On the Practicality of Motion Based Keystroke Inference Attack

Liang Cai and Hao Chen

University of California, Davis
lngcai@ucdavis.edu, hchen@cs.ucdavis.edu

Abstract. Recent researches have shown that motion sensors may be used as a side channel to infer keystrokes on the touchscreen of smartphones. However, the practicality of this attack is unclear. For example, does this attack work on different devices, screen dimensions, keyboard layouts, or keyboard types? Does this attack depend on specific users or is it user independent? To answer these questions, we conducted a user study where 21 participants typed a total of 47,814 keystrokes on four different mobile devices in six settings. Our results show that this attack remains effective even though the accuracy is affected by user habits, device dimension, screen orientation, and keyboard layout. On a number-only keyboard, after the attacker tries 81 4-digit PINs, the probability that she has guessed the correct PIN is 65%, which improves the accuracy rate of random guessing by 81 times. Our study also indicates that inference based on the gyroscope is more accurate than that based on the accelerometer. We evaluated two classification techniques in our prototype and found that they are similarly effective.

1 Introduction

Modern mobile devices, such as smartphones and tablets, are equipped with multiple sensors. While these sensors enable exciting new applications, they also pose new security and privacy risks. The risks of some of these sensors are easily understood. For example, when an attacker can access the microphone, camera, or GPS, she can eavesdrop on the sound, image, and location of the user [5,24,23]. Therefore, most mobile platforms protect these sensors by requiring access permissions to these sensors. By contrast, the security risks of motion sensors, such as the accelerometer and gyroscope, are not as well understood. For example, applications need no permission to access motion sensors on Android. As another example, W3C's *DeviceOrientation* Event Specification [19] allows any web application to access the accelerometer and gyroscope, which was adopted by both Android since version 3.0 and iOS since version 4.2.

However, recent researches have shown that motion sensors can leak sensitive information [4,17]. The attacker may use motion sensors as a side channel to infer keystrokes typed on on-screen keyboards, which may help the attacker recover important information about the user, such as his passwords or credit

card numbers. This *motion-based keystroke inference attack* is based on the observation that device vibration during a keystroke is correlated to the key typed. Although previous studies showed that motion sensors leak information about keystrokes, they have yet to demonstrate the practicality of this attack. Those studies were based on a single smartphone and a few users. However, for this attack to be practical, we must evaluate whether it is robust against:

- **Hardware variation:** Different devices may use different sensor chips, which may have different sampling rates and precisions. Also the motion sensors may be embedded at different locations on the mobile devices. Does this attack work on different devices?
- **Dimension variation:** Previous work studied only smartphones. Lately, larger devices, such as tablets, are becoming popular. Does this attack work better or worse on these larger devices?
- **Keyboard layout variation:** Device vibration during a keystroke is correlated to the location of the key, which is determined by both the key and the keyboard layout. Furthermore, keyboard layout often affects how the user holds the device and types. During our experiment, we observed that on regular keyboards in portrait mode, users usually held the device in one hand and typed with fingers in the other hand; however, on split keyboards in landscape mode, users usually held the device using both hands and typed with both thumbs. Does this attack work on different keyboard layouts?
- **User variation:** Device vibration during keystrokes may depend on the user's typing style, such as the force of her finger, the tilt angle of the device, and anchor of her holding hand on the device. Does this attack work on different users?

Besides the above questions regarding the robustness of this attack, a successful attack must address the following questions in its design and implementation:

- **Extracting keystroke-relevant signal from motion sensor data.** Although the attacker, through his malware installed on the victim device, can read from the motion sensors, he does not know when a keystroke starts and ends in the continuous data stream. To recognize keystrokes, he must divide the continuous sensor data stream into segments for each keystroke.
- **Selecting the motion sensor.** A device may have multiple motion sensors, such as an accelerometer and a gyroscope. Previous work studied only the accelerometer, but a shrewd attacker would choose the sensor that provides the best results.
- **Selecting the inference techniques.** Multiple techniques exist for inferring keystrokes based on device motion. No previous study compared the alternative techniques, but a shrewd attacker would choose the best technique.

To answer the above questions and to evaluate the practicality of the motion-based keystroke inference attack, we conducted a user study where 21 participants typed on four different mobile devices consisting of two smartphones and

two tablets. We asked each participant to type in each of six settings to evaluate various factors affecting the attack and collected a total of 47,814 keystrokes. We developed a prototype attack and applied the attack on the collected keystrokes. We evaluated how variations in hardware, device dimension, keyboard layout, and user habit affect the attack. To make the attack more effective, we investigated how to extract data segments representing keystrokes from a continuous stream of motion sensor data, the difference between different motion sensors, and the difference between two classifiers. Our evaluation shows that on a number-only keyboard, after the attacker tries 81 4-digit PINs, the probability that she has guessed the correct PIN is 65%, which improves the accuracy rate of the random guessing attack by 81 times.

2 Background

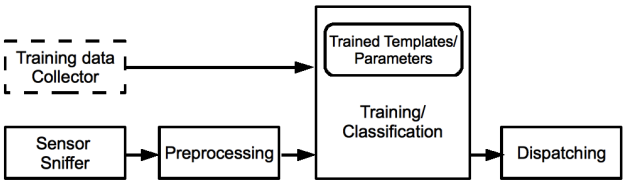


Fig. 1: Architecture of a motion based keystroke inference app

2.1 Motion based keystroke inference attack

Keyboards are the most common input device. We use keyboards to input a variety of information, some of which is highly valuable, such as passwords, PINs, social security numbers, and credit card numbers. It came as no surprise that keystroke logging [1] is a favorite tool of the trade by attackers. The attacker can install a Trojan program on the victim computer to log keystrokes, or use out-of-band channels to infer keystrokes. An acoustic key logger, for example, can infer keystrokes from acoustic frequency signatures [2], timings between two keystrokes [8], or language models [25]. Electromagnetic emanations of keyboards have also been studied for keylogging [21].

Touch screen mobile devices have changed the paradigm of user interaction. Most touch screen mobile devices have no physical keyboard. Instead, the user types on the software keyboard on the screen. Since there is neither sound nor electromagnetic emanation from a virtual keyboard, the attacker can no longer infer keystrokes based on these signals. Moreover, mobile operating systems, such as Android and iOS, have security design that thwarts Trojan based keyloggers. For instance, on the Android platform, each app runs in its own Linux process and is assigned a unique user ID. An application cannot read keystrokes unless it is active and receives the focus on the screen. In most cases, it seems that key

loggers, at least the traditional ones described above, face severe obstacles on touch screen mobile devices.

However, a new approach for keystroke logging on touch screen smart phones has been recently proposed in [4,17]. The new attack exploits the output of motion sensors, such as accelerometers, to infer keystrokes. When the user types on the soft keyboard on her smartphone (especially when she holds her phone by hand rather than placing it on a fixed surface) it causes slight phone vibrations, which can be detected by motion sensors. The keystroke induced vibration on touch screens is correlated with the location of keys being typed. This can be observed from the shifting reflection of distant objects on the device screen when we type. It is possible to estimate the approximate location where a user's finger hit the screen by analyzing the output of motion sensors. Given the keyboard layout is known, it is then straightforward to infer the keystroke value from the location.

A keystroke inference malware should have at least four components– sensor sniffer, preprocessor, classifier and dispatcher, as shown in Figure 1. The sniffer reads the motion sensor output from a background process. After preprocessing, the sensor data is sent to the classifier, which extracts features and maps the data to a keystroke value. The dispatcher eventually sends the inference result to a remote server controlled by the attacker. If the attack is user dependent, the attacking program should also contain a component to collect a certain amount of training data (sensor data labeled with key values). The training data set is sent to generate templates or parameters used in classification.

2.2 Motion sensor data

There are two possible hardware motion sensors available on mobile devices: accelerometer and gyroscope. Accelerometers are already widely adopted by mobile devices. Recently, a gyroscope has been integrated in a number of smartphones and tablets to allow for more accurate recognition of rotating movement within a 3D space. The device movement caused by keystroke is the combination of both shift and rotation. However, since we have observed that the rotation is more related to the key locations, precisely capturing the device rotation is of more interests in keystroke inference. In general it is believed that the accelerometer is designed for recognizing the linear shift component of device movement and the gyroscope is better at recognizing device rotation. But the reality is both of them can detect rotation. With a fixed reference from gravity, accelerometers provide a better measurement tracking pitch and roll when the device is not moving. Gyros provide a higher accuracy when the device is in motion [15]. In this paper, we compared the keystroke inference results based both on accelerometer and gyroscope.

Another important specification of motion sensor data is the sampling rate. Unlike audio input, the motion data is not sampled in a fixed rate. Instead, all of the motion sensors return multi-dimensional arrays of sensor values in terms of sensor events, i.e. new sensor values are reported only when they are different from those reported in the previous event. We list the average and standard

Device	Accelerometer		Gyroscope	
	average (ms)	stdev	average (ms)	stdev
Google Nexus S	20.07	0.77	1.18	0.11
HTC Evo 4G	22.04	1.93	n/a	n/a
Galaxy Tab 10.1	10.10	0.23	10.10	0.23
Motorola Xoom	10.05	0.36	1.15	0.18

Table 1: Interval of motion sensor output for difference devices

deviation of intervals in motion sensor data from different devices we used for evaluation in Table 1.

Mobile platforms allow applications to specify different data delays when reading motion sensor output to trade off between efficiency and accuracy. In this paper we focus mainly on the inference rate rather than the efficiency so that the sensor data delay has been always set to zero.

3 Related works

Previous research [5] has raised the awareness of privacy attacks on smartphone sensors. Besides the obvious privacy concern over the GPS sensor, researchers have shown attacks using the camera [23] and microphone [24]. These attacks are less insidious because these sensors are protected with access permission by mobile operating systems.

Researchers have studied keystroke inference based on side channels, such as sound [2,25], electromaganetic wave [21], and timing [20]. Since these attacks exploit characteristics of physical keyboards, they become ineffective on smartphones with soft keyboards.

Applications exploiting motion sensors have been extensively researched. Most of these works focus on human activity or gesture recognition. Activity recognition is an important topic in the area of pervasive computing. Researchers have proposed schemes to detect user's activity in choreography [3], food preparation [18], and in medical research [14]. In [11], Lester, etc. tried to determine whether two devices are worn by the same person based on motion signals. The main application for gesture recognition is user interaction [12,6,10,22]. Some also use it for authentication[16,7]. In [13], users can authenticate two devices by attaching them together and shaking. It is also based on accelerometer data. These works use a wide variety of approaches for classification, ranging from frequency domain analysis[11,3,13], Time series analysis [14], Template matching[12,18] to statistical learning[22]. Although their experience on processing motion sensor signal can be borrowed, several major differences between these researches and motion based keystroke inference must be noticed. First, the duration of keystroke induced device movement is much shorter than that caused by a gesture or human activity such as walking or dancing. Second, many of the previous research collect sensor data from a customized devices or Wii remote. The motion sensor signals from these devices usually have constant sampling rate. On a smartphone, the motion data is reported via motion events asynchronously. Fi-

nally, the movement caused by keystrokes are not perceptible as user activity or gesture. For example, it is hard for users to control the magnitude of movement.

This paper is directly related to [4] and [17], as they all focus specifically on motion based keystroke inference attack. However, our paper provides a more thorough investigation on the practicality of such attack. We conducted a user study of many more users, devices and settings, and compared the performance of different classification schemes. To the best of our knowledge, this paper is also the first one to investigate output of gyroscopes on mobile devices.

4 Methodology

4.1 Data acquisition

The attacker can read the motion sensor data through either a web application or an application installed on the victim mobile device. For example, the attacker can embed the code for sniffing the motion sensors in an otherwise legitimate application. Since Android requires no permission for reading motion sensors, these applications are unlikely to raise suspicion.

We record the stream of motion sensor events in a sequence of tuples $(t^i, V^i = \{v_x^i, v_y^i, v_z^i\}), i = 1 \dots N$, where t_i is the time when the i_{th} sensor event occurs, V^i contains sensor reading on three dimensions, and N is the total number of sensor events. For the accelerometer, V_i contains the acceleration force in m/s^2 along the x, y and z axis, respectively. For the gyroscope, V^i contains the rate of rotation in rad/s around the x, y and z axis.

4.2 Preprocessing

De-jittering: Many signal analysis methods require constant-interval sampling. However, motion sensors usually do not generate new events until the reading has changed (Table 1). Therefore, we dejitterize the motion events as follows:

1. Calculate the average interval Δ of sensor events in each stream.
2. For any event $e^i = \{t^i, V^i\}$, if $t^i - t^{i-1} > \frac{4}{3}\Delta$, we insert M events evenly between e^{i-1} and e^i such that $\frac{2}{3}\Delta \leq \frac{t^i - t^{i-1}}{M+1} < \frac{4}{3}\Delta$. We set the sensor values in all these new events to be equal to that in V^{i-1}, because a long interval with no event indicates that the sensor reading has not changed.
3. For any event $e^i = \{t^i, V^i\}$, if $t^{i+1} - t^{i-1} < \frac{2}{3}\Delta$, then we delete e^i.

Low-pass Filtering: The interpolation in the previous step converts the stream of sensor events into a time series. To remove spurious high frequency spikes, we apply an IIR Low-pass filter whose cutoff frequency is 30Hz.

Calibration: When the motion data is received from the accelerometer, we must calibrate it to remove the projection of gravity on each axis. Although typing may cause slight device movement, the total rotation and shift of the device are negligible during the short time of each keystroke. Therefore, we calibrate the accelerometer data by subtracting the average value from each data point on each axis, resulting in $(t^i, V^i = \{v_x^i - \bar{v_x}, v_y^i - \bar{v_y}, v_z^i - \bar{v_z}\})$. In theory, the average gyroscope value on each axis is zero. In our experiments, however, we observed that the average values were small but non-zero, possibly due to hardware or driver imprecision. Thus we must calibrate the gyroscope data similarly.

Segmentation: After calibration, we obtain a series of motion sensor data, from which we must extract segments of motion data where each segment corresponds to one keystroke. In other words, we must recognize the start and end of each keystroke from the motion data series. We build a library of waveform patterns of keystroke motion and use them to determine the segment of each keystroke in the motion data.

4.3 Classification

We use and compare two classification techniques: Dynamic Time Warping(DTW) and Support Vector Machine(SVM). They have been extensively used in user activity and gesture recognition. DTW is a template matching technique that uses a time function as the feature, while SVM is a statistical learning technique that uses a vector of parameters as the feature.

Feature selection Feature selection extracts relevant information from input data to feed to classifiers. The input to our keystroke inference tool is motion data, which may look similar to the input to user activity or gesture recognition superficially. However, a key difference is in the magnitude and stability of the data. In user activity and gesture recognition, the user perceives and controls the device motion consciously; therefore, the magnitude of motion data can often be used as a good feature in recognizing activities or gestures. By contrast, magnitude is a poor feature in keystroke inference, as motion is a byproduct of typing and is never controlled by the user consciously. Therefore, we need to explore features other than magnitude.

The motion data on the z-axis from the accelerometer mainly reflect the shift component of the device movement, and the motion data on the z-axis from the gyroscope reflect the rotation around the z-axis. Since neither of them is closely related to the keys being typed, we drop them from further consideration.

Dynamic Time Warping Dynamic Time Warp is a common template matching approach for motion sensor analysis [12,18,6,10]. Likely because of the relatively high variance in the sampling rate of our data, we find that DTW works better than other template matching algorithms, such as Euclidean Distance. Another factor in favor of DTW is the varying number of motion data points

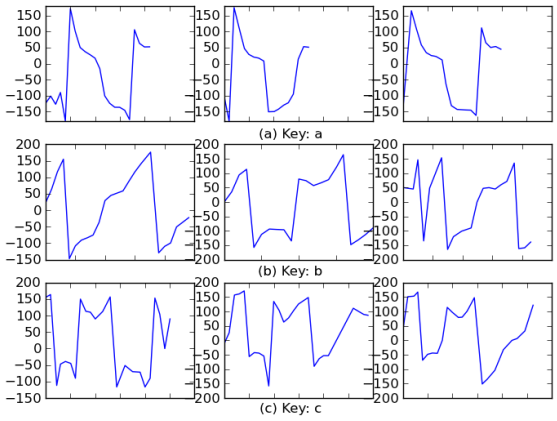


Fig. 2: Sample DTW features extracted from the data by same user.

for different key presses. For example, when the user types continuously and quickly, a new keystroke can interrupt the device vibration caused by the previous keystroke. We observed in our data that the duration of a keystroke can be as short as 100ms, less than half of the duration of a typical keystroke. DTW substring matching handles varying length of input nicely.

Existing works on activity or gesture recognition based on DTW use the magnitude of motion data on the three dimensions as input to DTW. However, as we discussed earlier, magnitude of motion data is not a good feature in inferring keystrokes, and neither is motion data on the z-axis. Therefore, from the motion data, we compute $h_i = arctan(v_y^i/v_x^i) \times 180/\pi$ as the DTW feature. Figure 2 plots the values of this feature on sample keystrokes. Our experiments show that this feature gives better results than the magnitudes on three axes.

During training, among all the data segments of the same key, we choose as a template the segment that minimizes its total distance to all the other segments.

Support Vector Machine Support Vector Machine(SVM) is a statistical learning technique used in related research [22]. Unlike template matching, Support Vector Machine uses parameter features extracted from the motion data. Common features used in SVM can be either from time domain or frequency domain. We choose to use time domain features only because the relatively small number of data points in the motion data segment of each keystroke makes frequency domain features unreliable. We also avoid features that are determined solely by the magnitude of motion, as discussed earlier. Our features include:

- **Segment duration:** the duration of the motion data segment.
- **Peak time difference:** $p_x - p_y$, where $v_x^{p_x}$ and $v_y^{p_y}$ are the first peaks on the x-axis and y-axis respectively.
- **Spike number on X (and Y):** the number of spikes on X(and Y) axis.
- **Peak interval on X (and Y) axis:** $p_{x'} - p_x$ (and $p'_y - p_y$), where $v_x^{p_x}$ and $v_x^{p_{x'}}$ (and $v_y^{p_y}$ and $v_y^{p_{y'}}$) are the first and second peaks on X (and Y) axis.

– **Attenuation rate on X (and Y) axis:** $v_x^{p_x}/v_x^{p_{x'}}$ (and $v_y^{p_y}/v_y^{p_{y'}}$).
– **Vertex angles:** $arctan(v_y^p/v_x^p)$ and $arctan(v_y^{p'}/v_x^{p'})$, where p and p' is the time of the first and second peaks on $(v_x)^2 + (v_y)^2$.

The basic form of SVM makes binary classification decisions. To apply SVM as a multi-classifer to infer keystrokes, we build a binary decision tree [9] based on the geometric distribution of keys on each keyboard.

5 Evaluation

To answer the questions raised in Section 1, we conducted a user study in which we collected typing-induced motion data from 21 users on 4 mobile devices in 6 settings. We designed and implemented a prototype system for keystroke inference as described in Section 4. We ran the system on the data collected in our user study.

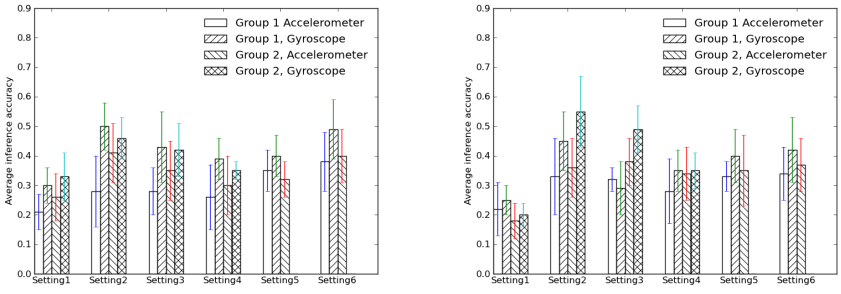
5.1 User study

Participants With the approval of our university IRB, we recruited 21 participants for our user study. They were all undergraduate students. Before the user study, we told them that the purpose was to study the usability of onscreen keyboards. We purposely did not disclose the true purpose of this study so as not to prime the participants to our security evaluation. All the participants have used smartphones and 1/3 of them have also used tablets.

Procedure We developed an application for recording keystrokes and their corresponding motion data and installed it on two smartphones and two tablets all running Android. We gave each participant a set of random strings and ask him to type each string in six different settings with regard to device type, key set, device orientation, and keyboard layout (Table 2). In each setting, we collected around 30 keystrokes per key from each participant.

It took each participant around one hour to finish the study. To prevent fatigue, our application reminded the participant to take a break after every few strings. Before the application started to record keystrokes, we allowed all the participants enough time to play with the devices to find the most comfortable way to type. The only restriction is that they could not place the devices on any fixed surfaces. We found that all the participants held the devices with one hand and typed with the other in every setting except the one that used a split software keyboard. However, the typing styles, such as the tilt of the devices and the anchor points of their hands on the devices, varied greatly between different participants and even between different strings typed by the same participant.

Settings Participants type each string in each of six settings, which differ in device type, key set, device orientation, and keyboard layout.



(a) Using dynamic time warping

(b) Using supporting vector machine

Fig. 3: Keystroke inference accuracy in different settings. There are only three bars in Setting 5 and 6 of Group 2 because HTC Evo has no gyroscope.

- *Device types*: We used four Android devices in the user study: two smartphones (Nexus S and HTC Evo) and two tablets (Motorola Xoom and Samsung Galaxy Tab 10.1). All the devices except the HTC Evo have both an accelerometer and a gyroscope. Table 1 shows that the motion sensors in different devices have different sampling rates. The OS on both smartphones is Android 2.3.1 (Gingerbread) and on both tablets is Android 3.0 (Honeycomb). We randomly divided all the users into two groups: 10 Users were in group 1 while the remaining were in group 2. Users in group 1 typed on Nexus S and Motorola Xoom while users in group 2 typed on HTC Evo and Samsung Galaxy Tab.
- *Key sets*: The keystroke inference attacker may know the set of keys in certain scenarios. For example, during phone calls the user can type only numbers because phone dialing pads have only numbers. Intuitively, one expects lower inference rate on an alphabet-only keyboard than on a number-only keyboard because the former has more keys to distinguish between. To evaluate this conjecture, we chose only alphabet characters in all the strings in setting 1 of our study, and chose only numbers in all the rest 5 settings of our study.
- *Screen orientation*: All software keyboards have different layouts for different orientations of the screen. Typically the keyboard is larger in landscape mode than in portrait mode. The screen was in portrait mode in one setting and in landscape mode for the other five.
- *Keyboard layout*: On an Android smartphone, the layout of the default software keyboard can be configured. For instance, an app can display the keyboard with the QWERTY layout by choosing *text* class, or with phone dialing pad layout by choosing *phone* class. Users can enter numbers in either layout. In the *text* class, number keys are located only in the first row of the keyboard while in the *phone* class, number keys occupy most area of the keyboard. In our user study, uses entered numbers in both keyboard layouts on smartphones.

  We compared two keyboard layouts on tablets. One is the QWERTY layout of default Android keyboard. The other is a split layout provided by

	Devices		Key set	Orientation	Keyboard Layout
	Group 1	Group 2			
1	Motorola Xoom	Galaxy tab 10.1	alphabet only	landscape	default keyboard
2	Motorola Xoom	Galaxy tab 10.1	number only	landscape	default keyboard
3	Motorola Xoom	Galaxy tab 10.1	number only	portrait	default keyboard
4	Motorola Xoom	Galaxy tab 10.1	number only	landscape	split keyboard
5	Nexus S	HTC Evo	number only	landscape	default keyboard, text class
6	Nexus S	HTC Evo	number only	landscape	default keyboard, phone class

Table 2: Users type each key in all six settings, varied by device, orientation, keyboard layout, and key set.

a third party input method called *Tablet Keyboard Free*. In the split layout, the QWERTY keyboard is divided into a left pane and a right pane, located in the lower left and right corners of the screen, respectively. A split keyboard allows users to hold the device with two hands and to type with both thumbs.

Table 2 lists the six settings for both participant groups. The order of settings in which each participant types is randomized.

5.2 Finding

Overview We collected valid data for 47,814 keystrokes in total. Figure 3 shows the inference accuracy rate for each setting. It shows that we correctly inferred 30% - 33% of the keystrokes within 26 letters (from the gyroscope reading), which is more than 8 times as good as a random guess. The average inference accuracy on number only keystrokes is as high as 55%, which is 5.5 times as good as a random guess. Even on a smartphone with a smaller screen, the inference accuracy on number-only keystrokes is 49%. These results confirmed that motion sensors are a significant side channel for leaking sensitive information.

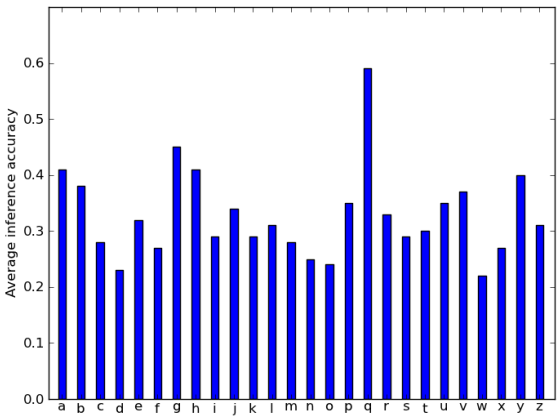


Fig. 4: Average inference accuracy of each key on the default QWERTY keyboard in setting 1 on Motorola Xoom using dynamic time warping.

Figure 4 shows that the average inference accuracies of different keys on the default keyboard are close to each other except for one key. We have found no evidence to suggest that inference accuracy differs on different areas on the keyboard.

User dependency The above results are based on user-dependent inference, where the training and testing data sets are from the same user. Figure 5 compares the accuracy of user dependent inference with that of user independent inference. In user independent inference, we picked a random user and used his data to train all the classifiers, and then tested the classifiers on all the other users' data. Figure 5 shows that user independent inference has much lower accuracy. It indicates that keystroke inference depends heavily on the user's typing style. However, even though user independent inference is less accurate, it still leaks useful information about keystrokes. For example, the average accuracy rate in Setting 1 is 12%, which is 3 times as good as random guessing.

Minimum training set size To evaluate the effect of training set size on the accuracy in user dependent inference, we repeated the classification with training sets of different sizes. For each size, the test was done in repeated random sub-sampling cross validation. Figure 6 shows the results using dynamic time warping. Initially, the inference accuracy increases when the training set becomes larger. However, the curves become flat when the training set reaches a certain size (12 for the alphabet-only keyboard and 8 for the number-only keyboard). We found a similar correlation between training set size and inference accuracy when using support vector machine as the classifier.

Device variation The participants in our user study were divided into two groups. Each group was assigned a different set of devices, which use different motion sensor chips. Although the precision and sampling rate of sensor data that we obtained from the two groups are different, the results on keystroke inference were very close.

Layout variation In Figure 3 we can see the accuracies in setting 2 is slightly higher than those in setting 3. It suggests that keystroke inference is more accurate on a keyboard in landscape mode than in portrait mode. This is not surprising because the keyboard in landscape mode is larger and the keys are separated farther.

Device dimension variation Comparing the results in setting 2 and those in setting 5 shows that the inference accuracy is affected by device dimension. In both settings the users were typing number-only strings on the default keyboard (text class) in landscape mode. Using dynamic time warping, the inference accuracy based on the output of tablet gyroscope is 50%, while that of inference

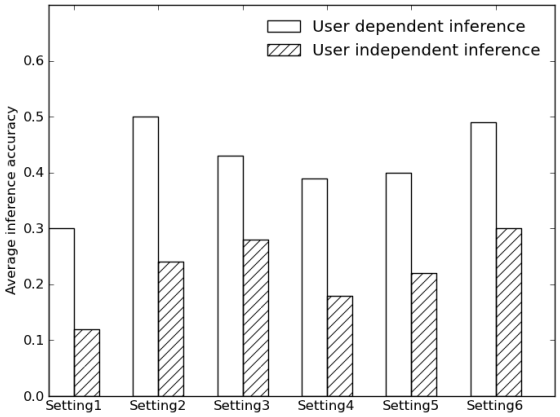


Fig. 5: Average inference accuracy is much higher when the attack is user dependent.

based on the smartphone gyroscope is 41%. Using support vector machine, the accuracies are 45% and 36% respectively. In both cases, the inference accuracy on a tablet is higher. Because all other factors — keyboard layout, key set, device orientation, data sampling rate, and users — are identical, we believe the difference in accuracy is caused by device dimension.

The variables in setting 5 and 6 are all the same except the keyboard layout. In setting 5, the number keys use only one row of the QWERTY keyboard. By contrast, the number keys almost occupy the whole keyboard in setting 6. Intuitively, the inference accuracies in setting 6 are higher than those in setting 5, which is confirmed in Figure 3. Comparing the results in setting 2 and 4 further supports our conclusion.

Finally, keystroke inference is affected by the size of the key set, as we expected. The users typed alphabet-only strings in setting 1 and number-only strings in setting 2, with all other variables identical. The inference accuracies in setting 1 are always lower than those in setting 2.

5.3 Motion Sensor Selection

Figure 3 suggests that the gyroscope is a better side channel than the accelerometer for keystroke inference. In almost every setting, gyroscope data result in higher inference accuracy. In the beginning, we suspected that it is due to the higher sampling rate of the gyroscope sensors in both Motorola Xoom and Nexus S, but comparing the results between Setting 1 and Setting 4 of Group 2 disapproved our suspicion because both motion sensors on Samsung Galaxy Tab 1.0 have exactly the same sampling rate. One possible explanation for the superiority of gyroscope data is the effect of gravity on the accelerometer data. We can see from the recorded data that the projection of gravity on each axis of the accelerometer data is changing over time, which makes it hard to eliminate the gravity during data calibration. It suggests that the angle between the device and the desk surface is changing when users type. Such movement introduces

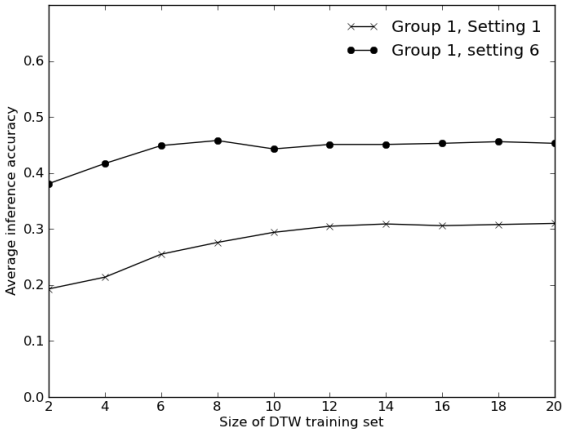


Fig. 6: Average inference accuracy by the size of training set used in dynamic time warping when the inference is user dependent.

noise to the accelerometer output and reduces keystroke inference accuracy. On the other hand, type induced device movement includes both rotation and shifting. Ideally we want to extract the rotation to differentiate keystrokes because it is better related to the location of the key on the screen. Both the accelerometer and the gyroscope can be used to measure device rotation, but the gyroscope is better at capturing high frequency rotation (> 0.5Hz) while the accelerometer is more accurate when the rotation has a lower frequency (< 0.1Hz) [15]. The data we collected indicate that typing-induced movement lasts only about 200ms, which supports the observed superiority of gyroscope data. In the rest of the paper, we focus on keystroke inference based on gyroscope data.

5.4 Classification Techniques

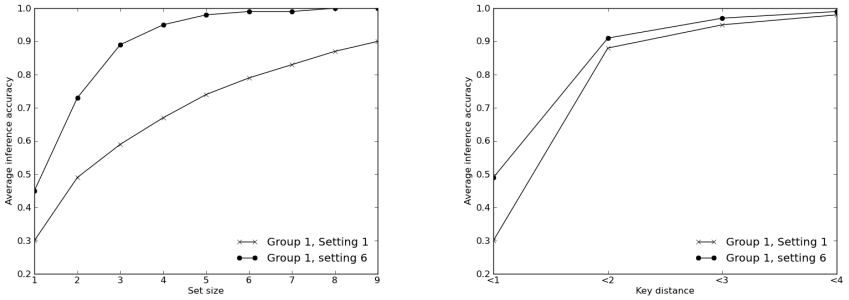
We chose both dynamic time warping and support vector machine as the classifer in our prototype. Our results show no strong evidence that one is superior to the other. Other than these two classifiers, we also tried time series analysis techniques, such as Linear predictive coding. All of them have inferior performance.

6 Discussions

6.1 Inference precision

Our evaluation shows that the accuracy for inferring a single keystroke is about 33% for the alphabet only keyboard and about 50% for the number only keyboard. Moreover, when the inference is incorrect, the probability that the falsely inferred key belongs to a small set of keys surrounding the correct key is high.

Figure 7a shows that on a number-only keyboard, the probability that the inferred key belongs to a set of three keys (including the correct key) is about 90%. Therefore, after the attacker records the motion data of a four-key PIN on this keyboard, he can try $3^4 = 81$ different PINs and the probability that one of these PINs is correct is $0.9^4 = 0.65$. By comparison, when the attacker has no motion data and therefore has to guess each key randomly, after 81 tries the probably that he has guessed a correct PIN is $0.3^4 = 0.0081$. Our motion-based keystroke inference has improved the success probability by 81 times.



(a) (b)

Fig. 7: Inference precision: (a)Probability that the inferred key belongs to a set of keys (the correct key and its neighboring keys) using dynamic time warping; (b)Accuracy rate described by the distance of the inferred key from the correct key using dynamic time warping.

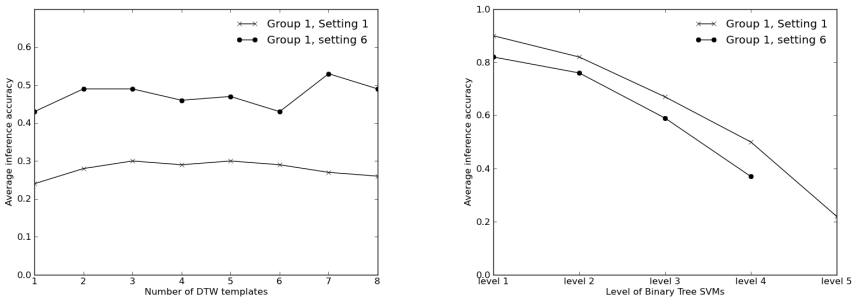
The incorrectly inferred keys are usually nearby the actual key. Figure 7b shows the distribution of *key distance* [17]. It indicates that almost 90% of inferred keys are either the actual key pressed or only one key distance away.

6.2 Multiple templates in DTW

We observed in the user study that many participants switched among a set of fixed typing style rather than changing randomly. This reminds us that a user may feel comfortable typing in several styles. To account for this, we tried matching multiple templates in dynamic time warping classification and found that it works better than matching a single template. Figure 8a shows the DTW classifier has a higher accuracy when the data is matched against multiple templates. However, the classification takes longer as the template number increases. We chose to use three templates in setting 1 and two templates in setting 6 even though the accuracy is higher if 7 templates are used as in setting 6.

6.3 Multi-Class SVM

SVM is a binary classifier. To apply it to keystroke inference, a variety of techniques are available for decomposition of the multi-class problem. We compared



(a) (b)

Fig. 8: Classifier parameters slection: (a)Average inference accuracy by different template number using dynamic time warping; (b)Average inference accuracy by the level binary tree SVMs(BTS). The more levels of the BTS the lower the accuray.

two approaches: One-against-all(OvA) and Binary tree of SVM (BTS) and the latter shows a much better performance. Moreover, BTS has a useful feature in keystroke inference. The SVM on the first level only determines whether the key is on the left side or the right side of the keyboard. But the SVM on the last level need to make a decision between two adjacent keys. Thus it is reasonable that SVM on the lower level node has lower inference accuracy. As shown in Figure 8b, the inference accuracy decreases when the input goes through more classifiers. The results suggest that the medium output of BTS can be used for determining approximation of the key. For example, the accuracy inferring a key on the alphabet only keyboard with a BTS classification of 5 levels is only 22%, but the accuracy after the 4th SVM is 50%, i.e, the chance that the actual key is one of two keys is 50%. This is consistent to what we observed from Figure 7a.

7 Conclusion

To evaluate the practicality of motion-based keystroke inference attack, we conducted a user study where 21 participants typed a total of 47,814 keystrokes on four different mobile devices in six settings. We developed a prototype attack and applied the attack on the users' keystrokes. Our results show that this attack remains effective even though the accuracy is affected by user habits, device dimension, screen orientation, and keyboard layout. On a number-only keyboard, after the attacker tries 81 4-digit PINs, the probability that she has guessed the correct PIN is 65%, which improves the accuracy rate of random guessing by 81 times. Our study also indicates that inference based on the gyroscope is more accurate than that based on the accelerometer. We evaluated two classification techniques in our prototype and found that they are similarly effective.

8 Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant Numbers 0644450 and 1018964. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Keystroke logging wiki page. `http://en.wikipedia.org/wiki/Keystroke_logging`.
2. Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *Proceedings of IEEE Symposium on Security and Privacy, 2004.*, pages 3 – 11, may 2004.
3. R. Aylward, S.D. Lovell, and J.A. Paradiso. A compact, wireless, wearable sensor network for interactive dance ensembles. In *International Workshop on Wearable and Implantable Body Sensor Networks, 2006.*, BSN 2006, pages 4 pp. –70, april 2006.
4. Liang Cai and Hao Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Proceedings of the 6th USENIX conference on Hot topics in security*, HotSec'11, pages 9–9, 2011.
5. Liang Cai, Sridhar Machiraju, and Hao Chen. Defending against sensor-sniffing attacks on mobile phones. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, MobiHeld '09, pages 31–36, 2009.
6. Bong Whan Choe, Jun-Ki Min, and Sung-Bae Cho. Online gesture recognition for user interface on accelerometer built-in mobile phones. In *Proceedings of the 17th international conference on Neural information processing: models and applications - Volume Part II*, ICONIP'10, pages 650–657, 2010.
7. Ming Ki Chong, Gary Marsden, and Hans Gellersen. Gesturepin: using discrete gestures for associating mobile devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 261–264, 2010.
8. Denis Foo Kune and Yongdae Kim. Timing attacks on pin input devices. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 678–680, 2010.
9. Dejan Gjorgjevikj Gjorgji Madzarov and Ivan Chorbev. A multiclass svm classifier utilizing binary decision tree. In *Informatica33*, pages 233–241, 2009.
10. Gerhard P Hancke. Gesture recognition as ubiquitous input for mobile phones. 2008.
11. Jonathan Lester, Blake Hannaford, and Gaetano Borriello. Are you with me? using accelerometers to determine if two devices are carried by the same person. In *Proceedings of Second International Conference on Pervasive Computing*, Pervasive 2004, pages 33–50, 2004.
12. Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5:1–9, 2009.
13. Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *Proceedings of the 5th international conference on Pervasive computing*, Pervasive 2007, pages 144–161, 2007.

14. Cheol-Hong Min and Ahmed H Tewfik. Automatic characterization and detection of behavioral patterns using linear predictive coding of accelerometer sensor data. In *Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, volume 2010, pages 220–223.
15. Steve Nasiri, David Sachs, and Michael Maia. Selection and integration of mems-based motion processing in consumer apps. `http://invensense.com/mems/gyro/documents/whitepapers/Selection-and-integration-of-MEMS-based-motion-processing-in-consumer-apps-070809-EE-Times.pdf`, July 2009.
16. Yuan Niu and Hao Chen. Gesture authentication with touch input for mobile devices. In *3rd International Conference on Security and Privacy in Mobile Information and Communication Systems*, MobiSec '11, May 2011.
17. Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, HotMobile '12, pages 9:1–9:6, New York, NY, USA, 2012. ACM.
18. Cuong Pham, Thomas Plotz, and Patrick Olivier. A dynamic time warping approach to real-time activity recognition for food preparation. In *Ambient Intelligence*, volume 6439, pages 21–30, 2010.
19. Andrei Popescu and Steve Block. DeviceOrientation event specification, editor's draft 9. `http://dev.w3.org/geo/api/spec-source-orientation.html`, February 2011.
20. Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, pages 25–25, 2001.
21. Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 1–16, 2009.
22. Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, and Shijian Li. Gesture recognition with a 3-d accelerometer. In *Ubiquitous Intelligence and Computing*, volume 5585, pages 25–38, 2009.
23. Nan Xu, Fan Zhang, Yisha Luo, Weijia Jia, Dong Xuan, and Jin Teng. Stealthy video capturer: a new video-based spyware in 3G smartphones. In *Proceedings of the second ACM conference on Wireless network security*, WiSec '09, pages 69–78, 2009.
24. Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Proceedings of the 18th Annual Networkand Distributed System Security Symposium*, NDSS '11, 2011.
25. Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security*, 13:3:1–3:26, November 2009.