When Good Becomes Evil: Keystroke Inference with Smartwatch [*]

Xiangyu Liu
The Chinese University of Hong Kong
lx012@ie.cuhk.edu.hk

Zhe Zhou
The Chinese University of Hong Kong
zz113@ie.cuhk.edu.hk

Wenrui Diao
The Chinese University of Hong Kong
dw013@ie.cuhk.edu.hk

Zhou Li
ACM Member
lzcarl@gmail.com

Kehuan Zhang
The Chinese University of Hong Kong
khzhang@ie.cuhk.edu.hk

ABSTRACT

One rising trend in today's consumer electronics is the wearable devices, e.g., smartwatches. With tens of millions of smartwatches shipped, however, the security implications of such devices are not fully understood. Although previous studies have pointed out some privacy concerns about the data that can be collected, like personalized health information, the threat is considered low as the leaked data is not highly sensitive and there is no real attack implemented. In this paper we investigate a security problem coming from sensors in smartwatches, especially the accelerometer. The results show that the actual threat is much beyond people's awareness.

Being worn on the wrist, the accelerometer built within a smartwatch can track user's hand movements, which makes inferring user inputs on keyboards possible in theory. But several challenges need to be addressed ahead in the real-world settings: e.g., small and irregular hand movements occur persistently during typing, which degrades the tracking accuracy and sometimes even overwhelms useful signals.

In this paper, we present a new and practical side-channel attack to infer user inputs on keyboards by exploiting sensors in smartwatch. Novel keystroke inference models are developed to mitigate the negative impacts of tracking noises. We focus on two major categories of keyboards: one is numeric keypad that is generally used to input digits, and the other is QWERTY keyboard on which a user can type English text. Two prototypes have been built to infer users' banking PINs and English text when they type on POS terminal and QWERTY keyboard respectively. Our results show that for numeric keyboard, the probability of finding banking PINs in the top 3 candidates can reach 65%, while for QWERTY keyboard, a significant accuracy improvement is achieved compared to the previous works, especially of the success rate of finding the correct word in the top 10 candidates.

[*]We have got the IRB approval from the authors' institutes before performing any experiment related to human subjects.

CCS'15, October 12–16, 2015, Denver, Colorado, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813668.

Categories and Subject Descriptors

D.4.6 [**Operating System**]: Security and Protection—*Invasive software*

Keywords

Smartwatch; Keystroke Inference; Side-channel Attacks

1. INTRODUCTION

Keyboard is extensively employed for inputting sensitive information, like PINs and secret documents, and it is naturally a very attractive target to cyber-criminals. Keylogger can directly steal the keystrokes but it has to maintain footage in victim's machine. On the other hand, side-channel based keystroke inference attack collects the signals emanated from keyboards using external devices and is stealthier. There have been a plenty of works exploring feasible channels for such attacks. For example, the work by Asonov and Agrawal demonstrated that sound produced by hitting different keys is unique and can be exploited for keystroke inference [20]. Following their work, a broad spectrum of attack surfaces were identified, including electromagnetic emanations [38], optical emanations [35], acoustic emanations [29, 23, 42], and even the vibration of wooden desk [32].

A large corpus of existing attacks are only feasible after the attacker deploys signal collector (e.g., parabolic microphone [20]) in the near range of the victim. Such constraint seems to be removed if the attacker is able to control victim's smartphone and abuse it as the signal collector. However, the result from previous works suggests the smartphone has to be placed close enough to the keyboard by the victim to enable such attacks [32, 42], which is not practical in most cases.

Smartwatch, on the other hand, appears to be a better option for such attacks. Featuring diversified apps and sensors, it greatly extends the functionality of traditional watch and quickly gains popularity in recent years. But if the smartwatch is controlled by an attacker, e.g., through malicious app, user's movement could be monitored to infer the typed keys. The bar for launching such attacks is also lower: the adversary does not need to deploy signal collector or expect the user putting one nearby. In this paper, we explore the attack surface from smartwatch and try to answer these questions: *Is there an effective side-channel source from smartwatch for such task? Will it lead to more accurate keystroke inference?*

The answer appears to be "yes". Sensors like accelerometer, gyroscope and microphone provide continuous data streams directly

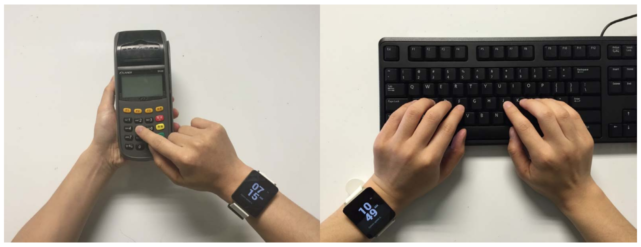
Figure 1: The scenarios of attacking POS terminal (Left) and attacking QWERTY keyboard (Right).

related to user's activities. The adversary can intercept the stream from either motion sensors or acoustic sensors for keystroke inference. Yet, it is not straightforward to transform the stream data into precise keystrokes. Motion sensors can be used to capture hand movements but several issues have to be addressed before using the data. First, there is a big variance in hand movements. For instance, the moving speed of hand is not constant between different keys: people type fast for familiar words but slow for unfamiliar ones. Second, the data collected is noisy. Small and irregular hand movements always accompany with typing keys, making the data input unstable. Third, the sampling rate is limited for smartwatch sensors, which causes movement information uncollected occasionally. Regarding acoustic sensors, previous works leverage the ones on mobile phones for keystroke inference [42], but the same techniques cannot be applied here. Previous works assume the mobile phones are placed steadily in fixed positions during attack to reduce the variance of acoustic signals produced by hitting the same key. Such an assumption does not hold in our scenario: smartwatch always moves with hand and the acoustic signals oscillate for the same key due to the ever-changing distance and relative position between keyboard and sensors.

In this paper, we develop a set of new techniques to decode keystrokes using sensors on smartwatch. Instead of reconstructing the whole precise track of hand movements, we only capture hand movements between successive keystrokes and model them using displacements or motion directions. A transition diagram is built to assign probability to different combination of keys. This new model turns out to be quite effective. We tested two commonly used keyboards, including a normal QWERTY keyboard and a numeric keypad of a handheld Point of Sale (POS) device. The user's input falls into a small candidate set produced by our approach with high probability. This result suggests using smartwatch for keystroke inference is totally feasible. To notice, the security implications regarding smartwatch has been discussed before, but mainly revolving in leaking health index [13, 2]. Our research shows smartwatch could bring in more severe threat if abused by attackers. We summarize our contributions as follows:

- We present a new and practical side-channel attack against keyboards for decoding keystrokes using a smartwatch, which can infer banking PINs from a POS terminal and recover English text from a QWERTY keyboard.
- We develop novel schemes to capture and model two different typing movements, i.e., displacement mode for numeric keypad and acceleration mode for QWERTY keyboard, which alleviate the negative impact of noises, like shaking of hand, different typing styles, etc. In addition, a modified k-NN algorithm and an optimization scoring algorithm are also proposed to improve the inference accuracy.
- We build two prototypes and thoroughly evaluate them. The result shows that for numeric keyboard, the probability of finding banking PINs in the top 3 candidates can reach 65%. While for QWERTY keyboard, a significant accuracy improvement (i.e., 50% improvement in accuracy for top 10 candidates) can be achieved compared to the previous works [23, 32].

Roadmap. The rest of this paper is organized as follows: We begin with background introduction in Section 2, as well as an overview of our attacks in Section 3. Section 4 shows our attack on POS terminal keypad. Inferring English text when user types on QWERTY keyboard is introduced in Section 5. Section 6 proposes several approaches for mitigation and in Section 7, we discuss limitations of this paper and future works. We compare our work with the previous related research in Section 8. Finally, Section 9 concludes this paper.

2. BACKGROUND

2.1 Smartwatch

It has been a long history for watch manufacturers to extend the capability of watch beyond basic time keeping function, and smartwatch is just another trend that migrates functionalities from smartphones. To support such rich functions, a smartwatch generally has a built-in modern Operating System, such as Android Wear in Samsung Galaxy Gear and Watch OS in Apple Watch. Such framework enables users to install various applications (apps) to perform advanced tasks, like making phone calls, checking SMS, etc.

Besides, smartwatches also have multiple sensors to collect information about the user and the environment. Sensors include accelerometer, microphone, gyroscope, heart rate sensor, and so on. In this work, we focus on the security implications of the sensors in smartwatches, in particular keystroke inference issue based on information from those sensors. *LG G WATCH* [25] was chosen for our evaluation purpose. This watch is powered by Android Wear OS [1], a variation of standard Android OS supporting a subset of full Android functionalities but suiting better to limited processing power and battery life. We mainly extract the data from accelerometer for the purpose of attack. The built-in accelerometer is manufactured by InvenSense [10], and its resolution and maximum measured value are 0.04 m/s^2 and 19.6 m/s^2 respectively. Similar to accelerometers in mobile phones, this sensor can measure the accelerations of movement in x-, y- and z-axis regardless of the orientation of watch (see Figure 2).

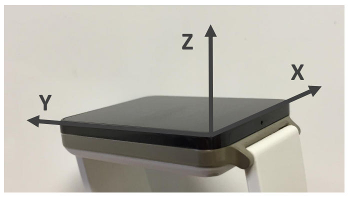

Figure 2: The x-, y- and z-axis of accelerometer in smartwatch.

2.2 Adversary Model

We assume that a user (the victim) wears a smartwatch while she is typing, and the adversary studied here is interested in inferring the inputs the victim has typed when wearing the smartwatch. The keyboards we studied include normal QWERTY keyboard and numeric keyboard on POS terminal. Both of them are frequently used.

In order to get data from sensors on a smartwatch, it is assumed that the adversary has already tricked the victim to install a malicious app onto her smartwatch, which is parallel to other attacking papers on Android smartphones. To propagate and install a piece of mobile malware, one common practice for adversaries is repackaging a popular paid app with malicious code embedded, then uploading to and making it free in one or more Android markets. Careless users will quickly get their devices compromised after downloading and installing the masqueraded malicious app.

We also assume that the malicious app can access smartphone sensors, including the accelerometer (for both cases) and microphone (for QWERTY keyboard case). This is a reasonable assumption. On Android, access to accelerometer is not fettered by any permission. Access to microphone is mediated by permission `RECORD_AUDIO`, but the malicious app could trick the user to believe such access is necessary and legitimate [1].

Since there is a significant difference between the two types of keyboards, we study them separately under different attacking scenarios. For numeric keyboard, the adversary's goal is to infer PIN code which consists of 6 digits. For QWERTY keyboard, the adversary's goal is to recover the English text input by the victim [2]. Given the nature of this type of attack (i.e., side-channel information with lots of noises), it is almost impossible to get the exact user inputs, so we take the same strategy as previous works [29, 23, 32]: the algorithm produces a set of possible user inputs, and see how likely the real input appears in this set. Like the previous keystroke inference attacks [20, 29], we store the data locally on the device during the collection process and upload them later to our server, where Matlab is used to perform the offline analysis.

3. ATTACK OVERVIEW

In this section, we briefly describe the steps of our attack against numeric keypad and QWERTY keyboard. Then, we highlight the advantages of our attack.

3.1 Attacking Numeric Keypad

In this paper, we take POS terminal as an example to show how a victim's digital inputs can be inferred, but the same techniques could be applied to devices with similar inputting interface, like ATM, door access control system, telephone, etc. The attack includes two phases: learning phase to build a model, and attacking phase to infer victim's input based on the generated model.

- *Learning phase*. We recruit a group of participants and ask them to press sequences of numbers with smartwatch worn on one wrist. The acceleration data collected by our app will be processed to extract the data points relevant to movements between keystrokes. These data points are finally decomposed along x- and y-axis of smartwatch as features and fed into a modified k-NN algorithm for model training.

- *Attacking phase*. Similar to learning phase, the raw acceleration data of the participants are recorded, processed and converted into features. We then apply the model generated in training stage on the features and produce a set of PIN candidates ranked by their possibilities.

Advantages. Accelerometer has been exploited by previous works to decode PINs [30, 33, 40]. However, these works all utilize the data from the accelerometer embedded within the targeted devices, and as such, the attacks are only effective against smart devices (e.g., smartphones). Instead, the accelerometer we exploit comes with the smartwatch which stays away from the targeted devices, hence, any keypad, including the one on POS terminal and smart devices, is potentially under threat.

3.2 Attacking QWERTY Keyboard

To demonstrate the broad attack surface by using smartwatch, we also attempt to recover English text typed through QWERTY Keyboard. In addition to information from accelerometer, we extract acoustic signals from the embedded microphone to improve accuracy. By combining the data collected from two sensors, our attack can accurately determine hand movements even without training phase. The flow of this attack is divided into three steps and they are elaborated below.

- *Detecting keystrokes*. Comparing to typing numeric keypad, the hand movements of a common user tend to be slimmer when typing adjacent keys on QWERTY keyboard, which makes separating keystrokes much more difficult if using acceleration information from accelerometer solely. Hence, in addition, we use the acoustic signals collected from embedded microphone to identify the window for individual keystroke.

- *Modeling keystrokes*. The information collected from the hand wearing smartwatch and the other hand is quite different, so we model two hands differently. Assume the user wears smartwatch on her left hand. When a keystroke is detected, we first determine if it is pressed by left hand using z-axis accelerations. If so, we represent the key pairs using the direction of hand displacement inferred from x- and y-axis acceleration variations. Otherwise, the key is pressed by right hand and we use "R" to label it[3]. After the sequence of keystrokes is processed, multiple sequences of labels are generated (called predicted profiles) to cover all the possible combinations of hand movements.

- *Word matching*. A labeling process is first performed to tag each word in a dictionary and then we develop an optimization algorithm to score likelihood of the words in the dictionary by comparing their tags with the predicted profiles. The outcome is a list of words ranked by probability.

Advantages. Compared to the previous works on QWERTY keyboards, our attack has several advantages:

Non-intrusive. Previous works collect acoustic signals or vibrations with the help of external devices, like microphone or smart phones [20, 32, 42]. The adversary has to deliberately set up these devices to be placed around the keyboard. To the opposite, smartwatch is worn by the user herself and the distance between the watch and keyboard is small enough naturally.

Strong tolerance to acoustic noise. Since the smartwatch worn on user's wrist is very close to keyboard, the keystroke sound in general is much stronger than the ambient noise (e.g., noise from air conditioner) when sensed by smartwatch. Besides, people tend to keep quiet and reduce prominent body movement when typing, which makes the sound of key presses more distinguishable.

High accuracy. The mechanism we developed takes both acoustic signals and accelerations as input, which produces high entropy and improves the accuracy comparing to previous works [23, 32].

[1]Analysis result from our repository of collected apps shows that this permission was requested by a large corpus of legitimate apps.

[2]Password inference on QWERTY keyboard is not considered in this work.

[3]"L" letters include {a b c d e f g q r s t v w x z} and "R" letters include {h i j k l m n o p u y} [12].

4. INFERRING PINS FROM NUMERIC KEYPAD

In this section, we present the attack stealing banking PINs from numeric keypad of POS terminal. The attack can succeed by just analyzing the acceleration data collected from the victim's smartwatch. Our empirical study suggests it is not a good choice to directly infer the keys tapped from the acceleration data for two reasons. First, slight shake of user's hand frequently happens during typing and finger moving, injecting a lot of noise. Second, as we observed from empirical study, user tends to move and type slow when using POS terminal, and the duration for each action differs prominently, making it more difficult to profile keystrokes. So instead, we convert the accelerations into displacements along x- and y-axis using integration techniques. Finally, we feed the displacement vectors into a state-transition model to obtain PIN candidates ranked with possibilities.

Settings. The POS terminal we attacked is LANDI E530 [16], which is used by a large number of stores in China. The layout of keypad is also similar to other POS terminals. We assume each PIN contains 6 digits, a typical setting adopted by financial systems around the world [14]. The malicious app we built collects accelerations using the Android sensor sampling interface: `TYPE_LINEAR_ACCELERATION`, and the sampling rate is set to mode `SENSOR_DELAY_FASTEST` (`FASTEST` for short) by specifying it when registering the sensor listener.

In this study, we assume the user follows the common typing style shown in Figure 1 (Left). Specifically, the POS terminal is held by the user's left hand, and she presses keys with the index finger of her right hand wearing the smartwatch. We also assume the user does not make huge movements during typing.

4.1 Movement Modeling

We model the displacements of movements in 2-dimensional Cartesian coordinate system, elaborated in Figure 3. Specifically, we use the geometric center of each key as the point coordinate and set the origin to key 1. The horizontal direction aligns with y-axis and the vertical direction aligns with x-axis. Then, the displacement between two keys can be represented by a vector. For example, the coordinates of key 1 and key 2 are (0, 0) and (0, 1), and the displacement between them is a vector [0, 1]. The number of all unique displacement vectors between number keys is 31, less than the number of all unique number key pairs (100 for 10 number keys), because different pairs of keys might have the same vector (e.g., {[0, 2]: key 1 $\rightarrow$ key 3, key 4 $\rightarrow$ key 6, key 7 $\rightarrow$ key 9}). A label $Label\ i$ is assigned to each vector, where $i \in [1,\ 31]$.

In addition to modeling number keys, we also need to model "Enter" key, which is used to submit the previous 6-digit number. Since the size of "Enter" key is two times as the number key (see Figure 3), we use (2.5, 3) to represent its coordinate. The vectors between each number key to "Enter" key are different, counting up to 10 vectors, which are denoted as $Label\ i$, where $i \in [32,\ 41]$.

4.2 Attack Steps

Our attack first learns the mapping model between accelerations and displacements through a learning phase. Then the model is applied in the testing stage to infer PINs.

4.2.1 Learning Phase

Data collection. Since user's hand movements are classified into 41 vectors ($Label$ 1 to $Label$ 41), we collect accelerations from participants for each vector. We did not collect accelerations for the combinations of vectors since user tends to pause a while between

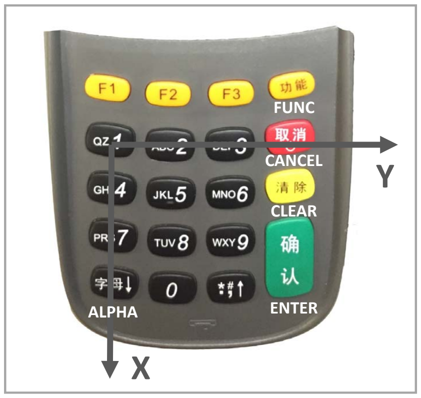

Figure 3: The x-, y-axis of POS terminal.

pressing one key and moving to the next key (also proved by our pilot experiment), introducing clear separation between consecutive vectors. To reduce the workload for each participant, we cluster the vector and its reverse one (e.g., [2, 2] for key 1 $\rightarrow$ key 9 and [-2, -2] for key 9 $\rightarrow$ key 1) into one group. After that, only 26 groups of accelerations need to be collected instead of 41 vectors. During this stage, each participant is instructed to complete 15 times for each group. One extra requirement for participants is to keep hand static for a period before/after the whole input, creating a "silence" start/end period with unobservable accelerations. To notice, this is only for facilitating the later extraction process and it is not required in testing stage.

Extraction. For the accelerations logged for each group, we extract the acceleration segment during which the 15 times input is completed. Such a process is performed automatically with the help of the "silence" start and end period.

Pre-processing. We perform the following two phases to filter out noises and improve signal-to-noise ratio (SNR).

Re-sampling. Due to the limited accuracy of accelerometer on smartwatch, the time interval between two successive sampled data points is not constant. Such non-uniform sampling data impacts the performance of latter stage, and should be refined. For this purpose, we log the time stamp of each sampled acceleration as well besides its magnitude. Then, cubic spline interpolation [6], a commonly used signal processing technique, is utilized to obtain acceleration magnitudes on regular sampling intervals throughout the whole acceleration segment.

Filtering. The collected accelerations mainly contain two types of noises, including the linear noise (also known as signal trends) and noise in high-frequency range, which is usually caused by the small but random movement of hand. We filter them using an Fast Fourier Transform (FFT) filter. In particular, we first use FFT to detect dominant frequencies corresponding to user's movements. Then we set the amplitudes of the linear part and high frequency part as zero, and apply Inverse FFT (IFFT) on the remaining frequency data to recover the time-domain signal.

Feature extraction. Our feature extraction consists of three phases: *Movements capturing*, *Calculation*, and *Optimization*. We elaborate them separately as below.

Movements capturing. After the accelerations are pre-processed, we map them to the displacements in order to reconstruct the hand movements. As a first step, we need to correctly identify the time window for each movement since different users usually type keys with different time span. Moreover, a single user may type keys differently each time. Fixed-size time window is not a viable solution and we detect the abrupt changes of energy to infer the time window instead. In particular, assume $accX(i)$, $accY(i)$ and $accZ(i)$ represent pre-processed accelerations on x-, y-, and z-axis for sample point i in time domain. We compute the acceleration $E_{XYZ}(i)$ using the equation below:

$$E_{XYZ}(i) = accX(i)^2 + accY(i)^2 + accZ(i)^2 \tag{1}$$

Then, we compute the accumulated energy $A_{XYZ}(i)$ of the accelerations in a sliding window (size equals to 10):

$$A_{XYZ}(i) = \sum_{n=i}^{i+10} E_{XYZ}(n) \tag{2}$$

Each $A_{XYZ}(i)$ is compared to a pre-defined threshold A^{th}_{XYZ} (set to 0.7 based on the empirical analysis). If $A_{XYZ}(i)$ surpasses the threshold, the user is supposed to begin to move finger to the next key, and we take i as the start location in time domain. A user moves finger 6 times to finish PIN typing and the start location for the remaining movements is identified similarly. To reduce the computation overhead, we skip the time window gap_time after one start location is detected and scan for the new start location. We set gap_time to 1 second (200 samples) based on our observation that it typically takes more than 1 second between two keystrokes. Hence, there is at most one start location within one-second time range. After that, a list of start locations is detected and we define the start location of the k^{th} movement as Loc_{m_k}.

For the k^{th} movement, we infer its accurate start and end acceleration location along x-axis using Algorithm 1, which is inspired by one prominent acceleration pattern: the accelerations will first follow the direction of the movement, and then change to the reverse direction as there should be a sharp deceleration before the finger hovers on the next key. Particularly, we elaborate Algorithm 1 with the diagram shown in Figure 4 as below: we first scan the samples in the range $[Loc_{m_k} - L1, Loc_{m_k} + L2]$ and find the location of maximum and minimum (see P2 and P3 in Figure 4). Then, we define the smaller location as Loc_{small} (P2) and the larger one as Loc_{large} (P3). Finally, a forward and backward search are performed based on P2 and P3 respectively to obtain $Loc^{start}_{m\_x_k}$ (P1) and $Loc^{end}_{m\_x_k}$ (P4). Similarly, the accurate start and end location of the k^{th} movement along y-axis can be obtained by applying these steps and denoted as $Loc^{start}_{m\_y_k}$ and $Loc^{end}_{m\_y_k}$.

Calculation. The array of accelerations in x- and y-axis corresponding to each movement will be integrated into a "coarse" displacement array using a double integral (trapezoidal integration). The equation of calculating displacement array for x-axis is shown as below:

$$S^j_{m\_x_k} = cumtrapz(t, cumtrapz(t, accX(j))), \quad j \in [Loc^{start}_{m\_x_k}, Loc^{end}_{m\_x_k}] \tag{3}$$

t is a list of timestamps corresponding to the acceleration sample points and the initial velocity is set as 0 since each movement starts with a relatively static posture. $cumtrapz$ is the function for trapezoidal integration. For each acceleration sample point with index j, we can find its corresponding displacement. Therefore, the **last element** of x-axis displacement array ($S^{j=Loc^{end}_{m\_x_k}}_{m\_x_k}$) is taken as x-axis displacement of the k^{th} movement. Similarly, $S^j_{m\_y_k}$, $j \in [Loc^{start}_{m\_y_k}, Loc^{end}_{m\_y_k}]$ is computed and $S^{j=Loc^{end}_{m\_y_k}}_{m\_y_k}$ is the y-axis displacement of the k^{th} movement.

Optimization. The above "coarse" displacement array needs to be refined. It is inaccurate when the user moves slightly along the

Algorithm 1 : Algorithm to capture x-axis movements

```
Input: $Loc_{m_k}$, $L1 = 20\ samples$, $L2 = 180\ samples$
Output: The accurate start and location of a movement
1: $range = [Loc_{m_k} - L1, Loc_{m_k} + L2]$;
2: $Loc^{min}_{m\_x_k} = argmin\ accX(range)$;
3: $Loc^{max}_{m\_x_k} = argmax\ accX(range)$;
4: $Loc_{small} = min\ (Loc^{min}_{m\_x_k}, Loc^{max}_{m\_x_k})$;
5: $Loc_{large} = max\ (Loc^{min}_{m\_x_k}, Loc^{max}_{m\_x_k})$;
6: Do a forward search from $Loc_{small}$ and find the first sample in opposite sign. The location plus one is labeled as $Loc^{start}_{m\_x_k}$.
7: Do a backward search from $Loc_{large}$ and find the first sample in opposite sign. The location minus one will be marked as $Loc^{end}_{m\_x_k}$.
8: return $Loc^{start}_{m\_x_k}$, $Loc^{end}_{m\_x_k}$
```

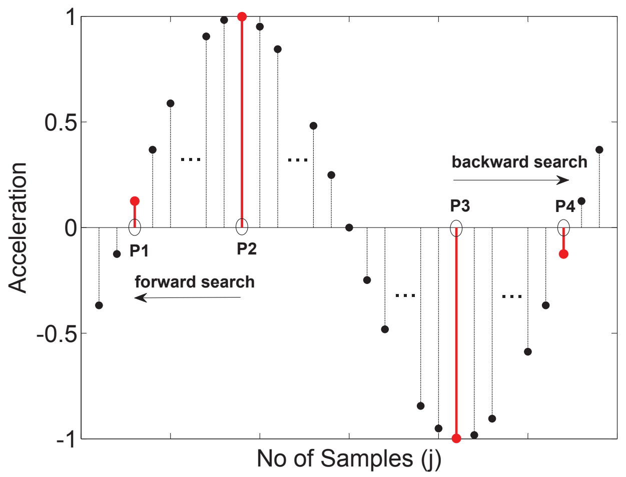

Figure 4: The diagram of processes described in Algorithm 1.

corresponding axis, though it is fairly accurate if the user moves significantly along the corresponding axis. Figure 5 shows an example of the displacement arrays for 15 movements from key 1 to key 9. The 15 x-axis displacement arrays (shown in top left) are close, and the displacements can be clustered. But the y-axis displacement arrays (shown in top right) differ prominently, though they should be close to 0 in theory. We think the errors can be attributed to the instability of user's hand and low sampling rate of accelerometer.

We aim to mitigate the errors caused by instable hand. The key insight we leverage here is that the direction of acceleration changes frequently when the hand moves along x- or y-axis, therefore, we search the location in time domain where the second alternation of the acceleration direction occurs and remove the data points after. Taking y-axis as an example, if the location exists, we denote it as $Loc^p_{m\_y_k}$ and use it to correct the end location of movement. If $S^{j'=Loc^p_{m\_y_k}}_{m\_y_k}$ is smaller than a threshold, i.e., 0.5 cm, we replace $Loc^{end}_{m\_y_k}$ with $Loc^p_{m\_y_k}$, and the new displacement array is denoted as $S^{j'}_{m\_y_k}$, where $j' \in [Loc^{start}_{m\_y_k}, Loc^p_{m\_y_k}]$.

Using the above approach, we can get the optimized integral displacement arrays, shown in Figure 5 (bottom left and bottom right). As expected, the x-axis displacement arrays keep unchanged, while the y-axis displacement arrays become much shorter and are more consistent. Finally, we define the last element of $S^{j'}_{m\_x_k}$ and $S^{j'}_{m\_y_k}$ as $S_{m\_x_k}$ and $S_{m\_y_k}$ respectively, and the displacemen-

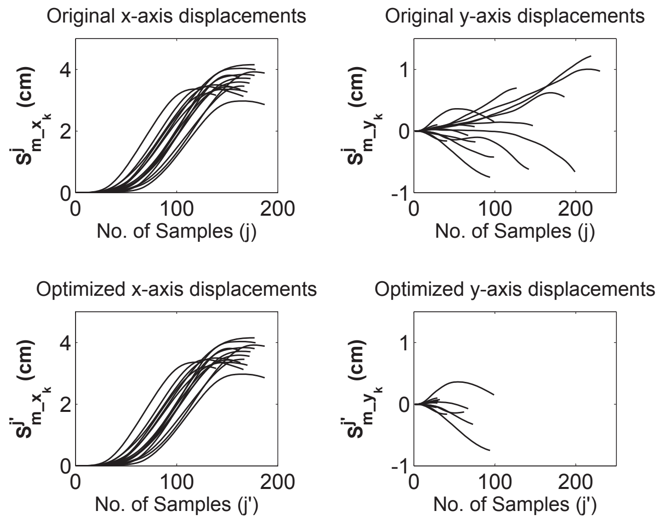

Figure 5: (Top Left) The x-axis original integral displacement arrays for 15 movements from key 1 to key 9; (Top Right) the corresponding y-axis original integral displacement arrays; (Bottom Left) The optimized x-axis displacement arrays; (Bottom Right) the y-axis displacement arrays after optimization. Note that the No. of samples after optimization is denoted as j'.

t vector [$S_{m\_x_k}$, $S_{m\_y_k}$] is taken as the feature to model the k^{th} movement.

4.2.2 Testing Phase

The displacement vectors generated in the learning stages are leveraged to infer the keys typed by a victim. We elaborate how the victim's movement data is processed and matched with the vectors as below:

Data collection. The app on smartwatch continuously record the accelerations, covering the movement of typing 6-digit PINs and "Enter" key. The accelerations are not separated into movement labels beforehand this time.

Extraction. Given the obtained accelerations, the first step is to extract the acceleration segment during which the PIN is entered. We fulfill this task through applying a heuristic that a movement usually starts and ends with low accelerations and such pattern will repeat 6 times during typing. Alternatively, we can extract accelerations using the sliding window technique, which has been widely used for recognizing gestures from accelerations [28].

Pre-processing. Similar to the processes conducted on training data, the resampling and FFT filter techniques are also used to process the testing data.

Feature extraction. We extract the same set of features, i.e., displacement vectors, as what are derived from the learning phase. The main goal of this testing phase is to find the correct $Label$ for each vector and we apply machine-learning based techniques for this purpose.

Classification. After evaluating several widely used classification algorithms, including Random Forest, K-Nearest Neighbor (k-NN), Support Vector Machine and Neural Network, we found k-NN is the best option since it is accurate and only incurs low computation overhead. The standard k-NN algorithm produces only one label for each input, which reduces the chance of success for adversary, since only all the 6 successive classifications were made correct decisions would result in the correct PIN code. Therefore, we modify the standard k-NN algorithm to output all the possible $Labels$ with their corresponding probabilities. For example, after [$S_{m\_x_k}$, $S_{m\_y_k}$] is obtained, a prediction array P will be generated. Each element P_j within P is the probability for the j^{th} $Label$, where $j \in [1,\ 41]$. Specifically, if there are n neighbors belonging to the j^{th} $Label$, $P_j = \frac{n}{k}$. Note that two models were built separately for the two types of movements (movements between numeric keys with $Label\ i$, where $i \in [1,\ 31]$ and the movements from numeric keys to the "Enter" key with $Label\ i$, where $i \in [32,\ 41]$) using the modified k-NN algorithm.

Selecting k. The parameter k is critical for k-NN algorithm as it directly decides the performance of a single classification. However, the accuracy of our attack depends on all the 6 successive classifications. In other words, only all the results of the 6 classifications are non-zero will lead to a non-zero final result. Therefore, k in this attack could not be determined by only testing a single classification. Instead, we take the process of determining k as part of our evaluation.

Cross-validation. To demonstrate the feasibility of our attack, we use 10-fold cross-validation (kfoldLoss function in Matlab) on the two groups of training data and the results are 0.2616 and 0.2975 respectively when k=5. Such loss values are acceptable and show the promising of our attack, especially considering that the modified k-NN algorithm could further improve the final accuracy.

PIN recovery. For each movement, non-zero probability will be generated for each valid $Label$. Each $Label$ includes several pairs of keys with the transition (e.g., key 0 $\rightarrow$ key 0), as shown in Figure 6. Therefore, the password recovery process is to calculate the probabilities of all the possible combinations for the 6 movements, which is similar to the process of probability calculation in Hidden Markov Model. Note that some combinations are not valid. Such examples are marked out with red cross in Figure 6. Finally, all the possible PINs with non-zero probabilities will be produced as candidates and sorted in descending order.

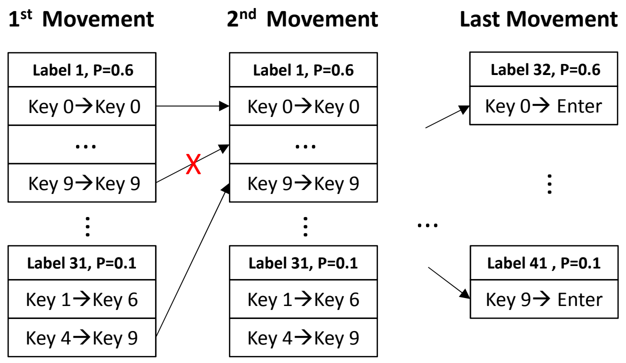

Figure 6: The diagram of transition between labels.

4.3 Experiment Results

The training data were collected from 8 volunteers (denoted as $P1$-$P8$), who were university students aged from 20 to 30. In total, we collect valid accelerations data for 4920 movements, in which 3720 movements correspond to the motion between two number key presses and 1200 movements about moving to last "Enter" operation.

We examined the effectiveness of our attack on three different participants, denoted as $User1$, $User2$ and $User3$. 300 6-digit

Table 1: The success rate of PIN inference attack.

	User 1					User 2					User 3				
k	**Top3**	**Top10**	**Top25**	**Top50**	**Top100**	**Top3**	**Top10**	**Top25**	**Top50**	**Top100**	**Top3**	**Top10**	**Top25**	**Top50**	**Top100**
5	60%	80%	88%	88%	88%	52%	81%	86%	86%	86%	51%	75%	82%	82%	83%
10	63%	77%	91%	93%	94%	57%	78%	94%	96%	96%	45%	71%	83%	89%	91%
20	65%	76%	91%	97%	99%	54%	77%	86%	95%	98%	50%	67%	77%	91%	94%
30	62%	78%	91%	94%	96%	63%	82%	89%	94%	98%	52%	69%	77%	83%	95%
40	61%	78%	87%	95%	97%	60%	83%	90%	93%	97%	47%	65%	76%	80%	89%
50	59%	73%	82%	90%	93%	60%	85%	91%	95%	98%	46%	68%	76%	79%	83%

PINs were randomly generated following the principles[4]. Each volunteer was instructed to type 100 PINs and each PIN was typed once. Finally, we logged 300 pieces of accelerations and each piece corresponds to one PIN input. The processes described in testing phase were leveraged to examine PINs inference.

Overall success rate. A PIN is correctly inferred if it appears in the candidate PIN list and the attack is deemed successful for this case. Table 1 describes the overall success rate for our experiments. We changes the size of candidate list and the parameter k for k-NN algorithm and obtains result for each combination. Our result shows that when limiting to 3 candidates, the highest success rates of $User$ 1, $User$ 2 and $User$ 3 are 65%, 63% and 52% with k set to 20, 30 and 30. Since most POS terminals allow one user to type a PIN three times, the attacker has decent chance of selecting the right PIN. The success rate largely increased when using top 10 candidates, such that the highest accuracies on the three users are 80%, 85% and 75%. When the number of candidates grows higher, the increase of success rate is relatively small, suggesting the possibilities assigned by the modified k-NN algorithm are reasonable.

Impact of k. As shown in Table 1, the value of k makes a big impact on the accuracy. In general, the accuracy rises when k increases from 5 to 30 and then decrease when k grows from 30 to 50. On one hand, smaller k clusters less neighbors, reducing the chance of finding the right label. On the other hand, larger k associates more labels to a movement, making the correct label less distinguishable.

4.4 Discussion about Attack Settings

We assume the victim uses the same hand to type PINs and wears smartwatch. This is a necessary condition as to collect accelerations regarding hand movement. A natural question is how many people share this typing style. It turns out to be quite difficult to tell the precise number. Instead, we discuss the population of potential victims as below:

Watch worn on the right wrist. Contrary to the traditional mechanical wristwatch which is usually worn on the left wrist due to time adjustment and manual winding issues, digital watch (e.g., smartwatch) is equally easy to use on either wrist [17]. Therefore, which wrist to be worn is more determined by user's personal preference, and in real world, it is not surprised some people do wear watch on the right wrist. We classify them into two groups and have a discussion.

Right-handers. In general, female right-handers are more likely to wear smartwatch on the right wrist. Additionally, many right-handers in the forums [11, 18] claim that they wear watch on the right hand and suggest some famous people also do the same, like Putin (President of Russia), Jean-Pierre Melville (French noir film director), etc.

Left-handers. Studies suggest that approximately 30% of population is mixed-handedness, whose hand preference is changed between tasks [19]. Therefore, a certain number of left-handers will type PINs using right hand.

Watch worn on the left wrist. Though we only evaluated our model when user wears smartwatch on her right hand, we believe these techniques are also promising if a user wears smartwatch on her left hand since the movement vectors are the same, which will further increase the number of potential victims.

5. INFERRING ENGLISH TEXT FROM QWERTY KEYBOARD

In this section, we present an English text inference attack against QWERTY keyboard. The attack scenario is shown in Figure 1 (Right), in which the smartwatch is worn on the left wrist of user. We elaborate the left-wrist case and the right-wrist case is discussed briefly. We assume the user follows the standard typing pattern [12] in general, but several overly restricted rules are changed to accommodate the style of more users[5]. We combine both acoustic signals and accelerations as input to limit the negative impact of various noises, thus making our attack more robust. Solely relying on accelerations or acoustic signals is not a good option and the causes are described below.

Accelerations only. Different from the case of PIN inference, when a user types English text, accelerations are always interfered with strong noises due to user's movements are more delicate and change in motion direction happens much more frequently. What's worse, the low sampling rate of accelerometer will amplify such noises and lead to high error rate. To address this problem, we record acoustic signal with the embedded microphone simultaneously, which helps to select the right time window when a key is pressed. In addition, the acoustic signal also facilitates the detection of "R" letters, when the acceleration information is not available.

Acoustic signals only. A large corpus of previous works about keystroke inference on QWERTY keyboards exploit acoustic emanations, and those works assume the spying devices are placed at a standstill within certain range to keyboard. Such an assumption is invalid for our setting since user's hand is always moving when she types text. We evaluate the impact of moving sensors by redoing the experiment conducted in [20] using sound collected from

[4]Some financial institutions do not give out or permit PINs where all digits are identical (such as 111111, ...), consecutive (123456, ...), numbers that start with one or more zeroes [14]. In addition, we add one requirement that no more than two successive digits are identical, which in order to introduce more randomness.

[5]The original standard typing pattern: the user places her fingers on "home row" ({a s d f j k l ;} [12]) initially, and moves fingers back after pressing keys in other rows. The modifications: (i) user will not return to "home row" if the next letter is located in the same row of the current key; (ii) user's fingers will not pause on the "home row" when moving from the top row to the bottom row or vice versa.

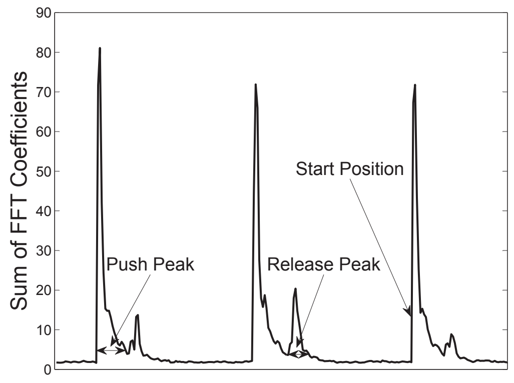

Figure 7: Energy levels over the duration of 3 keystrokes.

watch. Specifically, each alphabetic key was pressed 100 times, and the acoustic signal was then processed for creating FFT feature vectors. We trained a neural network model and tested using 70%/30% data split, the overall accuracy was only 35.75%, significantly lower than 78.85% reported in [20]. The result clearly suggests that using acoustic signals alone is not sufficient. It is also worth noting that there are often quite dramatic differences between acoustic emanations generated by different people and keyboards, which prevents the prevalence of this kind of attacks.

5.1 Detecting the Start of Keystrokes

As a start, we need to identify the time windows corresponding to key presses. Often, there exists a big gap between consecutive keystrokes. The study by Asonov et al. [20] showed that the period from pushing to releasing is about 100 milliseconds. Typically, a user spends 200 milliseconds to type one character (300 characters per minute) [20]. Such gap ensures that the keystrokes can be separated.

We use microphone on the smartwatch to collect sound signals in order to identify start of each key press. The main challenge is to prune ambient sound noises. We distinguish keystroke sound and ambient noise using energy levels in time windows. Specifically, we calculate the energy level of signal using windowed discrete Fourier transform (DFT) and take the sum of FFT coefficients (denoted S) in the range [0.4, 12] KHz. We only sum over the coefficients in this specific range since we observed that the energy of keystroke duration mainly lies in the frequencies between 400Hz and 12KHz. During our experiments, as the audio sampling rate is set to 44.1kHz, we set the window size of DFT as 256 samples and the overlap is 50%, which is able to correctly frame keystrokes. Figure 7 shows the energy curve of three keystrokes, and each keystroke contains push phase and release phase. The start of keystrokes is detected by comparing S to a threshold, which is empirically set to 7. We evaluate this step over a number of keystroke samples, and it turns out the start can be correctly identified as the push peak of keystroke.

5.2 Modeling Keypress Events

After obtaining the time window of a keystroke, we further infer the letter pressed using accelerations recorded simultaneously.

5.2.1 Detecting "L" Letters

As described in Section 3.2, the letters on the keyboard are clustered into "L" and "R" letters. Hitting both "L" letters and "R" letters will produce noticeable sound. Here, we exploit the fact that only "L" letters introduces huge z-axis acceleration and measure this variable.

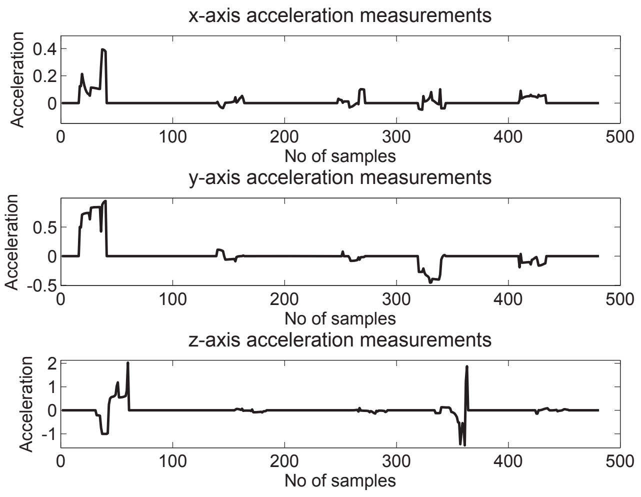

Figure 8: The x-, y-, and z-axis accelerations when user types the word "quick". The accelerations have been processed using the corresponding approaches described in Section 5.2.1 and Section 5.2.2.

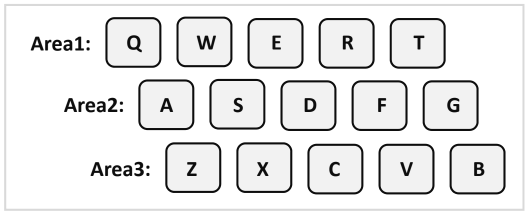

Figure 9: The three areas of the "L" letters.

Signal pre-processing. The variation of acceleration sampling rate will decrease the accuracy of the synchronization between acoustic signal and acceleration signal, thus pose negative impact on later stages. Similar to the $Resampling$ process described in Section 4.2.1, the cubic spline interpolation technique is leveraged again to obtain consistent sampling intervals throughout the whole timestamped accelerations. In addition, we apply the method described in [8], which in general is used to remove the mean value or linear trend from a vector, to erase the linear noise.

Method. Since the start time of keystrokes in acoustic signal have been detected (see Section 5.1), the z-axis accelerations in a short time near keystrokes could be extracted, and we distinguish "L" letters and "R" letters using their z-axis acceleration range values. Specifically, we use K_i to denote the i_{th} letter in the word, and $accZ_{K_i}$ to represent the z-axis accelerations during the time [ST_{K_i} - $50ms$, ST_{K_i} + $100ms$], where ST_{K_i} is the acoustic start time of K_i. We consider K_i is typed by left hand if $Range(accZ_{K_i})$ beyond the threshold, i.e., 0.25, and $Range(accZ_{K_i})$ is defined as below:

$$Range(accZ_{K_i}) = (max(accZ_{K_i}) - min(accZ_{K_i}))^2 \quad (4)$$

To illustrate our approaches, we take a word with length 5 as an example. By analyzing the z-axis accelerations when user types

Table 2: The relationship between each acceleration variation and its corresponding movements.

Movements	Acceleration variation
Area1→Area2, Area1 →Area3, Area2 →Area3	negative (`-1`)
Area1→Area1, Area2 →Area2, Area3 →Area3	stable (`0`)
Area2→Area1, Area3 →Area1, Area3 →Area2	positive (`+1`)

Algorithm 2 : Algorithm to classify "L" letters

```
Input: $accX_{K_i}$, $accY_{K_i}$, $threshold = 0.2$, $st\_location$
Output: The classes of "L" letters
 1: $sumX_{K_i} = sum(abs(accX_{K_i}))$;
 2: $sumY_{K_i} = sum(abs(accY_{K_i}))$;
 3: $[max\_value, max\_location] = max\ accY_{K_i}$;
 4: $[min\_value, min\_location] = min\ accY_{K_i}$;
 5: if $max\_value == 0$ then
 6:     $min\_location = st\_location$;
 7: end if
 8: if $min\_value == 0$ then
 9:     $min\_location = st\_location$;
10: end if
11: if $sumX_{K_i} + sumY_{K_i} \geq threshold$ then
12:     if $max\_location < min\_location$ then
13:         return +1
14:     else
15:         return -1
16:     end if
17: else
18:     return 0
19: end if
```

this word, shown in Figure 8, we could model these keypresses events as "LRRLR".

5.2.2 *Classifying "L" Letters*

Only dividing letters into two classes ("L" and "R") is not sufficient to correctly infer words. Therefore, we classify "L" letters into sub-classes to produce more entropy. User moves hand to different directions when typing different letters, leading to distinct patterns of x- and y-axis accelerations, and we exploit the moving direction for classification.

The randomness of user's hand movements makes it impossible to calculate accurate displacements, we therefore divide all the "L" letters into three horizontal areas, denoted as Area1, Area2 and Area3, shown in Figure 9, and attempt to recognize movements between areas. We abstract the movement into three modes: positive (`+1`), negative (`-1`), and stable (`0`). The corresponding movement patterns are shown in Table 2.

We first extract $accX_{K_i}$ and $accY_{K_i}$ which are the x- and y-axis accelerations within the time window [ST_{K_i}-125ms, ST_{K_i}], and extract the timestamp of the first acceleration in time window as $st\_location$. Then, the accelerations with absolute values less than 0.2 will be set as 0. Finally, we classify the "L" letters following the steps in Algorithm 2. Particularly, the letter will be labeled as `0` if the the sum of $sumX_{K_i}$ and $sumY_{K_i}$ (sum of $accX_{K_i}$ and $accY_{K_i}$ respectively) is less than the threshold, i.e., 0.2. Otherwise, we determine the label (`+1` or `-1`) **only** using the y-axis accelerations, since the areas are divided vertically. When a user moves hand, the accelerations will first follow the direction of the movement, and then change to the reverse direction. Therefore, if the timestamp of the maximum acceleration is less than the timestamp of minimum acceleration, the hand movement should be upward and we mark it as `+1`. Otherwise, the letter is labeled as `-1`.

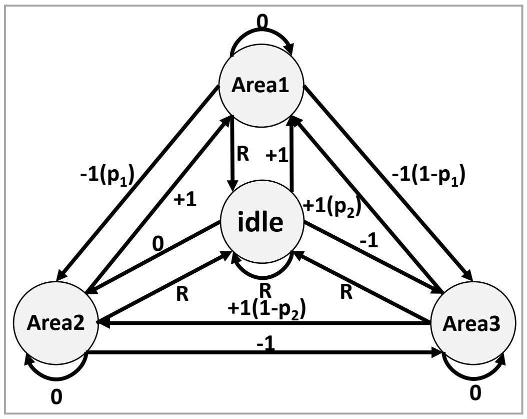

Figure 10: The diagram of labeling process.

Table 3: The number of tags that are associated with 1-5 words, including the scenarios that the smartwatch is worn on user's left wrist and right wrist.

	The number of tags	
The number of words	**Left wrist**	**Right wrist**
1	26143	16817
2	3722	2981
3	1323	1184
4	671	594
5	376	376

For the word "quick" which is labeled as "LRRLR" in the previous step, the the x- and y-axis accelerations within time window [ST_{K_i}-125ms, ST_{K_i}] are shown in Figure 8. The label is refined to "+1,R,R,-1,R", which is used as predicted profile to infer the typed word later.

5.3 Word Matching

In this section, we demonstrate how to recover word from the predicted profiles with the help of dictionary. The words in the dictionary are all translated into tags consisted of {`-1, 0, +1, R`}, and then matched against the predicted profile. A matching score will be assigned to each word.

5.3.1 *Labeling Words*

We assume the user types keyboard following the typing style described before. Therefore, we define the state transitions corresponding to the movements as shown in Figure 10. The initial state of typing is defined as idle state when fingers are lying on "home row" and transited to other states depending on the keys pressed subsequently. When "R" letters are typed, the state is transited to idle. As an example, "security" will be labeled as {"security":["0,+1,-1,R,+1,R,+1,R"]} using this state machine.

This model in essence labels a letter by the vertical direction of hand movement. However, hand moving in the same direction could finally end up in different areas. For instance, assuming the user moves her hand downward from Area1 (acceleration variation is `-1`), her hand could be placed at Area2 or Area3. There is no clear way to exactly differentiate these two cases, due to the limited capability of watch sensors. To solve this issue, we define such type movements using probabilities.

We assign each word with a movement pattern, but it is unclear how unique is one pattern, i.e., how many words are associated with the same pattern. It is less likely to derive the correct word if one pattern is associated with many words in average. To answer this question, we chose an aggregation of the "corncob" wordlist [5] as our testing dictionary and labeled them by virtually simulating the typing movement. For the 58110 dictionary words, in total 34121 tags are generated, and as many as 26554 tags are associated with a single word. Table 3 describes the distribution of number of tags to the number of words (limited to 1 to 5), which suggests this labeling approach could produce sufficient entropy.

Watch worn on the right wrist. The model described above assumes the watch is worn on the left wrist. For the case that the watch worn on the right wrist, we classify the "R" letters into three areas [6], then perform a similar labeling process to estimate the effectiveness. The accelerations and acoustic signals should have the similar pattern and the only factor deciding the result is the distribution between tags and words. We again scan the dictionary to generate tags for all words, and finally 23449 tags are generated, in which 16817 tags correspond to a unique word. The details are shown in Table 3. Such an analysis shows that a user is under threat no matter if she wears the smartwatch on her left wrist or right wrist, but it is more dangerous when the watch is worn on her left wrist since more entropy will be generated.

5.3.2 Matching Process

Obviously, the predicted profiles and the matching algorithm are two key factors in the matching process. We describe the method of optimizing the predicted profiles and the matching algorithm as below.

Optimizing predicted profiles. In the step of "L" letters classification, we use z-axis accelerations to decide if a letter is typed by left hand. In reality, user's left hand might also move when typing "R" letters. To make our scheme robust to this type of noise, we create additional predicted profiles based on the initial one. Specifically, we search for the letter (if exist) whose $Range$ (defined in equation 4) is closest to the threshold for detecting "L" letters, and then flip its label from "L" to "R" or "R" to "L" to create another predicted profile. Another source of incorrect labels come from long words. For words with more than 5 letters, we identify two letters whose $Range$ are smaller than the threshold and two letters whose $Range$ are larger than the threshold, and flip the labels separately to generate 4 other predicted profiles. All of the predicted profiles will go through the "L" letters classification process. Finally, a list of the predicted profiles are generated, with the original one as the "best" and the others as candidates.

Matching algorithm. We use a scoring mechanism to improve the success rate of matching process. Particularly, Algorithm 3 is leveraged to complete such process, it takes the predicted profiles as input, and then scores the similarity between the predicted profiles and the words in the dictionary with the same length. This process produces a list of candidate words which are presented to attackers with the scores sorted in descending order. The words with the same "L/R" profile to the "best" and other potential predicted profiles will be assigned with two and one more scores respectively. Words with only one letter (i.e., only 'a' and 'I') are exempted from being processed since they can be differentiated based on "L/R" profiles.

Algorithm 3 : Word matching process

```
Input: $dic$, $profiles$, $score = \{\}$
1:  for each $word\ in\ dic.keys$ do                    ▷ The dic with len(n)
2:      $score.word = 0$;
3:      for $i = 1$ to $len(profiles)$ do
4:          if $dic.word.LR == profiles[i].LR$ then
5:              if $i == 0$ then
6:                  $Mark(2)$;                          ▷ Add 2 bonus marks
7:              else
8:                  $Mark(1)$;
9:              end if
10:         else
11:             $Mark(0)$;
12:         end if
13:         if $Mark(result) > score.word$ then
14:             $score.word = Mark(result)$;
15:         end if
16:     end for
17: end for
18: return $sorted(score)$;
19:
20: procedure MARK($result$)
21:     $flag = 0$; $temp\_score = 0$;
22:     for $j = 1$ to $len(word)$ do
23:         if $word.LR[j] == R$ and $profile.LR[j] == R$ then
24:             $temp\_score + +$;                      ▷ One mark for equal R
25:             $result+ = temp\_score$;
26:             $flag = 0$; $temp\_score = 0$;
27:         end if
28:         if $word.LR[j] == L$ and $profile.LR[j] == L$ then
29:             $temp\_score + +$;                      ▷ One mark for equal L
30:             if $word.Acc[j] == profile.Acc[j]$ then
31:                 if $flag == 0$ then
32:                     $temp\_score++$;                ▷ One mark for equal Acc
33:                 end if
34:             else
35:                 $flag = 1$;                         ▷ Pause the comparison of Acc
36:             end if
37:         end if
38:     end for
39:     $result+ = temp\_score$;
40: end procedure
```

5.4 Evaluation

We measure the accuracy of our approach through two experiments. In the first experiment, we re-created the work conducted by Berger et al. [23] and compared with our result. We recruited 5 volunteers to complete the evaluation of this experiment, denoted as $User$ 4, 5, 6, 7, 8. In our second experiment, we examined the accuracy when user typed the sentences which were randomly chose from BBC News [3], and the dictionary was constructed using the context of the related articles. Another volunteer ($User$ 9) completed this experiment.

5.4.1 Implementation

A malicious app was developed and installed on the *LG G WATCH*, which records the acoustic signals and accelerations simultaneously. We set the audio sampling rate to 44.1kHz, and accelerations were logged using `FASTEST` mode.

We collected testing data from 5 volunteers, and all the participants were students in our university with ages between 20 and 30, and have the ability to type in words following the standard type method [12] fluently. To ensure the consistency across our data collection processes, all participants use the same keyboard (Dell Keyboard KB212-B) in all experiments. The participants were also instructed to keep the wrist always above the desk and avoid huge body movements when typing words.

[6] Area1*:{y u i o p}, Area2*:{h j k l}, Area3*:{n m}.

Table 4: A comparison of accuracies using smartwatch and the dictionary attacks proposed by Berger et al [23] and Marquardt et al [32].

User #	Top5	Top10	Top25	Top50	Top100	Top200
User 4	55%	64%	77%	83%	87%	90%
User 5	61%	69%	82%	87%	89%	94%
User 6	52%	60%	71%	80%	83%	88%
User 7	49%	58%	69%	79%	83%	87%
User 8	57%	64%	76%	83%	88%	93%
Mean	54.8%	63%	75%	82.4%	86%	90.4%
Berger	N/A	43%	61%	73%	87%	93%
Marquardt	N/A	43%	50%	57%	60%	80%

5.4.2 English Text Recovering

In order to compare our approach with the previous works in this space, we re-created the work proposed by Berger et al. [23], which attempted to recover 27 test words with length from 7 to 13 characters by analyzing acoustic emanations. The dictionary used in this work was the "corncob" wordlist [5], which contains 58110 words and has been discussed in Section 5.3.1. Intuitively, the word length is critical for dictionary attacks, longer words are easier to be identified due to fewer potential matches in the dictionary. While in our experiments, besides re-using the 27 words [7] and the "corncob" dictionary, we extended the 27 test words to 35 words by adding another 8 words with length from 5 to 6 characters. Such an extension is for the purpose of comparing with Marquardt's work [32] at the same time, which introduced more shorter words. Each participant was instructed to type each word 10 times.

The overall success rate of our experiments are shown in Table 4, which also describes that of Berger's work. Our techniques achieved better accuracy even when we added another 8 shorter words. For example, with only 43% probability Berger's work could find the correct word in the top 10 potential words, while in an average of 63% of our tests, the correct word was located within the top 10 candidates. When examining the number of correct words in the top 25 and top 50 candidates, our mean accuracy are 75% and 82.4% respectively, which are also better than the results achieved in the work of Berger. Our approach only performs slightly worse when considering the top 100 and top 200 potential words. Table 4 also describes the results of Marquardt's work, which leverages the accelerometer in a smartphone to monitor the vibrations of desk when user types text. Again, our approach produces better result, proving that using acoustic signals can greatly improve the accuracy.

5.4.3 Context-aware Text Inference

We assume the attackers in previous experiments have no prior knowledge of the victim and matches predicted word profiles against a dictionary with a large volume of words. Though the big dictionary covers the nearly all possible words user could type, it also increases the chance that multiple words share the same tag and therefore reduces accuracy. In real-world scenarios, it is not difficult for an attacker to collect context information about the victim, e.g., her occupation and residency, and leverage these information to reduce the guessing range. Expectedly, the success rate should increase and we evaluate this hypothesis through a modified version of the prior experiment. In particular, we constructed a dictionary using 4 news reports [4, 15, 9, 7] from BBC News, all were written by Smitha Mundasad, a health reporter specialized in medical field, in which we assume the victim is also engaged. Through the labeling process described in Section 5.3.1, 672 different tags were generated for all the 765 words, and as many as 615 tags were only associated with a singe word.

We evaluated the accuracy of our inferring approach on 20 sentences (463 words in total) randomly selected from the 4 reports. 57% of the words are correctly ranked top in the candidate list. If taking the top three candidates into consideration, the accuracy was raised to 88%, even significantly higher than the accuracy of using top 5 candidates (54.8% listed in Table 4). Note that our evaluations only focused on words with length no less than 4 letters, which were expected to contain more valuable information. Taking the sentence "*The results will be compared to see if either vaccine offers any meaningful protection against the virus*" as an example, the main sensitive information like "results, compared, vaccine, protection, virus" were all recovered. The result suggests our attack can be more effective when combining context information.

6. MITIGATIONS

The proposed attacks in this paper rely on two types of side-channel information: accelerometer data and acoustic emanations, so the most effective measure to mitigate the attack is to prevent malicious apps to get such data. There are several ways to achieve this goal.

Controlling accelerometer data. Current Android OS does not provide any permission to mediate the access to accelerometer, and a malicious app can read accelerations without any constraint. It turns out to be quite difficult to develop a mitigation approach without considerable cost. A new permission controlling such access could be introduced into Android OS, but then the user has to make the choice of whether granting the permission. A user can be easily fooled if the malicious app masquerade itself as a legitimate one. Reducing the sampling rate does not defeat our attack (QWERTY keyboard) either, since our approach models the hand movement from the directions instead of accurate readings. In addition, the functionality of the legitimate apps will be impaired. Probably the most reliable and secure solution is to take off the smartwatch while typing, which inevitably introduces usability problems.

Limiting acoustic emanations. It is equally difficult to mitigate the threat by limiting acoustic emanations. Mechanisms that introduces new permission or reduces sampling rate can be circumvented as well. What's worse, removing the watch does not solve the problem this time: the malicious app can still record the sound when running at background. On possible and interesting solution is to have keyboard or PC/laptop speakers generating white noise during user's typing, which can disturb the malicious app to some extend if it needs accurate acoustic data stream.

Our solution: dynamic permission management based on context. We propose to add permissions mediating access to sensors and dynamically grant or revoke them based on context information. Clearly, the access to the sensors should be limited when sensitive operation is performed by the user. To this end, profiles of permission configurations (e.g., "secure mode" which blocking access to all sensors) can be built ahead and activated under certain context (e.g., typing). Such context information can be explicitly obtained from signals broadcast by external objects [36] or inferred from the movement of user.

7. DISCUSSION

In this section, we discuss several limitations of our study and then describe the future plan.

[7]We replace the word "obfuscating" with "obfuscation", since we could not find the word "obfuscating" in the dictionary.

Sample size. Since each participant is required to type many PINs or phrases, each run of test takes significant time and we are unable to evaluate our attack on many users. But on the other hand, the scale of our evaluation is comparable to or even larger than many of the previous works in this space [20, 29, 32, 23, 42].

Computational overhead. We evaluate the computational overhead of the attacks in two ways. First, we queried the participants about noticeable phenomena but no prominent slowdown or temperature rises of watch were observed during testing. Second, we estimated the extra power consumption due to the attacks. Specifically, we compute the overhead by comparing the power consumption by the app when running and turning off. To keep consistent of our comparison, each experiment starts with a fully charged device and runs for 35 minutes (with 5 halts, each one consumes 1 minute). The results show that about 4% and 5% battery life are more consumed for attack scenario 1 (numeric keypad) and attack scenario 2 (QWERTY keyboard) respectively, which suggest our attack is hard to be detected by users.

Typing styles. We assume that users follow standard typing styles, but in the real-world scenario, a user may choose to type in ad hoc ways (e.g., starting with different posture when typing PINs or typing overly fast), increasing the difficulty of accurately inferring the right PINs/texts. It is very hard to accommodate our algorithm to all typing styles, but we can circumvent this issue by profiling the victim ahead and launch a targeted attack by applying the relevant movement model.

Future work. Some of the previous works [30, 34, 40] have demonstrated the feasibility of inferring keystrokes on devices with touch screen using their built-in sensors, e.g., accelerometer of smartphone. We believe the data from the sensors in touch-screen devices and smartwatches can be combined together and produce more entropy, such that the accuracy can be further improved. We plan to investigate the right way of combining them and the performance gain in the future.

The candidate list might not correctly rank the possibility of each candidate word or PIN, making them indistinguishable or in the wrong order. To improve the accuracy, the distribution of the usage of words and PINs in the real world can be incorporated. For example, we could increase the probability if the word or PIN is a popular choice.

Though we prune ambient noise based on energy level before further processing, certain noises might still be left and impact the result negatively. Another noise filter that can be considered is related to the consistency of acoustic signals. If sound with consistent frequency is detected (noises generated by vibration of air conditioners), we can label it as ambient noises and filter them out from the overall acoustic signals collected.

8. RELATED WORK

8.1 Body Movements Monitoring

Monitoring user's body movements is necessary for rehabilitation patients, and such researches are widely studied in medical field. For example, using miniature gyroscopes and accelerometers, Luinge et al. [31] proposed an approach for accurately measuring the orientation of human body segments. Friedman et al. [27] developed a device called "manumeter" to monitor the wrist and hand movements through capturing angular distance traveled by wrist and finger joints, which are useful for stroke rehabilitation. In addition, body movements monitoring is also a key factor for the applications heavily relying on human-computer interactions. For instance, game developers need to monitor user's movements and recognize their gestures accurately [26, 24].

The technique for monitoring body movements is definitely helpful in many aspects, but it is also a double-edged sword which can be abused to breach user privacy. As shown in this paper, it can be leveraged to infer sensitive data of smartwatch users.

8.2 Emanation-based Attacks

The emanations produced by electrical devices were known to leak information regarding users' activities. In particular, various emanation attacks that aim to infer keystrokes have been proposed and proved to be effective.

Electromagnetic emanations. By analyzing electromagnetic emanations produced by wired and wireless keyboards, M. Vuagnoux et al. [38] demonstrated the feasibility to recover keystrokes. This attack discovered four different kinds of compromising electromagnetic emanations that could lead to successful keystroke inference.

Acoustic emanations. Asonov and Agrawal [20] were the first to present a concrete keystroke inference attack using keyboard acoustic emanations. They extracted FFT values as the features of keystrokes, and used supervised learning technique to classify and recognize keystrokes. However, the performance of this approach largely depends on training dataset and is less likely to succeed when testing on different users. Zhuang et al. [29] performed a keystroke inference attack through unsupervised learning technique, which leveraging the cepstrum features of keystrokes and HMM model. A dictionary-based attack was then proposed by Berger [23], which exploited the similarity of keystrokes in a word and the constraints learned from dictionaries. More recently, Zhu et al. [42] presented a context-free and geometry-based approach to recover the keystrokes by combining sensor data from several smartphones near the keyboard.

Optical emanations. Researchers also discovered that optical emanations can lead to information leakage. By exploiting the reflections of a computer monitor on glasses, tea pots, spoons, plastic bottles, and eyes of the user, Backes et al. [22, 21] successfully recovered the content displayed on the computer monitor. With the popularity of touch screen devices, e.g., smartphone, [35] infers user inputs on a virtual keyboard by observing refections of a device's screen on a victim's glasses or other objects. Xu et al. [39] broadened the scope of such vision-based attacks by inferring the text input by user through tracking the movement of user's fingers over the screen. This attack was able to reconstruct the typed input even when the available video recording was in low resolution and recorded in a very long distance away from the victim. More recently, Yue et al. [41] improved the performance of the prior attack and made the attack feasible even when the user touches the screen with both hands and multiple fingers. Without leveraging the optical emanations from screen, i.e., reflections of the screen, Shukla et al. [37] proposed an attack to reconstruct the smartphone PIN, which entirely relies on the spatio-temporal dynamics of the hands during typing to decode the typed text.

Mechanical emanations. The mechanical vibrations can be leveraged to infer user's activities as well. For example, Marquardt et al. [32] achieve such goal by placing an iPhone near the keyboard and used accelerometer to collect vibration data of desktop to recover typing context. The work addressed the issue of low sampling rate of accelerometer (100 Hz) by modeling pairs of keypresses instead of individual keypress. Similarly, we have to deal with low sampling rate and we choose to model the displacement or motion direction of hand movement.

9. CONCLUSION

With the increasing popularity of smartwatch, concerns were raised about their capability of collecting and sharing user's private data, e.g., health index. However, little has been discussed about the threats introduced by its sensors. In this paper, we make the first step of exploring such threat space, and we show it is feasible to infer user's highly sensitive information, e.g., PINs and typed texts, through data collected from the built-in sensors, including accelerometer and microphone. In particular, we propose a set of new techniques to model user's hand movement and reduce the interference from noises, and our attack is able to achieve high accuracy in keystroke inference. As demonstrated by our research, the threat is real and we propose several countermeasures in addressing such threat. We also hope our research could raise more awareness of the security community and users in the security issues underlying smartwatch.

10. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the ECS/GRF grant from Research Grants Council, University Grants Committee, Hong Kong (with project number 24207815) and the Direct Grant from The Chinese University of Hong Kong (with project number C001-4055047).

11. REFERENCES

[1] Android wear. https://developer.android.com/wear/index.html.

[2] As smartwatches gain traction, personal data privacy worries mount. http://www.computerworld.com/article/2925311/wearables/as-smartwatches-gain-traction-personal-data-privacy-worries-mount.html.

[3] Bbc news. http://www.bbc.com/news/.

[4] Cancer patients with depression 'are being overlooked'. http://www.bbc.com/news/health-28954661.

[5] The corncob list of more than 58 000 english words. http://www.mieliestronk.com/wordlist.html.

[6] Cubic spline data interpolation. http://www.mathworks.com/help/matlab/ref/spline.html.

[7] 'deaths averted' at hospitals put into special measures. http://www.bbc.com/news/health-31166211.

[8] Detrending data. http://www.mathworks.com/help/matlab/data_analysis/detrending-data.html.

[9] Ebola crisis: Experimental vaccine 'shipped to liberia'. http://www.bbc.com/news/health-30943377.

[10] Invensense. http://www.invensense.com/.

[11] Is it acceptable to wear a watch on the right wrist? http://www.askandyaboutclothes.com/forum/showthread.php?116570-Is-it-acceptable-to-wear-a-watch-on-the-right-wrist.

[12] Learn how to touch type. http://www.ratatype.com/learn/.

[13] A new wave of gadgets can collect your personal information like never before. http://www.businessinsider.com.au/privacy-fitness-trackers-smartwatches-2014-10.

[14] Personal identification number. https://en.wikipedia.org/wiki/Personal_identification_number.

[15] Poor water and hygiene 'kills mothers and newborns'. http://www.bbc.com/news/health-30452226.

[16] Pos terminals e530 pos. http://landicorp.en.frbiz.com/group-pos_systems/34719013-pos_terminals_e530_pos.html.

[17] Watch handedness. https://en.wikipedia.org/wiki/Watch#Handedness.

[18] why wear a watch on the wrist where you're hand dominant? http://www.reddit.com/r/Watches/comments/1wzub5/question_why_wear_a_watch_on_the_wrist_where/.

[19] ANNETT, M. *Handedness and brain asymmetry: The right shift theory.* Psychology Press, 2002.

[20] ASONOV, D., AND AGRAWAL, R. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy* (2004), IEEE Computer Society.

[21] BACKES, M., CHEN, T., DUERMUTH, M., LENSCH, H., AND WELK, M. Tempest in a teapot: Compromising reflections revisited. In *Security and Privacy, 2009 30th IEEE Symposium on* (2009), IEEE, pp. 315–327.

[22] BACKES, M., DURMUTH, M., AND UNRUH, D. Compromising reflections-or-how to read lcd monitors around the corner. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 158–169.

[23] BERGER, Y., WOOL, A., AND YEREDOR, A. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM conference on Computer and communications security* (2006), ACM, pp. 245–254.

[24] BIANCHI-BERTHOUZE, N. Understanding the role of body movement in player engagement. *Human–Computer Interaction 28*, 1 (2013), 40–75.

[25] ELECTRONICS, L. Lg g watch | powered by android wear. http://www.lg.com/global/gwatch/one/index.html#main, 2015.

[26] FOTHERGILL, S., MENTIS, H., KOHLI, P., AND NOWOZIN, S. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012), ACM, pp. 1737–1746.

[27] FRIEDMAN, N., ROWE, J. B., REINKENSMEYER, D. J., AND BACHMAN, M. The manumeter: A wearable device for monitoring daily use of the wrist and fingers.

[28] KWON, D. Y., AND GROSS, M. A framework for 3d spatial gesture design and modeling using a wearable input device. In *Wearable Computers, 2007 11th IEEE International Symposium on* (2007), IEEE, pp. 23–26.

[29] LI, Z., FENG, Z., AND TYGAR, J. Keyboard acoustic emanations revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security* (2005).

[30] LIANG, C., AND CHEN, H. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *6th USENIX Conference on Hot Topics in Security, HotSec* (2011).

[31] LUINGE, H. J., AND VELTINK, P. H. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and computing 43*, 2 (2005), 273–282.

[32] MARQUARDT, P., VERMA, A., CARTER, H., AND TRAYNOR, P. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 551–562.

[33] MILUZZO, E., VARSHAVSKY, A., BALAKRISHNAN, S., AND CHOUDHURY, R. R. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 323–336.

[34] OWUSU, E., HAN, J., DAS, S., PERRIG, A., AND ZHANG, J. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications* (2012), ACM, p. 9.

[35] RAGURAM, R., WHITE, A. M., GOSWAMI, D., MONROSE, F., AND FRAHM, J.-M. ispy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 527–536.

[36] ROESNER, F., MOLNAR, D., MOSHCHUK, A., KOHNO, T., AND WANG, H. J. World-driven access control for continuous sensing. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1169–1181.

[37] SHUKLA, D., KUMAR, R., SERWADDA, A., AND PHOHA, V. V. Beware, your hands reveal your secrets! In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 904–917.

[38] VUAGNOUX, M., AND PASINI, S. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security Symposium* (2009), pp. 1–16.

[39] XU, Y., HEINLY, J., WHITE, A. M., MONROSE, F., AND FRAHM, J.-M. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 1063–1074.

[40] XU, Z., BAI, K., AND 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* (2012), ACM, pp. 113–124.

[41] YUE, Q., LING, Z., FU, X., LIU, B., REN, K., AND ZHAO, W. Blind recognition of touched keys on mobile devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1403–1414.

[42] ZHU, T., MA, Q., ZHANG, S., AND LIU, Y. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM.