Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 4627108, 19 pages
https://doi.org/10.1155/2018/4627108

Research Article

Niffler: A Context-Aware and User-Independent Side-Channel Attack System for Password Inference

Benxiao Tang, Zhibo Wang, Run Wang, Lei Zhao, and Lina Wang

School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Zhibo Wang; wzb.zju@gmail.com

Received 2 March 2018; Accepted 26 March 2018; Published 8 May 2018

Academic Editor: Ximeng Liu

Digital password lock has been commonly used on mobile devices as the primary authentication method. Researches have demonstrated that sensors embedded on mobile devices can be employed to infer the password. However, existing works focus on either each single keystroke inference or entire password sequence inference, which are user-dependent and require huge efforts to collect the ground truth training data. In this paper, we design a novel side-channel attack system, called Niffler, which leverages the user-independent features of movements of tapping consecutive buttons to infer unlocking passwords on smartphones. We extract angle features to reflect the changing trends and build a multicategory classifier combining the dynamic time warping algorithm to infer the probability of each movement. We further use the Markov model to model the unlocking process and use the sequences with the highest probabilities as the attack candidates. Moreover, the sensor readings of successful attacks will be further fed back to continually improve the accuracy of the classifier. In our experiments, 100,000 samples collected from 25 participants are used to evaluate the performance of Niffler. The results show that Niffler achieves 70% and 85% accuracy with 10 attempts in user-independent and user-dependent environments with few training samples, respectively.

1. Introduction

With the rapid development of embedded systems and mobile computing, mobile devices (e.g., smartphones) become more and more powerful and make our life much more convenient than before. Thousands of Apps have been developed for mobile devices which rely on the embedded sensors (e.g., camera, accelerometer, and compass) to provide specific functions. Unlike certain data, such as photo, which is protected strictly and can be stored in cloud [1, 2], mobile systems provide kinds of interfaces for developers to access sensors with little restrictions. Although providing convenience [3] and protection [4] for developers and mobile users, this leaves vulnerability to attackers who intend to construct side-channel attacks with embedded sensors to recover users' sensitive information, such as the location [5], screen lock password, and credit card numbers [6–8]. These attacks based on motion sensors usually work in background and the perceived risk of motion sensors among users is very low [9]. Considering the popularity and universality of mobile devices, this kind of attacks has become a significant threat to sensitive information leakage.

Although there exist other secure authentication methods (e.g., fingerprint reader, graphical password), digital password still remains a vital place and is commonly used in screen lock and mobile payment. Many works have leveraged motion sensors to launch side-channel attacks to infer password lock. Some works divide the unlocking process into several tap events on keystrokes and explore the process of tapping a single keystroke to infer the lock password [10–13]. They found that tapping different positions on touchscreen causes particular reflections on the readings of accelerometer and gyroscope. Based on this observation, they inferred every single keystroke separately and combined them together to recover users' passwords. In contrast, Aviv et al. refer the process of tapping one entire password as a category to infer the password, instead of individual keystroke [14]. However, it is difficult to build a ground truth training set that contains all possible categories (e.g., there are 10^4 categories for four-digit password). Researches have proved that users have their

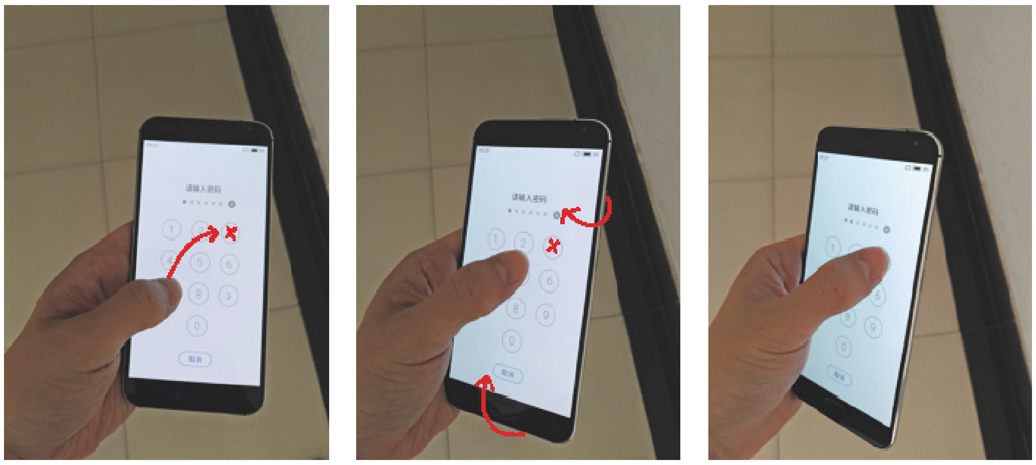

(a) (b) (c)

Figure 1: An illustration of grip change for a movement from button "7" to button "3." (a) The user taps button "7" and wants to tap button "3"; (b) the device is tilted to a suitable angle for the user; (c) the user taps button "3" with his thumb.

own unique behavioral patterns when tapping keystrokes on the touch screens [15], and the authentication mechanism [16, 17] exploring motion sensor is based on this theory; therefore, all of the above side channels are sensitive to users. The common assumption that attackers can collect sufficient data in target's device in training stage is difficult in practice. Whether accessing to mobile sensors within app or browser [18], attackers must build a unique model for certain target.

Note that existing attack methods focus on either single keystroke inference or whole password inference, which are user-dependent so that they are not scalable in practice and require a huge effort of collecting the ground truth training set [19]. In this paper, we explore the user-independent feature during the password unlocking process to accurately infer password with little effort and good scalability. We consider a common scenario that a user holds and unlocks its smartphones with one hand (according to the research of Hoober, 49% users hold mobile devices with one hand as they are the right size for that [20]). If a user moves his thumb from one keystroke to another, he needs to change the grip of the smartphone. We observe a fact from the experiments that same movements from one keystroke to another of different users have similar grip changes which can be reflected by similar changing trends of motion sensor readings. As shown in Figure 1, when a user moves his thumb from button "7" to button "3", the device will be tilted to a suitable angle for the thumb. Drawn from the observation and the characteristics of password unlocking process which consists of alternative movement and tap event, we argue that password unlocking is a Markov process and the movements of tapping two consecutive keystrokes are context-aware and user-independent, and the experimental results shown in Section 3 validate our argument.

In this paper, we design a novel side-channel attack system, called Niffler, that leverages accelerometer to infer unlocking passwords on smartphones by exploiting user-independent movements of tapping consecutive buttons. The basic idea of Niffler is to infer every movement (e.g., movement "7-3" in Figure 1) of the password sequence with accelerometer readings and then combine them together to find the most possible candidates for the password. We extract angle features to reflect the changing trends of similar grip changes. With these features, we build a multicategory classifier to infer the probability of each movement of password sequence and employ the Markov model to reconstruct the whole password sequence. The sequences with the highest probabilities will be considered as the possible candidates for the password. In particular, Niffler provides a feedback component that can improve the accuracy of classifier by feeding the successful attack results back to the classifier.

The data of our experiments are collected from twenty-five volunteers over two months. We evaluate the performance of Niffler in both user-dependent and user-independent environments. In user-dependent experiments, with a few training samples, Niffler achieves an average accuracy about 85% for single movement inference with one attempt. The accuracy of the password inference is near 40% with only one attempt and reaches 84% with 10 attempts. In user-independent environments, the accuracy of the password inference is about 75% with 10 attempts.

Our contributions are summarized as follows:

(i) We observe a fact from experiments that movements during password unlocking are context-aware and user-independent. The same movement of different users has similar grip changes which lead to similar changing trends of accelerometer readings.

(ii) We design and implement a side-channel attack system to infer digital password on smartphones. Different from existing works, Niffler infers every movement of the password based on the user-independent feature and employs the Markov model to reconstruct the whole password sequence.

(iii) We build a multicategory classifier that divides all the centroids of the 110 movements into 11 categories to avoid the ambiguity due to similar movements and adopt the dynamic time warping (DTW) algorithm to infer the probability of each movement, where each category contains 10 centroids of movements with the same starting button.

(iv) We conduct extensive experiments to evaluate the performance of Niffler in both user-dependent and user-independent environments. The experimental results demonstrate that Niffler is an effectively user-independent attack system.

The rest of this paper is organized as follows. We describe the system model in Section 2 and present the observation of internal relation of movements in Section 3. We present the overview and the design of the attack system in Section 4. The experimental results are shown in Section 5. The literatures on password unlock are briefly discussed in Section 6. Finally, Section 7 concludes the paper.

2. System Model

2.1. Digital Lock. Digital lock and pattern lock are still the most popular authentication methods for touch screen devices such as smartphones and tablets even if some new secure authentication methods have arisen. A typical layout of screen lock is shown in Figure 2. A digital password for screen lock is usually 4 or 6 in length where each number can be randomly selected from "0" to "9." Unlike the rules of pattern lock, each number in a digital password can appear more than once, so there will be 10^4 possibilities for a 4-digit password. Given that the Android operating system usually allows at most 20 attempts of entering wrong passwords before locking the device, it is impossible to unlock the device by random guess. In this paper, we focus on the inference of 4-digit password on smartphone, but the attack model can be easily extended to infer 6-digit password.

Generally speaking, Android allows 5 consecutive attempts within a specified time window by default when a user attempts to log in with his/her password. If the user fails to pass the authentication within 5 attempts, he/she is not allowed to try again until a certain time period passes. The interval is usually one minute, and after that another 5 attempts will be given. Most manufacturers allow 20 attempts in total, and the others allow at least 10 attempts.

2.2. Sensor Monitoring. In Android, developers can easily access to sensors in application layer by calling system APIs without claiming any permissions [21]. Furthermore, the sensor data is regarded as public source which is not protected by sandbox. Although users can outsource these data to a cloud server to preserve privacy [22–24], there are many ways for attackers to monitor sensor readings legally. For example, if a user permits the browser to access some harmless information, such as geographic location and sensor readings, the website can record these data stealthily.

Using Java, we designed a simple application which keeps running at background in Android application layer as a monitoring module to record accelerometer readings during user unlocking process. The program is hidden in a pedometer application which needs sensor data to avoid security inspection. Since the W3C specifications allow access to the motion sensors from JavaScript [9], more complex monitoring can be implemented via websites. Otherwise, researches have found approach to sniff and manipulate protected sensors on unrooted Android devices [25].

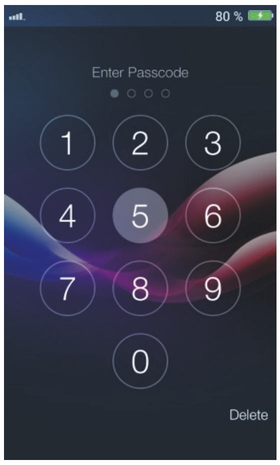


Figure 2: A typical layout of screen lock.

2.3. Scenarios and Assumptions. The main stream smartphones are designed with screen size about 4.7 inches to 5.5 inches which are suitable for single hand holding. We focus on the scenario that users hold mobile devices with one hand only because they are the right size for that, and a user can use the other hand to do other things simultaneously when he/she plays on the smartphone with one hand, according to the research of Hoober [20]. Niffler works for both left hand holding and right-hand holding, but in this paper, we use the scenario that users hold and unlock smartphones with their left-hands only as an example. The readings of accelerometers on smartphones are used to capture grip changes and infer the 4-digit password.

Definition 1 (movement). A movement is defined as the process of moving from one button to another on the screen during password unlocking.

If two movements start from the same keystroke and end at the same keystroke, they are called the *same movement*.

The stream of accelerometer readings can be represented as $\{a_1, a_2, \ldots, a_n\}$, where $a_i = (a_i^x, a_i^y, a_i^z)$ is the reading of accelerometer when the i_{th} sensor event occurs. a_i^x, a_i^y, and a_i^z express the acceleration in the x-axis, y-axis, and z-axis, respectively. The mean interval between every two tap events is about 500 ms and the typical tapping duration is between 100 ms and 200 ms [26], so we sample the accelerometer readings with a frequency of 50 Hz.

3. Context-Aware and User-Independent Movement

In this section, we use experimental results to show that the movements during unlocking process are context-aware and

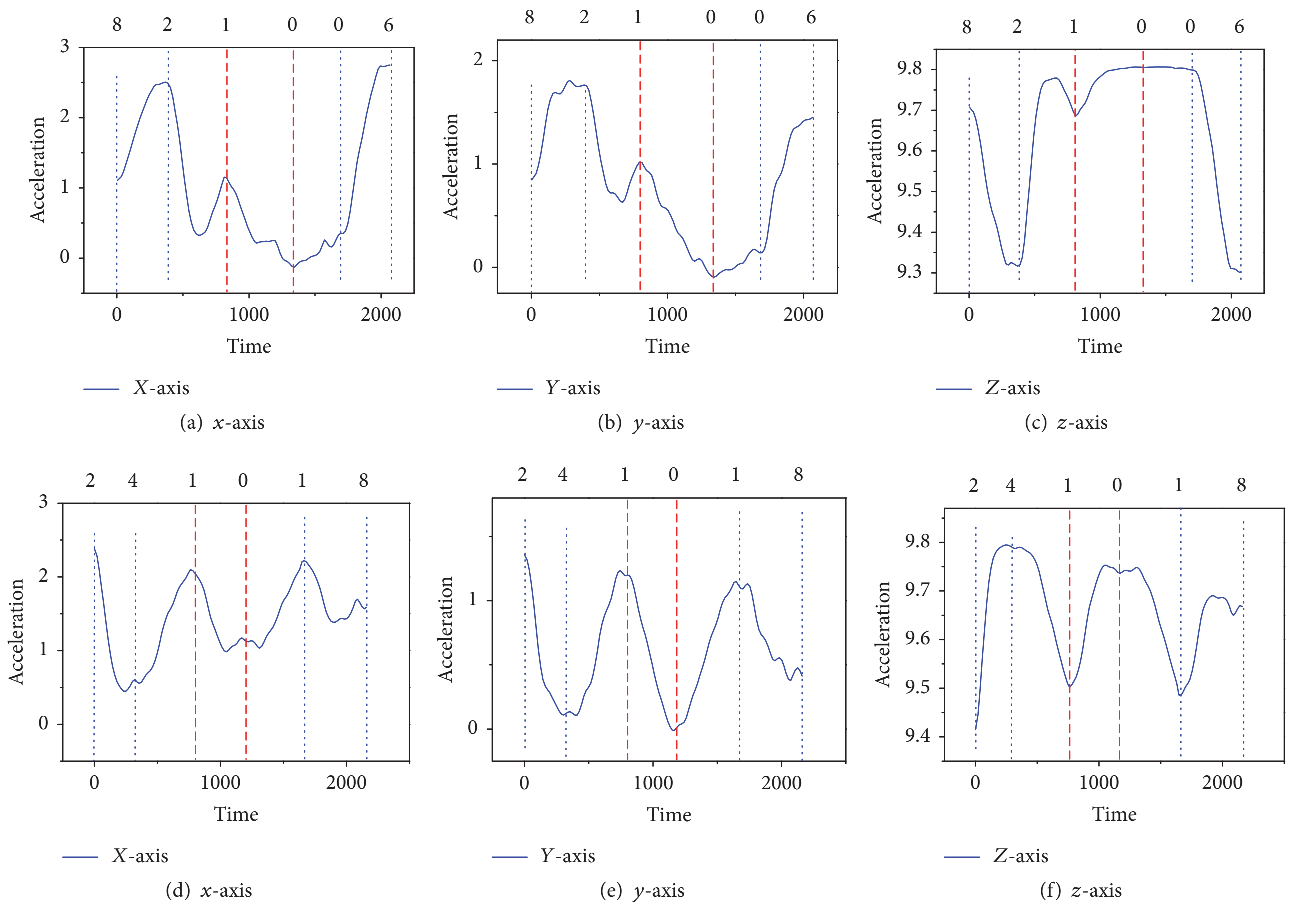


Figure 3: (a–c) Accelerometer readings in three axes of sequence “8-2-1-0-0-6” of user *a*. (d–f) Accelerometer readings in three axes of sequence “2-4-1-0-1-8” of user *b*.

user-independent, which is an ideal observation that can be used to recover passwords. The previous studies of sensor-based side-channel attacks [27] mainly focus on one user’s unique features (e.g., pressure, touched size, key hold time, and interkey time) and use these user-dependent features to crack passwords [14] and help authentication [15] of specific target. However, the context-aware and user-independent features of movements can be employed to build a general side-channel attack model that is scalable to users.

Although smartphones with screen size of 4.7 inches to 5.5 inches are suitable for single hand holding, users still need to tilt and rotate smartphones when they are tapping keystrokes [28]. For example, as shown in Figure 1, a user holds his smartphone with his left hand and wants to type a two-keystroke sequence from button “7” to button “3” with his thumb. The right side of this smartphone has been lied down to uplift the bottom left corner when he types button “7.” When he moves his thumb from button “7” to button “3” at the top right corner, the other four fingers uplift the right side of the smartphone to make the thumb reach the target position easily. We can see that the grip changes are context-aware that are related to the starting position and ending position of the movement. Due to similar structures of human hands as well as the layouts of virtual keyboards, we argue that the same movement on smartphones of different users leads to similar grip changes, which can be reflected by similar changing trends on accelerometer readings.

We perform extensive experiments and the results validate the correctness of our argument. In the following, we use two examples to better explain the context-awareness and user-independence of movements. The sensor sample frequency is 50 Hz, and we leave out tap events to put more focus on movements.

Figures 3(a)–3(c) show the accelerometer readings in three axes of a sequence “8-2-1-0-0-6” of user *a*, and Figures 3(d)–3(f) show the accelerometer readings in three axes of a sequence “2-4-1-0-1-8” of user *b*. These two users have one same movement “1-0”. Although the data are from two different users, we can see that the readings of the same movement “1-0” have the same changing trends in each axis. In contrast, different movements, such as movement “2-1” of user *a* and movement “4-1” of user *b*, show quite different changing trends. These figures also imply the relation between the acceleration variation and the movements. For example, if the movement is from the bottom to the top (e.g., “8-2”), the readings in x-axis and y-axis have increasing trends, while the readings in the z-axis are decreasing.

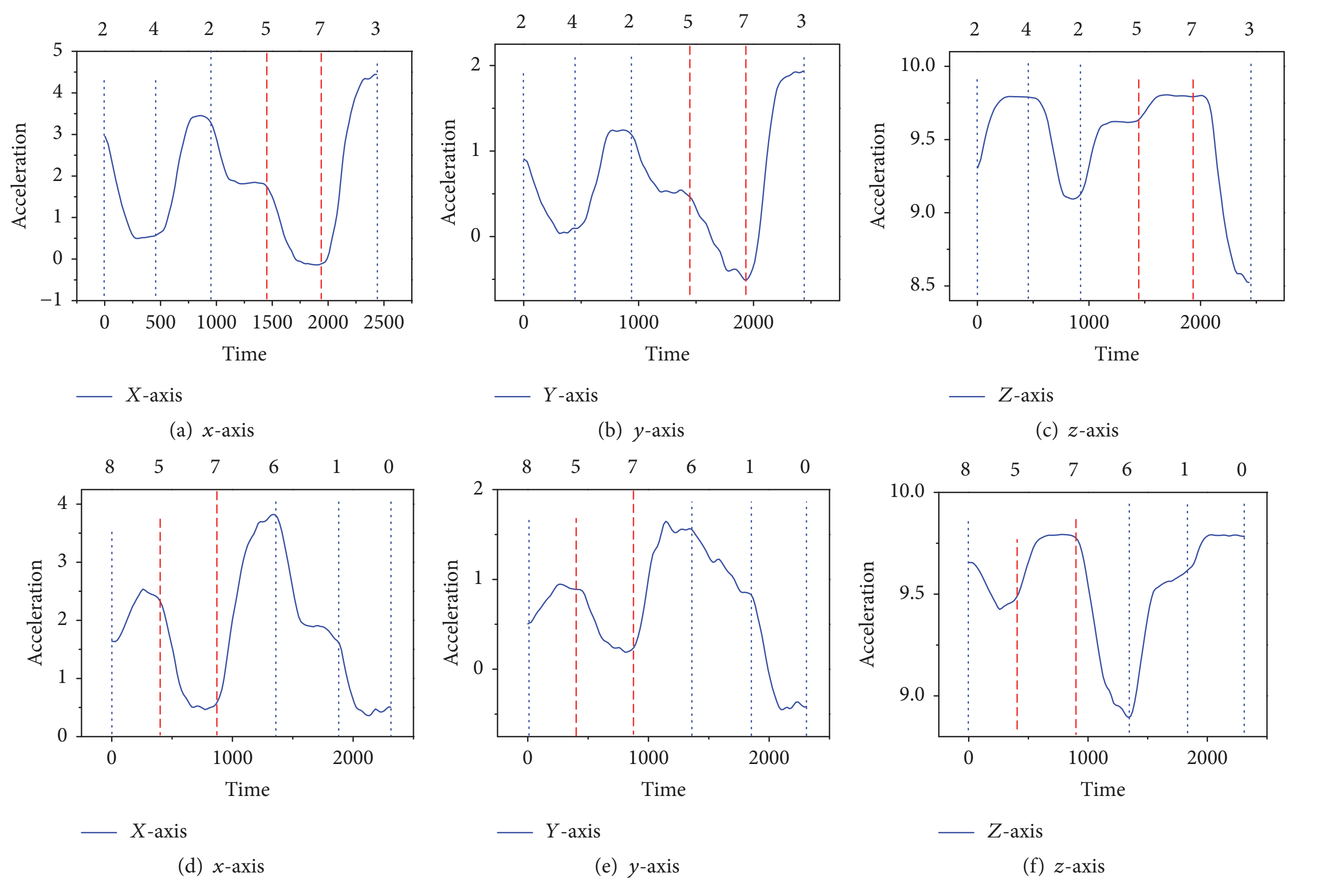


FIGURE 4: (a–c) Accelerometer readings in three axes of sequence "2-4-2-5-7-3" of user *c*. (d–f) Accelerometer readings in three axes of sequence "8-5-7-6-1-0" of user *d*.

Figure 4 is another example that Figures 4(a)–4(c) show the accelerometer readings in three axes of a sequence "2-4-2-5-7-3" of user *a*, and Figures 4(d)–4(f) show the accelerometer readings in three axes of a sequence "8-5-7-6-1-0" of user *b*. We can see that the same movement "5-7" of two different users also have similar changing trends in each axis.

Figures 3 and 4 validate our argument that the same movement leads to similar grip changes and therefore similar changing trends on accelerometer readings. This implies that the changing trends of accelerometer readings due to movements are user-independent, which in turn can be used to infer movements of a password sequence.

It is worth noting that, besides the same movement, similar movements may also lead to similar changing trends on accelerometer readings. For example, the movement of "8-2" and the movement of "4-1" are similar movements because both of them move upward straightly, which leads to quite similar grip changes and therefore similar changing trends on accelerometer readings in each axis, as shown in Figure 3. This introduces challenges to us when we infer movements from changing trends of accelerometer readings. In the Niffler, we design a multicategory classifier to avoid the ambiguity due to similar movements by putting the movements with the same starting button into one category.

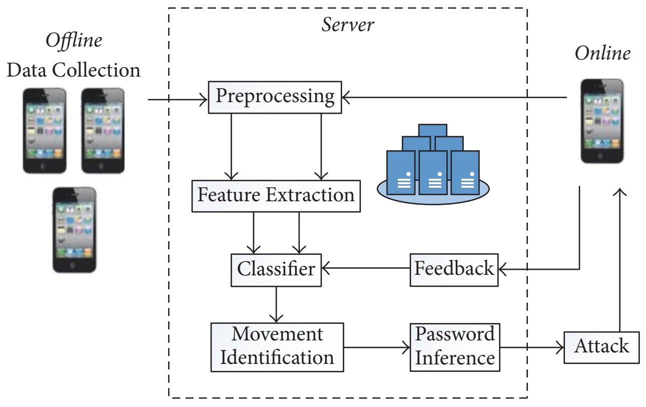
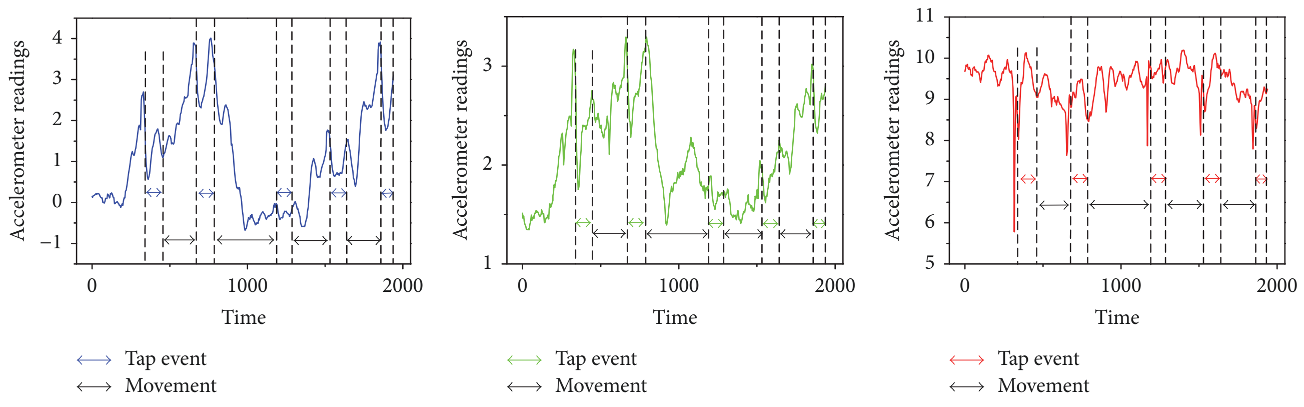


FIGURE 5: System architecture of Niffler.

4. System Design

In this section, we first give a high-level overview of Niffler, which uses accelerometer readings to infer unlocking passwords, and then describe the design of important components in Niffler.

Figure 5 shows the system architecture of Niffler. All the data collected on smartphones during the unlocking



(a) Accelerometer readings in x-axis (b) Accelerometer readings in y-axis (c) Accelerometer readings in z-axis

FIGURE 6: Accelerometer readings of password unlocking process.

process will be uploaded to the server and processed by Niffler. Niffler first preprocesses the collected data to remove noise and segment movements. Since the same movement leads to similar grip changes, angle feature vectors are extracted to represent grip changes of movements. During the offline phase, data of different movements from users are collected to train a classifier which can be further used to infer the movement during online phase. The dynamic time warping (DTW) algorithm is combined with the classifier to infer the possibility and probability of each movement of a password sequence. We further use the Markov model to model the unlocking process with multiple movements and use the sequences with the highest probabilities as the attack candidates. In particular, Niffler provides a feedback component that can improve the accuracy of classifier by feeding the successful attack results back to the classifier.

4.1. Preprocessing. All the raw data collected on smartphones during the unlocking process must be preprocessed at first, which aims to remove noise and segment original data into movements.

Noise Removal. The hand trembles during user tapping introduce noise to accelerometer readings which can affect the password inference. Since the frequency of hand trembles is much higher than that of grip changes, we use a median filter to remove noise. The sampling rate in Niffler is about Hz, so the window size of the median filter is 50 ms and moves with a step size of 20 ms.

Data Extension. The difference among devices' hardware leads to inconsistent sampling rate, even if using the sample app. In order to keep the consistency of frequencies of accelerometer readings, we use cubic spline interpolation to interpolate the vacant values. Data augmentation [29] is also used to leverage limited data in user-independent environment by transforming existing samples of same category to create new ones.

Movement Segmentation. We segment the accelerometer readings into movements in order to further extract features from movements. For the data collected offline, since we know the ground truth of password unlocking, we can easily segment the readings of each password into several movements. Note that, besides the movement between two keystrokes, we also consider a special movement that happens before tapping the first keystroke. Our experiments show that no matter where movements start from in the air, the grip changes are similar if the landing positions are the same. This observation is used for the case that the thumb hangs in the air. Because the average duration between two keystrokes is about 500 ms, we choose the time 300 ms before tapping the first keystroke as the time this movement starts.

Compared to the offline collected data, the segmentation is more difficult for online data, since it is impossible to record the ground truth of tap events legally by using a third-party application due to sandbox. We first identify the start of unlocking process by monitoring the power event. Based on our investigation, users should press the power button to turn on touch screen before they unlock mobiles for most of Android systems. We also use the sum of square root of acceleration in three dimensions to distinguish unlocking process from other activities (i.e., walking or running) [11]. The great background noise will interfere segmentation; thus, Niffler collects data with small standard deviation of acceleration readings.

Since a complete input process consists of tap event and movement in turn, we segment movements by identifying tap event, and the readings between two tapping events will be recognized as a movement. The methods of tap event detection in researches [11, 13] work poorly in our experiment; thus, we employ a new observation to measure the occurrence of tap events. We observe that accelerometer readings have a unique trend in the time of tapping a keystroke, which can be used to estimate the approximate time of tapping a keystroke with a high accuracy. The observation is shown in Figures 6 and 7.

Figure 6 presents accelerometer readings of a complete unlocking process with a higher frequency of 200 Hz to show the detail of tap events and movements. Although the acceleration fluctuations caused by tap events in x-axis (Figure 6(a)) and y-axis (Figure 6(b)) can hardly be identified

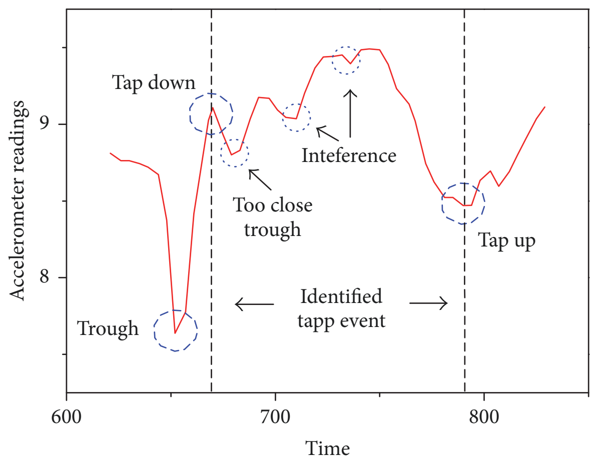


Figure 7: Accelerometer readings of a certain tap event in z-axis.

from movements, the tap events cause a unique and obvious change in accelerometer readings in z-axis (Figure 6(c)). At the moment someone taps the screen, he gives a downward (in z-axis) force to the device and this force will cause an instantaneous acceleration which includes gravity. Figure 7 shows accelerometer readings of one of tap events from Figure 6(c). There is a great trough before tap down, and the first peak point following this trough is the start of tap event. When the tap is finished and the user lifts his finger, an opposite direction acceleration is generated. This action causes another obvious trough which is the sign of the end of tap event. Note that if the following trough is too close (less than 15 ms) to the first trough, it will be regarded as noise. In addition, background noise and shaking of hand also cause some small waves within the tap event, and we lower these interferences by noise removal. The readings from the end of one tap event to the start of next tap event are segmented as a movement.

4.2. Feature Extraction. As we mentioned, the same movement of different users leads to similar grip changes and thus similar changing trends on accelerometer readings. With this observation, we use angle features from acceleration to form a feature vector to represent the trends of grip changes. The pitch angle and the roll angle represent the tilt degree of a device in forward-backward direction and the right-left direction, respectively. We ignore the azimuth angle because the experimental results show that there is little azimuth change during password unlocking process. Thus, we use pitch angles and roll angles as the features to represent grip changes.

The pitch angle and roll angle can be calculated through the gravity acceleration because it is a great reference system to calculate the angle of a device. As Figure 8 shows, the approximate pitch angle is expressed by arc-tangent between the components in z-axis and y-axis, and the approximate roll angle is computed by arc-tangent between the components in z-axis and x-axis.

Each accelerometer reading a_i collected from the device is a proper acceleration, which is a combination of gravity acceleration a_{gi} and linear acceleration a_{li}, where $a_{li} = (a_{li}^x, a_{li}^y, a_{li}^z)$ and $a_{gi} = (a_{gi}^x, a_{gi}^y, a_{gi}^z)$. We have to remove the linear acceleration in order to calculate the pitch angle and roll angle from the gravity acceleration. The linear acceleration in a fixed orientation is a transient variation and is sensitive with the force acted on the device, which means the linear acceleration changes with higher frequency than gravity acceleration. Therefore, we can use a discrete-time low-pass filter to remove the linear acceleration, which is shown as follows.

$$a_{gi} = \rho \times \left[a_{g(i-1)} + \left(a_i - a_{i-1}\right)\right], \tag{1}$$

where $\rho \triangleq c/(c + 1/f)$ controls the frequency of passing acceleration. c is a time constant and f is the data sampling rate. ρ ranges from 0 to 1, and the closer to 1, the higher frequency of passing acceleration. In our experiment, the value of ρ is 0.8.

With the gravity acceleration a_{gi}, the pitch angle and roll angle of a_i are $\arctan(a_{gi}^z, a_{gi}^y)$ and $\arctan(a_{gi}^z, a_{gi}^x)$, respectively. a_{gi}^x, a_{gi}^y, and a_{gi}^z are the components of a_{gi} in x-axis, y-axis, and z-axis, respectively.

Let $\mathbf{F} = [\mathbf{F}_p; \mathbf{F}_r]$ denote the feature vector of a movement, where $\mathbf{F}_p$ and $\mathbf{F}_r$ denote the pitch vector and the roll vector of the movement, respectively. The sampling rate for accelerometer readings is 50 Hz. Each movement may take different duration from another, so they have different number of readings. Thus, the sizes of feature vectors of different movements may be different. Suppose a movement has τ accelerometer readings. Its pitch vector and roll vector are represented as follows.

$$\begin{aligned} \mathbf{F}_p &= \left\{\arctan\left(a_{g1}^z, a_{g1}^y\right), \dots, \arctan\left(a_{g\tau}^z, a_{g\tau}^y\right)\right\}, \\ \mathbf{F}_r &= \left\{\arctan\left(a_{g1}^z, a_{g1}^x\right), \dots, \arctan\left(a_{g\tau}^z, a_{g\tau}^x\right)\right\}. \end{aligned} \tag{2}$$

We further normalize the values of these two vectors to the range of $[0, 255]$ to eliminate the impact of different value ranges while keeping the changing trends of movements.

4.3. Multicategory Classifier Construction. In this section, we describe how to build a multicategory classifier with the collected samples in the offline phase. Note that the classifier can be further improved with a real-time feedback mechanism in the online phase by providing samples of successful attacks to the classifier.

For each button on the keyboard, there are 10 possible movements starting from it. In our system, we also consider the process starting from the air to the first tapped position as a movement. Thus, there are a total of 110 movements. Some ground truth samples of movements are collected and the feature vector for each sample of a movement can be extracted. The sizes of feature vectors are different from each other since movements take different durations. We scale the pitch vector and the roll vector of each feature vector into size of 30 considering that the durations of movements are less than 600 ms. The scaling operation also removes user's individual characteristics in time intervals and keeps

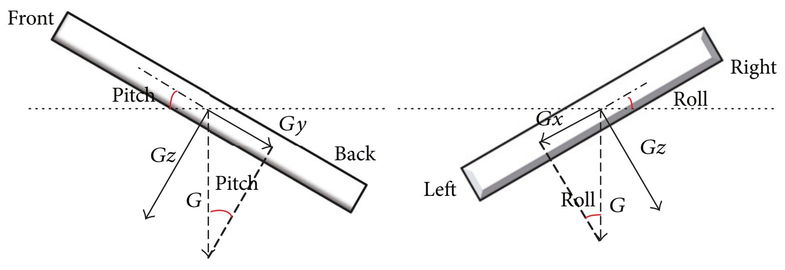


Figure 8: Angle calculation.

the general features between different users. Some researches have shown that the unique behavior of a user will be reflected in time intervals between two nearby keystrokes [15, 30]. Note that the feature vector of a movement is scaled only if it is used for building the classifier.

Let $C(\mathbf{F})$ denote the centroid of feature vectors of a movement, which consists of two vectors $C(\mathbf{F}_p)$ and $C(\mathbf{F}_r)$, where $C(\mathbf{F}_p)$ denotes the centroid of pitch vectors of the movement and $C(\mathbf{F}_r)$ denotes the centroid of roll vectors of the movement. Suppose there are a total of N samples for this movement; the centroid can be calculated as follows.

$$C\left(\mathbf{F}_p\right) = \frac{\sum \mathbf{F}_p}{N}, \qquad C\left(\mathbf{F}_r\right) = \frac{\sum \mathbf{F}_r}{N}. \tag{3}$$

After calculating the centroid for each movement, 110 movements have 110 corresponding centroids in the classifier. Once a new unknown movement comes, the most straightforward way to infer it is to calculate the similarity between its feature vector and each centroid in the classifier and the most similar one will be identified as the true movement. However, this straightforward way is not effective or accurate since there exist many similar movements that results in similar grip changes (e.g., "8-2" and "4-1"). It is difficult to accurately decide the true movement just by calculating the similarity to the centroid of each movement.

In order to reduce the influence of similar movements, we build the classifier in another way. Note that the movements starting from the same button to different buttons usually are not similar movements. Thus, we classify all the centroids into 11 categories according to the starting positions, and each category has 10 centroids for the corresponding 10 movements. Figure 9 shows the 10 centroids for the corresponding movements of the category starting from button "1." The ordinate is the range of normalization. The red solid line and the blue dashed line denote the roll vector and the pitch vector of the centroid, respectively. The centroids within one category are different from each other since they are not similar movements. Thus, if we know the starting position, we can achieve a high accuracy on the movement inference. However, this inference relies on the starting position, which may lead to cumulative errors. To solve this problem, several candidates for each starting position are taken into consideration during movement inference.

4.4. Movement Identification. In the classifier, 110 centroids of movements are divided into 11 categories. For a password sequence with four movements, we infer each movement separately by calculating the similarity between each unknown movement and the centroids of movements in each category.

First Movement Inference. First movement inference is indeed the first tapped position inference. Instead of matching the first unknown movement to all 110 centroids of movements, it will only be matched with 10 centroids at the specific category starting from the air. The smaller matching cost to a centroid, the larger similarity probability it has. Since movement inference severely depends on the starting position, we choose several movements with smallest matching costs rather than only one as the possible candidates to the unknown movement. The ending position of each candidate to the unknown movement will be regarded as a candidate for the first tapped position, which can further decide which category the following unknown movement belongs to.

Following Movements Inference. The following movements inference is similar to the first movement inference. The only difference is that the unknown movement will be matched with centroids at several categories since there may exist several possible candidates for the starting position. Each category is corresponding to one possible candidate for the starting position. Similarly, several movements with smallest matching costs are chosen by the possible candidates to the unknown movement, and the ending position of each candidate is regarded as the starting position to decide which category the next unknown movement belongs to.

Given one unknown movement, suppose μ possible candidates are selected for it. Let φ_i denote the matching cost between the unknown movement and the i_{th} candidate. The similarity probability of the unknown movement to the i_{th} candidate is calculated as $(1/\varphi_i)/(1/\varphi_1 + \cdots + 1/\varphi_\mu)$. That is, the smaller matching cost, the larger similarity probability.

Matching Cost Calculation. Let $\mathbf{U} = [\mathbf{U}_p; \mathbf{U}_r]$ denote the feature vector of an unknown movement consisting of a pitch vector $\mathbf{U}_p$ and a roll vector $\mathbf{U}_r$. We want to calculate the matching cost between $\mathbf{U}$ and each centroid $C(\mathbf{F})$ of movements. Note that the number of elements in $\mathbf{U}_p$ or $\mathbf{U}_r$ may not be 30, since movements take different durations and the sampling rate is fixed with Hz. Let m denote the actual number of readings in the pitch and roll vectors. We do not

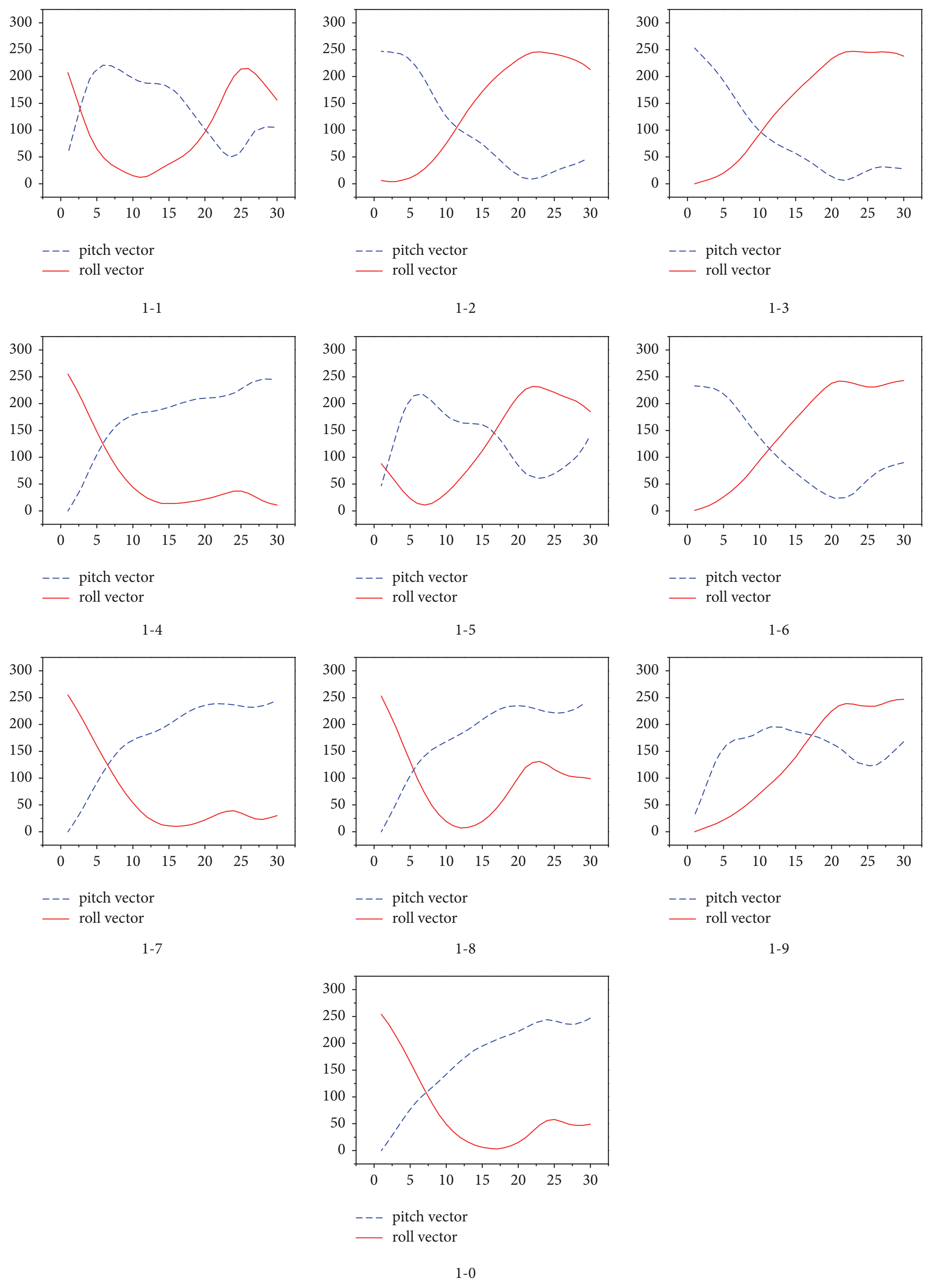


FIGURE 9: The centroids for the category starting at button "1" and ending at different buttons.

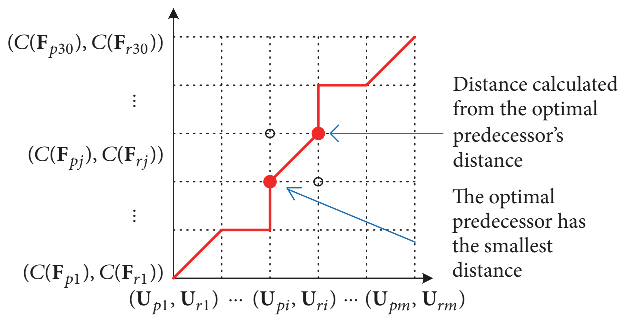

$(C(\mathbf{F}_{p30}), C(\mathbf{F}_{r30}))$
$\vdots$
$(C(\mathbf{F}_{pj}), C(\mathbf{F}_{rj}))$
$\vdots$
$(C(\mathbf{F}_{p1}), C(\mathbf{F}_{r1}))$
$(\mathbf{U}_{p1}, \mathbf{U}_{r1}) \cdots (\mathbf{U}_{pi}, \mathbf{U}_{ri}) \cdots (\mathbf{U}_{pm}, \mathbf{U}_{rm})$
Distance calculated from the optimal predecessor's distance
The optimal predecessor has the smallest distance

Figure 10: Illustration of matching cost calculation with the DTW algorithm.

scale $\mathbf{U}_p$ or $\mathbf{U}_r$ to the size of 30 elements for keeping data integrity as much as possible.

Figure 10 shows an example of calculating the matching cost of two vectors. The y-axis denotes the values of pitch and roll angles in $C(\mathbf{F})$, and the x-axis denotes the values of pitch and roll angles in $\mathbf{U}$. The matching cost is the length of the shortest path from coordinate $(1, 1)$ to coordinate $(m, 30)$. Euclidean distance is used to calculate the distance from one coordinate to another. Note that the matching cost exhibits suboptimal substructure. Therefore, we adopt the DTW algorithm [16, 31] which employs dynamic programming to calculate the matching cost.

Figure 11 shows an example of movement inference when one candidate position of the starting button is "1." The unknown movement will be matched with all the centroids in category starting from button "1." As shown in Figure 11, the red number on each button is the matching cost and smaller cost means higher similarity. The movement ending at button "7" has the smallest matching cost and is regarded as the most likely candidate. But the buttons in this layout are very close to each other, such that buttons "4," "8," and "0" are neighboring buttons. If we only select the label with the smallest cost as the candidate, the true movement may not be covered. The costs of movements ending in "4" and "0" are also much smaller than the other buttons. Therefore, we select three movements ending at "7," "4," and "0" as the candidates for the unknown movement, and their similarity probability can be calculated accordingly.

4.5. Password Inference. We regard password unlocking as a typical Markov process as the transition probability to the next button is only related to the current button. People tend to use some personal and important information as the password (e.g., birthday) [32]. By taking the side information [32] of users into consideration, we use a Markov chain to model the unlocking process and infer the most possible candidates for each password.

Figure 12 shows the Markov model for the 4-digit password. It contains 5 layers, where layer 0 represents the air and the other layers each correspond to one tapping position on the screen. Since there are 10 numbers ranging from 0 to 9 on the screen, there are 10 possible positions at each layer. As shown in Figure 12, a path from a position at layer 0 to a position at layer 4 represents a sequence of movements for a 4-digit password.

Let ℓ_k^i denote the button i at the kth layer. A movement "$i - j$" from kth layer to the next layer can be represented by (ℓ_k^i, ℓ_{k+1}^j), which is a directed line on the model. Let $T(\ell_k^i, \ell_{k+1}^j)$ denote the transition probability on the directed line, which is equivalent to the similarity probability of the unknown movement to this movement. That is,

$$T\left(\ell_k^i, \ell_{k+1}^j\right) = p\left(\ell_k^i, \ell_{k+1}^j\right), \tag{4}$$

where $p(\ell_k^i, \ell_{k+1}^j)$ denotes the similarity probability between the unknown movement and movement "$i - j$."

The probability of a path is calculated as the multiplication of the transition probability of each movement on the path.

$$P = \prod_{k=0}^{3} T\left(\ell_k^i, \ell_{k+1}^j\right). \tag{5}$$

For example, the probability of a sequence "4-8-2-6" is calculated as $P(4826) = T(\ell_0^4)T(\ell_1^4, \ell_2^8)T(\ell_2^8, \ell_3^2)T(\ell_3^2, \ell_4^6)$, as shown in Figure 12. There are 10^4 possible paths on the graph, but it is not necessary to calculate the probability of each path. Meanwhile, as we mentioned in movement inference, if we only consider one movement at each layer, it may result in cumulative errors due to inaccurate starting position.

In our system, we always use multiple candidates rather than one for each unknown movement to avoid the error due to inaccurate starting position. For an unknown movement, two centroids with the largest similarity probability to it will be selected as the candidates. In particular, we select 3 possible candidates for the first unknown movement considering its ending position is the first tapped position, in order to ensure that the true beginning position is covered. The experimental results in the evaluation section show that the accuracy of the first tapped position inference can reach 97% with three attempts. In this case, there are a total of 24 possible paths left on the graph for a 4-digit password and their probabilities can be calculated accordingly. Finally, the possible paths will be ranked in terms of the probability. We can try the sequence on each path one by one to see whether it is the true password.

5. Evaluation

In this section, we evaluate the performance of Niffler with real experiments in both user-dependent and user-independent environments. The former evaluation will answer the question of *how Niffler performs if an attacker can get sufficient samples from victim's device for training*, and the latter evaluation will answer the question of *how Niffler performs without victims' samples for training, which is the scalability of Niffler*.

In this paper, we compare Niffler with the native random guess method. Although several password inference models have been proposed, it is difficult to compare with them due to the following reasons. First, it is difficult to implement

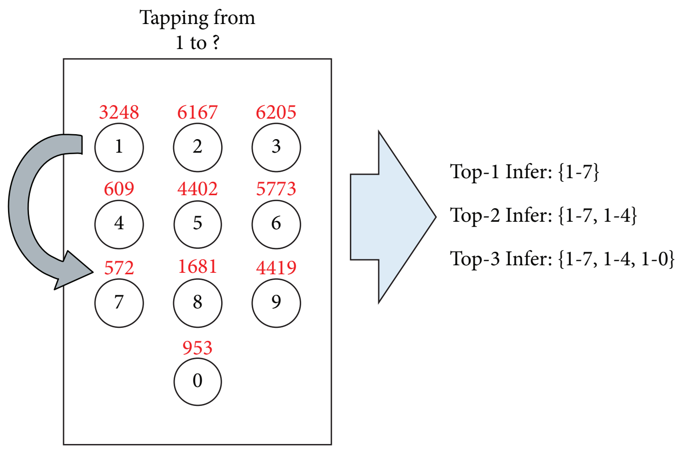


Figure 11: Example of movement inference when the starting position is "1."

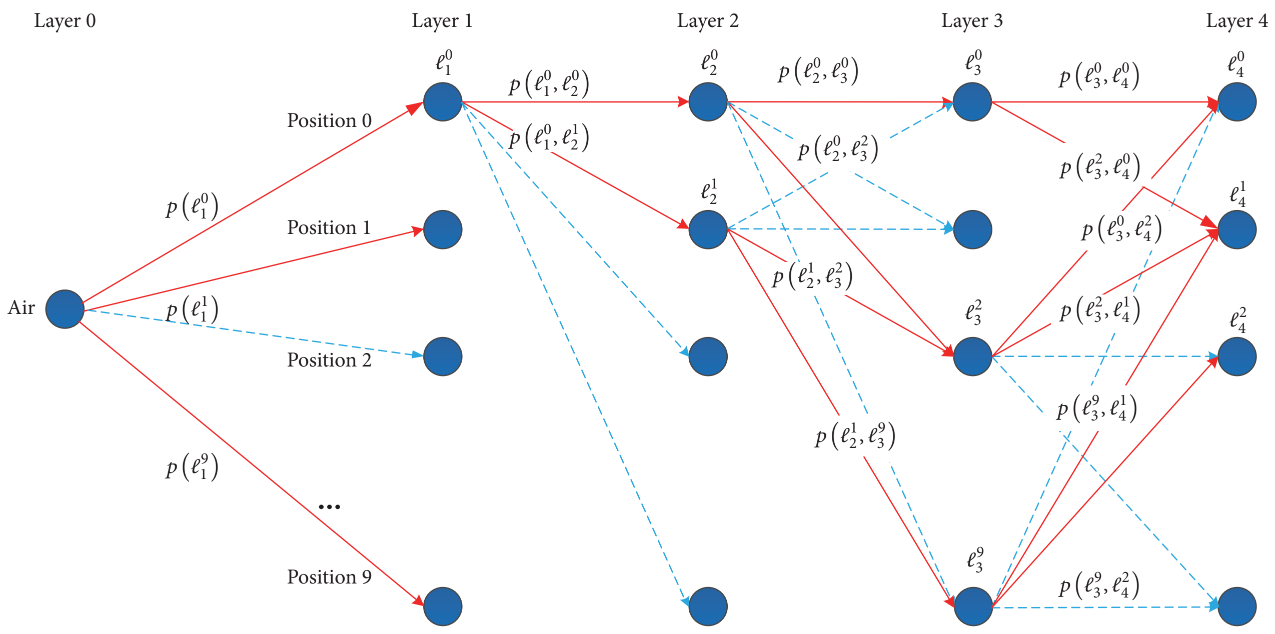


Figure 12: Illustration of password inference with the Markov model.

some of the previous works [10, 14] in real systems as their assumptions are strong. For example, they assume that an attacker can collect sufficient samples from a target user for training purpose, which is actually not realistic for a real attack. In contrast, Niffler is a general side-channel attack model without these assumptions. Second, some other models like [11, 12] demand very high precision in features extraction which are difficult to be extracted in practice due to the large noise in data with high sampling frequency. This also implies that these models are sensitive to noise and are not effective in real attacks.

5.1. Data Collection. We recruit 25 participants to collect samples for unlocking processes. There are 13 males and 12 females, and all of them are between 18 and 30 years old. They can try as many passwords as they want and the ground truth data are recorded as samples. The passwords they used are well designed: some of them are random and some of them are designed with side information. In addition, we also consider the extreme cases, such as "1111" and "9999." The participants are asked to hold and unlock their smartphone with left hand. As shown in Table 1, three types of smartphones with different mainstream screen sizes are used in our experiments. The sampling frequency is about 50 Hz. All participants are asked to keep their body static when they unlock smartphones. Finally, more than 100,000

Table 1: Device list.

Model	Screen size	CPU	Sample rate	Android
Meizu Mx5	5.5 inches	Helio X10	~50 HZ	V5.0
Nexus 4	4.7 inches	APQ8064	~50 HZ	V4.1
OPPO R8	5.2 inches	Cortex-A53	~50 HZ	V4.4

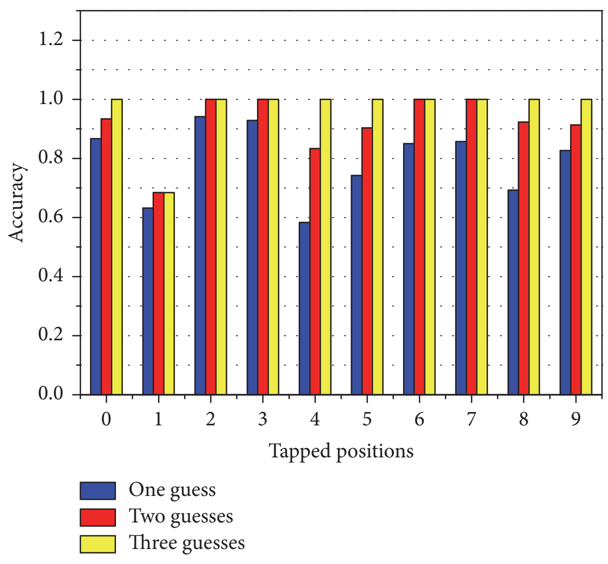


Figure 13: Inference accuracy of first tapped position in user-dependent environment.

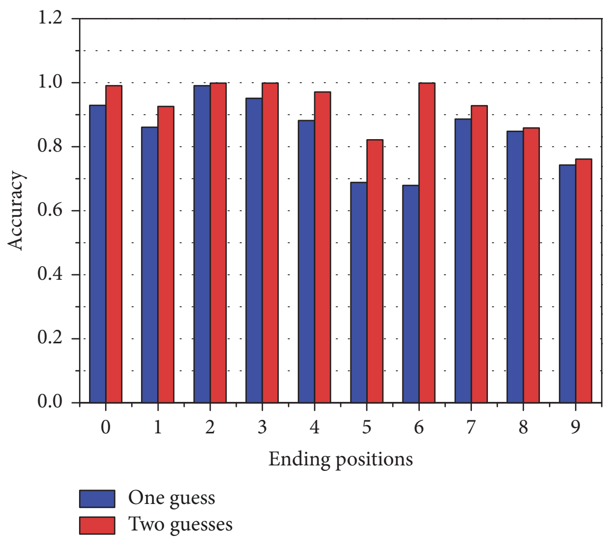


Figure 14: Inference accuracy of category “1” in user-dependent environment.

samples were collected over two months, which can be used for training and testing purposes.

5.2. Evaluation in User-Dependent Environments. The data used in this experiment were collected from 15 participants. Each of them provided a dataset containing 3,500 samples for training and another 300 samples for testing. We first evaluate the accuracy of identifying the movements starting in the air, because it is different from other movements and is very important for the following movements inference. We then present the identification of movements of the category “1” as an example. Finally, we show the results of whole password inference.

Figure 13 shows the inference accuracy of the first tapped position inference which is defined as the number of successful inferences divided by the total number of samples. With one guess, the inference accuracy of each position ranges from 58.33% to 94.12% and the average inference accuracy of ten positions is about 79.19% which is much higher than random guess (10%). The inference accuracy for each starting position increases with more number of guesses and can reach 97% with three attempts. This implies that 3 candidates with the largest similarity probability can cover the true first tapped position with almost 100% guarantee, so we use 3 candidates for the first tapped position inference during password inference.

Figure 14 shows the inference accuracy of movements at category “1.” The x-axis refers to the ending positions of the movements starting at button “1.” With one attempt, the average inference accuracy is about 84.58% which is much higher than TouchLogger 71.5% [12]. With one more attempt, the inference accuracy increases and ranges from 76.15% to 100%.

Figure 15 shows the average inference accuracy of movements in each category, except for the movements starting in the air. The x-axis refers to each category. We can see that the average inference accuracy is more than 80% for all categories with only one attempt, and the average accuracy reaches 90% given one more attempt. This implies that our inference model is robust to different movements and also implies that 2 candidates with the largest similarity probabilities can cover the true movement with more than 90% probability in user-dependent environments.

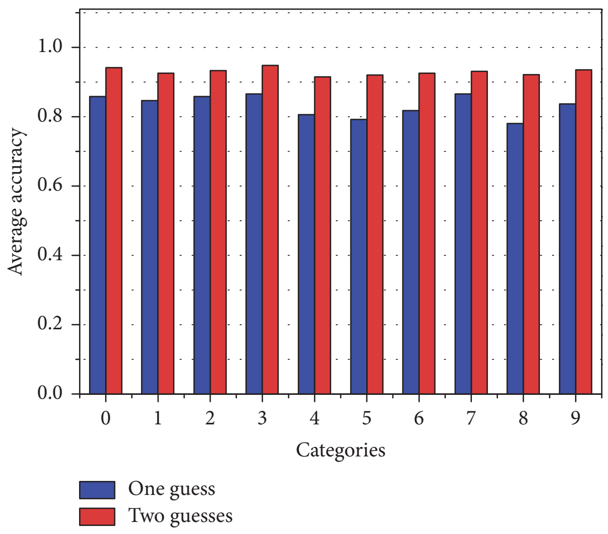


Figure 15: Average accuracy of 10 categories’ movement inference in user-dependent environment.

We present a comparison in terms of the experimental results among our work and typical side-channel attacks exploited motion sensors in Table 2. In our experimental environment, Niffler seams to achieve the highest average accuracy of 84.6% for one key inference on soft keyboards. But, for the inference of a movement depends on the result of last inference, the accumulated error would increase and

Table 2: Comparison with previous works that exploited motion sensors.

Source study	Sensor	Platform	Accuracy
Niffler	Accelerometer	Android	84.6% (72%)
Owusu et al. [8]	Accelerometer	Android	63%
Mehrnezhad et al. [18]	Accelerometer Linear accelerometer Gyroscope Orientation	Android	69%
Mehrnezhad et al. [9]	Accelerometer Orientation	Android	65%
Ping et al. [33]	Accelerometer Gyroscope	Android	75%
Miluzzo et al. [26]	Accelerometer Gyroscope	Android and iOS	72%
Xu et al. [11]	Accelerometer Gyroscope	Android	33.4%

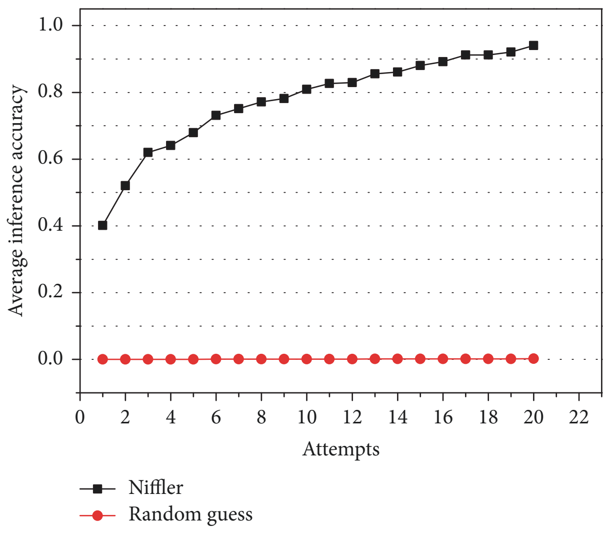


Figure 16: Average inference accuracy over multiple attempts for inferring 4-digit password in user-dependent environment.

Niffler obtains inference accuracy about 72% in effect. Nevertheless, Niffler still performs better than several previous works [8, 9, 11, 18] in user-dependent environment (the data of comparison experiment is collected by our participants, please contact us if you have any questions).

Figures 13–15 show the inference accuracy when the starting position is known to us. We then show the inference of the whole password without knowing the starting position. We ask each one of the five participants to randomly type 25 4-digit passwords. We use Niffler to infer the possible candidates for each password. We try each candidate one by one from the highest probability to the smallest and see whether it is the true password. Once a candidate matches the password, it is a successful inference.

As shown in Figure 16, we can see that the accuracy of password inference increases with more number of attempts. The inference accuracy is about 40% even with only one attempt and can reach 70% with 5 attempts, while random guess only has 0.1%. For a smartphone with Android operation system, the inference accuracy is about 95% with 20 attempts, which means that we can unlock smartphones with 95% successful rate. Therefore, the results indicate that our system based on grip changes achieves a very good attack performance.

5.3. Evaluation in User-Independent Environments. In this section, we evaluate the performance of Niffler in user-independent environments. We collected 50,000 samples in total from 10 participants and mix these data into one dataset to train the classifier. The testing samples are collected from another 5 participants.

Figure 17 shows the inference accuracy for category "1." The x-axis refers to the ending positions of the movements starting at button "1." With one attempt, the average inference accuracy is about 64.26% which is smaller than the average inference accuracy in user-dependent experiments (84.58%). The predication accuracy increases with more attempts, and the accuracy difference between user-dependent and user-independent becomes smaller with more attempts. The average inference accuracy reaches 90% with two attempts, which indicates that 2 candidates with the largest similarity probabilities also can cover the true movement with more than 90% probability in user-independent environments. That is why we only consider 2 possible candidates for each movement during password inference.

Figure 18 shows the inference of the whole password without knowing the starting position in user-independent environments. We can see that the accuracy of password inference increases with more number of attempts. The inference accuracy with one attempt is about 23%, which is smaller than 40% in user-dependent environments, but it is much higher than random guess. The accuracy reaches 55% with 5 attempts, which means there is a half probability we

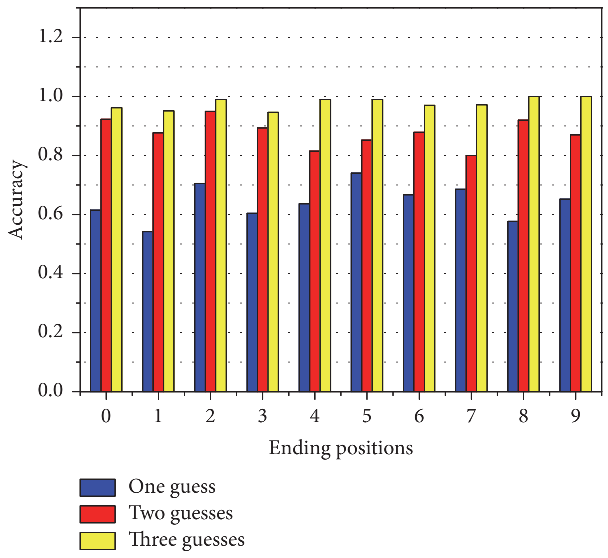


Figure 17: Inference accuracy for category "1" in user-independent environment.

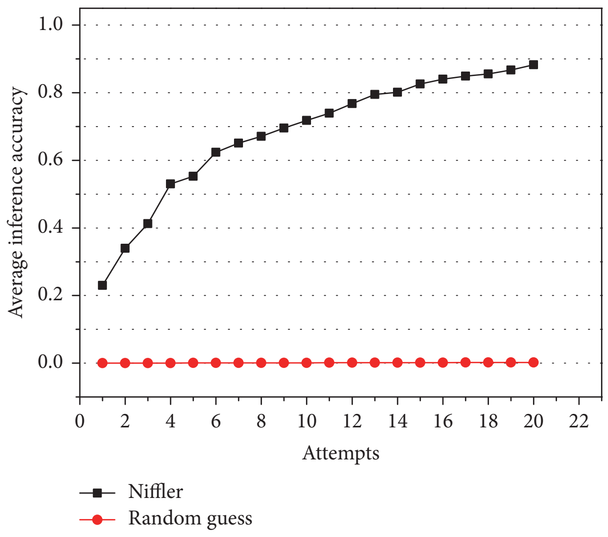


Figure 18: Average inference accuracy over multiple attempts for inferring 4-digit password in user-independent environment.

can unlock screen with 5 attempts. For a smartphone with Android operation system, the inference accuracy is about 74% with 10 attempts, which is close to the accuracy in user-dependent environments. We can conclude that the attack model is more effective in user-dependent environments and can also achieve a high inference accuracy in user-independent environments.

Table 3 shows the results of comparison with some other motion sensor based side channels in user-independent environment. The second column expresses the average inference accuracy for single typed position with one attempt based on our data set. Compared with Table 2, the performances of all of the attacks turn down sharply in user-independent environment, but Niffler still achieves the highest accuracy with 65% even considering accumulated error (42.3%).

Table 3: Comparison with previous works that exploited motion sensors in user-independent environment.

Source study	Accuracy
Niffler	65% (42.3%)
Owusu et al. [8]	40.9%
Mehrnezhad et al. [18]	30.2%
Mehrnezhad et al. [9]	34.8%
Ping et al. [33]	35.9%
Miluzzo et al. [26]	31.4%
Xu et al. [11]	29.6%

Summary. We conduct experiments to evaluate the performance of Niffler in both user-dependent and user-independent environments. The results show that Niffler achieves better performance in user-dependent environments since the testing data and the training date are from the same source. However, Niffler also achieves much better performance in user-independent environments than other side channels. It can achieve about 90% inference accuracy for movement inference with three attempts and 55% inference accuracy for the whole password sequence with 5 attempts. The inference accuracy increases with more attempts and can reach 74% with 10 attempts. We can see that our attack model based on grip changes can achieve a good inference accuracy and is user-independent.

5.4. Evaluation on Influence Factors

The Size of Training Samples. In the experiments above, each centroid is trained with about 100 samples. One question is: *How would our model perform as the number of training samples increases?* That is, we are interested in the performance of the attack model with more training samples. We first construct the classifier with 1,000 samples for each and increase the number of training samples to double size and triple size. We compare the average accuracy with one attempt, two attempts, and three attempts, respectively. All of the samples were collected from the participants with similar hand size and holding habit.

Figure 19 shows the inference accuracy under different sizes of training samples for one, two, and three attempts, respectively. The y-axis is the average inference accuracy and the x-axis is the ID of categories. As shown in Figure 19(a), the inference accuracy increases for most categories with more training samples. However, this is not true for some categories (e.g., categories 2, 3, 4, and 6), which may be because the added samples contain too many abnormal samples that influence the inference. We can also observe that the inference accuracy is better with more attempts, and most of them are larger than 90%. Therefore, the results suggest that the performance would be better with more training samples. The results suggest that Niffler performs better with more samples for training. Meanwhile, it can achieve a good attack performance even with limited number of samples.

Extreme Case. We also evaluate the performance of Niffler for extreme cases. In particular, ten special cases are evaluated

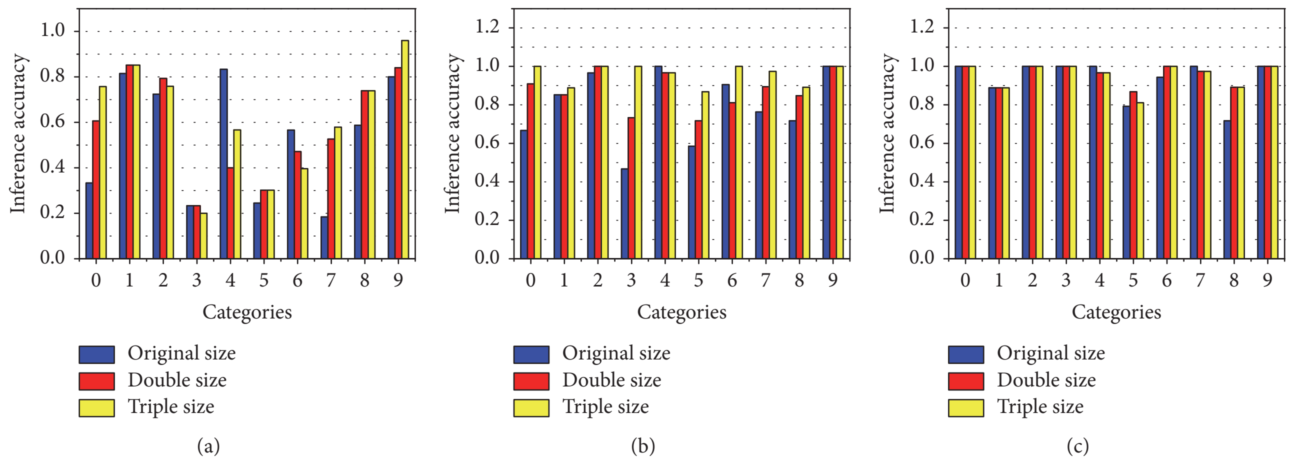


Figure 19: The inference accuracy under different sizes of training samples for (a) one attempt, (b) two attempts, and (c) three attempts, respectively.

Table 4: The impact of hand size on movement identification.

Approximate ratio	Average accuracy
0.93	77.2%
0.96	82.4%
0.98	91.1%
1.01	88.3%
1.04	84.3%
1.07	81.1%

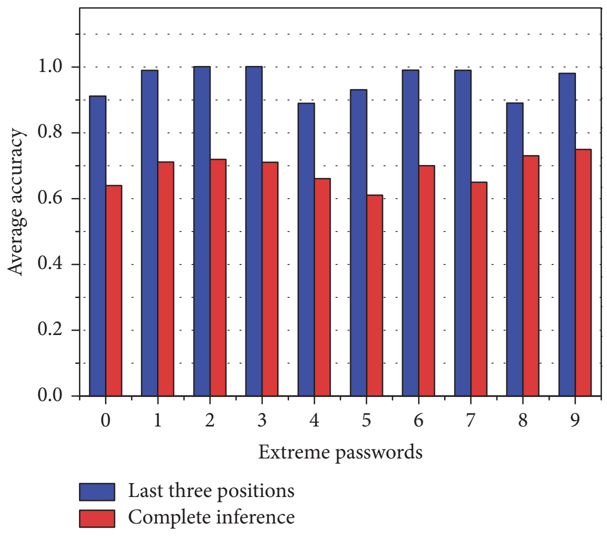


Figure 20: The average accuracy for extreme passwords within 10 attempts.

from "0000" to "9999." We consider two scenarios: the first tapped position is known to us and we infer the rest 3 positions, and none of the passwords are known to us and we infer the whole password. Figure 20 shows the average accuracy for the two scenarios with 10 attempts.

We can observe that the accuracy of password inference is better if the first tapped position is known to us, which is about 25% higher. The accuracy of password inference is larger than 90% if the first tapped position is known, while it is about 65% if not. Therefore, the performance of Niffler highly depends on the inference of the first tapped position for extreme cases. It is worth noting that Niffler performs better for general passwords than passwords in extreme cases, which can be observed from Figures 18 and 20. This is because the passwords in extreme cases are too extreme that it is difficult to leverage other methods, such as side information or screening, to improve the inference accuracy.

Hand Size. The basic idea of Niffler is based on the grip change of a smartphone during user holding and tapping. The size of hands might affect the performance of Niffler. In our experiments, we also evaluate the performance of Niffler under different sizes of hands. Table 4 shows the results of the impact of hand size on movement identification. The right column is the average movement identification accuracy with two attempts, and the left column is the approximate ratio between the size of testing sample providers' hands and that of training sample providers' hands. We found that users with similar hand size produce similar grip changes for the same movement, but the difference of grip changes among different sizes of hands is not obvious. Therefore, Niffler is a general attack model that is robust to the sizes of hands.

5.5. Evaluation on Power Consumption. We also evaluate the power consumption of the monitoring module of Niffler. Power consumption is an important factor [34] for successful attack since a high power-consuming service will be more possible to attract the user's attention and be killed [35]. It is difficult to accurately measure the consumption of the monitoring module. In our experiments, we compare the power consumption of the monitoring module of Niffler with several common applications and services including a preinstalled music player, QQ, a system service, a system

Table 5: Power consumption comparison when the monitoring module is only in monitoring state.

Application	Run time	Consumption
Music player	1 h	14.0%
QQ	1 h	7.2%
Monitoring module of Niffler	1 h	5.3%
System service	1 h	7.2%
System interface	1 h	5.6%
Input method	1 h	1.3%

Table 6: Power consumption comparison when the monitoring module is listening to accelerometer readings.

Application	Run time	Consumption
Music player	1 h	21.3%
QQ	1 h	5.3%
Monitoring module of Niffler	1 h	13.6%
System service	1 h	4.0%
System interface	1 h	3.2%
Input method	1 h	1.0%

interface, and SOGOU input method. Three participants were asked to do the same experiments on the same Meizu Mx5 mobile phone. For each participant, we first asked him to use the phone for 1 hour. During the 1 hour, all of the tested applications or services were kept running in the background. Specifically, the music player was in active state but did not play any music, and the monitoring module of Niffler did nothing except waiting for the unlocking process. After that, each participant was asked to use the phone for 1 hour. During this 1 hour, the music player started to play music for another 1 hour, and the monitoring module registered *Sensor Listener* and kept listening to accelerometer readings.

Table 5 shows the comparison of power consumption of these services for the first 1 hour, and Table 6 shows the comparison for the other 1 hour. As shown in Table 5, we can see that the monitoring module of Niffler has almost the same power consumption as QQ and is much less than the music player which did not play music. As shown in Table 6, the power consumption of Niffler when it is listening to accelerometer readings is about half of that of music player in playing mode but is also double that of QQ. Note that the monitoring module was kept listening to accelerometer readings for 1 hour, but in real system it only lasts for a few seconds. Therefore, we can conclude that Niffler is not power-consuming which would not significantly attract users' attentions.

6. Related Work

6.1. Side-Channel Attacks. Many kinds of side channels have been used in mobile attacks. Free floating (FF) windows in Android have been used to infer a user's input by Ying et al. [30]. The attackers can pass sandbox mechanism and get time intervals between two nearby keystrokes with a well-designed FF windows application to guess input sequences. Aviv et al. cracked graphical password about a recent user's input by analyzing oily residues on touch screen [36]. According to their methods, an attacker must approach the victim's device before oily residues being destroyed. It seems to be impractical in real attacks. Similarly, fingerprint is used to construct side-channel attacks [27]. They need the support of additional hardware, such as fingerprint powder, to dust the touch screen to reveal fingerprints left from tapping fingers. Zhang et al. proposed WiPass, a system that breaks Android pattern lock at a high success rate by analyzing how the finger movement affects the WiFi signal while drawing the pattern lock [37]. Ambient sensor can also be used for keylogging attacks. Simon and Anderson [38] presented an attack that exploits an ambient sensor to infer user's PIN input. Narain et al. [39] and Gupta et al. [40] showed that tap sounds recorded via stereo-microphones can be used to infer typed text on the touchscreen.

Vision information has also been studied to construct side-channel attacks recently. Raguram et al. [41] used the reflection of objects to recover the typed information. However, it needs a clear vision of the content displayed on the touch screen to realize accurate recovery. Yue et al. [42] proposed a method to break digit-based passwords by analyzing the shadow formation around the fingertip. Similar work in [43] reconstructed patterns by using reflection even when the adversary is around a corner. Shukla et al. [44] exploited video-based side-channel model to decode PIN entry process which relies on the spatiotemporal dynamics of the hands during typing. It is the first work to reconstruct typed text from a video recording of the hands without using any information about the keys being pressed or the content displayed on the screen. Ye et al. proposed a vision-based attack to crack pattern based lock [45]. They exploited the geometric information exposed by fingertip movement to identify patterns without knowing the console geometry, such as the screen size. Hansch et al. [46] demonstrated that the front camera can be used to capture the screen reflections in the eyeballs, which allows inferring user input.

6.2. Motion Sensor-Based Side-Channel Attacks. With the development of embedded systems and mobile devices, sensors have been widely exploited to detect the motion information [47] and infer passwords of touch-enabled screen devices, including accelerometer, orientation sensor, gyroscopic, and other motion sensors [10, 48]. Cai and Chen proposed *TouchLogger* [12] which uses the gyroscope and the orientation sensor as a side-channel attack to infer the location user tapped. They pointed out that keystroke vibration on touch screens is highly correlated to the keys being typed. This discovery became one of most important bases of sensor-based attacks. *Taplogger* [11], a side-channel model constructed with the gyroscope and the accelerometer data, used similar theory to infer tapped sequences. Both of them focus on the relationship between sensor readings and each tap event, which are too sensitive to data source. Subsequent publications [10, 26] also considered the combination of the accelerometer and the gyroscope in order to improve the performance as well as to infer even longer text inputs [33]. Besides text password, motion sensors are also used to infer pattern password. Andriotis et al. [49] established a scheme, which combines a behavior-based attack and a physical attack on graphical lock screen methods.

Aviv et al. [14] referred to research object as the process of tapping one PIN, instead of individual digits. Their work can reach high accuracy. However, it is almost impossible to build the large corpus containing all possible categories. *ACCessory* [8] proposed by Owusu et al. is similar to above work. The authors demonstrated that the accelerometer can be used as a side-channel attack to infer short sequences of touches on screen and employed standard machine learning techniques to infer input passwords. Our work differs from these pervious methods in that we focus on the movement between nearby keystrokes. Unlike individual tap events or entire process of tapping PINs, we observe that movements contain general characteristics among different users. Using this observation, our side-channel model, Niffler, can be easily implemented without facing previous limitations that training data and inferring data must be collected from the same source.

Noor et al. [50] outlined a machine learning approach modeling the grip changes to predict the resulting touchdown point while the finger is still in the air. They detected physical grip and grip change over time through adding extra hardware (additional sensor arrays on the back or side of the device). Ulteriorly, Negulescu and McGrenere [6] showed that internal motion sensors can achieve similar results in the air predictions of touch points. Similarly, we employ internal sensors to reflect grip changes during user tapping password. Our work differs from Noor et al. and Negulescu and McGrenere in that we discover the general features in grip changes to infer movements between nearby keystrokes and use them to infer the tapped sequences, for example, PIN.

7. Conclusions

In this paper, we design and implement a new sensor-based side-channel attack system, called Niffler, to accurately infer the unlocking passwords on smartphones. Our system is based on the observation that movements during the unlocking process are context-aware and user-independent. We collected 100,000 samples from 25 volunteers over two months for training and testing purposes. Angle feature vectors are extracted to build a multicategory classifier which can further use the DTW algorithm to calculate the similarity between an unknown movement and the centroids in each category. The Markov model is used to model the unlocking process and the paths with the largest probabilities are selected as the candidates for the true password. We conduct experiments in both user-dependent and user-independent environments. The results show that Niffler is a user-independent system and achieves a high accuracy on password inference.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. U1536204, 61502352, 61672394, 61373169, and 61672393), National High Technology Research and Development Program of China (Grant no. 2015AA016004), Natural Science Foundation of Hubei Province and Jiangsu Province (Grants nos. 2017CFB503 and BK20150383), and Fundamental Research Funds for the Central Universities (Grants nos. 2042015kf0016 and 413000035).

References

[1] J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1206–1216, 2015.

[2] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-Preserving Patient-Centric Clinical Decision Support System on Naïve Bayesian Classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 2, pp. 655–668, 2016.

[3] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Friendbook: a semantic-based friend recommendation system for social networks," *IEEE Transactions on Mobile Computing*, 2014.

[4] W. Zhang, H. He, Q. Zhang, and T.-H. Kim, "PhoneProtector: protecting user privacy on the android-based mobile platform," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 282417, 10 pages, 2014.

[5] Z. Wang, R. Tan, J. Hu et al., "Heterogeneous incentive mechanism for time-sensitive and location-dependent crowdsensing networks with random arrivals," *Computer Networks*, vol. 131, no. 2, pp. 96–109, 2018.

[6] M. Negulescu and J. McGrenere, "Grip change as an information side channel for mobile touch interaction," in *Proceedings of the 33rd Annual CHI Conference on Human Factors in Computing Systems, CHI 2015*, pp. 1519–1522, Republic of Korea, April 2015.

[7] A. Nahapetian, "Side-channel attacks on mobile and wearable systems," in *Proceedings of the 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*, pp. 243–247, USA, January 2016.

[8] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: password inference using accelerometers on smartphones," in *Proceedings of the Proceeding of the 13th Workshop on Mobile Computing Systems and Applications (HotMobile '12)*, no. 9, New York, NY, USA, February 2012.

[9] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "Stealing PINs via mobile sensors: actual risk versus user perception," *International Journal of Information Security*, pp. 1–23, 2017.

[10] L. Cai and H. Chen, "On the practicality of motion based keystroke inference attack," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7344, pp. 273–290, 2012.

[11] Z. Xu, K. Bai, and S. Zhu, "TapLogger: inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 113–124, Tucson, Ariz, USA, April 2012.

[12] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion," *HotSec*, vol. 11, p. 9, 2011.

[13] C. Shen, S. Pei, Z. Yang, and X. Guan, "Input extraction via motion-sensor behavior analysis on smartphones," *Computers & Security*, vol. 53, pp. 143–155, 2015.

[14] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12)*, pp. 41–50, ACM, Orlando, Fla, USA, December 2012.

[15] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: user verification on smartphones via tapping behaviors," in *Proceedings of the IEEE 22nd International Conference on Network Protocols (ICNP '14)*, pp. 221–232, Raleigh, NC, USA, October 2014.

[16] A. de Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, pp. 987–996, ACM, May 2012.

[17] A. Das, N. Borisov, and M. Caesar, "Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA.

[18] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of user touch actions and PINs based on mobile sensor data via JavaScript," *Journal of Information Security and Applications*, vol. 26, pp. 23–38, 2016.

[19] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices," *IEEE Communications Surveys & Tutorials*, 2017.

[20] S. Hoober, *How do users really hold mobile devices*, vol. 18, Uxmatters, 2013, http://www.uxmatter.com.

[21] J. Song, "The design of bottom layer sensor interfaces based on android os," in *Proceedings of the International Proceedings of Computer Science and Information Technology*, vol. 58, p. 54, 2012.

[22] X. Liu, K. R. Choo, R. H. Deng, R. Lu, and J. Weng, "Efficient and Privacy-Preserving Outsourced Calculation of Rational Numbers," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 27–39, 2018.

[23] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[24] Z. Huang, S. Liu, X. Mao, K. Chen, and J. Li, "Insight of the protection for data security under selective opening attacks," *Information Sciences*, vol. 412-413, pp. 223–241, 2017.

[25] M. Mohamed, B. Shrestha, and N. Saxena, "SMASheD: Sniffing and Manipulating Android Sensor Data for Offensive Purposes," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 901–913, 2017.

[26] 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 (MobiSys '12)*, pp. 323–336, Ambleside, UK, June 2012.

[27] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, "Fingerprint attack against touch-enabled devices," in *Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM '12)*, pp. 57–68, Raleigh, NC, USA, October 2012.

[28] K.-E. Kim, W. Chang, S.-J. Cho et al., "Hand grip pattern recognition for mobile user interfaces," in *Proceedings of the AAAI*, vol. 21, p. 1789, 2006.

[29] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618," *Genetic Programming and Evolvable Machines*, pp. 1–3, 2017.

[30] L. Ying, Y. Gu, Y. Chengy, P. Su, Y. Lu, and D. Feng, "Attacks and defence on android free floating windows," in *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2016*, pp. 759–770, China, June 2016.

[31] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[32] SplashData, "Announcing our worst passwords of 2015," 2015, https://www.teamsid.com/worst-passwords-2015/.

[33] D. Ping, X. Sun, and B. Mao, "TextLogger: Inferring longer inputs on touch screen using motion sensors," in *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, USA, June 2015.

[34] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3236–3249, 2015.

[35] Y. Liu, C. Xu, and S. C. Cheung, "Where has my battery gone? Finding sensor related energy black holes in smartphone applications," in *Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications, PerCom 2013*, pp. 2–10, USA, March 2013.

[36] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," *WOOT*, vol. 10, pp. 1–7, 2010.

[37] J. Zhang, X. Zheng, Z. Tang et al., "Privacy leakage in mobile sensing: your unlock passwords can be leaked through wireless hotspot functionality," *Mobile Information Systems*, vol. 2016, Article ID 8793025, 14 pages, 2016.

[38] L. Simon and R. Anderson, "PIN skimmer: Inferring PINs through the camera and microphone," in *Proceedings of the 3rd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM 2013, Held in Association with the 20th ACM Conference on Computer and Communications Security, CCS 2013*, pp. 67–78, Germany, November 2013.

[39] S. Narain, A. Sanatinia, and G. Noubir, "Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning," in *Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2014*, pp. 201–212, UK, July 2014.

[40] H. Gupta, S. Sural, V. Atluri, and J. Vaidya, "Deciphering text from touchscreen key taps," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 9766, pp. 3–18, 2016.

[41] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "iSpy: Automatic reconstruction of typed input from compromising reflections," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*, pp. 527–536, Chicago, Ill, USA, October 2011.

[42] Q. Yue, Z. Ling, B. Liu, X. Fu, and W. Zhao, "Blind recognition of touched keys: Attack and countermeasures," https://arxiv.org/abs/1403.4829.

[43] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pp. 1063–1074, Germany, November 2013.

[44] D. Shukla, R. Kumar, V. V. Phoha, and A. Serwadda, "Beware, your hands reveal your secrets!," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 904–917, ACM, Scottsdale, Ariz, USA, November 2014.

[45] G. Ye, Z. Tang, D. Fang et al., "Cracking Android Pattern Lock in Five Attempts," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA.

[46] R. Hansch, T. Fiebig, and J. Krissler, "Security impact of high resolution smartphone cameras," in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[47] Y. J. Lee, "Detection of Movement and Shake Information using Android Sensor," in *Proceedings of the Multimedia 2015*, pp. 52–56.

[48] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11*, pp. 551–562, USA, October 2011.

[49] P. Andriotis, T. Tryfonas, G. Oikonomou, and C. Yildiz, "A pilot study on the security of pattern screen-lock methods and soft side channel attacks," in *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '13)*, pp. 1–6, April 2013.

[50] M. F. M. Noor, A. Ramsay, S. Hughes, S. Rogers, J. Williamson, and R. Murray-Smith, "28 frames later: Predicting screen touches from back-of-device grip changes," in *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, CHI 2014*, pp. 2005–2008, Canada, May 2014.