Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Input extraction via motion-sensor behavior analysis on smartphones

Chao Shen [a,*], Shichao Pei [a], Zhenyu Yang [a], Xiaohong Guan [a,b]

[a] Xi'an Jiaotong University, No.28 Xianning West Road, Xi'an 710049, China
[b] Tsinghua University, Haidian District, Beijing 10084, China

ARTICLE INFO

Article history:
Received 30 March 2015
Received in revised form
18 June 2015
Accepted 29 June 2015
Available online 6 July 2015

Keywords:
Human-computer interaction
Smartphone security
Behavior analysis
Motion sensor
Input inference
Performance evaluation
Algorithm comparison

ABSTRACT

Smartphone onboard sensors, such as the accelerometer and gyroscope, have greatly facilitated people's life, but these sensors may bring potential security and privacy risk. This paper presents an empirical study of analyzing the characteristics of accelerometer and magnetometer data to infer users' input on Android smartphones. The rationale behind is that the touch input actions in different positions would cause different levels of posture and motion change of the smartphone. In this work, an Android application was run as a background process to monitor data of motion sensors. Accelerometer data were analyzed to detect the occurrence of input actions on touchscreen. Then the magnetometer data were fused with accelerometer data for inferring the positions of user inputs on touchscreen. Through the mapping relationship from input positions and common layouts of keyboard or number pad, one can easily obtain the inputs. Analyses were conducted using data from three types of smartphones and across various operational scenarios. The results indicated that users' inputs can be accurately inferred from the sensor data, with the accuracies of 100% for input-action detection and 80% for input inference in some cases. Additional experiments on the effect of smartphone screen size, sampling rate, and training data size were provided to further examine the reliability and practicability of our approach. These findings suggest that readings from accelerometer and magnetometer data could be a powerful side channel for inferring user inputs.

1. Introduction

Keyboard has long been the most common target for external or insider attackers to intercept users' sensitive or important input, such as password, email content, and transaction code (Asonov and Agrawal, 2004; Aviv et al., 2012; Cai and Chen, 2012; Foo Kune and Kim, 2010; Schlegel et al., 2011; Vuagnoux and Pasini, 2009). Currently, smartphones have become omnipresent platforms of personal computing for users to access the Internet and online services, and more and more personal information has been stored in smartphones. The obsolescence of physical keyboard on smartphones seems to provide a solution for the information leakage of sensitive user inputs. Extant smartphone operating systems (e.g., Android) also constrain that the input information and touch-input position cannot be directly read from the hardware (Mylonas, 2008). However, current smartphones are commonly equipped with various motion and location sensors, which can be used to measure the motion and posture

* Corresponding author. Tel.: +86 82663939.
E-mail address: cshen@sei.xjtu.edu.cn (C. Shen).
http://dx.doi.org/10.1016/j.cose.2015.06.013
0167-4048/

change of the smartphone. Also, Android smartphones have no limitation for third-party applications to access and read the sensor data (Mylonas et al., 2013). Along with the fact that input actions on touchscreen will cause small shaking of the smartphone, these findings offer us an opportunity of analyzing the behavior data from motion sensors to infer the position of touch actions, thus further to acquire users' inputs.

In this paper, we present an empirical study of inferring users' input on Android smartphone by analyzing the characteristics of motion sensors' data. The rationale behind our work is that different input actions would cause different levels of posture and motion change of the smartphone, which is based on different positions and strength of touch actions. We utilized an Android application (run as a background job) to monitor the data of accelerometer and magnetometer sensors. We then employed Kalman filter and denoising methods to obtain unbiased sensor data, and developed a procedure to detect the occurrence of input actions by analyzing the change of accelerometer data. We next extracted descriptive statistics features and wavelet transform features from the accelerometer data and magnetometer data to characterize the motion and posture changes of the smartphone, and applied classification technologies to establish an inference model of the positions of input actions. Finally, through the mapping relationship from input positions and common layouts of keyboard or number pad, the user inputs could be easily speculated. Experimental results across various types of smartphones and different operational scenarios showed that user inputs on number pad could be accurately inferred from the data of accelerometer and magnetometer sensors, with the accuracies of 100% and 80% for input-action detection and user-input inference in some cases. We also examined our performance against different smartphone screen sizes, different sampling rates, and different training data sizes, to further testify its reliability and practicability in practice. The main purpose and contributions of this work are fourfold.

Firstly, we lay empirical work for user input inference on smartphones by a side channel that relies on the analysis of the characteristics of accelerometer and magnetometer sensors. We investigate the reliability and practicability of inferring user inputs on smartphones based on the change of the sensors' data, without dedicated and explicit attention from users.

Secondly, we model characteristics of the accelerometer and magnetometer sensors by proposing newly-defined procedural features, such as fluctuation features and wavelet features, to characterize different positions of input actions in an accurate and robust manner. These features can lead to a performance boost both in input action detection and input position inference. We also apply four types of two-class classifier [*Random Forest*, *Support Vector Machine*, *Neural Network*, and *Nearest Neighbor*] to build the inference model, so that we can examine whether an observed effect is specific to one type of classifier or holds for a range of classifiers.

Thirdly, we examine the proposed approach in terms of the sensitivity to training data size, scalability to screen size, and flexibility to sampling rate, to further examine the applicability and generalization capability of the proposed approach in practice.

Finally, this study systematically evaluates the user-input inference by analyzing data of accelerometer and magnetometer sensors on smartphone, and extensive experiments across various types of smartphones and different operational scenarios show the proposed approach can perform input inference with a high accuracy. These results suggest that readings from accelerometer and magnetometer data could be a powerful side channel for inferring user inputs.

The structure of the paper is as follows. Section 2 discusses related work. Section 3 describes data acquisition process. Section 4 details the proposed approach for user-input inference. Section 5 presents our experimental design and results. Section 6 discusses and concludes.

2. Background and related work

2.1. *Input inference on smartphone*

Virtual keyboard is the most common input interface on smartphones, by which users may enter some security and privacy information, such as password, PINs, credit card numbers, or phone numbers. Therefore how to sniff or infer these inputs has recently aroused the interest from both attackers and researchers. In 2004, Asonov and Agrawal (2004) found that user inputs could be inferred by analyzing the sound emanated by the pressed keys. Later on, Foo Kune and Kim (2010) found that the timing information between two adjacent keystrokes could be used to accurately speculate users' input. Aviv et al. (2010) investigated the feasibility of inferring the touch pattern from oily residues on the touchscreen by users' fingers. Then some researchers (Raguram et al., 2011) exploited the visual feedback of magnified keys provided by smartphones' virtual keyboards to infer users' input. These ideas can be used for shoulder-surfing, whereby an attacker uses his own smartphone to video-record other users while they type PINs or messages.

Yet only recently have researchers come to the find of smartphone sensors as a sensitive source for users' input sniffer or speculation. Mylonas (2008) first showed the privacy risk of sensors utilized in smartphones. Xu et al. (2009) provided a way by a Trojan with access to the camera sensor for extracting private information. Later, Schlegel et al. (2011) analyzed the audio sensor for the same purpose. Cai and Chen (2011) also raised the concerns regarding the camera, the microphone, and the GPS signal for input inference in modern smartphones. Recently, Simon and Anderson (2013) demonstrated such an attack by exploiting the camera and microphone sensors and Spreitzer (2014) demonstrated this attack by exploiting the light sensor. But these attacks are less insidious because these sensors are protected with access permission by mobile operating systems. Besides, some researchers also investigated the reliability of the usage of accelerometer sensor in smartphones to capture inter-key timing measurements for extracting users' input on a nearby PC-keyboard (Marquardt et al., 2011), or to detect motion changes caused by users' taps on the screen and further to infer users' input on smartphones (Aviv et al., 2012; Cai and Chen, 2011; Owusu et al., 2012; Xu et al., 2012).

Most recently, a full survey of existing work in smartphone sensor data for digital investigation (Mylonas et al., 2012, 2013) provided an in-depth analysis of smartphone sensors' data as a potential source for digital forensics, and identified the vulnerabilities in Android's security model that enables the ad-hoc acquisition of sensor data.

2.2. *Motion-sensor-based input inference on smartphone*

To our knowledge, few papers have targeted the usage of the change of motion sensors in smartphones for users' input inference, which will be the central concern of our study. Cai and Chen (2011) were some of the earliest researchers to make use of accelerometer and gyroscope sensors to infer pressed keys on Android smartphones. They extracted features from orientation data, and the analysis on 449 keystrokes showed that they could correctly infer 71.5% of the typed keys on a number-only soft keyboard. Later on, they (Cai and Chen, 2012) extended their work with more keystrokes to further examine its effectiveness on different mobile devices, and reported a 65% of prediction accuracy for 4-digit PINs after 81 attempts.

Owusu et al. (2012) employed the accelerometer sensor to infer passwords entered on touchscreens. They created a predictive model and trained it only on the accelerometer measurements. The results on 2700 keypresses from 4 participants showed that 6 of the 99 passwords could be correctly deduced in a median 4.5 trials, and 59 of 99 passwords needed approximately 215 median trials to be cracked.

Aviv et al. (2012) investigated the practicality of using only accelerometer sensor for inferring PINs or graphical password patterns on Android smartphone. They extracted sampling-rate independent features from accelerometer data to learn user tap and gesture-based input. Using data from 24 users, their approach could classify the entered PINs and graphical patterns with accuracies of 43% and 73% within 5 attempts in a controlled setting; but in uncontrolled settings, the accuracy would reduce to 20% for the PINs and 40% for the graphical patterns.

Miluzzo et al. (2012) went one step further and investigated the usage of accelerometer and gyroscope sensors in general to infer the location of taps on both smartphones and tablets. They applied machine learning algorithms to model the sensor data from each tap input, and tested it for the English alphabet (26 letters) and icons on the screen of smartphones. Analysis on 10 participants showed that the recognition of tap locations and the inference of English letters could be done with high accuracies, which are adequate to extract users' passwords. Recently, Xu et al. (2012) further demonstrated such an attack by exploiting the accelerometer and gyroscope sensors to get the tap inputs. They learned motion change patterns of smartphones for the corresponding tap events, and achieved a good predication accuracy of 40% with 1 attempt.

These efforts confirm that motion sensors (i.e., accelerometer and gyroscope) have rich potential to provide side-channel attacks for compromising users' privacy on smartphones, but there remains other smartphone sensor to be unexplored. Based on the facts that the smartphone gesture can be represented by magnetometer data and the smartphone motion can be represented by accelerometer (Liu, 2013), here we attempt to present a possibility of using the combination of magnetometer and accelerometer data to infer user inputs on smartphones with a high accuracy, and to provide a more thorough analysis. In addition, no previous work examined its reliability and applicability across different operational scenarios and different types of smartphones, and identified its sensitivity and scalability to touchscreen size, data sampling rate, and training data size. Thus, our paper addresses these issues.

3. Data acquisition

3.1. *Operational scenario*

To systematically investigate the reliability and practicability of our approach, we deployed three different operational scenarios for collecting motion-sensor data, which would roughly cover user's routine touch input environments.

Hand-hold-input scenario: subjects hold the smartphones and perform touch-input actions with sitting-still or standing-still postures.
Table-hold-input scenario: smartphones are placed on the desktop, and subjects were asked to perform touch-input actions using a single hand.
Hand-hold-walk scenario: subjects hold the smartphones and performed touch-input actions when walking.

3.2. *Data capture process*

A free experimental environment was established on Android smartphones to collect sensor data in this evaluation. We developed a data capture application that runs as a background job, which starts monitoring users' touch-input actions when the screen is on and stops when the screen is off. The application is transparent and does not affect other applications. Sensor data were collected during users' routine touch-input activities, which mainly cover the data under the applications of phone dialing, password entering, and instant chatting. Whenever the user touched the screen to input some information, the data capture application recorded the accelerometer data, magnetometer data, and the sampling timestamp.

Specifically, data acquisition can be divided into two stages: training stage and testing stage. In the training stage, the sensor data were captured when participants typed random strings on the number pad (mainly under the applications of phone dialing and password entering), and the data capture application also recorded the location of input actions on touchscreen, and the timestamps of action starting (ACTION-DOWN) and action ending (ACTION-UP) (e.g., the down and up of a click action). The application would continuously record the data of input actions on touchscreen, and transferred the data to a remote server for training the input detection model and input inference model. In the testing stage, the application ran in the background, and the data capture process was similar to the training stage. But the location of input actions and the timestamp of ACTION-DOWN and ACTION-UP cannot be obtained, due to security restriction of Android system. However, the sensor data could

be easily acquired and written into a data file, which could be transferred to the remote attacker for inferring user inputs.

3.3. Participants

We recruited 30 volunteer students (22 males and 8 females) from the university, with ages ranging from 20 to 35 years (mean = 25.5, s.d. = 3.8). All participants were skilled smartphone users, and 3 of them were left-handed. They were told to conduct nine rounds of data collection (three rounds for each operational scenario), and waited at least one day between two rounds (ensuring that some day-to-day variation existed within our data). In each round of data collection, participants were asked to type random strings (with the length varying from 4 to 8) 60 times on the number pad (under the applications of phone dialing and password entering). All 30 participants remained in the study, each subject contributed around 3240 taps (mean = 3181, median = 3135, min = 2928, max = 3387, and s.d. = 457.3).

To prevent fatigue, our application allowed participants to take a break after every few strings. We also found that all the participants typed with her/his dominant hand with the smartphone in the portrait mode. Specifically, in the table-hold-input scenario, participants tapped with her/his dominant hand with the smartphone placed on the table. While in other two scenarios, participants held the devices with one hand and typed with the other. Besides, in order to trigger the interaction in a natural way, the users were explained the purpose of the study after finishing all data collection.

3.4. Apparatus

We set up three different types of smartphones, each operating on Android 4.1.x. The smartphones are Samsung Galaxy N7102 with a 5.5-inch screen, 1.6 GHz processor, and 2 GB of RAM; Huawei Ascend P6 with a 4.7-inch screen, 1.5 GHz processor, and 2 GB RAM; and ZTE V800 with a 3.5-inch screen, 0.6 GHz processor, and 256 M of RAM. Each type of smartphones was assigned with nearly equal numbers of subjects to conduct all nine rounds of data collection.

4. User input inference architecture

As shown in Fig. 1, our proposed user-input inference approach mainly consists of five modules – recorder, preprocessor, input action detection, input position inference, and input mapping. The design of the first component is straightforward. The main task of the recorder is to record the raw data of motion sensors. The focus of this section is on the data preprocessing, input action detection, input position inference, and input mapping.

4.1. Preprocessing

4.1.1. Gravity filtering

Generally, raw accelerometer data include the gravity component, which may make the obtained sensor data hard to accurately reflect the motion change of smartphones. Since researchers usually consider the gravity component as a constant component and the acceleration data as alternating components, the filtering technique is then employed to remove the gravity components. Here we employed the *Kalman* filter method, a recursive way of estimating optimal value of system state variables, to obtain an unbiased estimated value of accelerometer data. Specifically, the gravitational component embedded in the raw accelerometer data could be reduced in each of three smartphone axes (X, Y, Z) by the following steps.

Step 1: We computed the predicted value of accelerometer data P at time t by using the estimated value at time $t-1$.

$$P(t|t-1) = A \cdot P(t-1|t-1) + B \cdot U(t). \quad (1)$$

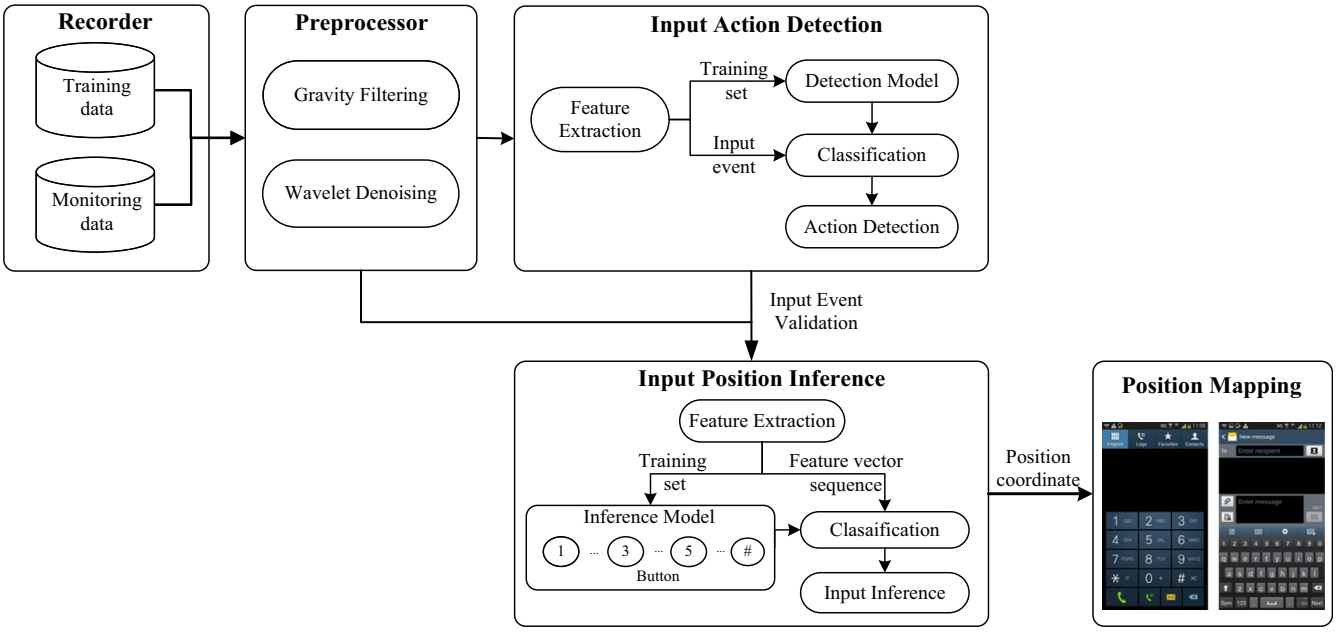


Fig. 1 – System architecture of our proposed input inference approach.

where $P(t|t-1)=\{X(t|t-1), Y(t|t-1), Z(t|t-1)\}$ is the predicted value of accelerometer data at three smartphone axes, $U(t)$ is the control value at time t, and A, B are the coefficient matrices.

Step 2: We calculated the deviation D of the predicted value at time t.

$$D(t|t-1) = A \cdot D(t-1|t-1) \cdot A' + Q. \tag{2}$$

where A' is the transposed matrix of A, Q is the process deviation.

Step 3: We obtained the optimal and unbiased estimated value of accelerometer data $P(t|t)$ at time t based on the measured value $Z(t)$ and the predicted value $P(t|t-1)$, and also updated the deviation of $P(t|t)$.

$$P(t|t) = P(t|t-1) + Kg(t) \cdot (Z(t) - H \cdot P(t|t-1)), Kg(t) = \frac{D(t|t-1) \cdot H'}{(H \cdot D(t|t-1) \cdot H' + R)}, D(t|t) = (I - Kg(t) \cdot H) \cdot D(t|t-1). \tag{3}$$

where Kg is the Kalman Gain coefficient, R is the deviation of the measurement, and H is the system matrix.

4.1.2. *Wavelet denoising*

Most often, sensor signal inevitably contains non-stationary noise which makes the signal exhibits multiple peaks or mutations. This would directly lower the accuracy of input action detection and input inference. Thus we applied the wavelet denoising method to mitigate the signal mutation instead of traditional Fourier analysis method, since the later one converts a signal in the frequency domain at a certain time point, but the mutation and noise usually affect the entire spectrum of the signal. Here we consider how the wavelet denoising method is applied to sensor signal.

Step 1: We selected suitable wavelet functions to decompose the signals into N levels and extracted the low frequency coefficients of every level and the high frequency coefficient of the N-th level.

Step 2: We employed the threshold analysis to filter decomposed signals.

Step 3: We used inverse wavelet transform on the filtered-decomposed signals to reconstruct the original signal.

4.2. *Input action detection*

In this section, we first characterize the input action for the purpose of input detection, and extract the features for measuring the occurrence of input actions. We then present how to detect users' input action in the testing stage based on the accelerometer data.

4.2.1. *Input action measurement*

To accurately detect the occurrence of an input action, we used $ACCEsum = |A|^2 = A_x^2 + A_y^2 + A_z^2$ as the metric, where A_x, A_y, and A_z are the accelerometer values in three axes. *ACCEsum* represents the magnitude of the external force F on the touchscreen. It equals the magnitude of gravity in the vertical direction when the smartphone is placed at the desktop, and turns to zero when the smartphone is in a free fall. Fig. 2 shows the change curve of *ACCEsum* with the growing numbers of input actions. The curve exhibits periodic and obvious peaks, which can be used to measure the occurrence of input actions with a high accuracy.

4.2.2. *Feature extraction*

Fig. 2 revealed that the *ACCEsum* curve of one input action has three typical fluctuation processes, which corresponds to a three-stage physical process of smartphone movement: (1) the smartphone moves downward with increasing acceleration when a finger touches the screen, then the smartphone turns to the deceleration motion until the speed reduces to zero; (2) the smartphone moves upward, and the speed increases first and then reduces to zero (the finger leaves the screen at the end of this stage); (3) the hand returns to the initial position on the basis of inertia (with a slight motion of the smartphone which corresponds to the third mutation in Fig. 2b). Our preliminary analysis showed that the relevant measurements extracted from these sub-processes appeared more stable than those from the whole process. We extracted procedural features from these processes to analyze the accelerometer data, which could accurately reflect the motion habits of smartphones by different input actions.

$F1$: the maximum value of the first fluctuation;
$F2$: the maximum value of the second fluctuation;

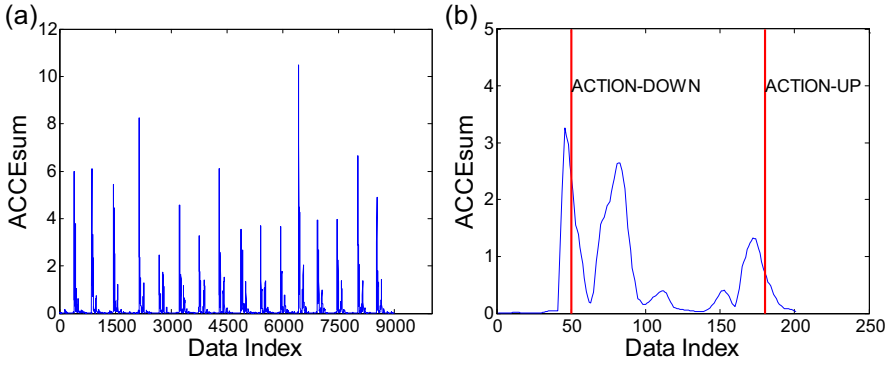


Fig. 2 – The curve of *ACCEsum* with respect to input events. Panel (a) depicts the change of *ACCEsum* curve with the increase of input events; panel (b) shows the *ACCEsum* curve within a single input event. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

F3: the maximum value of the third fluctuation;
F4: the *ACCEsum* value from the timestamp of start to the timestamp of end;
F5: the difference of the timestamps between *ACTION-DOWN* and *ACTION-UP*;
F6: the difference between the timestamps of *F1* and *F2*;
F7: the difference between the timestamps of *F2* and *F3*;

Given the statistical analysis by using the kernel density estimation method (Bowman and Azzalini, 2004), we found the probability density function (PDF) curves of *F1–F7* obeyed the normal distribution. Thus to accurately recognize an input action, we chose the range of *F1–F7* between the Lower Extreme and the Upper Extreme (Ross, 2009) as the standard set, represented by [*L*, *U*]. An input action then can be easily identified if the features are within the standard set. Specifically, we set $L = Q_1 - 1.5 \times IQR$ and $U = Q_3 + 1.5 \times IQR$, where Q_1 is the lower quartile, Q_3 is the upper quartile, and $IQR = Q_3 - Q_1$ is the interquartile range. In this way, every input action can be represented by $F_s = \{I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$, where $I_i = \{L_i, U_i\}$.

4.2.3. *Input action detection*

In the training stage, we had obtained the timestamps of *ACTION-DOWN* and *ACTION-UP* for each input action. However, according to Fig. 2b, the timestamps of *ACTION-DOWN* and *ACTION-UP* (represented by red vertical line) do not coincide with the start time of first fluctuation and the end time of third fluctuation in the *ACCEsum* curve, which may be due to the hardware noise and processing speed of smartphones. Thus to get the accurate timestamps of input action, we used the difference of change of *ACCEsum* values between adjacent points to estimate start point of input action, and then determined the end point to get accurate duration time of an input action by examining the change rate of *ACCEsum* curve. Next we extracted features (*F1–F7*) from the sensor data that occurred within the duration of an input action, and then one can compare the features with those obtained in the training stage to determine the occurrence of an input action.

4.3. *Input position inference*

4.3.1. *Screen division*

Subjects usually type inputs on touchscreen through man–machine interfaces, such as PIN number and Squared Up and Dial (as shown in Fig. 3). These interfaces are composed of similar structures, which commonly include input interface and display interface. The input interface can be easily divided into independent areas, each of which corresponds to a button. Besides, from the perspective of attackers, they would carefully consider the input interface to ensure similar regional division and input interface layout when inferring user inputs in attacks.

4.3.2. *Posture change analysis*

When a subject taps on touchscreen for inputting, the posture of the smartphone will change, which results in the change of values of pitch, roll, and azimuth for orientation data (in Android 4.x.x, the magnetometer and accelerometer data are used to calculate the value of orientation data). The pitch axis measures degrees of rotation around the smartphone's *x* axis, which is positive when the positive z axis rotates toward the positive y axis, and it is negative when the positive *z* axis rotates toward the negative y axis; the roll axis measures degrees of rotation around the smartphone's *y* axis, which is positive when the positive z axis rotates toward the positive *x* axis, and it is negative when the positive z axis rotates toward the negative *x* axis. It is noted that input actions usually do not cause changes in the azimuth axis of orientation sensor because the corresponding rotation degree around the z axis is very small, thus here we ignored the sensor change of azimuth axis. Based on our empirical observation that the sensor data in the pitch and roll axes are different at different screen positions, we used numerical changes of sensor data in these two axes to represent the posture change of a smartphone.

4.3.3. *Feature extraction*

Our preliminary analysis found that the data of acceleration and orientation would exhibit significant changes when input actions occurred. Thus we used the accelerometer data in all

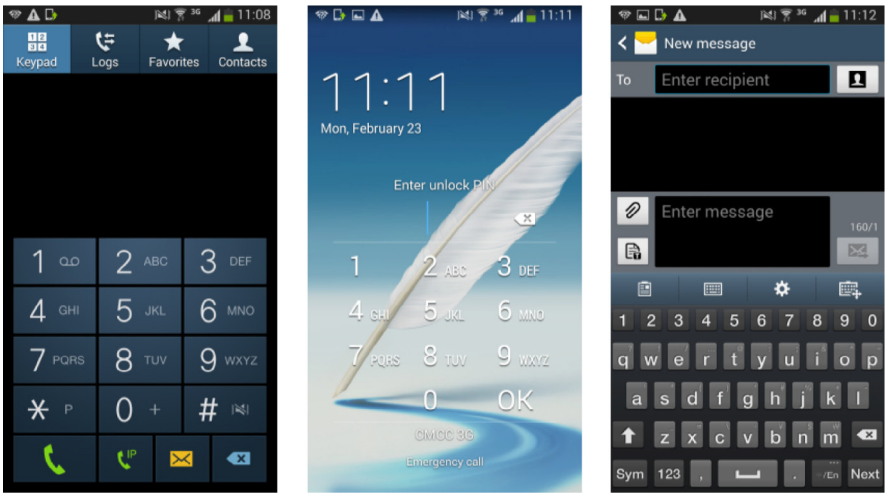


Fig. 3 – Different input interfaces in smartphones.

three axes and the orientation data in two axes as a source to represent the posture change of a smartphone. We employed descriptive statistics (Ostle, 1963) to characterize the central tendency, distribution, and discrete degree of the sensor data. Specifically, we extracted the mean, median, mode, skewness, kurtosis and standard deviation from the sensor data of five axes to depict the macroscopic information of the posture change. Secondly, we employed wavelet analysis method (Zhu et al., 2009) to extract the high frequency coefficient of every level and the low frequency coefficients of the last level as the features to characterize the microscopic content of the posture change. These two types of features were then combined together to form a feature vector for accurately representing the posture and motion change caused by a single touch-input action on smartphones.

4.3.4. *Position inference*

Four classification algorithms were implemented to analyze motion sensor data for input position inference task. Specifically, we applied four types of two-class classifier [Random Forest, Support Vector Machine (SVM), Neural Network, and Nearest Neighbor] to build the position inference model for users' inputs, so that we can examine whether an observed effect is specific to one type of classifier or holds for a range of classifiers.

We considered the position inference task as a multi-class classification problem which adopts the one-against-many approach. Given k classes, one for each number (or symbol) on the number pad, k component classifiers are constructed for each of four multi-class classifiers, one for each class. Each multi-class classifier must analyze sensor data and recognize a valid number (or symbol). Each multi-class classifier expects the sensor data to be encoded in the feature vectors, and has two phases: training and testing. For each individual user, during the training phase, a set of feature vectors from all numbers (or symbols) is utilized to train k component classifiers for each multi-class classifier; while during the testing phase, k component classifiers compare the new vector (either from the same user or different users) with their trained profiles, and produce a classification score for each of k component classifiers. The resulting label is the class with the largest score.

4.4. *Input mapping*

As shown in Fig. 3, the input interface usually consists of several buttons for users to enter information. Hence based on the inferred positions on the touchscreen, one could obtain the tapped button by simply mapping the inferred positions to the input interfaces. Taking the phone dialing as an example, the input interface can be divided into twelve areas, each of which represents a number button and corresponds to a relatively fixed position range on the touchscreen. Specifically, in the training stage, when the subject tapped on the number pad, we recorded the pressed number (or symbol) and its corresponding position, and labeled the tap actions on the same number (or symbol) as the same class. An inference profile was then built for each class. In the testing stage, the feature vector of sensor data was taken as the input to the inference model and the classification label would indicate the position of the input action. Through the mapping relationship between tap positions and common layouts of input interfaces, one can easily obtain the input content.

Table 1 – Results of input action detection across various operational scenarios.

Operational scenarios	Accuracy
Hand-hold-input	100%
Table-hold-input	97%
Hand-hold-walk	82%

5. Proof of concept

This section evaluates the reliability and applicability of the proposed approach through a series of experiments.

5.1. *Input inference across various operational scenarios*

In each operational scenario (as discussed in Section 3.1), we collected data set with about 32,400 keypresses on the number pad, and extracted 600 samples of each key as training data to establish the inference model. Besides, we used the HUAWEI P6 with the screen size of 4.7 inches as test bed, and adopted the data sampling rate of 50 Hz (which is the default setting for most legitimate applications in Android smartphones) to obtain the sensor data.

5.1.1. *Input action detection*

Table 1 shows the accuracies of input action detection across three different types of operational scenarios. The detection accuracies are 100% in the hand-hold-input scenario and 97% in the table-hold-input scenario. In these two scenarios, the input action can be accurately detected due to obvious fluctuations and low noise of the sensor data. But the accuracy becomes worse when subjects operate the smartphone in their walking. This may be due to the reason that the fluctuation caused by finger pressure and walk are mixed together, and introduce more noise into the raw data, thus leading to a low detection accuracy.

5.1.2. *Input inference*

We used standard number pads (as shown in Fig. 3) to perform the input inference task on smartphones. Fig. 4 shows the recognition rates for each button on the number pad across different operational scenarios. Each panel displays the results by using each of four classifiers. We observed that while the recognition rates for input inference differ across numbers (or symbols) on the number pad, they perform appreciably better than chance. The recognition rates for most of numbers (or symbols) are over 70%, which indicate that users' input can be inferred with a high accuracy using motion sensor data. Specifically, in the table-hold-input scenario, because the smartphone is placed horizontally on the desktop and the support force of desktop is uniformly distributed over the back of the screen, the smartphone would generate a smaller posture change than that in the hand-hold-input scenario when input action happened, which leads to a decline of

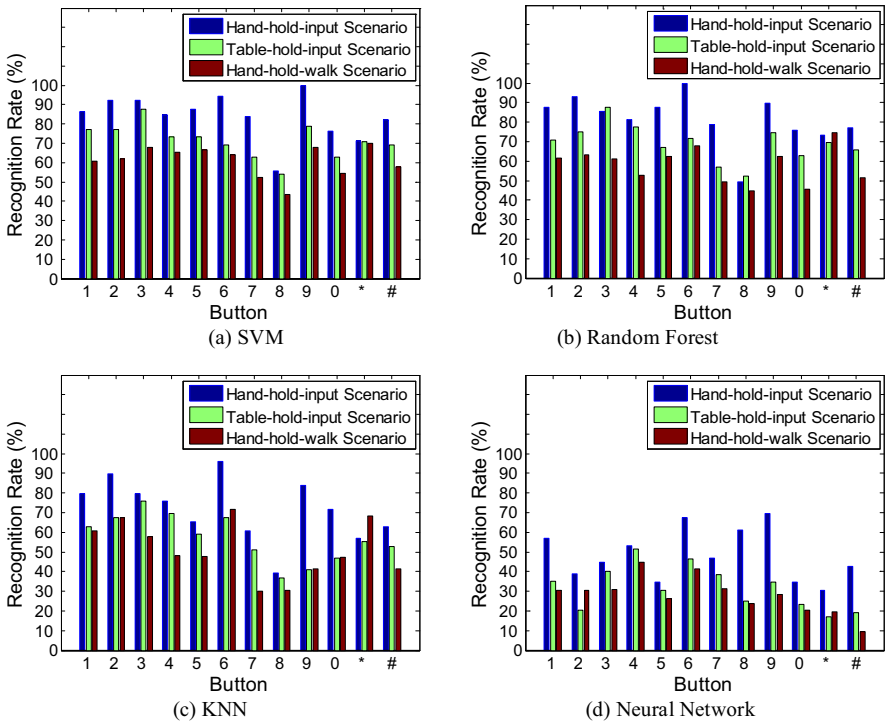


Fig. 4 – Recognition rates for each button on the number pad across different operational scenarios, using four types of classifiers: (a) SVM, (b) random forest, (c) nearest neighbor, and (d) neural network.

inference accuracy. While in the hand-hold-walk scenario, the sensor data may get affected by people moving, which would introduce additional noise into the accelerometer data. This may be the reason why the recognition rates in this scenario are worse than those in other two scenarios.

The SVM classifier has a better performance than other classifiers considered in this study. This may be due to the fact that its kernel functions and support vectors enable the SVM classifier to naturally detect outliers in the case where the data are not linearly separable and to find informative features with relative insufficiency of prior knowledge, and meantime maintain high stability and stability. Besides, the worst case is the neural network classifier. We conjecture that the reason is its strong sensitivity to outliers and variability in sensor data.

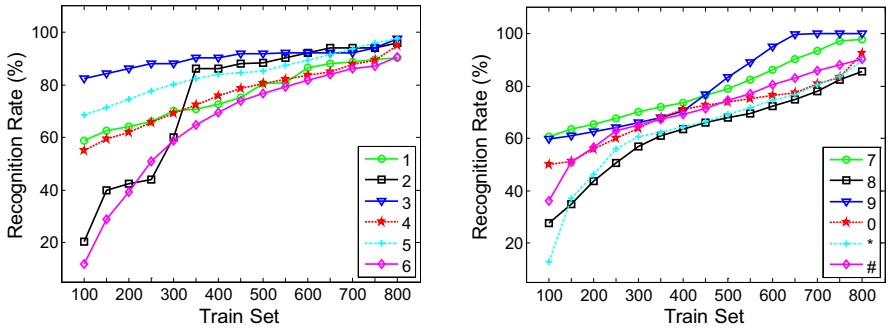
5.2. *Sensitivity to training data size*

This experiment examined the input inference performance at varying training data sizes. Training data size might have an important effect on the effectiveness and practicability of the proposed approach: if the training data are with a small size or not enough, it will generate a significant difference between the generalization error and the inference error on the training data; if training data are with a large size or too much, it will be hard to implement this attack in practice. In this study, we set the size of training data as a variable from 100 to 800 with a step of 50, and trained and tested the inference model in the hand-hold-input scenario using the SVM classifier. The experimental settings are same as the one in Section 5.1.

Fig. 5 shows the inference accuracy of the keys on number pad against varying training data sizes in the hand-hold-input scenario (similar observations in other two scenarios). Each panel presents the curves for six adjacent keys at the same scale. The inference results show the similar trend for all keys on number pad – the inference rates perform poorly with a small training data size, but will get improved as the training data size increases. Taking the inference of button 1 as an instance, as shown in Fig. 5a, the inference rate on the training data size of 100 is 58.82%; while the training data size increases to 800, the inference rate rises to 90.64%. Besides, these curves show that as the training data size is larger than 600, the inference rates for most keys on the number pad are over 70%, and only little accuracy improvement is gained by increasing the data size. These results suggest that the proposed approach can be a potential threat from a practical view in which it supports the usage of a relatively small set of training data to accurately infer users' input.

5.3. *Scalability to screen size*

This section explores the effect of smartphone screen size on performance of our proposed approach. The recognition results may be influenced by the different sizes of input areas because different smartphones would have different screen sizes. Intuitively, the area of an input button with a smaller screen size is expected to result in decreasing the uniqueness of the key, and may decline the inference accuracy. Here we used three smartphones with different screen sizes: Samsung Galaxy N7102 with a 5.5-inch screen, Huawei Ascend P6 with a 4.7-inch screen, and ZTE V800 with a 3.5-inch screen. We then



(a) 1-6 (b) 7-0/*/#

Fig. 5 – Inference accuracy of each button at varying training data size in hand-hold-input scenario.

employed SVM classifier to conduct input inference tasks across these smartphones in the hand-hold-input scenario. The evaluation methodology is similar to the one in Section 5.1.

Fig. 6 shows the inference results of the keys on number pad against three different smartphone screen sizes. The results indicate that the smartphones with larger screen sizes (Samsung Note 2 and Ascend P6) have a better recognition performance than the smartphone with a smaller screen size (ZTE V800). This is probably due to the reason that buttons with a larger active area have better uniqueness and appear more robust than the ones with smaller active areas, which may achieve better inference accuracy. Besides, the figure reveals that the performance of HUAWEI Ascend P6 is a little superior to that of Samsung Note 2 at a few numbers (i.e., number 3 and 7). The cause of this issue is not obvious; it could be an artifact of our smartphones' motion characteristics and would disappear with different or more smartphone screen sizes. More input actions and analysis would be necessary to determine whether such issues appear consistently for a particular smartphone, but the existence of this issue does suggest that the effects of factors like screen size are not always easy to predict.

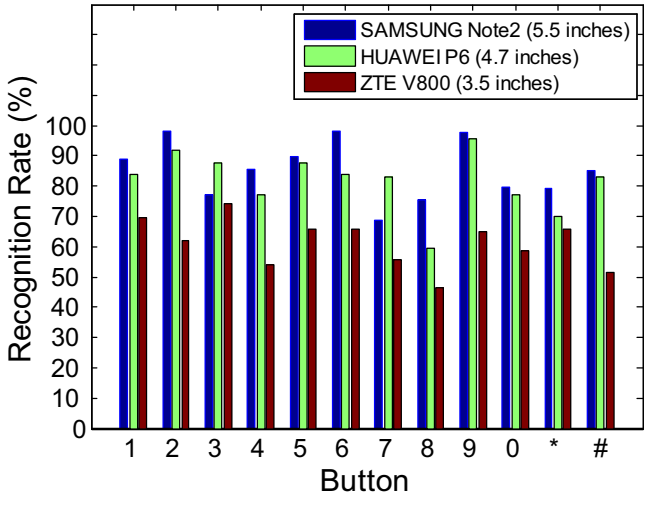


Fig. 6 – Inference accuracy for each button on the number pad across different smartphone screen sizes.

5.4. *Flexibility to sampling rate*

In this experiment, we investigated the effect that data sampling rate has on the performance of our proposed approach. The sampling rate of the sensor data is defined as the number of samples collected in one second, and is used for timing the sensor data in our evaluation. It almost certainly has an effect on the inference performance. But the extent of this effect has never been quantified or measured. Here we considered four typical sampling rates in Android system for sensor data collection – *SENSOR_DELAY_FASTEST*, *SENSOR_DELAY_GAME*, *SENSOR_DELAY_NORMAL*, and *SENSOR_DELAY_UI* – with the sampling rate of 1000 Hz, 50 Hz, 16.67 Hz, and 5 Hz respectively. We then conducted the input detection and input inference experiments in the hand-hold-input scenario, with the same performance assessment method in Section 5.1.

Fig. 7 shows the change of *ACCEsum* curves at different data sampling rates. It can be observed that the change of *ACCEsum* curve becomes less apparent with the decrement of the sampling rate. The curves with high sampling rates (as shown in Fig. 7a and 7b) present evident peaks and three fluctuation processes (as shown in Fig. 2), by which the input actions can be easily and accurately detected. While for the lower sampling rates, the curves become more flat and fewer peaks can be identified.

Besides, Table 2 and Fig. 8 show the results of input action detection and input inference across these sampling rates. It can be easily observed that both the accuracies of input action detection and input inference get worse as the sampling rate decreases. The accuracy with the high sampling rate of *SENSOR_DELAY_FASTEST* and *SENSOR_DELAY_GAME* would be up to 100% of input action detection and over 80% of input inference; even for the low sampling rate of *SENSOR_DELAY_FASTEST*, the accuracy of input inference can be up to over 70% for most keys, which suggests that the proposed approach could be effective in many application settings.

5.5. *Comparison with previous work*

We presented a qualitative comparison in terms of the experimental setting and results among our approach and extant work in the literature, which is shown in Table 3. Cai and Chen (2011) and Owusu et al. (2012) both relied upon the accelerometer for key inference on soft keyboards, and obtained

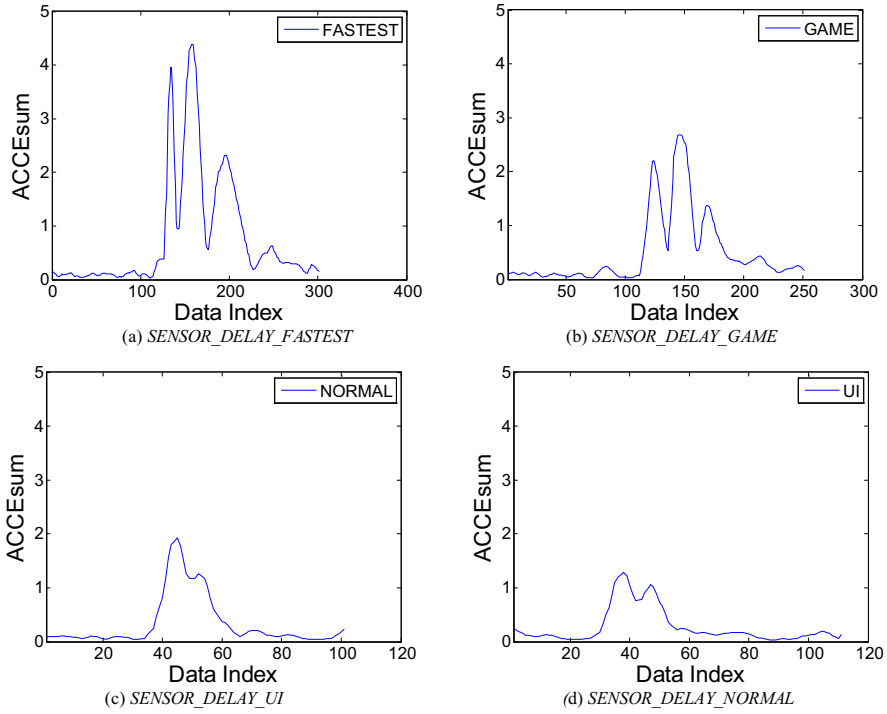


Fig. 7 – *ACCEsum* curves of a single input action across four typical sampling rates.

inference accuracies of 71.5% and 60%. But these results were obtained on a relatively small dataset (449 taps in Cai and Chen, 2011 and 4 users in Owusu et al., 2012), which may be insufficient to obtain a steady result. Aviv et al. (2012) and Cai and Chen (2012) examined the accelerometer-based input inference on larger datasets and across two different input modes, but their success rates were relatively low at 49% and 43%.

Xu et al. (2012) and Miluzzo et al. (2012) exploited both accelerometer and gyroscope for inferring users' input, and obtained the inference accuracies of 67.1% and 40% on Android smartphones. But these results were obtained on relatively small datasets (10 users in Miluzzo et al., 2012 and 3 users in Xu et al., 2012). Additionally, most extant studies explored such attacks on simple application scenarios, which may make them less diagnostic in practice.

However, in our approach, we had a larger subject pool (30), more keypresses (97,200) than previous research did, and developed a combined approach that uses both the accelerometer and magnetometer for achieving better accuracy. We could achieve better inference accuracies of 83.9% for standing mode and 71.4% for walking mode. Besides, our study provides a more thorough investigation on the practicality of such attacks. We conducted performance evaluations against different smartphone screen sizes, sampling rates, and training data sizes, to comprehensively testify its reliability and practicability in practice. To the best of our knowledge, this paper is also the first one to investigate output of magnetometer on Android smartphones. Although this result could be improved, we believe that, at our current performance level, magnetometer data suffice to have rich a potential of compromising smartphone users' privacy.

Table 2 – Accuracy of input action detection against various sampling rates.

Sampling rate	Accuracy
SENSOR_DELAY_FASTEST	100%
SENSOR_DELAY_GAME	100%
SENSOR_DELAY_UI	82%
SENSOR_DELAY_NORMAL	61%

6. Discussion and conclusion

In this paper, we presented a study of analyzing accelerometer and magnetometer data to infer user input on Android

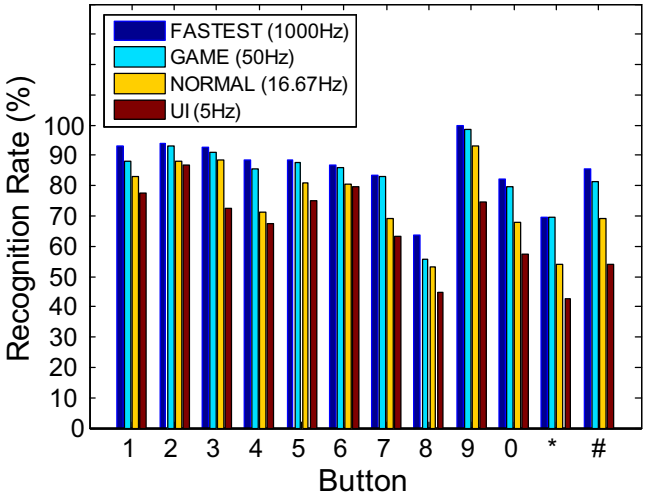


Fig. 8 – Inference accuracies across various sampling rates.

Table 3 – Comparison with previous work that exploited motion sensors.

Source Study	Sensors	Platforms	Data size	Users	Typing scenario	Accuracy
Our work	Accelerometer Magnetometer	Android	97,200 taps	30	Standing	83.9%
					Walking	71.4%
					Placing on table	61.0%
Cai and Chen (2011)	Accelerometer	Android	449 taps	N/A	N/A	71.5%
Cai and Chen (2012)	Accelerometer	Android	47,814 taps	21	Sitting or standing	49%
Owusu et al. (2012)	Accelerometer	Android	2700 taps	4	N/A	60% on 2^{15} trials
Aviv et al. (2012)	Accelerometer	Android	7200 taps	24	Sitting	43% on 5 trials
					Walking	20% on 5 trials
Miluzzo et al. (2012)	Accelerometer Gyroscope	Android and iOS	40,000 taps	10	Sitting or standing	78.7% on iOS 67.1% on Android
Xu et al. (2012)	Accelerometer Gyroscope	Android	N/A	3	N/A	40% on 1 trials

smartphones. Extensive analyses were conducted to examine the reliability and practicability of the proposed approach across various operational scenarios, and showed that the approach could achieve accuracies of 100% and 80% for input-action detection and input-content inference in some cases. Besides, we used soft keypad entries as the test bed in the evaluation, but our approach does not make any assumptions on the entered text – it simply and efficiently project sensor data to screen positions and to input keys. Thus this method could also be utilized to infer other kinds of text (e.g., e-mail or chatting messages), in which an approximate translation is sufficient to enable the method to obtain the complete understanding of sensitive information.

We modeled sensor behavior of accelerometer and magnetometer by capturing both the macroscopic characteristics (i.e., descriptive statistics of sensor data) and microscopic details (i.e., wavelet features from sensor data) for input inference task. First, descriptive-statistical attributions were analyzed to characterize the mean, median, mode, skewness, kurtosis and standard deviation of the sensor data. These features depict the central tendency, distribution, and discrete degree of the macroscopic information of the smartphone-posture change, and could generate a better match for statistical classifiers. Second, wavelet analysis technique was employed to investigate the structure of the sensor data. We extracted the high frequency coefficient of every level and the low frequency coefficients of the last level as the features, which may provide a procedural description of the microscopic content of the smartphone-posture change. Besides, we implemented four types of two-class classifier [Random Forest, Support Vector Machine (SVM), Neural Network, and Nearest Neighbor] to establish the inference model. By comparing various types of classifier, it is helpful to find what classification strategies could express well in characterizing sensor behavior data, and what strategies perform poorly.

We examined the effect of training data size on the inference performance, to investigate the effectiveness and practicability of the proposed technique. The results showed that the inference rates get improved as the training data size increases. In our evaluation, the inference rates for most keys are over 70% when the training data size is up to 500, which indicates this technique supports the usage of a relatively small set of training data to accurately infer users' input, and can be a potential threat to users' privacy on smartphones.

We explored the effectiveness of the proposed approach across three types of smartphone screen sizes. The input inference with larger screen sizes exhibits a better accuracy than that with smaller screen sizes. This may be due to the fact that buttons with a larger active area have better uniqueness and robustness than the ones with smaller active areas. Currently, the mainstream smartphones have screen size larger than 5 inches; while in our evaluation, the inference accuracies on the smartphones with screen sizes larger than 4.7 inches are all over 70%, which further presents a proof-of-concept analysis to demonstrate that this is a real threat.

We analyzed the inference accuracy against different sampling rates of sensor data. The three typical fluctuation processes for each input action are distinct at high sampling rate, but become less apparent with the decrease of the rate. Also the accuracies of input action detection and input inference get worse as the sampling rate decreases. However, it should be noted that our experiments showed that the inference accuracy of most keys are over 80% with a relatively low sampling rate of 16.67 Hz, and remains high when the sampling rate decreases to 5 Hz. This demonstrates the validity of such an attack against different sampling rates of sensor data, even with the default or lower sampling rate for most legitimate applications on smartphones.

This study provided a further empirical analysis of accelerometer and magnetometer data for input inference on smartphones. As of now, the camera, the microphone, the accelerometer, the gyroscope, and the magnetometer are known to help infer PINs on Android smartphones. The fundamental problem here is that sensing is unmanaged on existing Android smartphone platforms. People are still unaware of potential risks of unmanaged sensors on smartphones. To avoid such attacks, we can see researchers put more efforts in sensing management and permission restriction systems on the existing Android smartphone platforms (Dietz et al., 2011; Enck et al., 2014; Fang et al., 2014; Jeon et al., 2012), but these may greatly affect usability and convenience. A more drastic solution is to get rid of the passwords. This could be achieved by biometric features (Karthikeyan et al., 2014) (e.g., fingerprints, face recognition), smart devices with connection to the smartphone (Stajano, 2011) (e.g., smart watches, smart glasses, and smart wristband), and progressive authentication mechanisms (Riva et al., 2012).

Acknowledgements

The authors are grateful to several anonymous reviewers for their helpful comments. This research was supported in part by National Natural Science Foundation of China (61403301, 61221063), China Postdoctoral Science Foundation (2014M560783), Special Foundation of China Postdoctoral Science (2015T81032), Natural Science Foundation of Shaanxi Province (2015JQ6216), Application Foundation Research Program of SuZhou (SYG201444), and Fundamental Research Funds for the Central Universities (xjj2015115).

REFERENCES

Asonov D, Agrawal R. Keyboard acoustic emanations. In: IEEE symposium on security and privacy. IEEE Computer Society; 2004. p. 3–11.

Aviv AJ, Gibson K, Mossop E, Blaze M, Smith JM. Smudge attacks on smartphone touch screens, vol. 10. WOOT; 2010. p. 1–7.

Aviv AJ, Sapp B, Blaze M, Smith JM. Practicality of accelerometer side channels on smartphones. Proceedings of the 28th annual computer security applications conference. ACM. 2012. p. 41–50.

Bowman AW, Azzalini A. Applied smoothing techniques for data analysis. Clarendon Press; 2004.

Cai L, Chen H. TouchLogger: inferring keystrokes on touch screen from smartphone motion. Proceedings of HotSec. 2011.

Cai L, Chen H. On the practicality of motion based keystroke inference attack. Springer; 2012.

Dietz M, Shekhar S, Pisetsky Y, Shu A, Wallach DS. Quire: lightweight provenance for smart phone operating systems. Proceedings of the 20th USENIX conference on Security. San Francisco, CA: USENIX Association. 2011. p. 23–36.

Enck W, Gilbert P, Han S, Tendulkar V, Chun B-G, Cox LP, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst 2014;32:5.

Fang Z, Han W, Li Y. Permission based Android security: issues and countermeasures. Comput. Secur 2014;43:205–18.

Foo Kune D, Kim Y. Timing attacks on pin input devices. Proceedings of the 17th ACM conference on computer and communications security. ACM. 2010. p. 678–80.

Jeon J, Micinski KK, Vaughan JA, Fogel A, Reddy N, Foster JS, et al. Dr. Android and Mr. Hide: fine-grained permissions in android applications. Proceedings of the second ACM workshop on security and privacy in smartphones and mobile devices. Raleigh (North Carolina, USA): ACM. 2012. p. 3–14.

Karthikeyan S, Feng S, Rao A, Sadeh N. Smartphone fingerprint authentication versus PINs: a usability study. CMU-CyLab-14-012. 2014.

Liu M. A study of mobile sensing using smartphones. Int. J. Distrib. Sensor Netw 2013;2013:272916.

Marquardt P, Verma A, Carter H, Traynor P. (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. Proceedings of the 18th ACM conference on computer and communications security. Chicago, Illinois, USA: ACM. 2011. p. 551–62.

Miluzzo E, Varshavsky A, Balakrishnan S, Choudhury RR. Tapprints: your finger taps have fingerprints. Proceedings of the 10th international conference on mobile systems, applications, and services. ACM. 2012. p. 323–36.

Mylonas A. Smartphone spying tools. Master's thesis]. Royal Holloway, University of London. 2008.

Mylonas A, Meletiadis V, Tsoumas B, Mitrou L, Gritzalis D. Smartphone forensics: a proactive investigation scheme for evidence acquisition. In: Gritzalis D, Furnell S, Theoharidou M, editors. Information security and privacy research. Heidelberg: Springer Berlin; 2012. p. 249–60.

Mylonas A, Meletiadis V, Mitrou L, Gritzalis D. Smartphone sensor data as digital evidence. Comput. Secur 2013;38:51–75.

Ostle B. Statistics in research. 1963.

Owusu E, Han J, Das S, Perrig A, Zhang J. Accessory: password inference using accelerometers on smartphones. Proceedings of the twelfth workshop on mobile computing systems & applications. ACM. 2012. p. 9–14.

Raguram R, White AM, Goswami D, Monrose F, Frahm J-M. iSpy: automatic reconstruction of typed input from compromising reflections. Proceedings of the 18th ACM conference on computer and communications security. Chicago (Illinois, USA): ACM. 2011. p. 527–36.

Riva O, Qin C, Strauss K, Lymberopoulos D. Progressive authentication: deciding when to authenticate on mobile phones. In: USENIX security symposium; 2012. p. 301–16.

Ross SM. Introduction to probability and statistics for engineers and scientists. Academic Press; 2009.

Schlegel R, Zhang K, Zhou X-Y, Intwala M, Kapadia A, Wang X. Soundcomber: a stealthy and context-aware sound trojan for smartphones. Proceedings of NDSS. 2011. p. 17–33.

Simon L, Anderson R. PIN skimmer: inferring PINs through the camera and microphone. Proceedings of the third ACM workshop on security and privacy in smartphones & mobile devices. ACM. 2013. p. 67–78.

Spreitzer R. PIN skimming: exploiting the ambient-light sensor in mobile devices. Proceedings of the 4th ACM workshop on security and privacy in smartphones and mobile devices. Scottsdale (Arizona, USA): ACM. 2014. p. 51–62.

Stajano F. Pico: no more passwords!. In: Security protocols XIX. Springer; 2011. p. 49–81.

Vuagnoux M, Pasini S. Compromising electromagnetic emanations of wired and wireless keyboards. In: USENIX security symposium; 2009. p. 1–16.

Xu N, Zhang F, Luo Y, Jia W, Xuan D, Teng J. Stealthy video capturer: a new video-based spyware in 3g smartphones. Proceedings of the second ACM conference on Wireless network security. ACM. 2009. p. 69–78.

Xu Z, Bai K, Zhu S. Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. Proceedings of the fifth ACM conference on security and privacy in wireless and mobile networks. ACM. 2012. p. 113–24.

Zhu K, San Wong Y, Hong GS. Wavelet analysis of sensor signals for tool condition monitoring: a review and some new results. Int. J. Mach. Tools Manuf 2009;49:537–53.

Chao Shen is currently an Assistant Professor in the School of Electronic and Information Engineering, Xi'an Jiaotong University of China. His research interests include insider/intrusion detection, behavioral biometric, and measurement and experimental methodology.

Shichao Pei is currently an undergraduate student in the School of Software Engineering, Xi'an Jiaotong University of China. His research interests include mobile security, machine learning, and intrusion detection.

Zhenyu Yang is currently an undergraduate student in the School of Electronic and Information Engineering, Xi'an Jiaotong

University of China. His research interests include mobile security, machine learning, and intrusion detection.

Xiaohong Guan received the B.S. and M.S. degrees in automatic control from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical engineering from the University of Connecticut, Storrs, in 1993. He is currently a Cheung Kong Professor of Systems Engineering and the Dean of School of Electronic and Information Engineering in Xi'an Jiaotong University. His research interests include allocation and scheduling of complex networked resources, network security, and sensor networks.