TapLock: Exploit Finger Tap Events for Enhancing Attack Resilience of Smartphone Passwords

Hongji Yang[1], Lin Chen[1,2], Kaigui Bian[1], Yang Tian[1], Fan Ye[3], Wei Yan[1], Tong Zhao[1], and Xiaoming Li[1]
[1]Institute of Network Computing and Information System, School of EECS, Peking University, Beijing, China
[2]Department of Electrical Engineering, Yale University, New Haven, CT, USA
[3]Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA

***Abstract*—In this paper, we present *TapLock* as a smartphone password system that exploits the finger tap events on capacitive touch screens for increasing the password's resilience to shoulder-surfing attacks (where the password input by a user can be easily observed by a bystander over the user's shoulder). TapLock captures the size and the axis length of the finger touch area on the phone screen for creating a password, which cannot be easily observed by a shoulder surfer. Our user study shows that TapLock has several advantages over existing smartphone password systems, including its strong attack resilience, small authentication delay, and haptic input feedback that improves the usability.**

I. Introduction

Smartphones are popular and consumers are increasingly using their phones to access and store private information (e.g., bank accounts). A vast majority of users want to prevent a third party from gaining unauthorized access to their phones. Modern smartphones commonly use a conventional password system, which allows users to set a Personal Identification Number (PIN) or a text/graphical password that must be entered to unlock the phone or to use a specific application [3], [6], [7], [10], [11][1]. Most recently, Apple's Touch ID sensor [2] enables the fingerprint when unlocking iPhone 5s/6 (plus), or an app over the iOS devices, while such a fingerprint sensor is not available on most of today's smartphones. Recent research reveals that those conventional password systems on smartphones are far from being secure.

As known, PIN/text/graphical password systems are subject to the *shoulder-surfing* attack where the password input by a user can be easily observed by a bystander over the user's shoulder. Various types of password inputs have been tried to make the password system shoulder-surfing resistant, such as tactile/haptic pattern (e.g., vibration) [4], eye-gaze entry [6][2], and the pressure of a digital pen on touch screens [7]. However, the vibration pattern based password depends on a customized input hardware; the eye-gaze based password incurs an increased authentication delay (about 10 seconds); the input of a pressure-based password requires additional accessaries (i.e., a digital pen), and the pressure value may not available on today's smartphones with capacitive touch screens that can be pressure-insensitive.

TapLock is a smartphone password system that exploits the finger tap events on capacitive touch screens for increasing the password's attack resilience. Finger tap events on a touch screen can produce a rich set of information, including location coordinates, touch area size, orientation, the length of the major/minor axis of an ellipse that describes the touch area, of a given finger tap. We perform a systematic study on the feasibility of capturing these featured values on smartphones. Based on those available featured values, each finger tap can be assigned a label of "big" or "small" according to a machine learning algorithm, and the finger tap label cannot be easily observed or perceived by a third party other than the phone user. TapLock exploits the finger tap label for creating a password, thereby enhancing its shoulder-surfing resistance.

We prototype TapLock on Android smartphones as a PIN-based screen unlock tool. The experimental results show that:

1) TapLock greatly enhances the attack resilience of the smartphone password to known attacks without requiring any hardware modification or additional accessary;
2) Most users are able to correctly memorize the TapLock password that contains a PIN and a sequence of finger tap labels;
3) TapLock increases the authentication delay by approximately half a second, which has little impact on user experience.

The rest of the paper is organized as follows. We systematically study the feasibility of exploiting finger tap events for creating high-dimensional passwords in Section II. Section III describes the proposed TapLock system, including the design and implementation of TapLock passwords. In Section IV, we analyze its password entropy and conduct extensive user studies to evaluate the the TapLock system in terms of usability, increased authentication delay, and shoulder-surfing attack resilience. We conclude this paper in Section V.

II. A Study of Finger Tap Events

In this section, we conduct a systematic study on the feasibility of exploiting finger tap events for creating the high-dimensional password. Finger tap events have been exploited for enhancing the user experience of single-handed smartphone operations [5].

A. A Taxonomy of Finger Tap Events

We develop an Android application based on the Motion-Event API [1] to capture the featured values of finger tap events, which include:

[1]PIN is a simplified text password system.
[2]Users can gaze at the letters on a keyboard for the password input.

TABLE I
Statistics of Featured Values of Finger Tap Events on Android Devices. Values appended to 'f:' and 'm:' represent the results of female and male volunteers respectively.

Featured value (Android)	**Supporting Devices**	**(Avg, Var.) on I9100**	**(Avg, Var.) on Droid X**
Orientation (getOrientation)	None	f: $(0, 0)$ m: $(0, 0)$	f: $(0, 0)$ m: $(0, 0)$
Pressure (getPressure)	N/A on S5368, I919	f: $(0.04, 0.02)$ m: $(0.07, 0.02)$	f: $(0.11, 0.03)$ m: $(0.20, 0.02)$
Touch size (getSize)	All	f: $(0.19, 0.04)$ m: $(0.28, 0.07)$	f: $(0.09, 0.04)$ m: $(0.15, 0.05)$
Major axis len. (getTouchMajor)	All	f: $(33, 7.5)$ m: $(52, 10.15)$	f: $(21, 5.8)$ m: $(30, 8.3)$
Minor axis len. (getTouchMinor)	All	f: $(33, 7.5)$ m: $(52, 10.15)$	f: $(21, 5.8)$ m: $(30, 8.3)$

- The "orientation" of a finger tap measures the radians relative to the vertical plane of the device;
- The finger tap "pressure", although it may not be available on capacitive touch screens, is returned by an Android method "getPressure", generally ranging from 0 to 1;
- The "size" of the finger touch area is in relation to the maximum detectable size for the device, and it is normalized to a range from 0 to 1;
- The "length of major/minor axis" of the touch area ellipse is an integer and measured by display pixels.

B. Featured Values of A Finger Tap

We recruited 20 volunteers (ten females, and ten males), and each of them was instructed to perform ten finger taps on off-the-shelf Android devices (Moto Droid X, HTC G11, Samsung S5368, Samsung I919 S7562, Samsung I9100). We show the results in Table I, and summarize our observations below.

Availability of featured values. The orientation of the finger tap is unavailable on all devices. The finger tap pressure is unavailable on a few devices. In contrast, the size, and the length of major/minor axis of the finger touch area are available on all devices.

Average and variance of available featured values. The length of major/minor axis has a greater variance than the value of touch size. Moreover, the length of major axis is exactly the same as that of the minor axis on all devices. This implies that all devices' touch screens treat the finger touch are as a round shape.

III. The TapLock System

The goal of TapLock is to extract the featured values of the finger tap events, and combine these values with the conventional key inputs (e.g., PIN/text/graphical inputs).

Suppose a TapLock password, $\mathbf{P}$, is m-bit long[3] and n dimensional, and can be represented as a n-tuple:

$$\mathbf{P} =< \mathbf{p}_1, ..., \mathbf{p}_i, ..., \mathbf{p}_n >,$$

[3]An m-bit long password is also known as an m-digit password.

where $\mathbf{p}_i = \{p_{i,1}, ..., p_{i,j}, ..., p_{i,m}\}$ denotes the i^{th} dimension of the password that contains m bits of input. Most smartphone password systems have $n = 1$, and $\mathbf{p}_1$ contains a number of keypad inputs, such as PIN values, text inputs, etc. For example, the 4-digit PIN system is a password system with $m = 4$ and $n = 1$.

However, such a single dimensional password input clearly exposes the x, y location coordinates of a finger tap on the touch screen (i.e., the password) to attackers. For this reason, we set out to add more input dimensions in the TapLock password—i.e., featured values of the finger tap events—to complement the x, y position coordinates for enhancing the password's attack resilience.

A. TapLock Password Design

Based on the available featured values, it is easy to create the m-bit multi-dimensional password for TapLock.

Choice of binary input for TapLock password. It is difficult for a user to input the exact value of either the touch size or axis length via a simple finger tap. On Android platform, the getSize() method returns a value in the range of $[0, 1]$; the getTouchMajor() and getTouchMinor() methods return a positive integer upper-bounded by the maximum number of display pixels.

Naturally, it is relatively easy for a user to input a binary value that indicates the current finger tap is "big" or "small". Hence, we define a coarse-grained measure for the finger tap area's size, called *the tap label*. Based on the historical featured values of touch size and axis lengths, TapLock employs a classifier to determine the tap label of each user input as big or small.

As a result, we devise the TapLock password as an m-bit two-dimensional one:

$$\mathbf{P}_t =< \mathbf{u}, \mathbf{b} >,$$

where $\mathbf{b} = \{b_1, ..., b_m\}$ is the second dimension of the password input, and every $b_i, i \in [1, m]$ takes a binary value: $b_i = 1$ indicates that the i-th finger tap's label is big; and $b_i = 0$ indicates that the i-th finger tap's label is small.

System challenges. There are two system challenges that need to be addressed before TapLock becomes usable.

1) *Attack-resilient input feedback*. It is necessary to employ an input feedback such that a user could realize whether her/his current finger tap input is big or small. Without such an input feedback, it is difficult to implement the password setup or recognition. Meanwhile, the input feedback must be resistant to known attacks.
2) *Diversity of users' finger taps*. Intuitively, a male's finger tap size is greater than a female's finger tap size. More generally, the finger tap sizes for any two users are different. Thus, TapLock should be able to learn from the featured values of individuals' past finger tap events, and then determine the current finger tap's label as either big or small. Next, we discuss the solutions to these challenges.

Three steps of TapLock implementation. The implementation of a TapLock system consists of three steps, namely:

1) *Password setup*: a user needs to tell the phone the size of his big/small finger tap, as well as his password digits.
2) *Password recognition*: the phone is able to correctly detect a big/small finger tap as well as the input digit from a user.
3) *Input feedback*: upon recognizing each big/small finger tap, the phone immediately sends a haptic feedback to the user indicating this tap pattern, such that the user can be aware of his previous inputs and proceed with the next-digit input.

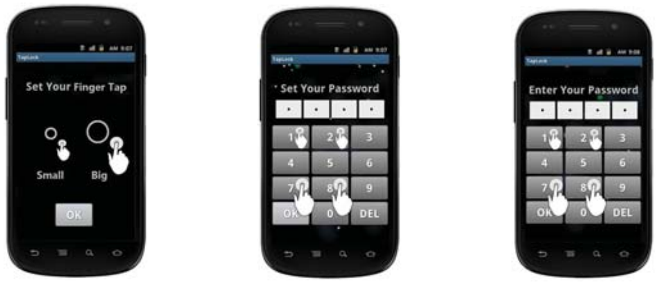
B. One-time Password Setup

At this step, a user has to perform a one-time setup for the finger tap's label and a number of keypad inputs. To recognize an input finger tap' label, a user has to provide the system with the knowledge about the featured values of a big or small finger tap event. Note that a smartphone is typically owned by individuals, and thus the one-time setup is sufficient for a phone to recognize the TapLock password set by its owner. Another user of the phone has to repeat this step to set his/her own password.

1) *Setup of the finger tap's label*. Figure 1(a) shows the user interface and two circles are used to define the finger touch area where the featured values of finger tap events can be captured.
   - To set up the finger tap's label, the user needs to place one finger on the display and press each of two circles for at least once. Upon each touch, the user is required to alter the finger touch size to match the size defined by each circle as much as possible.
   - Each finger tap is classified by the TapLock system with a label of big or small according the following rule: (1) the finger tap matching the small circle is labeled as small; and (2) the finger tap matching the big circle is labeled as big.
2) *Setup of a TapLock password*. To set up a TapLock password, a user intentionally presses the keypad button of each password bit with a big or small touch area, which allows the TapLock system to assign a label of big or small to each password bit. Figure 1(b) illustrates this procedure when the password is

$$< \{1, 2, 7, 8\}, \{small, small, big, big\} > .$$

Note that, for high-end smartphones, it is easy for the system to capture two featured values for each finger tap—the user's finger touch size and major/minor axis length of the touch area. However, low-end smartphones may fail to provide the two aforementioned featured values, and TapLock will collect the distance between two boundary points of the touch area (e.g., the distance between the upper-left-most and lower-right-most boundary points). Intuitively, a greater distance of boundary points in the of the touch area indicates a "bigger" finger tap.



(a) (b) (c)

Fig. 1. Screenshots of TapLock as a 4-digit PIN system. (a) Setup of a finger tap's label: The finger tap matching the small (or big) circle is automatically assigned a small (or big) tap label. (b) Setup of a TapLock password: The big (or small) finger icon shows a big (or small) touch area. The icons in the figure are for illustration purposes, which are not visible to users. (c) Enter password on a keypad with a fixed layout.

C. Learning-based Password Recognition

The TapLock system is required to not only recognize every keypad input of the password, but also the binary label value of each finger tap (either big or small). We formulate the password recognition problem as a classification problem, which classifies a finger tap event based on its featured values and the historical featured values of past finger tap events.

The goal of the TapLock password recognition problem is to determine the label of a new finger tap event (a.k.a. a *test instance*). During the password setup, we have collected a set of past finger tap events (a.k.a. *training instances*), and let D denote such a set. Each instance $d \in D$ can be described by two featured values and a tap label. The two featured values are $A_{d,1} = getTouchMajor()$, $A_{d,2} = getSize()$, and the tap label $L_d = 0$ (or $L_d = 1$) if the instance d is classified as small (or big).

Given D and a new password input, the TapLock system needs to determine the label of each password bit as big or small. We use a simplified k-Nearest Neighbor Classification (kNN) algorithm to address this problem (i.e., the 1-NN algorithm over D that has a fixed size). Let t denote a test instance that represents the i-th finger tap event of a password (the i-th bit of the password input).

1) Compute the distance values between t and each training instance $d_i \in D$ as below:

$$\delta_1(t, d_i) = |A_{t,1} - A_{d_i,1}|, \delta_2(t, d_i) = |A_{t,2} - A_{d_i,2}|.$$

2) Choose the training instance, $d_1 \in D$ that is nearest to t in terms of distance value δ_1. Similarly, choose the training instance, $d_2 \in D$ that is nearest to t in terms of distance value δ_2.
3) If $L_{d_1} = L_{d_2}$, assign t the label of the two chosen training instances, d_1 and d_2.
4) If $L_{d_1} \neq L_{d_2}$, choose another training instance, $d_3 \in D$ that is nearest to t in terms of distance value δ_1, and assign t the label of d_3.

D. *Haptic Input Feedback*

The haptic input feedback, such as the phone vibration upon the user's input, cannot be easily observed by a third party. When a finger tap is determined as big by TapLock, the phone provides a vibration feedback which helps the user to memorize the password. Specifically, the user realizes the keypad input via the visual perception by looking at the button being pressed, and realizes the tap label of each finger tap event through the haptic perception of whether the current finger tap causes a vibration.

IV. Evaluation

In this section, we first analyze the security of the TapLock password, and then show the experimental results. TapLock is easy to be integrated with any existing smartphone password system. In the experiment, we simply prototype TapLock as a 4-digit PIN system for screen unlock on Android platforms.

A. *Password Entropy*

TapLock is useful to enhance the attack resilience of a password system based on its already established password entropy. Let $\mathcal{E}$ denote the entropy of an m-bit password based on the keypad input. By enabling TapLock in this m-bit password, each password bit (i.e., each keypad button being pressed) has a binary-valued label, and the password entropy becomes $\mathcal{E}\cdot 2^m$. In other words, TapLock enhances the entropy of an m-bit password by a factor of 2^m.

For example, the 4-digit PIN system (each digit is chosen from $\{0, 1, ..., 9\}$) has a password entropy of 10^4, and thus the 4-digit TapLock system has a password entropy of 20^4. Similarly, TapLock works well with the Android "draw-a-pattern" system (with nine round buttons), and it enhances the "draw-a-pattern" password by a factor of up to 2^9. Many smartphone password systems require a text-based password with a length of more than eight characters, which implies its password entropy can be increased by TapLock with a factor of at least 2^8. Ideally, in a graphical password system, the user is allowed to freely draw any pattern without being restricted to fixed buttons. If a big or small label can be assigned to the pattern at any time during the drawing process, the password entropy becomes ∞.

There is always a tradeoff between the complexity of a password and the user effort of memorizing the password, which exists in TapLock as well. The longer a password is, the more efforts a user needs to pay to memorize the password. A user can choose an appropriate length of a password to balance the tradeoff. We evaluate the performance of TapLock via experiments over a PIN system, and similar experiments can be easily done when TapLock is combined with other password systems.

B. *Usability*

In the first experiment, we invite 30 participants and categorize them in three age groups (each group has ten users): between 20-29, 30-39, and 40-49 years old.

We first study the difficulty of memorizing and using the two-dimensional TapLock password, and the results are shown in Figure 2. All the users in the first two age groups are able to correctly setup their passwords and use a TapLock password to unlock the phone. However, there are three users in the third group (10% of all users) respond that it is somewhat difficult to memorize the big/small pattern of their PIN's. During the password input phase, all users are able to perceive the vibration feedbacks when a big finger tap occurs.

C. *Detection Rate of Big/small Finger Taps*

In the second experiment, we require participants to press the buttons of digits 0 through 9 with their thumbs over all of our Android devices mentioned in Section II, and then these tap events are classified by the aforementioned classification algorithm (see Section III) and assigned with a label of "big or small".

The results show that the learning-based password recognition algorithm of TapLock achieves a detection rate of 98% for recognizing small finger taps and 88% for recognizing big finger taps. This thereby validates the accuracy of TapLock's classification algorithm when detecting big/small finger taps.

D. *Increased Authentication Delay*

Adding the big/small labels will increase the time for completing the password input process. We compare the average time of inputting a 4-digit PIN and a 4-digit TapLock password in this experiment.

Delay of entering a simple and designated PIN. We first create a simple 4-digit PIN $\{1, 2, 3, 4\}$, and then assign $\{small, small, big, big\}$ labels to the four digits to form a 4-digit TapLock password. We require the participants to input the 4-digit PIN and the TapLock password respectively.

In Figure 3, the increased delay when entering a TapLock password in the first two age groups is less than half a second. However, the delay increases by nearly one second in the third group. Overall, the discrepancy of delay between the two systems increases as the user age grows, and all of the participants respond that they can tolerate such a small delay.

Delay of entering a user-selected PIN and a randomly generated PIN. We require the ten users whose ages range from 20 to 29 to input user-selected and randomly-genrated passwords via the classical PIN system and our TapLock system, respectively. This study is conducted on Motorola Droid X. The statistics of average input latency for entering different types of PINs are shown in Figure 4. The results indicate that there is no significant impact of different passwords on the password input latency (the difference is less than 0.2 seconds), although it takes a shorter time for a user to input a PIN chosen by himself/herself.

E. *Shoulder-surfing Attack Resilience*

Vulnerability of phone vibration. In quiet environments, one may argue that TapLock's haptic input feedback (phone vibrations) may be heard by others so that a side-channel attacker can tell the current input's big/small label.

To investigate the vulnerability of phone vibration to this side-channel attack, we conduct a series of audibility experiments in quiet (noiseless and silent offices) and noisy

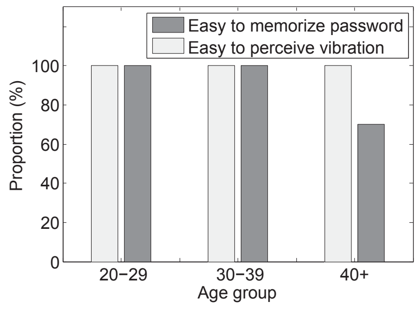


Fig. 2. Results of the user study on usability.

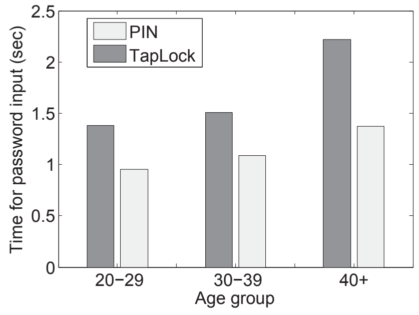


Fig. 3. Delay of entering a simple 4-digit PIN.

environments (at restaurants) to test the audibility of phone vibrations with different durations: short (0.01s), medium (0.05s) and long (0.1s) duration vibration, where the attacker is at a distance of 0.3m from the tested smartphone (Motorola Droid X) and the user. The experiment results are presented in Figures 5 and 6. We are able to observe that

1) A phone vibration as short as 0.01s is imperceptible even in quiet environments;
2) A phone vibration that lasts longer than 0.05s is audible in quiet environments while inaudible in noisy environments.

Short vs. medium phone vibration. A phone vibration as short as 0.01s, however, can be excessively weak for a user's hand to detect, as reported by 30% of the participants. Thus, a short vibration in turn cannot function as a feedback avenue for those who have problem in perception of short vibrations.

However, a phone vibration that lasts approximately 0.05s can effectively provide haptic feedbacks for users; meanwhile, it can be imperceptible for eavesdroppers in most scenarios, which enables TapLock's shoulder-surfing attack resistance.

Attack resistance. We recruit ten users as observers/attackers who stand at 0.3 m from the user's smartphone to launch the shoulder-surfing attack against the password input process by users of all age groups in the traditional PIN system and the proposed TapLock system, respectively. These observers/attackers are aware of whether the PIN system or the TapLock system is employed in each test. We assume an attcker has no way of touching the phone to detect a phone vibration or not. Experiment settings remain the same as the previous experiment.

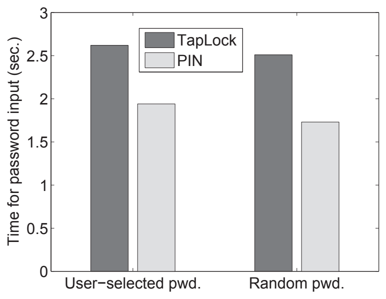


Fig. 4. Delay of entering different 4-digit PINs (user-selected or randomly generated).

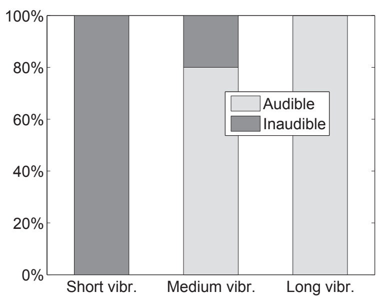


Fig. 5. Audibility of phone vibrations with different durations in quiet environments.

In this experiment, we analyze statistically the success rate of shoulder-surfing attacks in three different scenarios: big/small finger taps only, digit inputs only, and TapLock password input (big/small taps plus digits).

- Results show that observers can very easily make a correct guess of input PINs that consist of only digits (see light-grey bars in Figure 7), as the buttons that a user presses during a password input process can be clearly identified by an attacker at a distance of 0.3 m from the user's smartphone.
- However, during a TapLock password input process, one may have difficulty in distinguishing whether a finger tap is small or big, as reported by all attackers. For example, 30% and 90% of attackers fail to guess the big/small finger taps when the length of password is 4 and 8 digits, respectively (see dark-grey bars in Figure 7).
- Meanwhile, all attackers report that they fail to remember anything as long as they attempt to pay attention to both of the digit input and the big/small taps simultaneously—i.e., they cannot infer the exact content of a TapLock password.

Impact of the length of a password. As Figure 7 shows, when the length of password is four, only inferring a sequence of big/small taps yields a lower success rate compared with inferring a sequence of PIN digits. As the password length increases, there a significant reduction in the success rate for inferring a sequence of big/small taps; in contrast, a 8-digit PIN is still vulnerable to shoulder-surfing attacks (the attack

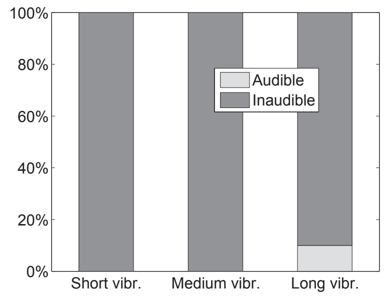


Fig. 6. Audibility of phone vibrations with different durations in noisy environments.

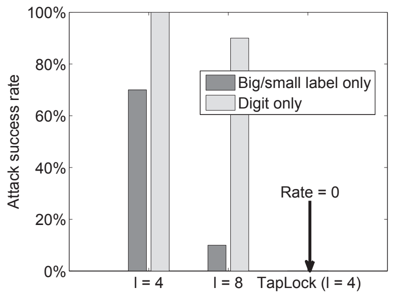


Fig. 7. Success rates for shoulder-surfing attack: l denotes the number of input digits in a password; the dark grey bars represent the results of big/small label only passwords; the light grey bars represent the results of conventional PIN (digit-only) passwords.

success rate is around 90%). Again, none of attackers succeed in inferring a 4-digit TapLock password.

Extreme cases of password input. According to the statistics, two users in the third age group spend more than five seconds to finish the input process, which may provide a sufficiently long time for the attacker to observe the finger taps and infer the big/small tap labels. However, they finally fails in capturing the password.

Furthermore, the attackers in the experiment report that they try to infer big/small tap patterns via both finger tapping movements and the sound of phone vibrations. Therefore, the worst case would be the combination of a user with a slow input speed, and an attacker with an extraordinary memory or a camera. However, using a camera will alert the user to protect his/her password input from an observer.

F. Discussions on Side-channel Attacks

Side-channel attacks are feasible against the smartphone password system when an information leakage occurs during the password input procedure. For example, embedded motion sensors on modern smartphones unexpectedly leave a loophole for launching a motion-based side channel attack [9] based on the keystroke biometrics [8]. Specifically, motion sensors are used to collect the smartphone motion data (e.g., orientation, acceleration, direction), and thus it is possible for a third party to predict the positions of password keystrokes with up to 90% accuracy. Another type of side channel attacks can be launched when oily fingers leave smudges on the touch screens [3]. As a side effect, smudges may be usable to infer frequently touched areas of the screen—a form of hint for predicting the password.

Please note that the defense against the side-channel attack is out of the scope of this work. However, we find that the TapLock password system with a randomized keypad layout can be robust to both the motion- and smudge-based side channel attacks. Our study shows that a row-wise randomization of keypad layout could completely remove the relationship between the location of finger tap and the key input.

V. Conclusion

In this paper, we present *TapLock*, a smartphone password system that exploits the finger tap events on capacitive touch screens for enhancing the password's resilience to the shoulder-surfing attacks. The size and major/minor axis length of the finger touch area on touch screens are useful to create a multi-dimensional password that is shoulder-surfing resistant. Our user study shows that the TapLock system is attack resilient, easy to use, and its authentication delay is small.

VI. Acknowledgement

This work was partially sponsored by Beijing Natural Science Foundation under grant 4143062, National Natural Science Foundation of China under grant number 61201245 and 61272340, and Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) under grant 20120001120128.

References

[1] Android Developers. MotionEvent. http://developer.android.com/reference/android/view/MotionEvent.html.
[2] Apple. Touch ID. http://www.apple.com/iphone-6/touch-id/.
[3] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge attacks on smartphone touch screens. *WOOT*, 10:1–7, 2010.
[4] A. Bianchi, I. Oakley, and D. S. Kwon. The secure haptic keypad: a tactile password system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1089–1092. ACM, 2010.
[5] S. Boring, D. Ledo, X. Chen, N. Marquardt, A. Tang, and S. Greenberg. The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 39–48. ACM, 2012.
[6] A. Forget, S. Chiasson, and R. Biddle. Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1107–1110. ACM, 2010.
[7] B. Malek, M. Orozco, and A. El Saddik. Novel shoulder-surfing resistant haptic-based graphical password. In *Proc. EuroHaptics*, volume 6, 2006.
[8] R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 201–210. IEEE, 2010.
[9] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 323–336. ACM, 2012.
[10] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on Advanced visual interfaces*, pages 177–184. ACM, 2006.
[11] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 113–124. ACM, 2012.