Secure Keyboards Against Motion Based Keystroke Inference Attack

Shaoyong Du, Yue Gao, Jingyu Hua, and Sheng Zhong(✉)

State Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology,
Nanjing University, Nanjing, China
shaoyong.du.cs@gmail.com, njucsmoon@gmail.com,
{huajingyu,zhongsheng}@nju.edu.cn

Abstract. Nowadays, attackers seek various covert channels to access the users' privacy on the mobile devices. Recent research has demonstrated that the built-in motion sensors can be exploited to monitor the users' screen taps and infer what they have typed. This paper presents several practical and convenient countermeasures against this attack in terms of the soft keyboard. We find that this attack is sensitive to the motion noise of the mobile device and the layout variation of the soft keyboard. We, thus, present two kinds of countermeasures against this attack by introducing vibration noise in sensor readings and dynamics in the keyboard layout, respectively. We implement these countermeasures on Android platform and recruit 20 volunteers to evaluate these countermeasures' effectiveness and usability on both the smartphones and tablets. The results show that the proposed countermeasures can effectively reduce the attackers' keystroke inference accuracy without significantly hurting the typing efficiency.

Keywords: Keystroke inference attack · Motion sensor · Mobile device · Countermeasure · Soft keyboard

1 Introduction

The mobile devices' popularity makes themselves become one of the key targets of the attackers. To collect the users' privacy on the mobile devices, the attackers seek various covert channels, for example, the on-board sensors. It has been demonstrated that the cameras [9], gyroscopes [12], microphones [16,19] and ambient-light sensors [21] all can be used directly to exhibit the users' sensitive information.

Recent research [1,2,4,5,8,14,17,23] has demonstrated that the motion sensors can be utilized to record the user's screen taps so as to further infer what the

This work was supported in part by the Jiangsu Province Double Innovation Talent Program and in part by the National Natural Science Foundation of China under Grant NSFC-61300235, Grant NSFC-61321491, Grant NSFC-61402223, and Grant NSFC-61425024.

user has typed on the soft keyboard, which is called the motion based keystroke inference attack (MoBaKIA attack). It is first proposed by Cai *et al.* in [4] and they give a detailed presentation about it in [5]. Owusu *et al.* have proved that it is possible to infer the 6-character passwords in as few as 4.5 trails with the accelerometer readings [17]. Meanwhile, it can also get high accuracy on inferring English words on both the smartphone and tablet [14]. Aviv *et al.* enhance MoBaKIA attack with the polynomial fitting and signal processing techniques, making it possible to infer 40% of the patterns and 20% of the PINs within 5 attempts when users are walking [2]. A real Android Trojan application about MoBaKIA attack, TapLogger [23], has been implemented by Xu *et al.*

Nowadays, besides the standard soft keyboard, the users can utilize a variety of novel soft keyboards in the mobile application market, such as Swype[1] and Dynamic Keyboard[2], to type something on the mobile devices. Despite the innovative user experience provided, these soft keyboards pay little attention to defending against the covert channel attacks on the mobile devices, for example, MoBaKIA attack. They are still vulnerable to MoBaKIA attack, since the operations on them depend heavily on the entered content, which can be recorded by the motion sensors.

Prior work [2,14,17,23] has presented some countermeasures on MoBaKIA attack such as reducing the sampling rate, requiring specific permission on the motion sensors and so on, but they leave these countermeasures alone without any further implementation or evaluation. Some researchers attempt to provide a dynamic, flexible and fine-grained control on the access to the motion sensors [3,6,7,13,18,22,24]. However, these countermeasures are highly specified, which are implemented on the specific operating systems and require the alterations on the Android framework as well as the Linux kernel that can only be done by *ROOT*. With rooted Android, the users will encounter some practical problems, such as invalidating warranty and causing update issues[3]. As a result, these countermeasures cannot be widely applied to current Android versions.

To make the countermeasures more practical, a number of researchers seek to defend against MoBaKIA attack on the application layer in terms of keyboard [10,11,15,25–27], instead of security framework. Since these keyboards are much different from the standard keyboard, the users have to spend much longer time to learn about them and be more concentrative when they type.

In this paper, we take both the countermeasure's effectiveness and usability into account, and propose some practical and convenient countermeasures against MoBaKIA attack in terms of keyboard.

When we analyze the process of launching MoBaKIA attack, we have the following two observations:

I. *MoBaKIA attack is sensitive to the shaking noise of the mobile device.* When the motion data are used to infer the typed content, the motion noise of the

[1] https://play.google.com/store/apps/details?id=com.nuance.swype.trial.

[2] https://play.google.com/store/apps/details?id=com.alastairbreeze.dynamickeyboard&hl=en.

[3] http://betanews.com/2013/10/01/5-reasons-not-to-root-android/.

mobile device has a great influence on the inference result. For example, the shaking caused by the user's movement usually has a greater influence on the mobile device's state than just tapping the screen, which makes it more difficult to infer the typed content [23].

II. *MoBaKIA attack relies on the fixed screen area and constant layout of the soft keyboard.* It is the soft keyboard's fixed position and constant layout that enable MoBaKIA attack to infer the content that the user has typed with high accuracy [4]. Therefore, the fixed mappings from the keys to their screen locations are very critical to MoBaKIA attack. We can try to break these fixed mappings by dynamically modifying the soft keyboard's layout.

Our countermeasures against MoBaKIA attack are just based on these two observations. In this paper, we make the following contributions:

1. Driven by the first observation, we propose our first kind of countermeasures that we take advantage of the vibrator in the mobile device to add noise to the motion sensor readings. We make a detailed analysis of the vibration noise in terms of correlation and frequency spectrum, and we find that it can be difficult for the attackers to remove this noise.
2. Based on the second observation, we propose the second kind of countermeasures, which defends against MoBaKIA attack through dynamically modifying the layout of the keyboard. We make use of the entropy theory to analyze and select the modification strategies. Meanwhile, we present several effective strategies to reduce the side effects on the usability.
3. Based on Google's sample soft keyboard project, we implement these countermeasures and evaluate their effectiveness and usability on both the smartphones and tablets. With the experiment conducted among 20 volunteers and more than 90,000 keystrokes collected, we can see that the proposed countermeasures can effectively reduce the attackers' keystroke inference accuracy without significantly affecting the typing efficiency.

As for the soft keyboard, it can be unavoidable to sacrifice some usability to protect the user's privacy. We try our best to keep the soft keyboard's usability while modifying it. What is more, we do not want to persuade the users to use our countermeasures all the time. It depends on the specific contexts. When the users type their account numbers, passwords and other sensitive information, it could be acceptable for them to sacrifice the usability to protect their private information. At other time, they can still use the standard keyboard without any modification.

The rest of this paper is organized as follows. We start our work with the realization of MoBaKIA attack in Sect. 2. In Sects. 3 and 4, we give a detailed description of our two kinds of countermeasures against MoBaKIA attack respectively. We present our experiments and evaluations of these countermeasures in Sect. 5. We show the related work in Sect. 6. In Sect. 7, we conclude the whole paper.

2 MoBaKIA Attack Introduction

MoBaKIA attack can be considered as a problem of classification, and it can be divided into two stages: training stage and inferring stage. Through a well-designed malicious application (e.g., a mobile game application), the attacker can collect the labeled motion data sequences, which have been associated with the accurate touched areas on the mobile device. And then, the attacker builds the classifiers with these labeled motion data sequences. When the malicious application runs in the background, it stealthily accesses the motion data sequences when the user types something on the mobile device. With these unlabeled motion data sequences, the attacker can infer the original content that the user has typed.

To make an evaluation of our countermeasures' effectiveness, we implement and launch MoBaKIA attack on Android platform first:

Step 1. **Motion sensor selection:** We use the accelerometer and orientation sensor, which are utilized in prior work [2,4,5,14,17,23], to launch this attack.
Step 2. **Configuration of motion sensor:** To get a detailed presentation about the touch event, we set the sensor's receiving rate to the fastest one, 100 Hz.
Step 3. **Touch event extraction:** We record the exact time when a key is pressed and released so that we can easily locate the touch event in the motion data.
Step 4. **Feature selection:** Following the prior work [14,17], we extract both time domain and frequency domain features from the motion data.
Step 5. **Classifier selection:** Different classification algorithms' effect on MoBaKIA attack has been compared by Owusu *et al.* in [17] and they have claimed that the Random Forest classification algorithm has the best effect, so in this paper we adopt the Random Forest classification algorithm.

In the following parts, we will give a detailed description about our countermeasures which are based on the observations presented in Sect. 1. In this paper, we only focus on the alphabetic soft keyboard. Other soft keyboards can utilize the same strategies.

3 Countermeasure Based on Observation I

The prior work [23] has pointed out that MoBaKIA attack is sensitive to the motion noise of the mobile device. Based on this point, we can add some noise to the motion data before they are delivered to the registered *SensorListeners*. There are two measures to add noise. The first one is to modify the real sensor data by the programs. Raghavan *et al.* in [18] propose to add Gaussian noise to the acceleration data when the soft keyboard is activated. However, it requires the modifications of current operating systems, contradictive to our intention. The second one is to dynamically change the state of the mobile device. It can depend on the users to make extra motion when they operate the mobile devices, which can be inconvenient for the users. A more convenient way is to utilize the vibrator in the mobile device. Once a key is pressed, the vibrator is started. In Fig. 1, we can see that the noise produced by the vibrator can absolutely disturb the motion sensor readings. In this part, we mainly consider the accelerometer's readings, since the vibration noise has the similar influence on the other motion sensors' readings, according to our observation.

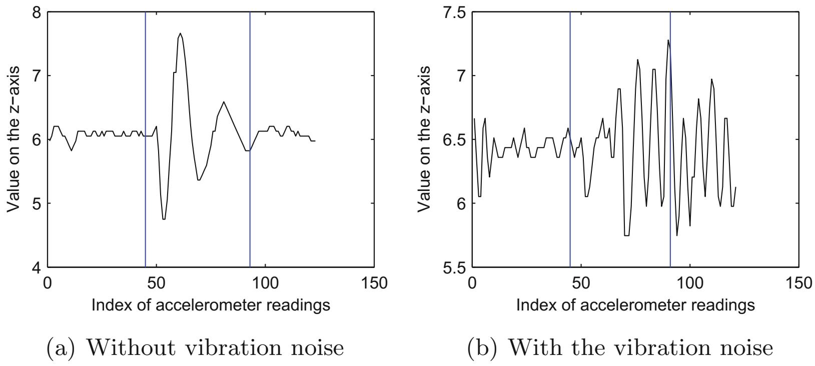


Fig. 1. Comparison of the accelerometer readings on the z-axis with the key "U" typed (The blue lines mark the time when the touch events start and stop.) (Color figure online)

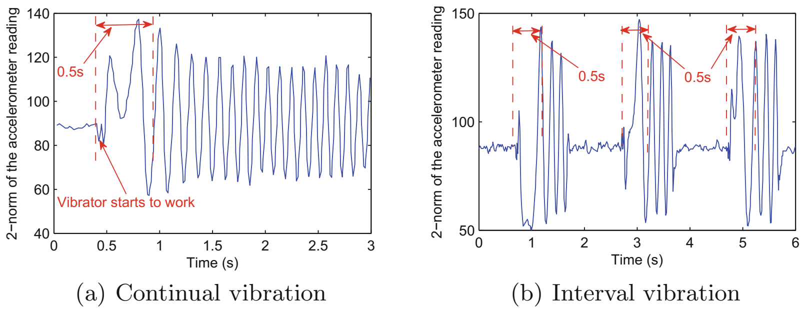
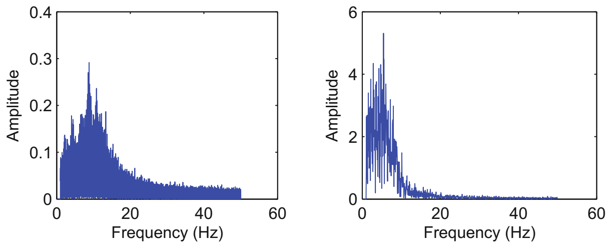


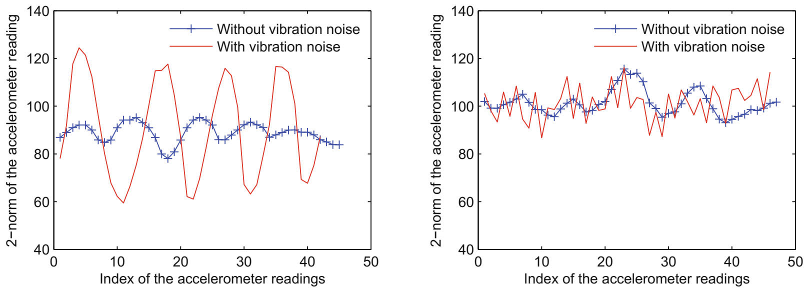
Fig. 2. Vibration noise analysis

Before fully utilizing the vibration noise, we start with an investigation to learn about the features of the vibration noise. In Fig. 2(a), we can see that the noise is periodic when the vibrator keeps working for some time, except for the first 0.5 s when the vibrator starts to work. Our extra experiments' result in Fig. 2(b) validates our observation. Based on our observation, we can see that the correlation coefficient of the vibration noise in the first 0.5 s each time follows the uniform distribution over $[-1, 1]$, making it hard to filter out this kind of noise. With 27,307 touch events collected, we obtain the average touching time which is just 0.48 s, shorter than 0.5 s. Therefore, we can take advantage of the irregular vibration noise in the first 0.5 s to disturb the original motion data about the touch events. Moreover, with the spectral analysis, we can see that the frequency spectrum of the vibration noise in Fig. 3(a) is much similar to that of the touch events in Fig. 3(b). Therefore, even with the band-pass filter, it is still difficult to remove the vibration noise from the motion sensor readings.



(a) Touch event (b) Vibration noise

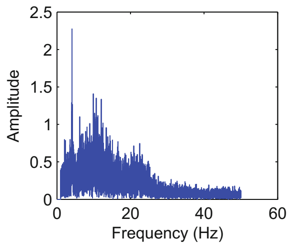
Fig. 3. Spectral analysis of the touch event and the vibration noise



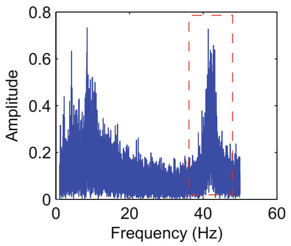
(a) Motorola XT910 (b) Mi One S

Fig. 4. Comparison of the touch events with different smartphones

Now, we test the vibration noise on different smartphones to gain the further insights about it. We find that it highly relies on the smartphones, which can be classified into two categories. To make a brief presentation about them, we take the representative one from each category in this part. Compared with the motion data without noise, some of the noise, in Fig. 4(a), can be strong enough to disturb the original motion data, while some of them, in Fig. 4(b), cannot. Moreover, with the spectral analysis of the touch events with different smartphones' vibration noise in Fig. 5, we can see that some vibration noise's



(a) Motorola XT910



(b) Mi One S

Fig. 5. Spectral analysis of the touch events with different smartphones' vibration noise (Mi One S vibration noises frequency is marked by the red dash rectangle.) (Color figure online)

frequency in Fig. 5(b) is so high that it can be filtered out by the band-pass filter, when compared with the touch events' frequency in Fig. 3(a). Therefore, this countermeasure's effect heavily depends on the smartphones. In the next section, we introduce some countermeasures that can be used on the whole smartphones.

4 Countermeasures Based on Observation II

The soft keyboard's fixed position on the screen and constant layout establish the fixed relations between letter keys and screen locations, which makes it possible to infer the typed information with the motion data. Therefore, we can dynamically adjust the keyboard's layout to break the fixed relations so as to increase the difficulty of MoBaKIA attack. Before we describe our detailed schemes, we first present the principle guiding our design.

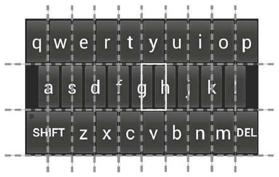


Fig. 6. The standard soft keyboard layout

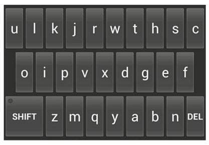


Fig. 7. Randomize the layout without improvement

In the standard soft keyboard in Fig. 6, there are a total of 28 keys. We represent them by $k_1, k_2, \cdots, k_{28}$. If the user taps on k_i, a classifier may falsely recognize it as a different key based on the eavesdropped motion data. We denote by P_{ij} the probability that the tapped key k_i is falsely recognized as k_j. To break the fixed layout of the keyboard, we aim to find a randomization strategy to dynamically change the representing letter of each key. Supposing that after applying this strategy, the probability that k_i represents letter l_j is $Q_{k_i l_j}$. Note that the value of P_{ij} is determined by the locations of k_i and k_j, which have nothing to do with the randomization of the representing letters. Taking the inference error into consideration, if the classifier claims that the user has tapped on k_j in the last tapping event, the probability that letter l_s is entered is

$$P_{k_j \to l_s} = \sum_{i=1}^{i=28} (P_{ij} \times Q_{k_i l_s}). \tag{1}$$

The values of $P_{k_j \to l_s}$ for $s = 1, 2, \cdots, 28$ can be regarded as the probability distribution of a stochastic event, which is denoted by E_{k_j}. Then, the entropy of this event is

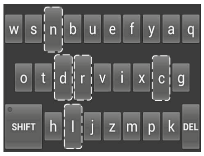
$$H(E_{k_j}) = -\sum_{s=1}^{s=28} (P_{k_j \to l_s} \times log(P_{k_j \to l_s})). \tag{2}$$

As we know that a greater entropy indicates that the stochastic event is more difficult to predict. Thus, we can claim that the greater the value of $H(E_{k_j})$ is, the more difficult it is for the attacker to predict which letter has been typed when the MoBaKIA attack classifier infers that k_j has been tapped on. Therefore, we should find a randomization strategy of the letter keys to maximize $H(E_{k_j})$ for each i for security. Usability has a great influence on the users' acceptance of these strategies. Therefore, when we design these strategies, we take full consideration of the usability and try to make the strategies user-friendly by following users' typing habits.

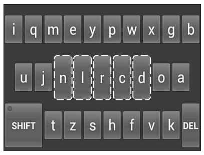
4.1 Completely Randomize the Layout

When we want to dynamically adjust the keyboard's layout, the most direct way is to completely randomize it. However, when the keyboard's layout is completely randomized, it makes it difficult for the user to pick up the target key. Therefore, in this part, besides the basic strategy to randomize the layout, we come up with some improved strategies to accelerate the user's typing speed.

Basic Strategy. According to the entropy theory, $H(E_{k_j})$ reaches its maximal value when $P_{k_j \to l_s} = 1/28$ for all $s = 1, 2, \cdots, 28$. We can easily prove that this condition is satisfied when $\forall i$, $Q_{k_i l_1} = Q_{k_i l_2} = \cdots = Q_{k_i l_{28}} = 1/28$, which means that the letters are completely randomly distributed to the keys. Therefore, our first countermeasure follows this observation and completely randomizes the layout of the alphabet keyboard as shown in Fig. 7. Note that our randomization does not include "Shift" and "Del" keys. When this strategy is applied, once a touched screen area is identified, the attacker can predict nothing but randomly select one letter to report. The probability that one input letter is inferred correctly is *3.85%* ($\frac{1}{26}$). This is the best strategy as for security. Nevertheless, this solution may make it difficult for the user to find out the target letter, slowing



(a) Decentralized



(b) Centralized

Fig. 8. Randomize the layout with improvement (keys marked with the white dash rectangles are the keys with high probabilities to be pressed.)

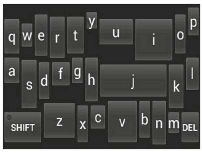


Fig. 9. Randomly resize the keys

down the typing speed. In our opinion, it can be acceptable for the users when they are typing very sensitive information on the screens such as bank accounts and passwords.

Improved Strategies. To provide a comprehensive and acceptable protection on the typed content no matter whether it is sensitive or not, we need to work further on the basic strategy to improve its usability. With the occurrence frequencies of words and the letter sequence entered by the user, we can predict the next letters with some probability and highlight them so as to speed up typing. The first improvement that we make is to resize the letter keys according to the probabilities that they are going to be pressed with. The higher probability the letter key is going to be pressed, the larger its size will be made. Once the user types one letter, we follow the basic strategy to randomize the letter keys' positions and then resize each key according to the associated probability. It keeps the basic strategy's effectiveness. Figure 8(a) shows the soft keyboard when the letter key "A" is touched. However, the predicted letter keys are decentralized, which is still hard for the user to find out his target letter key. Therefore, we can add an additional step to move the predicted letter keys with high probabilities to the center of the soft keyboard so that the user can pick up his desired letter key more easily. The initial keyboard's layout is just as the one of the basic strategy. With more letters typed, the attacker still can learn nothing since he cannot build a robust classifier, in advance, to infer the associated letters, except for random guess. Therefore, this improved strategy still keeps the basic strategy's effectiveness. Figure 8(b) shows the soft keyboard when the letter key "A" is touched in this case. To make the countermeasures not only effective in resisting MoBaKIA attack but also user-friendly, we should consider more information, such as the keys' proximate relations.

4.2 Randomly Resize the Keys

Randomly resizing the keys can adjust the soft keyboard's layout as well as keep the proximate relations between the keys in some degree in Fig. 9. Therefore, we treat it as one of the promising strategies, but we should pay attention to its side effects such as it can be difficult for the user to touch the key accurately and it may be impossible to present the whole soft keyboard on the screen. With the minimum key size considered, we resize the keys. We firstly randomize the keys' width in each row and guarantee that they cover the whole space in the horizontal direction. And then, we modify the keys' height in each row. When we modify the keys' width and height, we also adjust the keys' positions to avoid the block between the keys. Based on the remained space in the vertical direction, we randomly modify all the keys' y-coordinate values in the first row so that all the keys are randomly shifted in the vertical direction.

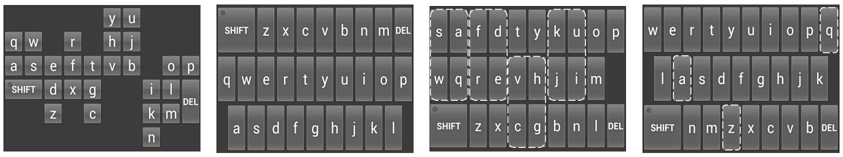
4.3 Heuristically Adjust the Layout

The neglect of the users' habits formed on the standard soft keyboard leads to the bad user experience. To guarantee the usability of the soft keyboard, we focus our attention on the users' habits. In Sect. 4.2, we randomly resize each key. It keeps the proximate relations between the keys in some degree. However, when the user tries to type the key with the small size, he may accidentally type the key nearby instead of it. It is just because of the key's small size. In this part, we still adjust the keyboard's layout, but we adjust the keys' positions in the local areas around their original positions, following some specific regulations. Based on the entropy theory discussed in Sect. 4, it is easy to see that the larger the key's adjustment area is, the larger the key's entropy value is. Driven by this observation, we come up with the following strategies:

H1. *Shift each column within a random distance.* We make uniform modification of the whole letter keys' height so that the original 3 rows in Fig. 6 are divided into 6 rows. In this way, each column can be shifted within a random distance in the vertical direction in Fig. 10(a).
H2. *Randomize the row order.* We just randomly resort the three rows each time a key is pressed just as shown in Fig. 10(b).
H3. *Randomize the keys within a local area.* We dynamically take the four keys nearby as a group, and randomly rearrange them among these four positions, as shown in Fig. 10(c).
H4. *Randomly shift the keys in each row.* We randomly select a constant for each row and all the keys in the same row are moved with the randomly selected constant in a circle, just as shown in Fig. 10(d).

We can see that there can be various heuristic strategies to adjust the soft keyboard's layout and we just enumerate some of them. However, we want to find a representative one among them. Now we give analysis about the effectiveness of the above schemes based on the entropy theory that we described earlier.

We first divide the keyboard area into a grid of 3×10 *cells* as shown in Fig. 6. Each cell represents a key (i.e., the minimal recognizable area that we consider in this analysis). For simplicity, we think that the keystroke inference accuracy reaches 100%, i.e., $P_{ii} = 1$ and $P_{ij} = 0$ $(i \neq j)$. We take the key $k_{example}$ highlighted in Fig. 6 as an example to compare the effectiveness of different schemes.



(a) H1 (b) H2 (c) H3 (d) H4

Fig. 10. Heuristic strategies ((c) is a moment that the keys marked within the white dash rectangles are randomly rearranged. The keys marked with the white dash rectangles in (d) are the first keys in each row in the standard soft keyboard.)

We first consider the strategy H1 shown in Fig. 10(a). In this scheme, if $k_{example}$ is considered to be tapped on, the input letter has three possibilities: *"Y"*, *"H"*, and *"V"*. We can derive that $Q_{k_{example}Y} = 1/3$, $Q_{k_{example}H} = 1/3$ and $Q_{k_{example}V} = 1/3$. Therefore, according to Eq. 2, the entropy value is $H(k_{example}) = -3 \times \frac{1}{3}log\frac{1}{3} = 1.58$. Similarly, we can compute $H(k_{example})$ with the other heuristic strategies H2–H4, which are equal to 1.92, 2.50, and 3.17, respectively. We can see that H4 have the largest entropy value, so we choose H4 as the representative of the heuristic strategies and make further evaluation on it in the following section.

5 Experiment and Evaluation

From the aforementioned countermeasures, we select the representative ones and implement them on Android platform to evaluate their effectiveness and usability. For simplicity, we index all these countermeasures as Table 1. We use *BASIC*, the standard soft keyboard without any modification, as a baseline to make comparison with other countermeasures.

Table 1. Indexed countermeasures

Index	Countermeasure
BASIC	The standard soft keyboard without modification
C1	Add noise with the vibrator
C2	Randomize the layout without improvement
C3	Randomize the layout with improvement decentralized
C4	Randomize the layout with improvement centralized
C5	Randomly resize the keys
C6	Randomly shift the keys in each row

1. **Participants:** We recruit 20 volunteers (10 males, 10 females), whose average age is *24.50 years old*, to make evaluations of all these countermeasures on both the smartphones and tablets.

2. **Test Devices:** We conduct our experiments on 5 different smartphones (*Motorola XT910*, *Mi One S*, *Mi 2*, *Samsung Galaxy SII* and *Samsung Galaxy S4*) and 2 different tablets (*Samsung GT-N8000* and *Huawei MediaPad 10*).
3. **Further Illustrations:** During the experiments, the participants sit when they are typing and all the devices are kept in portrait mode. We do not have any further restrictions on the participants' typing manners so that they can follow their own habits to keep the mobile device and type the content on it.

5.1 Effectiveness

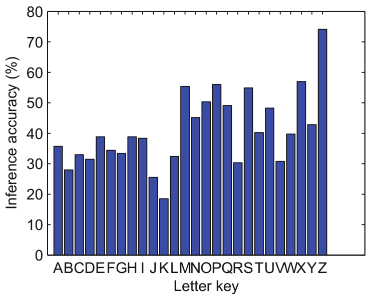
Settings. As our purpose in this paper is to resist MoBaKIA attack, we evaluate the countermeasures' effectiveness at first. Prior work [1,2,4,14,17,23] focuses on the keystroke inference accuracy, since it can directly affect the target key's rank among the candidates, which has a significant influence on the number of attempts needed to correctly infer the whole content. Therefore, we also focus on the keystroke inference accuracy when we evaluate these countermeasures' effectiveness.

We require the participants to touch the keys one by one ("*A*" → "*B*" → "*C*" → ... → "*Z*") in each round and go on for about 10 rounds with each countermeasure. With the captured motion data, we build the personal keystroke inference models for each participant and make *K-fold cross validations* (K=10), which are used in prior work [1,2,5,14,17], to obtain the inference accuracy on each key with each countermeasure.

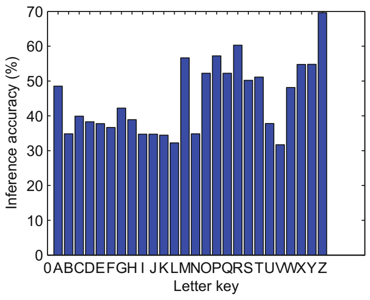
We do not take *C3* and *C4* into account in this subsection, as they are just made to improve the usability of *C2*, which does not work in this experiment. Just as we have talked in Sect. 3, *C1*'s effect highly depends on the mobile devices, in this part, we only evaluate it on the mobile devices whose vibration noise can be similar to that in Figs. 4(a) and 5(a).

Effectiveness on Smartphone. We firstly conduct the effectiveness evaluation on the smartphones. Before we begin to evaluate these countermeasures, we deploy *BASIC* at first to show MoBaKIA attack's keystroke inference accuracy without any countermeasures. Following the steps presented in Sect. 2, we obtain the result in Fig. 11(a), similar to the result of Owusu *et al.* in [17]. We can see that the inference accuracy of the letter keys varies, which is mainly because of the keys' positions in the soft keyboard. The average inference accuracy is *40.84%*.

Compared with the average keystroke inference accuracy obtained without any countermeasures, all the average keystroke inference accuracies are reduced in Fig. 12 when we apply our countermeasures. We can see that *C1* does have some influence on the keystroke inference accuracy. However, since we do not have any restrictions on the participants' typing manners, *C1*'s effect varies among the participants, which leads to the inconspicuous reducing on the average keystroke inference accuracy. Although *C1*'s effect is not very significant, under some conditions it can be still effective to defend against MoBaKIA attack as it



(a) On smartphone



(b) On tablet

Fig. 11. Keystroke inference accuracy of standard soft keyboard

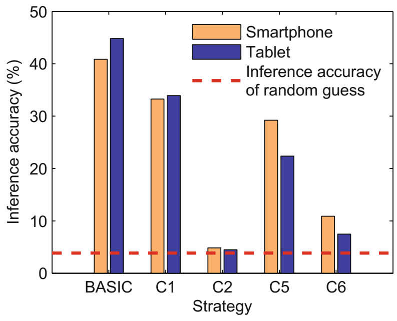


Fig. 12. Comparison of average inference accuracies on smartphone and tablet in the real scenario (*C3* and *C4* are excluded since they do not work when the keys are pressed one by one, "*A*" → "*B*" → ... → "*Z*".)

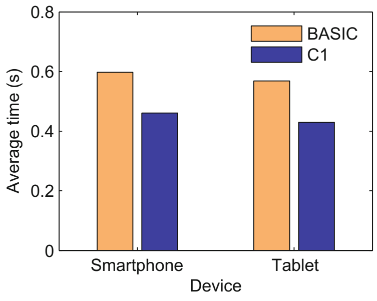


Fig. 13. Comparison of the average time of typing one letter between *BASIC* and *C1*

makes it hard to detect when a type starts and stops when the user types some information continually.

Among these countermeasures, *C2* is the most effective one to defend against MoBaKIA attack, as it reduces the average inference accuracy to about *3.85%*, the inference accuracy of random guess. Just as we have talked above, it may cost the user a much longer time to pick up the target letter key. If the content that is going to be typed is sensitive, it can be acceptable for the user in consideration of security. The second candidate is *C6*. Compared with *C2*, it preserves some proximate relations on the standard soft keyboard, but it may not be as effective as the first one, since it just reduces the accuracy to *10.89%*. We can see that *C5* does have some influence on the keystroke inference accuracy, but its influence is not very significant. It is mainly because that when we dynamically modify the keys' size, the majority of the keys' position do not change dramatically.

Therefore, each letter key is still located in a fixed area, compared with *C6* in which each letter key can appear at 7 possible areas at least.

Effectiveness on Tablet. Besides the smartphones, we also make an evaluation on the tablets. The process on the tablets is the same as that on the smartphones. With *BASIC*, we obtain the original keystroke inference accuracy as shown in Fig. 11(b). The average value is *44.82%*. The evaluation result of the proposed countermeasures on the tablets is also presented in Fig. 12, from which the similar result can be achieved.

Comparing the evaluation results on the smartphones and tablets, we can see that the average keystroke inference accuracy of *BASIC* on the tablets is higher than that on the smartphones, which is mainly because that the key's size on the tablets is larger than that on the smartphones, making it easier to infer the target key, just as Owusu *et al.* state in [17]. It is also the keyboard's larger display area on the tablets that makes *C2*, *C5* and *C6* more effective on the tablets, as the key's variation that can be made is much larger on the tablets than on the smartphones.

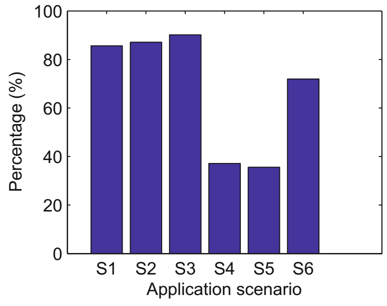
5.2 Usability

When we make some modifications of the soft keyboard's layout, we need to take the usability into account. In this paper, we pay attention to the time cost to type. In Fig. 13, we can see that *C1* does not introduce extra time when compared with *BASIC*. Therefore, in this part, we do not take *C1* into account and just list time statistics on *BASIC* and the countermeasures that modify the keyboard's layout.

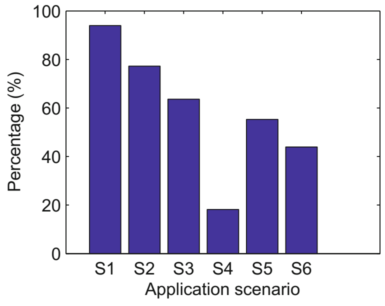
Table 2. Application scenarios

Index	Scenario
S1	Entering account numbers and passwords
S2	Writing SMSes and making phone calls
S3	On-line chatting through the mobile social network Apps
S4	Posting and replying on the social networks
S5	Sending E-mails
S6	Searching through the Internet

We start our evaluation with a survey, which fully considers the application scenarios in Table 2. It is about the keyboard's popular application scenarios as well as the sensitive information that the participants consider. There are 132 participants (96 males and 36 females, the average age of which is 22.27 years old) taking part in this investigation. The obtained result is presented in Fig. 14.



(a) Popularity



(b) Importance of entered content

Fig. 14. Voting results about the keyboard's application scenarios in Table 2

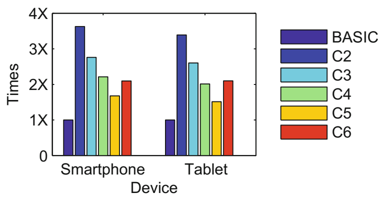


Fig. 15. Average typing time cost of different keyboards, compared with *BASIC*

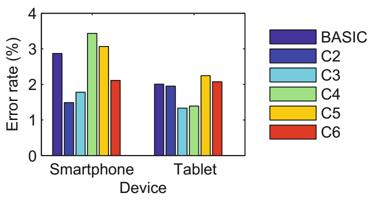


Fig. 16. Typing error with different keyboards

In Fig. 2(a), we can see that the top 3 popular scenarios are *S3*, *S2* and *S1*. Under these scenarios, it is common that the content cannot be too long to type at one time. In this way, we design 10 sentences for testing, the average length of which is 11 words (about 56 letters per sentence, except for the blanks). 20 participants use the devices that we provide for testing. Considering their familiarity with the keyboard as well as the content to be typed, which can have some influence on the time cost to type, we demand them to repeatedly type the sentence for 6 times with each countermeasure, and then we calculate and compare the average typing time. We take into account the time that they spend on correcting the mistakes that they accidentally made. The detailed result is presented in Fig. 15. *C2* is very effective to defend against MoBaKIA attack, but it costs the longest time to enter which is about 3.63X times longer than that cost by *BASIC* in Fig. 15 on smartphones, so it can be applied only when users are entering some sensitive information such as account numbers and passwords. With the improvement on usability, in Fig. 15, *C4* can reduce the time to be only about 2.21X times of that with *BASIC*. In this way, no matter what the users enter, they can choose *C4* that not only can resist MoBaKIA attack but also does not cost too much time. However, if users think that the content to be entered is long but not so sensitive, they can also select other secure strategies instead, like *C5*.

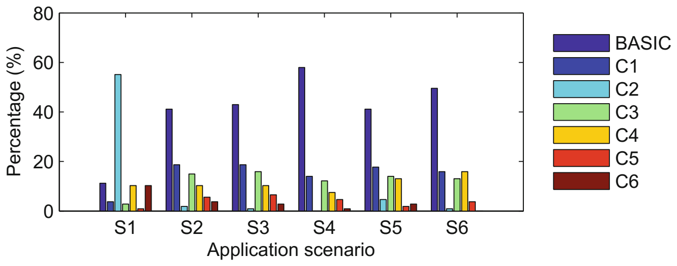


Fig. 17. Voting result about the keyboards over different application scenarios

What is more, we can see that the error rate that the participants tap the wrong keys by mistake varies over the keyboards in Fig. 16. The error rates of *C2* and *C3* are lower than that of *BASIC*, on both the smartphones and tablets. It is mainly because that the participants need to spend more time searching the target keys with *C2* and *C3* than with *BASIC*. *C5* has the higher error rate than *BASIC*, due to some keys' small size that cannot be accurately tapped by the users. On the smartphones, *C4* has the highest error rate, since the centralized target keys as well as the small key size lead to the users typing the wrong letters frequently.

We conduct a further survey on the countermeasures' adoption in different scenarios with the 132 participants. However, in this part, only 107 participants' feedbacks are analyzed since there are 25 participants who do not take MoBaKIA attack seriously and persist in using current keyboard on the whole scenarios. The obtained result is presented in Fig. 17. Comparing it with Fig. 14, we can see that the more sensitive the content to enter is, the more effective secure strategy they will choose. Under *S1*, 88.79% participants tend to adopt some countermeasures to defend against MoBaKIA attack and 55.14% participants directly select *C2* to protect their sensitive information at the cost of usability. Moreover, over the whole application scenarios, at least 40% participants tend to adopt some defenses against MoBaKIA attack. In fact, these countermeasures can be set to switch automatically based on the importance of the content to be typed and so is the refreshing rate to modify the soft keyboard's layout, which can further improve our countermeasures' usability as well as keep the effectiveness in defending against MoBaKIA attack.

6 Related Work

The keyboard based countermeasure can be widely applied to current Android versions, so it has attracted many researchers' attention. Making use of the dragging, dropping and tapping action, Kwon *et al.* propose Drag-and-Type [10], to improve the typing accuracy. To defend against MoBaKIA attack, they propose to randomize the keyboard layout based on their basic method. Furthermore, they propose a rolling image visual keyboard, RIK [15], to fully utilize dragging and dropping actions to enter, which can effectively counter with MoBaKIA attack. CoverPad [25] introduces *variants* that are randomly generated and only can be seen by the users when they enter the sensitive information to build a random mapping between the entering keys and the target keys so as to defend against MoBaKIA attack. PassWindow [26] guarantees the security of PIN with a moving grid-configured window over a virtual keypad. To defend against MoBaKIA attack, the rear camera is utilized to imitate the touch events instead of touching on the screen. This kind of keyboards is much different from the standard keyboard, so the users need to spend much longer time to learn about them and be more concentrative when they type.

Yue *et al.* in [27] propose Privacy Enhancing Keyboard (PEK), randomly shuffling the keys and introducing the Brownian motion, respectively. Chu *et al.* propose *para-randomized keyboard* with MoBaKIA attack considered when designing TrustUI [11]. While both PEK and *para-randomized keyboard* can be utilized to defend against MoBaKIA attack, the researchers do not pay much attention to the further improvement on usability.

The most similar work is what Song *et al.* do in [20]. They propose two kinds of defenses: reducing the accuracy of accelerometer readings with a kernel modification that sets their square sum to a constant value, and completely randomizing keyboard layout. Much different from their work, we make a deep investigation on the vibrator's effect on defending against MoBaKIA attack. What is more, when we randomize the keyboard layout, we apply the entropy theory to guide our design and analyze the proposed countermeasures' effectiveness. Meanwhile, we try our best to improve the countermeasures' usability.

7 Conclusion

In this paper, we are engaged in the practical countermeasures against MoBaKIA attack. In our opinion, it is the motion data without noise as well as the fixed screen area and constant layout of soft keyboard that provide the opportunity to launch this attack, based on which, we propose our countermeasures. We evaluate them on Android platform in terms of effectiveness and usability. The result shows that while all the countermeasures have some influence on the keystroke inference accuracy, some of them can reduce the keystroke inference accuracy of MoBaKIA attack without significantly hurting the typing efficiency.

References

1. Al-Haiqi, A., Ismail, M., Nordin, R.: On the best sensor for keystrokes inference attack on android. Procedia Technol. **11**, 989–995 (2013)
2. Aviv, A.J., Sapp, B., Blaze, M., Smith, J.M.: Practicality of accelerometer side channels on smartphones. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 41–50. ACM (2012)
3. Bugiel, S., Heuser, S., Sadeghi, A.R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Presented as Part of the 22nd USENIX Security Symposium (USENIX Security 2013), pp. 131–146 (2013)

4. Cai, L., Chen, H.: Touchlogger: inferring keystrokes on touch screen from smartphone motion. HotSec **11**, 9–9 (2011)
5. Cai, L., Chen, H.: On the practicality of motion based keystroke inference attack. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) Trust 2012. LNCS, vol. 7344, pp. 273–290. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30921-2_16
6. Cappos, J., Wang, L., Weiss, R., Yang, Y., Zhuang, Y.: Blursense: dynamic fine-grained access control for smartphone privacy. In: 2014 IEEE Sensors Applications Symposium (SAS), pp. 329–332. IEEE (2014)
7. Chakraborty, S., Shen, C., Raghavan, K.R., Shoukry, Y., Millar, M., Srivastava, M.: ipShield: a framework for enforcing context-aware privacy. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2014), pp. 143–156 (2014)
8. Damopoulos, D., Kambourakis, G., Gritzalis, S.: From keyloggers to touchloggers: take the rough with the smooth. Comput. Secur. **32**, 102–114 (2013)
9. Fiebig, T., Krissler, J., Hänsch, R.: Security impact of high resolution smartphone cameras. In: 8th USENIX Workshop on Offensive Technologies (WOOT 2014) (2014)
10. Kwon, T., Na, S., Park, S.H.: Drag-and-type: a new method for typing with virtual keyboards on small touchscreens. IEEE Trans. Consum. Electron. **60**(1), 99–106 (2014)
11. Li, W., Ma, M., Han, J., Xia, Y., Zang, B., Chu, C.K., Li, T.: Building trusted path on untrusted device drivers for mobile devices. In: Proceedings of 5th Asia-Pacific Workshop on Systems, p. 8. ACM (2014)
12. Michalevsky, Y., Boneh, D., Nakibly, G.: Gyrophone: recognizing speech from gyroscope signals. In: 23rd USENIX Security Symposium (USENIX Security 2014), pp. 1053–1067 (2014)
13. Miettinen, M., Heuser, S., Kronz, W., Sadeghi, A.R., Asokan, N.: Conxsense: automated context classification for context-aware access control. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 293–304. ACM (2014)
14. Miluzzo, E., Varshavsky, A., Balakrishnan, S., Choudhury, R.R.: Tapprints: your finger taps have fingerprints. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, pp. 323–336. ACM (2012)
15. Na, S., Kwon, T.: Rik: a virtual keyboard resilient to spyware in smartphones. In: IEEE International Conference on Consumer Electronics (ICCE), pp. 10–13 (2014)
16. Narain, S., Sanatinia, A., Noubir, G.: Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, pp. 201–212. ACM (2014)
17. Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J.: Accessory: password inference using accelerometers on smartphones. In: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, p. 9. ACM (2012)
18. Raghavan, K.R., Chakraborty, S., Srivastava, M., Teague, H.: Override: a mobile privacy framework for context-driven perturbation and synthesis of sensor data streams. In: Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones, p. 2. ACM (2012)
19. Schlegel, R., Zhang, K., Zhou, X.Y., Intwala, M., Kapadia, A., Wang, X.: Soundcomber: a stealthy and context-aware sound Trojan for smartphones. In: NDSS, vol. 11, pp. 17–33 (2011)

20. Song, Y., Kukreti, M., Rawat, R., Hengartner, U.: Two novel defenses against motion-based keystroke inference attacks. arXiv preprint arXiv:1410.7746 (2014)
21. Spreitzer, R.: Pin skimming: exploiting the ambient-light sensor in mobile devices. In: Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, pp. 51–62. ACM (2014)
22. Tong, T., Evans, D.: Guardroid: a trusted path for password entry. In: Proceedings of Mobile Security Technologies (MoST) (2013)
23. Xu, Z., Bai, K., Zhu, S.: Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 113–124. ACM (2012)
24. Xu, Z., Zhu, S.: Semadroid: a privacy-aware sensor management framework for smartphones. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 61–72. ACM (2015)
25. Yan, Q., Han, J., Li, Y., Zhou, J., Deng, R.H.: Leakage-resilient password entry: challenges, design, and evaluation. Comput. Secur. **48**, 196–211 (2015)
26. Yi, H., Piao, Y., Yi, J.H.: Touch logger resistant mobile authentication scheme using multimodal sensors. In: Jeong, H.Y., S. Obaidat, M., Yen, N.Y., Park, S.H. (eds.) CSA 2013. LNEE, vol. 279, pp. 19–26. Springer, Heidelberg (2014). doi:10.1007/978-3-642-41674-3_4
27. Yue, Q., Ling, Z., Liu, B., Fu, X., Zhao, W.: Blind recognition of touched keys: attack and countermeasures. arXiv preprint arXiv:1403.4829 (2014)