $[0, 1]$. One thing worth mentioning is that a finger may generate a series of touch events even though a user believes that she does not move her finger. The reason is that a touch event will be generated whenever there is a slight change in any of x, y, P, and S, which may be imperceptible.

III. System Overview of RhyAuth

In this section, we give an overview of the RhyAuth design. RhyAuth consists of two subsystems, TapAuth and SlideAuth, in which a rhythm is input via finger taps and slides on the screen, respectively. A user needs to choose one to proceed when RhyAuth is invoked. Whichever subsystem is chosen, the whole authentication process comprises two phases: an enrollment phase and a verification phase.

A. Enrollment Phase

During this phase, a user first needs to choose a melody, of which the rhythm becomes her password. A good melody should be sufficiently familiar to the user so that she has little difficulty in repeating her password. It should also be sufficiently random and thus cannot be easily figured out by an attacker. We will come back to this issue when analyzing the security of RhyAuth in Section V.

Fig. 2 shows an excerpt of "Amazing Grace," which we use to introduce some relevant musical terms. A *note* is a sign used in musical notation to represent the relative duration and *pitch* of a sound. A pitch is an auditory sensation in which a listener assigns musical tones to relative positions on a musical scale. Usually, we denote a pitch by one of the seven letters of the Latin alphabet, *i.e.*, A, B, C, D, E, F, and G. So a note is a pitch with a defined duration. For example, in Fig. 2, the first note in the first *measure* is a *quarter note* with a duration of $\frac{1}{4}$ and a pitch of C4, while the first note in the second measure is a *half note* with a duration of $\frac{1}{2}$ and a pitch of F4. In this paper, we are interested in the number of *"extended notes"* of a melody. An extended note refers to one note or multiple continuous notes of the same pitch connected by *ties*. A tie is a curved line connecting the heads of two notes of the same pitch and name, indicating that they are to be played as a single note with a duration equal to the sum of the individual durations. In practice, an extended note probably corresponds

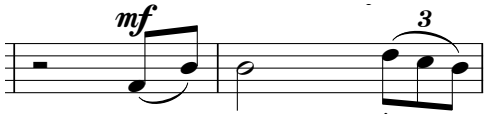


Fig. 2: An excerpt of "Amazing Grace" [16].

to one *tap* or *sub-slide* of RhyAuth, which will be explained shortly. For simplicity, we abbreviate "extened note" to " ex-note." Obviously, the number of ex-notes cannot be too small; otherwise, an attacker may figure out the rhythm easily. We assume that the melody has at least six ex-notes, and the user is asked about the number of ex-notes at the beginning of the enrollment phase.

Assume that the user chooses TapAuth. She continues to decide which finger(s) to tap on the screen. TapAuth gives the user full freedom to decide how she taps. A green user can choose and stick to one finger, while an advanced user may use multiple fingers and also switch fingers during her input. The user has to remember how many fingers she uses for each tap and inform RhyAuth about it. For example, if a melody has eight ex-notes, an advanced user may input the first four ex-notes with her middle finger, and the other ex-notes with her index and ring fingers together. Afterwards, the user needs to input the rhythm in the exact way. To avoid confusion, we refer to the tapping of one finger as a *touch*. A user may input an ex-note with multiple fingers, in which we refer to all the touches of an ex-note as a *tap*. Therefore, a tap can have one touch or multiple touches, and the number of taps is the same as the number of ex-notes.

Then a user inputs the rhythm by tapping on the screen according to her interpretation of her chosen melody. She needs to input the rhythm multiple times until a sufficiently good classifier can be obtained. As illustrated in Fig. 1, the touch-event data are first sent into a Data Processing module, which prepares the data for a Feature Extraction module. Then multiple distinguishable features are extracted and fed into a Metric Calculation module. Subsequently, a metric vector is generated and sent to a Classifier Training module. Finally, a binary classifier is generated for the verification phase to determine whether a new input is legitimate or not.

SlideAuth follows the same system architecture and only differs in some implementation details. Firstly, the rhythm in SlideAuth is input via continuous finger sliding on the screen. Therefore, the user does not switch the finger(s) while she is inputting the rhythm. Secondly, we need to divide a continuous slide into multiple sub-slides, each corresponding to an ex-note. The end of each ex-note is marked by an abrupt change of the sliding direction. Finally, some features of SlideAuth are different from those of TapAuth.

B. Verification Phase

In this phase, the user first chooses between TapAuth and SlideAuth, and then the user inputs her rhythm by tapping or sliding. The input goes through the same Data Processing, Feature Extraction, and Metric Calculation modules in sequence. The resulting metric vector is finally fed into the Verification module, where the established classifier is applied to determine whether the user is legitimate or not.

IV. Illustration of RhyAuth Modules

In this section, we detail each module of RhyAuth.

A. Data Processing

This module checks the consistency of the user input and prepares data for feature extraction. The steps below apply to each finger involved in either TapAuth or SlideAuth.

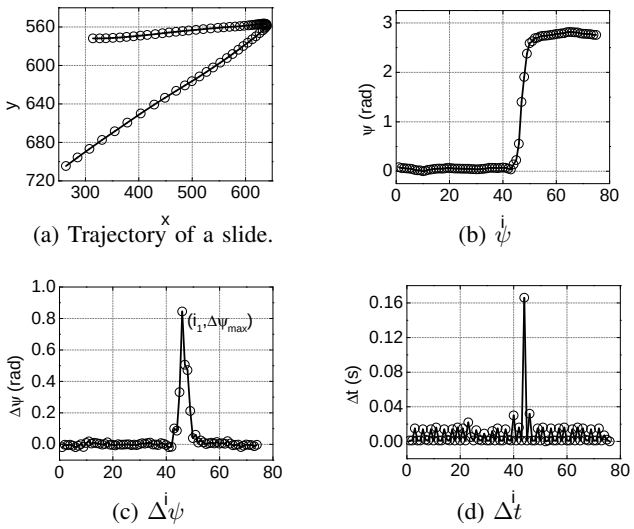
1) Data Processing for TapAuth: Firstly, as mentioned in Section II, a touch of a finger on the screen generates multiple info vectors of format $[t, \mathsf{fID}, x, y, P, S]$. These info vectors have the same fID and are slightly different in other fields. Let $\bar{x}$, $\bar{y}$, $\bar{P}$, and $\bar{S}$ denote the average x, y, P, and S values, respectively. To reduce the data redundancy, we merge these info vectors into a single one with the same fID, $\bar{x}$, $\bar{y}$, $\bar{P}$, $\bar{S}$, and all the t values remain intact.

Secondly, the number of taps and the number of fingers in each tap are extracted. If these numbers are not consistent with the user's setting in the enrollment phase, the user input is immediately considered invalid and not further processed.

2) Data Processing for SlideAuth: Firstly, we need to adjust the orientation of the slide to ensure that the device orientation and the starting direction of the slide have little effect on the authentication result. The orientation adjustment allows the user to input the rhythm more freely. We denote the coordinates of the slide as $\{(x_i, y_i)\}_{i=1}^{l}$, where l denotes the number of points of the slide. The slide orientation is adjusted such that the starting direction is aligned with the x axis, which is defined with the screen in the portrait mode. This is achieved in three steps. We first move the whole slide to make the coordinate of the first point $(0, 0)$ and change the coordinates to be $\{x'_i, y'_i\}_{i=1}^{l}$, where $x'_i = x_i - x_1, y'_i = y_i - y_1$. Then we calculate the angles of $(\eta_1 - 1)$ vectors which start from $(0, 0)$ and end at $\{(x'_i, y'_i)\}_{i=2}^{\eta_1}$, respectively, denoted by $\{\theta_i\}_{i=2}^{\eta_1}$. The starting direction of the slide is denoted by θ_s and defined as the average of $\{\theta_i\}_{i=2}^{\eta_1}$. Finally, the coordinates of a slide are transformed to $\{x''_i, y''_i\}_{i=1}^{l}$, where $x''_i = x'_i \cos\theta_s + y'_i \sin\theta_s$, $y''_i = -x'_i \sin\theta_s + y'_i \cos\theta_s$. Here η_1 is an empirical parameter, and it should be chosen such that the resulting θ_s is a good estimation of the direction of the first sub-slide. Note that η_1 cannot be too small, *e.g.*, two or three, to avoid instability. We use $\eta_1 = 5$ in our implementation.

Secondly, we smooth the trajectory of the slide because the collected data usually exhibit a jagged trajectory. We use a 10-point simple moving average (SMA) [17] filter for this purpose. After filtering, the coordinates become $\{\overline{x}_i, \overline{y}_i\}_{i=1}^{l}$.

Thirdly, we divide the whole smoothed slide into multiple sub-slides, each corresponding to an ex-note. This is equivalent to locating the last point of each sub-slide. Consider Fig. 3a as an example, where the slide consists of two sub-slides. A sharp change in the sliding direction indicates the end of the current ex-note or the beginning of the next ex-note. Given $\{\overline{x}_i, \overline{y}_i\}_{i=1}^{l}$, we first calculate the angles of the vectors connecting two consecutive points, *i.e.*, $\psi_i = \arccos(\frac{\overline{x}_{i+1} - \overline{x}_i}{\sqrt{(\overline{x}_{i+1} - \overline{x}_i)^2 + (\overline{y}_{i+1} - \overline{y}_i)^2}})$, $i = 1, \ldots, l-1$. If a sequence of vectors are associated with the same sub-slide, the corresponding ψs should be similar. Fig. 3b is the corresponding plot of ψ. We can see that ψ switches from one stable value to another through a sharp



(a) Trajectory of a slide. (b) $\overset{\mathsf{i}}{\psi}$ (c) $\Delta\overset{\mathsf{i}}{\psi}$ (d) $\Delta\overset{\mathsf{i}}{t}$

Fig. 3: An example on dividing a slide into two sub-slides.

transition phase, indicating a noticeable change of the sliding direction. To locate the last point of the first sub-slide, we further calculate $\Delta\psi_i = \psi_{i+1} - \psi_i, i = 1, \ldots, l-2$. Fig. 3c is the plot of $\Delta\psi$ in our example. Denote the index of the point with the largest $\Delta\psi$ by i_1. Then $\{i_1 - \eta_2, \ldots, i_1, \ldots, i_1 + \eta_2\}$ are indices of the points and include the last point of the first sub-slide. Here η_2 is an empirical parameter, and $\eta_2 = 10$ is adopted in our implementation. We proceed to calculate the time difference Δt of two consecutive points as $\Delta t_i = t_{i+1} - t_i$ as shown in Fig. 3d. The largest Δt corresponds to the last point. The above process can be easily extended to a slide with two or more sub-slides, in which case we just need to look for the last points of multiple sub-slides.

Finally, the number of fingers and the number of sub-slides are compared with the user's setting in the enrollment phase. If these numbers are not consistent, the sliding input is deemed invalid and not further processed.

Some parameters need to be adjusted to properly divide a slide in practice. First, we need to decide when a change of ψ (or sliding direction) occurs. A straightforward solution is that when $\Delta\psi$ is larger than some threshold φ, a change of ψ occurs. In general, the suitable φ for different users is different, and we can train it during the enrollment phase. More specifically, we let $\varphi = \frac{\pi}{2}$ at the beginning of the enrollment phase, and the number of sub-slides set by the user is n. After slide division as described above, the number of sub-slides, denoted by n', is the number of changes of ψ plus one. If $n = n'$, the training of φ ends, and the current φ is used during the verification phase. If $n > n'$, φ is decreased by $\frac{\pi}{32}$ and increased by $\frac{\pi}{32}$ otherwise until $n = n'$.

B. Feature Extraction

This module takes the processed data from the Data Processing module as input and extracts the features used in RhyAuth. Depending on which of TapAuth and SlideAuth is chosen, the corresponding features are different. Most features are extracted from the data of each individual fID, while others are extracted by combining the data of multiple fIDs. The following descriptions apply to each fID involved in either TapAuth or SlideAuth.

1) Features of TapAuth: Below, n denotes the number of ex-notes (or equivalently taps), each corresponding to an integrated info vector formed in Data Processing.

- Intra-tap and inter-tap intervals: We denote the intra-tap interval by $\{\Delta t_{1,i}\}_{i=1}^{n}$ and the inter-tap interval by $\{\Delta t_{2,i}\}_{i=1}^{n-1}$. Let $t_{f,i}$ and $t_{l,i}$ denote the time of the first and last touch events associated with the ith tap, respectively. Then we have $\Delta t_{1,i} = t_{l,i} - t_{f,i}$ and $\Delta t_{2,i} = t_{f,i+1} - t_{l,i}$.
- Maximum and minimum pressure: These two features are denoted by $P_{\max}$ and $P_{\min}$, respectively. Then $P_{\max} = \max\{\overline{P}_i\}_{i=1}^{n}$ and $P_{\min} = \min\{\overline{P}_i\}_{i=1}^{n}$.
- Maximum and minimum size: We denote maximum and minimum size by $S_{\max}$ and $S_{\min}$, respectively. We have $S_{\max} = \max\{\overline{S}_i\}_{i=1}^{n}$ and $S_{\min} = \min\{\overline{S}_i\}_{i=1}^{n}$.
- Maximum and minimum distance: We denote them by $D_{\max}$ and $D_{\min}$, respectively. Suppose that the ith tap is associated with m fIDs and thus m coordinates. The distance of the ith tap is defined as the average Euclidean distance of each pair of coordinates, and it equals zero if $m = 1$. Then $D_{\max}$ and $D_{\min}$ are the maximum and minimum of the n tap-distance values.
- Maximum and minimum areas: We denote them by $A_{\max}$ and $A_{\min}$, respectively. Suppose that the ith tap is associated with m fIDs and thus m coordinates. If $m \geq 3$, the area of the ith tap is defined as the area of the convex hull determined by the m coordinates; otherwise, it is defined as zero. $A_{\max}$ and $A_{\min}$ are the maximum and minimum of the n tap-area values.

2) Features of SlideAuth: Below, n denotes the number of ex-notes (or equivalently the number of sub-slides) and N denotes the number of touch events.

- Intra-slide and inter-slide intervals: We denote them by $\{\Delta t_{3,i}\}_{i=1}^{n}$ and $\{\Delta t_{4,i}\}_{i=1}^{n-1}$, which are defined similarly to those of TapAuth.
- Maximum and minimum pressure: They are defined as $P_{\max} = \max\{P_i\}_{i=1}^{N}$ and $P_{\min} = \min\{P_i\}_{i=1}^{N}$.
- Maximum and minimum sizes: They are define as $S_{\max} = \max\{S_i\}_{i=1}^{N}$ and $S_{\min} = \min\{S_i\}_{i=1}^{N}$.
- Maximum and minimum slide length: We denote them by $L_{\max}$ and $L_{\min}$, respectively. For each sub-slide, we define its slide length as the distance between the coordinates of the first and last associated touch events. Then $L_{\max}$ and $L_{\min}$ are the maximum and minimum of the n slide lengths, respectively.
- Slide direction: For the ith ($i \in [1, N-1]$) point of the slide, we denote its slide direction by the angle, θ_i, of the vector from the ith point to the $(i+1)$th point. Therefore, $\{\theta_i\}_{i=1}^{N-1}$ can be calculated as

$$\theta_i = \arccos\left(\frac{\overline{x}_{i+1} - \overline{x}_i}{\sqrt{(\overline{x}_{i+1} - \overline{x}_i)^2 + (\overline{y}_{i+1} - \overline{y}_i)^2}}\right).$$

- Curvature: It is denoted by $\{\kappa_i\}_{i=2}^{N-1}$ and computed as

$$\kappa_i = \frac{4\Psi_i^y \Delta_i^x - 4\Psi_i^x \Delta_i^y}{((\Delta_i^x)^2 + (\Delta_i^y)^2)^{3/2}},$$

where $\Delta_i^x = (\overline{x}_{i-1} + \overline{x}_{i+1})/2$, $\Delta_i^y = (\overline{y}_{i-1} + \overline{y}_{i+1})/2$, $\Psi_i^x = (\overline{x}_{i+1} - 2\overline{x}_i + \overline{x}_{i-1})$, and $\Psi_i^y = (\overline{y}_{i+1} - 2\overline{y}_i + \overline{y}_{i-1})$.

- Velocities along the x axis and y axis: We denote them by $\{v_x\}_{i=1}^{N-1}$ and $\{v_y\}_{i=1}^{N-1}$, respectively and compute them as

$$v_{x,i} = \frac{\overline{x}_{i+1} - \overline{x}_i}{t_{i+1} - t_i} \text{ and } v_{y,i} = \frac{\overline{y}_{i+1} - \overline{y}_i}{t_{i+1} - t_i}.$$

- Accelerations along the x axis and y axis: We denote them by $\{a_{x,i}\}_{i=1}^{N-2}$ and $\{a_{y,i}\}_{i=1}^{N-2}$, respectively, which are computed as

$$a_{x,i} = \frac{v_{x,i+1} - v_{x,i}}{t_{i+1} - t_i} \text{ and } a_{y,i} = \frac{v_{y,i+1} - v_{y,i}}{t_{i+1} - t_i}.$$

- Distance: We denote it by D. Suppose that the slide is associated with m fIDs. If $m \geq 2$, D is defined as the average pairwise Euclidean distance among the first points of all the trajectories. If $m = 1$, we let $D = 0$.

- Area: We denote it by A. Suppose that the slide is associated with m fIDs. If $m \geq 3$, A is defined as the area of the convex hull determined by the first points of their trajectories; otherwise, we let $A = 0$.

C. Metric Calculation

This module is to consolidate the output from the Feature Extraction module into a metric vector. For TapAuth, the extracted features include $\{\Delta t_{1,i}\}_{i=1}^{n}, \{\Delta t_{2,i}\}_{i=1}^{n-1}$, $P_{\max}, P_{\min}, S_{\max}, S_{\min}, D_{\max}, D_{\min}, A_{\max}$, and $A_{\min}$; for SlideAuth, the extracted features include $\{\Delta t_{3,i}\}_{i=1}^{n}$, $\{\Delta t_{4,i}\}_{i=1}^{n-1}, P_{\max}, P_{\min}, S_{\max}, S_{\min}, L_{\max}, L_{\min}, \{\theta_i\}_{i=1}^{N-1}$, $\{\kappa_i\}_{i=2}^{N-1}, \{v_{x,i}\}_{i=1}^{N-1}, \{v_{y,i}\}_{i=1}^{N-1}, \{a_{x,i}\}_{i=1}^{N-2}, \{a_{y,i}\}_{i=1}^{N-2}$, D, and A. Features like $\{\Delta t_{1,i}\}_{i=1}^{n}$ are in the vector form, and we would like to use a real number to denote each such feature for integration with non-vector features. This can be done by computing the distance between any vector feature and a reference vector. The comparison result (or the vector distance) is the real number we seek.

The vector features can also be divided into two categories, which require different comparison methods. Specifically, each feature vector of TapAuth is of length n, while that of SlideAuth is of length n or N. Both TapAuth and SlideAuth require two matching inputs to have the same number of exnotes. This consistency check is done in the Data Processing module. Therefore, we can use statistical models to compare such feature vectors of the same length. In contrast, different inputs in SlideAuth most likely generate different numbers of touch events, leading to feature vectors of different lengths. We adopt Dynamic Time Warping (DTW) [18] to compare such feature vectors of variable lengths.

1) Comparison Based on Statistical Model: We take $\{\Delta t_{1,i}\}_{i=1}^{n}$ as an example to explain how to calculate the distance based on the statistical model. Suppose that there are Q samples from one user with the same n. We treat each element of $\{\Delta t_{1,i}\}_{i=1}^{n}$ as a Gaussian random variable. Given Q samples of $\{\Delta t_{1,i}\}_{i=1}^{n}$, we can calculate the mean and variance of each element. That is, we will have $\{(\mu_1, \sigma_1^2), \ldots, (\mu_n, \sigma_n^2)\}$ as the statistical model for $\{\Delta t_{1,i}\}_{i=1}^{n}$. Given a new sample of $\{\Delta t_{1,i}\}_{i=1}^{n}$, it is converted into $d_{\Delta t_1}$ as

$$d_{\Delta t_1} = \left(\sum_{i=1}^{n} \frac{(\Delta t_{1,i} - \mu_i)^2}{\sigma_i^2} \right)^{\frac{1}{2}}. \quad (1)$$

The intuition here is that a new sample of the same user most probably well follows the statistical model built from her historical data well, leading to a small $d_{\Delta t_1}$. However, a sample from a different user is very likely to deviate much from this statistical model, resulting in a large $d_{\Delta t_1}$. In the same way, $\{\Delta t_{2,i}\}_{i=1}^{n-1}, \{\Delta t_{3,i}\}_{i=1}^{n}$, and $\{\Delta t_{4,i}\}_{i=1}^{n-1}$ are converted as $d_{\Delta t_2}, d_{\Delta t_3}$, and $d_{\Delta t_4}$, respectively.

2) Comparison Based on DTW: We use $\{\theta_i\}_{i=1}^{N-1}$ to explain how to calculate the distance based on DTW. Suppose that there are Q samples from one user. Assuming that the Q samples are quite similar, we randomly choose one as a reference and denote it by $\Theta^* = \{\theta_i^*\}_{i=1}^{N^*}$. DTW constructs a $(N-1) \times N^*$ matrix M with its (i,j) element $M(i, j) = |\theta_i - \theta_j^*|, i = 1, \ldots, N-1, j = 1, \ldots, N^*$. Then DTW looks for a non-decreasing path starting from $M(1, 1)$ to $M(N-1, N^*)$ along which the sum of all elements would be the minimum of all possible paths. This minimum sum is used as the transformed value of $\{\theta_i\}_{i=1}^{N-1}$ and denoted by d_θ. Similarly, $\{\kappa_i\}_{i=2}^{N-1}, \{v_{x,i}\}_{i=1}^{N-1}, \{v_{y,i}\}_{i=1}^{N-1}, \{a_{x,i}\}_{i=1}^{N-2}$, and $\{a_{y,i}\}_{i=1}^{N-2}$ are transformed into $d_\kappa, d_{v_x}, d_{v_y}, d_{a_x}$, and d_{a_y}, respectively.

D. Classifier Training

This module is to train a binary classifier from the metric vectors of the legitimate user and other users.

We use SVM as the classification algorithm and LibSVM [19] in our implementation, which has been widely used and proved to achieve satisfactory performance under various circumstances. The classifier we need is a binary classifier, which classifies a sample (or metric vector) into the positive class or negative class. We use f_i to denote the class label of the ith sample. If $f_i = 1$, the sample is classified into the positive class, meaning that the sample is legitimate. If $f_i = -1$, the sample is classified into the negative class, indicating that the sample is illegitimate. In order to train the classifier, we need a training dataset consisting of metric vectors of both the legitimate user and other users. For this purpose, a library of metric vectors of other users can be preloaded with each RhyAuth; it can also be downloaded in real time from a trusted server. Now suppose that we have n_s metric vectors or samples in total. Each of them is expanded into a sample-label pair (u_i, f_i), where u_i denotes the ith sample. $f_i = 1$ if u_i is a metric vector of the legitimate user and $f_i = -1$ otherwise. Given $\{(u_i, f_i)\}_{i=1}^{n_s}$, SVM solves the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{n_s} \xi_i \\ \text{subject to} \quad & f_i \cdot \left(w^T \phi(u_i) + b \right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned} \quad (2)$$

Here u_is are mapped into a higher dimensional space by the function ϕ , and $K(u_i, u_j) \equiv \phi(u_i)^T \phi(u_j)$ is called the kernel

function. SVM finds a linear separating hyperplane $w^T v + b = 0$ with the maximal margin in this higher dimensional space. Here v is a vector in the higher dimensional space, and $C > 0$ is the penalty parameter of the error term ξ_i. In our implementation, we choose the radial basis function (RBF) as the kernel function which has been proved to be a reasonable first choice. More specifically, the kernel function we choose is $K(u_i, u_j) = \exp(-\gamma \|u_i - u_j\|^2)$, where $\gamma > 0$ is the kernel parameter. The result of classifier training is a SVM model for the legitimate user, which predicts the class label of a new metric vector or sample.

E. Verification

A candidate user input goes through the same Data Processing, Feature Extraction, and Metric Calculation modules until a metric vector u is generated in either TapAuth or SlideAuth. The Verification module first verifies whether the user input has the same numbers of ex-notes and fingers as those of the legitimate user. If not, the user fails the authentication, and the verification stops. Otherwise, the Verification module tests the candidate metric vector using the SVM model of the legitimate user. The SVM model consists of the optimal w and b, which are obtained by solving the optimization problem in Eq. 2. Given a candidate vector u, the decision function is $\mathsf{sgn}(w\phi(u) + b)$. If the result is 1, the user is considered legitimate and illegitimate otherwise.

V. Security Analysis

In this section, we analyze the security of RhyAuth. Unlike conventional authentication schemes involving alphanumerical or patter passwords, we cannot answer the question: "What is the size of the password space?" The reason is that RhyAuth combines a user-chosen rhythm and the user's behavioral biometrics together. In [20], Sherman *et al.* studied the security and memorability of user-generated free-form gestures for authentication. The metric they used is to quantify the "*surprisingness*" of a given gesture, rather than the security of their authentication scheme. Similarly, we focus on the security of a rhythm, which can be regarded as a lower bound of the overall security assessment of RhyAuth.

First, we want to answer the question: "Given a melody of n ex-notes, how many rhythms can there be?" Here we assume that a melody chosen by a user follows the music convention. Specifically, each ex-note consists of multiple notes; the duration of a note is one of the 12 note values, from 8 (*i.e.*, 2^3) corresponding to a maxima to $\frac{1}{256}$ (*i.e.*, 2^{-8}) corresponding to a two-hundred-fifty-sixth note; and the duration of a note with zero dot can be further augmented by adding one dot, two dots, and three dots. Therefore, a note may have $12 \times 4 = 48$ possible duration values. Although there is no limit on how many notes an ex-note can consist of, we assume that an ex-note lasts no more than two measures, each comprising no more than 24 notes for usability concerns. Therefore, an ex-note can consist of up to 48 notes. It is worth noting that the number of possible duration values of an ex-note is not $48 \times 48 = 2304$, as some of the 2304 values are the same. For example, the duration of two notes with a note value of $\frac{1}{2}$ is the same as that of a single note with a note value of 1. So we further refine the 2304 values to eliminate redundant ones. Finally, we obtain 1002 unique possible duration values of an ex-note. Then for a melody of n ex-notes, the number of possible rhythms is simply $1002^n \approx 2^{10n}$. We should point out that this is an underestimation due to the assumption on how many notes an ex-note consists of. Here we give some numerical examples to briefly compare RhyAuth with the following schemes: (1) 4-digit PIN simple password of iOS, $10^4 \approx 2^{13}$; (2) n-character complex password of iOS, $77^n \approx 2^{6.27n}$; (3) Android Pattern Lock, around 2^{19} [21]. RhyAuth is obviously much more secure than all of them.

Secondly, we would like to answer the question: "Given an input rhythm, can the system suggest whether it is a good choice or not?" A firm answer to this question can help a user to choose a rhythm of high security strength. Inspired by [20], we use the surprisingness of a rhythm as a measure of its security strength. Specifically, we denote a rhythm of n ex-notes by $\{T_i\}_{i=1}^n$, where T_i is the duration of the ith ex-note and calculated as

$$T_i = \begin{cases} \Delta t_{1,i} + \Delta t_{2,i} & \text{if i < n,} \quad (3a) \\ \Delta t_{1,i} & \text{if i = n.} \quad (3b) \end{cases}$$

We assume $\{T_i\}_{i=1}^n$ follows a second-order autoregressive model as $T_i = \beta_0 + \beta_1 T_{i-1} + \beta_2 T_{i-2} + \epsilon_i$, where β_0, β_1, and β_2 are parameters to optimize by least squares fitting, and ϵ_i is the error term of T_i. Suppose that the least squares estimates are $\hat{\beta}_0, \hat{\beta}_1$, and $\hat{\beta}_2$ after parameter fitting. We then use $h(T)$ to denote the surprisingness of $\{T_i\}_{i=1}^n$, calculated as $h(T) = \left(\sum_{i=1}^n (T_i - \hat{T}_i)^2\right)^{\frac{1}{2}}$, where $\hat{T}_i = \hat{\beta}_0 + \hat{\beta}_1 \hat{T}_{i-1} + \hat{\beta}_2 \hat{T}_{i-2}$.

VI. Performance Evaluation

In this section, we report the performance evaluation of RhyAuth, which we implemented as an application on Google Nexus 7 tablets running Android 4.2. In the rest of this section, we describe the attacker models, experimental setup, performance metrics, and experimental results in sequence.

A. Attacker Models

In this paper, we consider the following models with increasingly capable attackers.

Type-I. The attacker knows neither the rhythm a user chooses nor how a user inputs the rhythm on the screen. Then the attacker's best effort is a brute-force attack.

Type-II. The attacker can observe how a user taps or slides on the screen multiple times, but he cannot figure out the exact rhythm the user chooses. The attacker can at best obtain a general idea of the rhythm through observations. For example, he may notice that the user taps eight times on the screen.

Type-III. The attacker knows the exact rhythm a user chooses and can also observe how the user taps or slides on the screen. Under this model, the attacker can input on the screen according to his own interpretation of the rhythm and his observations of the user.

Type-IV. An attacker knows exactly how a user taps or slides on the screen and the rhythm the user chooses. He, however, still needs to input the rhythm according to his own perception.

B. Experimental Setup

We recruited 32 volunteers for the experiments aged 18 to 35. Most of them are/were BS/MS students in Computer Science, Electrical Engineering, and Computer Engineering. These volunteers were divided into two groups. The first group consisted of 22 volunteers to emulate legitimate users, and the second comprised 10 volunteers to emulate various attackers.

Every user was asked to come up with one melody with at least six ex-notes s/he can easily memorize and repeat. Also, every user was asked to input her/his rhythm with one-finger tapping, multi-finger tapping, one-finger sliding, and multi-finger sliding. Then every user practiced less than five minutes until s/he was confident to input them. Finally, every user input her/his rhythm using each method for 25 to 45 times, and we obtained 888 one-finger tapping samples, 870 multi-finger tapping samples, 894 single-finger sliding samples, and 873 multi-finger sliding samples. In reality, a RhyAuth user only needs to tap or slide a few times during the enrollment phase, as shown later. In addition, all the users were asked not to look at the screen to emulate people with visual impairment.

We also conducted experiments to evaluate the resilience of RhyAuth to various attacks. First, we video-recorded the input process of 14 users, and we also made the sound tracks of their chosen rhythms. We divided the 10 attackers into two groups of equal size. Each attacker in the first group randomly chose three users, watched their one-finger-tapping videos, and mimicked them. Then s/he randomly chose another three users, watched their multi-finger-tapping videos, and mimicked them. In total, there were six videos for each attacker in the first group. Each attacker in the second group did the similar experiments after watching one-finger and multi-finger sliding videos only. In accordance with our attacker models, we simulated the following five attack scenarios and collected $5 \times 6 \times 5 \times 2 = 300$ attacker samples for each attack.

1. One-time observation. The attacker was shown the video once. Then the attacker decided how to mimic the user, practiced, and input the rhythm five times.

2. Four-time observations. For each video he watched in Scenario 1, the attacker watched it for three more times and made five more attempts.

3. Four-time observations and one-time listening. After watching the video for four times, the attacker was allowed to listen to the sound track of the corresponding rhythm. Then the attacker combined his observations of the video and his perception of the rhythm together to make five more attempts.

4. Arbitrary observations and listenings. The attacker was allowed to watch each video and listen to the corresponding sound track for arbitrary times. Again, he could control how to watch the video and listen to the sound track. Finally, he made five more attempts when he was ready.

5. Arbitrary observations and listenings as well as how the user inputs her/his rhythm. The attacker could watch each video and listen to the corresponding sound track for arbitrary time in his own way. Furthermore, we asked each user to write down how s/he input her/his rhythm, including how many taps in her/his rhythm and whether the time between two consecutive taps was short or long for one-finger tapping, which fingers s/he used for each tap for multi-finger tapping, how many sub-slides s/he drew and whether the time s/he used to draw each sub-slide was short or long for one-finger sliding, and which fingers s/he used to slide for multi-finger sliding. Finally, the attacker combined all these information and mimicked the user for five more times.

Attacks 1 and 2 correspond to Type-II attackers, Attacks 3 and 4 correspond to Type-III attackers, and Attack 5 corresponds to Type-IV attackers.

C. Performance Metrics

We use receiver operating characteristic (ROC) and `Precision-Recall` curves to evaluate RhyAuth.

ROC Curve. A ROC curve is used to illustrate the performance of a binary classifier as its discrimination threshold changes. We can plot a ROC curve by plotting true positive rate (`TPR`) with respect to false positive rate (`FPR`) at various threshold settings. Denote the number of true positives, false positives, true negatives, and false negatives by $\#\texttt{TP}, \#\texttt{FP}$, $\#\texttt{TN}$, and $\#\texttt{FN}$. Then `TPR` and `FPR` can be calculated as

$$\texttt{TPR} = \frac{\#\texttt{TP}}{\#\texttt{TP} + \#\texttt{FN}} \text{ and } \texttt{FPR} = \frac{\#\texttt{FP}}{\#\texttt{FP} + \#\texttt{TN}}. \quad (4)$$

`Precision-Recall` Curve. `Precision` represents the percentage of legitimate users out of all admitted users and can be calculated as

$$\texttt{Precision} = \frac{\#\texttt{TP}}{\#\texttt{TP} + \#\texttt{FP}}. \quad (5)$$

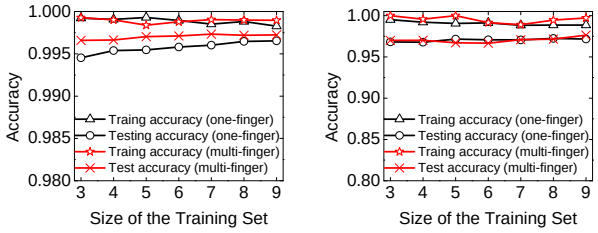
`Recall` in authentication systems is the same as `TPR`, which measures the proportion of legitimate users who are correctly identified as such.

Authentication Time. We also measure the time RhyAuth takes to determine whether a user is legitimate or not, which should be as short as possible.

D. Experimental Results

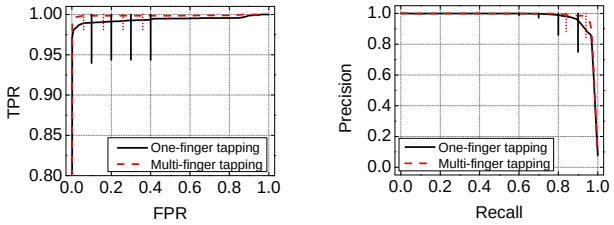
1) Performance Without Attackers: This section demonstrates the performance of RhyAuth without attackers. Recall that each user was required to input her/his rhythm by four methods. The evaluation for each method was done as follows. For each user, we randomly chose ω samples from all the legitimate users to form a training set of 22ω samples for classifier training. The remaining samples were treated as the testing set. We did this evaluation 30 times for each user, and the results are the average results over the 30 times.

We first report the impact of the size of the training set on classification accuracy which can further be divided into training accuracy and testing accuracy. Here we define accuracy as the ratio of correctly classified users among all users in a dataset. We changed the size of the training set by varying ω and showed the results in Fig. 4. We can see that when ω varies from 3 to 9, the training and testing accuracy of four input methods only vary slightly. A smaller ω means a legitimate user can input her/his rhythm fewer times in the enrollment phase. When ω is larger than 5, the training and testing accuracy of four input methods stay stable. Therefore, we chose $\omega = 5$ for the later evaluations.



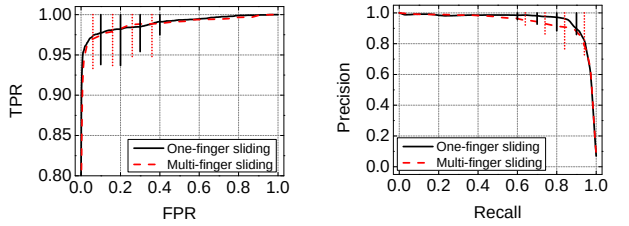
(a) Finger tapping. (b) Finger sliding.

Fig. 4: Impact of the size of the training set.



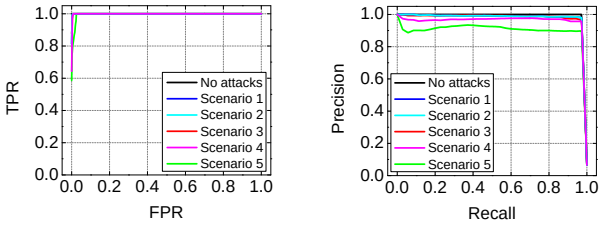
(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 5: Authentication results with finger tapping.



(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 6: Authentication results with finger sliding.



(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 7: Attack resilience with one-finger tapping.

Fig. 5 shows the authentication performance of one-finger and multi-finger tapping, including the average results and the upper and lower bounds in the `Precision-Recall` and ROC curves. The ROC curves of the two methods are close to the top-left corner, which indicates that RhyAuth can achieve high `TPR` with low `FPR`. The `Precision-Recall` curves are close to the top-right corner, meaning that our system can achieve high `Precision` and high `Recall` at the same time.

Similarly, Fig. 6 illustrates the authentication performance of one-finger and multi-finger sliding. We can see that the ROC curves are close to the top-left corner and the `Precision-Recall` curves are close to the top-right corner, indicating that RhyAuth performs well in distinguishing a legitimate user from others. In contrast to finger tapping, finger sliding has slightly worse performance. The reason is that tapping with a rhythm is more natural than sliding with a rhythm for most people. As a result, tapping inputs are more consistent than sliding inputs, leading to fewer classification errors and thus fewer false negatives.

2) Performance With Attackers: This section reports the resilience of RhyAuth to attacks. For each victim under attacks, we first trained the classifier and obtained the SVM model of the victim. Then we added the samples of the attackers into the testing set and evaluated the performance of RhyAuth.

Fig. 7 and Fig. 8 show the results for one-finger and multi-finger tapping, respectively. We can see that both are highly resilient to the attacks. In addition, their attack resilience both slightly decreases as the capability of the attackers increases. Compared with one-finger tapping, multi-finger tapping is more resilient to the attacks. The reason is that more features are extracted from multi-finger tapping inputs.

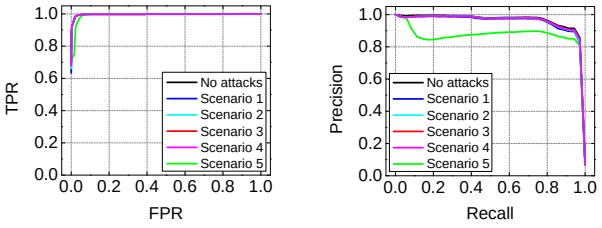
Fig. 9 and Fig. 10 show the results for one-finger sliding and multi-finger sliding, respectively. As we can see, both methods are also highly secure against the attacks, and their attacker resilience slightly decreases as well as the attacker becomes more capable. In addition, multi-finger sliding is more secure than one-finger sliding because more features are available in the former. Finally, finger sliding is more resilient to finger tapping, which is simply due to more features available in finger-sliding inputs.

3) Computation Time: We have also measured the computation time of RhyAuth. In particular, our measurements show that the one-time enrollment time of one-finger tapping and sliding are 14s and 200s, respectively, measured on a Dell desktop with 2.67 GHz CPU, 9 GB RAM, and Windows 7 64-bit Professional. Since the enrollment phase of RhyAuth takes relatively long time, we suggest that the classifier can be trained on a powerful desktop computer and then downloaded to the mobile device. This strategy has been advocated by many classifier-based mobile authentication schemes such as [15]. In addition, the verification time of one-finger tapping and sliding are 3.8ms and 500ms, respectively, measured on Google Nexus 7 tablet, which make them very practical. Finally, the computation time of multi-finger taping/sliding is simply that of one-finger taping/sliding multiplied by the number of fingers involved.

VII. Additional Related Work

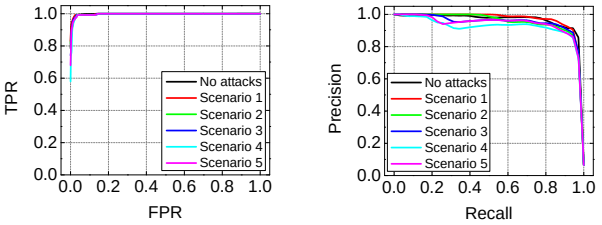
There are some rhythm-based authentication schemes. In [22], Wobbrock *et al.* used the tapping on a button as the input of a rhythm for user authentication. Their experimental results showed a relatively low successful acceptance rate of 83.2%. In addition, their scheme does not target multi-touch mobile devices. In [23], Marques *et al.* transformed the timing information of taps on a touch screen into a sequence and proposed a Hamming-distance-based matching approach for user authentication. However, the acceptance and rejection rates of their scheme are not reported. In [24], Lin *et al.* presented *RhythmLink*, a protocol to securely pair I/O-constrained devices by tapping.

Our work differs from the above schemes in many aspects. First, RhyAuth allows a user to input a rhythm by either tapping or sliding, while the above schemes only allow tapping. Second, RhyAuth additionally incorporates the behavioral biometrics of a user inputting the rhythm and thus can achieve much higher true acceptance and rejection rates. Finally, we conduct theoretical analysis of rhythm-based authentication for the first time in literature.



(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 8: Attack resilience with multi-finger tapping.



(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 9: Attack resilience with one-finger sliding.

Recent years have also seen many mobile authentication techniques based on behavioral biometrics [13], [15], [25], [26]. In [27], however, Abdul *et al.* suggested that a programmable Lego robot could emulate users' behavioral biometrics to some extent, which poses potential threats on such techniques. In contrast, RhyAuth combines an additional user-chosen secret rhythm with her behavioral biometrics, thus achieving stronger attack resilience.

Finally, TouchIn [28], [29] is a two-factor mobile authentication scheme that works by letting a user draw secret geometric curves on the device screen with one or multiple fingers. RhyAuth and TouchIn have similar authentication performance. But RhyAuth is more usable for people who have better memory for rhythms than for geometric curves.

VIII. Conclusion

In this paper, we presented the design and evaluation of RhyAuth, a novel rhythm-based two-factor authentication scheme for multi-touch mobile devices. Detailed security analysis and user experiments confirmed that RhyAuth is highly secure and usable for sighted and visually impaired people, thus has the great potential for wide adoption.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-1421999, CNS-1320906, CNS-1117462, and CNS-1422301. We would also like to thank anonymous reviewers for their constructive comments and helpful advice.

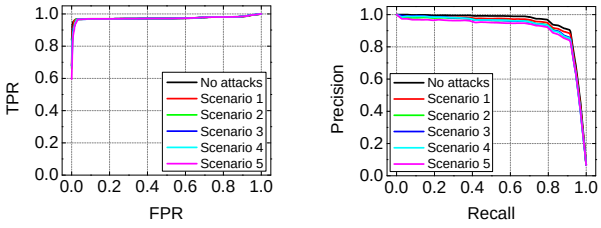
References

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[2] N. Sae-Bae, *et al.*, "Biometric-rich gestures: a novel approach to authentication on multi-touch devices," in *CHI'12*, Austin, TX, May 2012, pp. 977–986.

[3] A. Aviv, *et al.*, "Smudge attacks on smartphone touch screens," in *WOOT'10*, Washington, DC, Aug. 2010, pp. 1–7.

[4] http://www.who.int/blindness/en/.



(a) ROC curves. (b) `Precision-Recall` curves.

Fig. 10: Attack resilience with multi-finger sliding.

[5] http://www.afb.org/section.aspx?SectionID=15.

[6] H. Bojinov, *et al.*, "Mobile token-based authentication on a budget," in *HotMobile'11*, Phoenix, AZ, Apr. 2011, pp. 14–19.

[7] T. Vu, *et al.*, "Distinguishing users with capacitive touch communication," in *ACM MobiCom'12*, Istanbul, Turkey, Aug. 2012, pp. 197–208.

[8] http://gizmodo.com/hackers-iphone-5s-fingerprint-security-is-not-secure-1367817697.

[9] M. Jakobsson, *et al.*, "Implicit authentication for mobile devices," in *HotSec'09*, Montreal, Canada, Aug. 2009, pp. 9–9.

[10] E. Shi, *et al.*, "Implicit authentication through learning user behavior," in *ISC'10*, Boca Raton, FL, Oct. 2010, pp. 99–113.

[11] J. Mantyjarvi, *et al.*, "Identifying users of portable devices from gait pattern with accelerometers," in *ICASSP'05*, Philadelphia, PA, Mar. 2005, pp. 973–976.

[12] D. Gafurov, *et al.*, "Spoof attacks on gait authentication system," *IEEE TIFS*, vol. 2, no. 3, pp. 491–502, Sept. 2007.

[13] A. Luca, *et al.*, "Touch me once and i know it's you! implicit authentication based on touch screen patterns," in *CHI'12*, Austin, TX, May 2012, pp. 987–996.

[14] F. Sandnes, *et al.*, "User identification based on touch dynamics," in *UIC/ATC'12*, Fukuoka, Japan, Sept. 2012, pp. 256–263.

[15] L. Li, *et al.*, "Unobservable re-authentication for smartphones," in *NDSS'13*, San Diego, CA, Feb. 2013.

[16] http://www.malechoirmusic.co.uk/sheet_music/Amazing%20Grace.pdf.

[17] http://en.wikipedia.org/wiki/Simple_moving_average.

[18] http://en.wikipedia.org/wiki/Dynamic_time_warping.

[19] C.-C. Chang, *et al.*, "Libsvm: a library for support vector machines," *ACM TIST*, vol. 2, no. 3, p. 27, 2011.

[20] M. Sherman, *et al.*, "User-generated free-form gestures for authentication: security and memorability," in *MobiSys'14*, Bretton Woods, NH, Jun. 2014, pp. 176–189.

[21] S. Uellenbeck, *et al.*, "Quantifying the security of graphical passwords: the case of android unlock patterns," in *ACM CCS'13*, Berlin, Germany, Nov. 2013, pp. 161–172.

[22] J. O. Wobbrock, "Tapsongs: tapping rhythm-based passwords on a single binary sensor," in *ACM UIST'09*, Victoria, BC, Canada, Oct. 2009, pp. 93–96.

[23] D. Marques, *et al.*, "Under the table: Tap authentication for smartphones," *BCS-HCI'13*, London, UK, Sept. 2013, pp. 33:1–33:6.

[24] F. X. Lin, *et al.*, "Rhythmlink: securely pairing i/o-constrained devices by tapping," in *ACM UIST'11*, Santa Barbara, CA, Oct. 2011, pp. 263–272.

[25] N. Zheng, *et al.*, "You are How You Touch: User Verification on Smartphones Via Tapping Behaviors," in *IEEE ICNP'14*, The Research Triangle, NC, Oct. 2014, pp. 221–232.

[26] M. Shahzad, *et al.*, "Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it," in *MobiCom'13*, Miami, FL, Sep. 2013, pp. 39–50.

[27] A. Serwadda, *et al.*, "When kids' toys breach mobile phone security," in *ACM CCS'13*, Berlin, Germany, Nov. 2013, pp. 599–610.

[28] J. Sun, *et al.*, "TouchIn: Sightless two-factor authentication on multi-touch mobile devices," in *IEEE CNS'14*, San Francisco, CA, Oct. 2014, pp. 436-444.

[29] J. Sun, *et al.*, "TouchIn: Sightless two-factor authentication on multi-touch mobile devices," in *arXiv preprint arXiv:1402.1216*, 2014.