Context-free Attacks Using Keyboard Acoustic Emanations

Tong Zhu[1,2]
zhutong@greenorbs.com

Qiang Ma[1,2]
maq@greenorbs.com

Shanfeng Zhang[1,3]
shanfeng@greenorbs.com

Yunhao Liu[1]
yunhao@greenorbs.com

[1] School of Software and TNLIST, Tsinghua University, P. R. China
[2] Tsinghua National Lab IOT Center WuXi, P. R. China
[3] Department of Computer Science and Engineering, HKUST, Hong Kong

ABSTRACT

The emanations of electronic and mechanical devices have raised serious privacy concerns. It proves possible for an attacker to recover the keystrokes by acoustic signal emanations. Most existing malicious applications adopt context-based approaches, which assume that the typed texts are potentially correlated. Those approaches often incur a high cost during the context learning stage, and can be limited by randomly typed contents (e.g., passwords). Also, context correlations can increase the risk of successive false recognition. We present a context-free and geometry-based approach to recover keystrokes. Using off-the-shelf smartphones to record acoustic emanations from keystrokes, this design estimates keystrokes' physical positions based on the Time Difference of Arrival (TDoA) method. We conduct extensive experiments and the results show that more than 72.2% of keystrokes can be successfully recovered.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Unauthorized access; H.5.5 [**Sound and Music Computing**]: Signal analysis, synthesis, and processing

General Terms

Security

Keywords

Context-free attack; Keystroke recovery; Acoustic emanations

1. INTRODUCTION

The emanations of electronic and mechanical devices have been a major topic of concern in the security and privacy communities. As early as the 1910s, German scientists eavesdropped on French field phone lines [5]. In 1943, an engineer with Bell Telephone discovered electromagnetic signals by means of an oscilloscope while using a Bell Telephone model 131-B2, a top secret encrypted teletype terminal used by the US army and navy to transmit wartime communications against German and Japanese cryptanalysis [26]. Researchers eavesdrop on video display units by picking up and decoding the electromagnetic interference produced by the equipment [9]. Such eavesdropping can be achieved at great distances and through significant obstacles (e.g., brick walls). More recently, scientists recover sound by observing the small vibrations of an object's surface produced when sound hits the object [8]. This kind of emanation detection has been successful mainly due to the use of precise instruments, which also increases the difficulty of deployment.

Mobile phones are becoming increasingly powerful devices. In addition to being able to run applications ranging from email clients to online banking, the phenomenal growth of various sensors enable these devices to actively interact with the world around them and to be used in unintended ways [21, 19, 20, 23, 27, 11, 4]. Indeed, malware could potentially gain access to a smartphone's camera and take photos or shoot videos [31], or activate the microphones to record ambient sounds [7]. In recent years, the issue of most concern is that, contents typed via keyboard can be recovered from the acoustic emanations [37, 6, 2] and vibrations perceived by accelerometers [18]. The attackers plant malware into the target's smartphone. When the target user is typing the keyboard, the malware automatically records the signal information. Through WiFi, 3G or other interfaces, the data can be collected by the attackers to recover the keystrokes.

Most existing acoustic-based malicious applications assume that the typed contents are correlated, and they infer the content using machine learning technologies. The main disadvantages of those context-based approaches are three-fold. First, they suffer from the high cost at the context learning stage (e.g., word spelling, word frequency and language grammar). Second, the assumption is not valid in many cases, e.g., passwords are usually a complicated combination of random characters. Last but not least, any mistype may significantly decrease the detection accuracy. For instance, hitting the "delete" button on Mac keyboards (the "backspace" button on other keyboards) does not input any useful content, but deletes what was typed. Moreover,

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2660267.2660296 .

while context correlations are leveraged to infer typed text, they also increase the risk of successive false recognitions.

We present a context-free and geometry-based approach to recover the keystrokes even if they are randomly typed. By collecting acoustic emanations from keyboard using two or more smartphones, this approach analyzes acoustic signals, and determines the typed key's physical position in a geometrical way. Specifically, every keystroke is supposed to generate an acoustic signal, which is recorded by microphones on smartphones. For each pair of microphones, we measure the difference of distance range from the key to them. Therefore, a candidate set of key position can be estimated in the plane. Intuitively, a candidate set is narrowed by considering multiple pairs of microphones. To achieve this work, however, two main challenges have to be overcome. First, the deviation of distance estimation caused by the low sampling rate of adopted microphones is non-negligible, which significantly expands the candidate region of typed key positions in the plane, and increases the difficulty of recognition. Second, to make the approach generally applicable, we do not assume that the relative locations of the keyboard to the smartphones are known. Therefore, before recovering the typed content, we need to reconstruct the keyboard location first.

To the best of our knowledge, this is the first context-free and geometry-based approach to recover keystrokes by means of acoustic emanations. Major contributions of this work are as follows:

- We present a context-free and geometry-based approach for keystroke localization. By extracting the acoustic signals received by off-the-shelf phones, we estimate the candidate area of a typed key with Time Difference of Arrival (TDoA) values.
- We logically reconstruct the keyboard by aligning the keys and the *maximum intersections*, which are defined in a heatmap after overlaying the candidate areas.
- We have tested this design on commodity smartphones, and the experimental results show that more than 72.2% of keystrokes can be accurately recovered.

The rest of this paper is structured as follows. In Section 2, we describe an overview of this work, and introduce background knowledge about keyboard acoustic emanations. Section 3 presents the details of our technical contributions. We discuss some related research issues in Section 4. Our proposed approach is evaluated and validated in Section 5. Section 6 describes the related work, and Section 7 concludes the work.

2. OVERVIEW

In this section, we first give background information about off-the-shelf devices to support our attack, including hardware support and theoretical feasibility. We then introduce the general idea of our attack.

2.1 Hardware Support

Nowadays smartphones usually have more than one microphones to support beam-forming direcctional audio techniques and eliminate background noises. For example, iPhone 5 and iPhone 5s are equipped with three microphones: "front", "back" and "bottom" [1]. The sampling rate of the microphone on smartphone is up to 44.1kHz. The high sampling rate and the multi-microphone enable other novel applications, such as BeepBeep [22] (a pairwise acoustic ranging system for COTS devices), phone-to-phone 3D localization [24], ENSBox [12] (a distributed and self-calibrating localization system for outdoor environments), and driver detection [35, 34].

2.2 Theoretical Feasibility

2.2.1 Inevitable Error

In our attack, we record the acoustic signals caused by keystrokes, and compute the physical positions of the keys using a geometric method. *Is it feasible to estimate such a small key with acoustic signals?* Generally, the distance between the centers of two keys is 1.9 $\sim$ 2.2cm. Suppose that the sampling rate of the microphone on smartphones is 44.1kHz, then the minimal differential distance between two sound signal points is $(343m/s)/(44.1kHz) \approx 0.77cm$. Because $0.77cm < 1.9cm$, we conclude that it is theoretically feasible to use the sound to measure distance between the keystroke and the microphone, with an inevitable minimal resolution of $0.77cm$.

2.2.2 Keystroke Distinguish

A typical computer user types up to about 300 keystrokes per minute. Each keystroke consists of a push and a release. Fig. 4 shows the acoustic signal with a push peak and a release peak. We find that the period from push to release is typically about 100 milliseconds, which is consistent with that reported by Asonov and Agrawal [2]. This observation implies that more than 100 milliseconds is left between consecutive keystrokes, and this time gap is large enough to distinguish the consecutive keystrokes. We design an efficient signal segmentation algorithm to distinguish the consecutive keystrokes in Section 3.

2.3 Our Attack

2.3.1 Attack Scenario

We consider the following attack scenario: *in a public area such as a library, a target is using his/her laptop. We sit near to him/her, and try to reconstruct the user input, which can be understandable (e.g., chat messages and documents), or can be purely random keystrokes (e.g., passwords and account numbers).* Our geometry-based approach uses two or more mobile phones, and each phone is equipped with at least two microphones with a 44.1kHz sampling rate. The overall attack setting is illustrated in Figure 1.

2.3.2 Attack Work flow

The main observation to support our attack is that the acoustic signal of a keystroke reach different microphones asynchronously since the distances from the keystroke to these microphones differ. We use m to denote the number of smartphones and K to denote the location of the pressed key. Suppose each smartphone has two microphones, we use $M_1, M_2, \cdots, M_{2m}$ to denote the locations of $2m$ microphones. When a key is pressed, we do not know its location K, nor the absolute travel distances of its sound to the $2m$ microphones. However for any two microphones, say i and j,

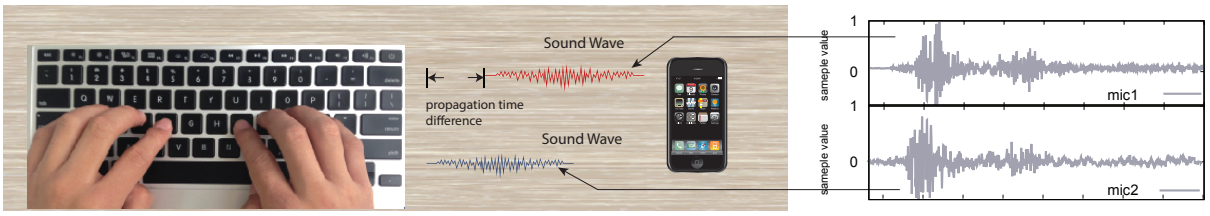


Figure 1: Attack overview. Context-free attacks to recover random text using keyboard acoustic emanations.

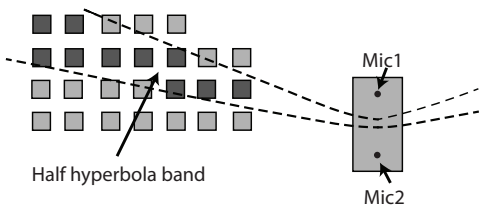


Figure 2: For a pair of microphones, the candidate set of typed key is a half hyperbola belt

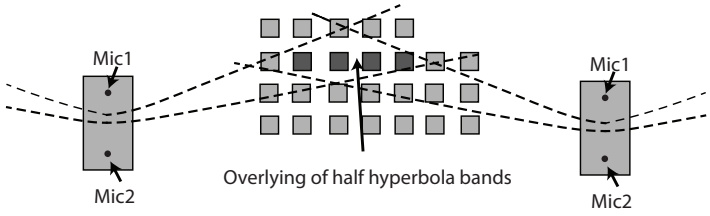


Figure 3: With more pairs of microphones, the candidate set of typed key can be narrowed.

Table 1: Summary of symbols

R	The rectangle area of a keyboard
r	The center location of the keyboard
α	The rotation angle of the keyboard in the 2D plane defined by smartphones
R_i	The rectangle area of a key
K, K_i	Key, or the center location of a key (dependent on the context)
P_i	Candidate area of a key
p_i	The gravity center of P_i
R'_i	A key region
I_i	A maximum intersection
M_i	The location of a microphone
W	Width of a acoustic signal piece
N_p	Size of the excessive value set
$\overline{KM_i}$	The distance from K to M_i
$x_i(t)$	Acoustic signal received on microphone i
$E_i(t)$	The energy level of an acoustic signal $x_i(t)$
$A_i(t)$	Accumulated energy of an acoustic signal $x_i(t)$

we can measure TDoA value, which can be further converted into the distance by multiplying $343m/s$:

$$\Delta d_{ij} = 343m/s \times \Delta t_{ij}, \tag{1}$$

and geometrically Δd_{ij} is defined as

$$\Delta d_{ij} = |\overline{KM_i} - \overline{KM_j}|, \tag{2}$$

where Δt_{ij} refers to the TDoA result and $\overline{KM_i}$ denotes the distance from the location of the key to the location of microphone i. Then the center location of the pressed key K is on a half hyperbola geometrically: M_i and M_j are two focal points, Δd_{ij} is distance difference of the hyperbola, and only the branch with the focal point closer to K is selected. Ideally K can be pinpointed as the intersection of 2 half hyperbolas, with at least three microphones. In practice, due to the estimation deviation, Δd_{ij} is in a range: $\Delta d_{ij} \in (l_{ij}, u_{ij})$, where l_{ij} and u_{ij} denote the lower and upper bound of difference, respectively. Therefore, K should be in a half hyperbola band (as shown in Fig. 2). 2m microphones can generate $\binom{2m}{2}$ half hyperbola bands. By overlying the bands, we can narrow the candidate area of the key's position (as shown in Fig. 3).

Table 1 lists the symbols used through the paper. We summarize the four main steps as follows and the details are given in the next section.

- *Pre-Processing.* In the pre-processing part, we segment the acoustic signal into small pieces. Each piece represents exactly one keystroke. As we have multiple microphones, we further group the signal pieces generated by the same keystroke together.
- *TDoA Estimation.* For every two signal pieces of the same keystroke, we propose to use a generalized cross-correlation(GCC) with PHAT weighting to calculate the time difference of the keystroke acoustic signal's arrival at different microphones.
- *Keystroke Pinpoint.* A number of microphone pairs have distance differences, this results in a number of half hyperbola bands. We choose a proper width for each hyperbola band, and overlay the bands to narrow down the candidate area of a keystroke.
- *Keyboard Reconstruction and Keystroke Recovery.* To make our attack more general, the relative location of the keyboard to mobile phones is assumed to be unknown. This assumption largely raises the difficulty of keystroke localization. After collecting the acoustic signals over a period of time, we draw a most likely region for each keystroke. Then, we estimate the keyboard location with an optimization algorithm by matching those regions to their most likely keys. With the location of keyboard and the pinpointed localization of the key, the keystrokes can be recovered.

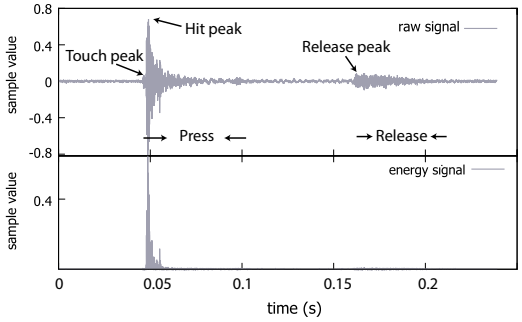


Figure 4: Example of a keystroke acoustic signal and its approximate energy signal.

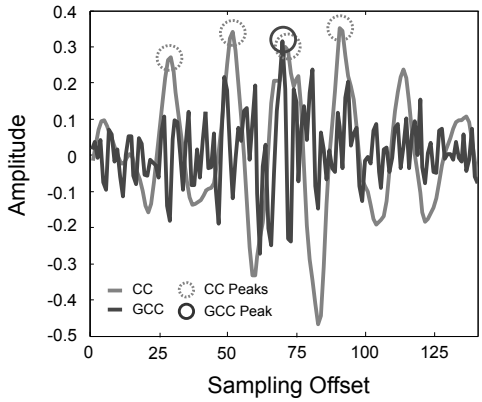


Figure 5: The result of cross-correlation and generalized cross-correlation of $a_1(t)$ and $a_2(t)$

3. TECHNICAL DETAILS

3.1 Pre-processing

In this part, we show how to segment the received acoustic signals into pieces so that each piece represents a keystroke. We use $x_i(t)$ to denote the acoustic signal, received by microphone i, indexed by t. As shown in [37, 2], one keystroke can be divided into two events: the press event and the release event. From Fig. 4, we can find three peaks: *Touch Peak*, *Hit Peak* and *Release Peak*. Among these three peaks, the *Touch Peak* is unconspicuous and may sometimes be ignored; the *Release Peak* may last for a long time and therefore is hard to pinpoint; the *Hit Peak* is most suitable to serve as a landmark of a keystroke. To search for the hit peak, we transform the signal sequence $x_i(t)$ into its energy level:

$$E_i(t) = kx_i(t)^2, \tag{3}$$

where $E_i(t)$ is the energy level and k is a constant. The lower part of Fig. 4 shows the energy level graph of the signal. We use a sliding window of 10 samples to calculate the accumulated energy:

$$A_i(t) = \sum_{n=t}^{t+10} E_i(n). \tag{4}$$

The timing of a *Hit Peak* (denoted by t_{hit}) is formally defined as follows:

$$t_{hit} = \underset{t}{\operatorname{argmax}}\, A_i(t), \tag{5}$$
$$\text{s.t. } A_i(t) \geq A_i(j) \text{ for } t - 10ms \leq j \leq t + 90ms,$$
$$\text{and } A_i(t) \geq A_\theta.$$

The search range is set to $100ms$ because the period of a keystroke from push to release is typically more than 100 milliseconds and there is at most one *Hit Peak* during 100 milliseconds. A_θ is empirically set to 0.1. Algo. 1 shows the details of how to find t_{hit}.

Finally, for each t_{hit}, we take $10ms$ before t_{hit} and $90ms$ after t_{hit} to form a piece of a acoustic signal covering a keystroke. We denote W as the width of the signal piece. Each piece contains $44.1kHz \times 100ms = 4410$ acoustic signal points.

3.2 TDoA Estimation

After processing the received signals, we get acoustic signals for a keystroke from each microphone. Suppose we have two acoustic signal pieces for the same keystroke, denoted by $a_1(t)$ from the first microphone and $a_2(t)$ from the second microphone. We let Δt refers to TDoA result between the two signals. A simple approach to estimate Δt is to utilize a landmark time stamp in both signals. Such a landmark time stamp can be anytime when a signal point occurs. Our first method leverages *Hit Peak* as the landmark to match the two signals and calculate the difference. According to our experimental results, hit peak matching approach, however, causes a deviation in Δt of over $0.44ms$, which is equivalent to an unacceptable deviation of over $15cm$ in distance. This is caused by the multipath effect and aliasing in the acoustic signal. Such an estimation deviation cannot be tolerated to pinpoint the keystroke. To address this issue, we use cross-correlation between signal $a_1(t)$ and $a_2(t)$ for the TDoA estimation.

Cross-correlation is a standard signal processing technique for searching a best match position between a signal and a pattern sample (The sharp peak of the cross-correlation shows the best match, as shown in Fig. 5). It is widely adopted in acoustic distance measurement [22, 36, 24]. Without loss of generality, the reference signal is the acoustic signal received by Mic 2, and the sliding signal is sampled in the signal received by Mic 1. The normalized cross-correlation

Algorithm 1 Algorithm to find *Hit Peaks*

Input: The acoustic signal $x_i(t)$, width of signal piece W.
Output: Time of *Hit Peaks*.

```
1: foreach t in the signal x_i(t) do
2:    A_i(t) = Σ_{n=t}^{t+10} k x_i^2(n).
3:    if A_i(t) > A_θ then
4:       for n > t and n < t + W do
5:          if A_i(t) < A_i(n) then
6:             break
7:          n++
8:       if n == t + W then
9:          t is the time of a Hit Peaks
10:         t = t + W
11:   t++
```

is computed as:

$$CC(t_0) = \frac{\sum_t \left[a_1(t) - \overline{a_1(t)}\right]\left[a_2(t-t_0) - \overline{a_2(t-t_0)}\right]}{\sqrt{\sum_t \left[a_1(t) - \overline{a_1(t)}\right]^2 \sum_t \left[a_2(t) - \overline{a_2(t)}\right]^2}}, \quad (6)$$

where $a_2(t)$ is the reference signal, and $\overline{a_1(t)} = \frac{\sum_t a_1(t)}{W}$, $\overline{a_2(t)} = \frac{\sum_t a_2(t)}{W}$. We denote t_{cc} as the sample offset that yields the maximum cross correlation:

$$t_{cc} = \underset{t_0}{\operatorname{argmax}}\, CC(t_0). \quad (7)$$

Fig. 5 illustrates the cross-correlation of two signal piece. Due to the reverberation effect, the peak of cross-correlation is unconspicuous, or there may be several peaks. Therefore the TDoA result is not accurate. To mitigate the deviation caused by the reverberation effect, we propose to use generalized cross-correlation [16]. Generalized cross-correlation improves cross-correlation (Equ. 6) by adding a weighting function in front of the correlation in the frequency domain:

$$GCC_{PHAT}(t_0) = \mathcal{F}^{-1}\left[\Phi_{12}(\omega)A_1(\omega)[A_2(\omega)]^*\right], \quad (8)$$

where $A_1(\omega)$ and $A_2(\omega)$ are the Fourier transforms of the two signals $a_1(t)$ and $a_2(t)$, and $[\cdot]^*$ denotes the complex conjugate. A number of weighting functions have been investigated in the existing literature. We use the heuristic-based Phase Transform (PHAT) weighting [16], which is defined as:

$$\Phi_{12}(\omega) = \frac{1}{|A_1(\omega)[A_2(\omega)]^*|} = \frac{1}{|A_1(\omega)||A_2(\omega)|}. \quad (9)$$

PHAT has been found to perform well under realistic acoustical conditions [32, 25]. We denote t_{gcc} as the sample offset that yields the maximum generalized cross correlation:

$$t_{gcc} = \underset{t_0}{\operatorname{argmax}}\, GCC_{PHAT}(t_0). \quad (10)$$

GCC-PHAT has an advantage of making the peak distinct and clear. As shown in Fig. 5, the result obtained by GCC-PHAT has only one peak, while the result obtained by CC has four peaks. In practice, however, GCC-PHAT can also yield multiple peaks, which then introduce variances into the estimation of Δt because each peak in GCC-PHAT result series indicates a possible Δt. We now present an empirical metric which can measure the deviation of Δt: the number of excessive values N_p. It is defined as:

$$N_p = |\{t \quad | \quad GCC_{PHAT}(t) \geq \beta \cdot GCC_{PHAT}(t_{gcc})\}|, \quad (11)$$

where $GCC_{PHAT}(t_{gcc})$ refers to the maximum value along GCC-PHAT result series and $\beta \in [0, 1]$ is a constant coefficient. In our experimental results (Section 5.2), we will show how N_p affects the variance of Δt.

3.3 Keystroke Pinpoint

Suppose the locations of the two microphones are M_i and M_j respectively and the measured distance difference is $\Delta\tilde{d}_{ij}$, the possible positions of the pressed key K are then along the half hyperbola defined by Eq. 2. Due to the uncertainty of TDoA estimation, we use a half hyperbola band to narrow down the keystroke in a region instead of the half hyperbola. The challenge is to estimate the hyperbola band width. A narrow band may lead to locating to a wrong key, while a wide band will cover too much key regions which cannot be distinguished. We propose a conservative method to estimate the hyperbola band. According to the distribution of measurement error, the third quantile w of the distribution is used to estimate the band width. We denote a lower bound of Δd_{ij} as $l = \Delta\tilde{d}_{ij} - w$, and a upper bound of Δd_{ij} as $u = \Delta\tilde{d}_{ij} + w$. w is discussed in Section 5.2. The candidate area of keystroke is now a hyperbola band:

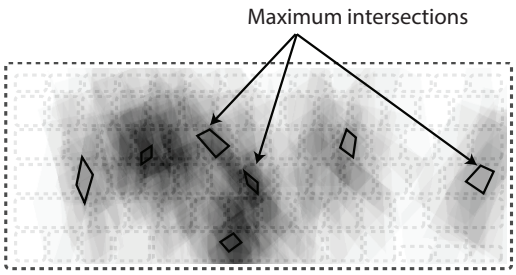


Figure 6: The heatmap of the intersection of candidate areas and the illustration of maximum intersections

$$l_{ij} \leq \overline{KM_i} - \overline{KM_j} \leq u_{ij}. \quad (12)$$

Using m smartphones with $2m$ microphones, we can generate $\binom{2m}{2}$ half hyperbola bands. We overlay those bands to narrow the candidate area of a keystroke position. An illustration is shown in Fig. 3 where two half hyperbola bands are overlayed (only two of the $\binom{4}{2} = 6$ bands are drawn).

3.4 Keyboard Reconstruction and Keystroke Recovery

In the previous section, we show that TDoA-based method can determine the candidate area for one keystroke using one or more smartphones. If the position and the layout of keyboard are known, we can easily map the candidate area onto the keyboard layout and find the pressed key. Though the layout of the keyboard is standardized with only a few variations, the position of a keyboard, such as its center position and relative angles to the smartphones, is not always known. Next, we present an estimation algorithm to compute the relative position of the keyboard with respect to the phones.

First, we formulate the keyboard reconstruction problem. The keyboard layout is a rectangular area R where there are n keys. Each key is a small rectangular R_i with a center point r_i to be determined after keyboard mapping. After the user has typed m keys $K_1, K_2, \ldots, K_m$ (the same key may be typed for more than once), we apply TDoA algorithm to find their corresponding candidate areas $P_1, P_2, \ldots, P_m$ in the 2D plane, and calculate the m gravity centers of the candidate areas $p_1, p_2, \ldots, p_m$. The goal is to map the actual keyboard layout (a rectangle) onto the 2D plane defined by the candidate areas, i.e., to find the center of the keyboard r and the rotation angle of the keyboard α.

Given the 2D plane with m candidate areas identified, there are infinite ways to map the keyboard layout onto this plane. Most of them are simply invalid because all the candidate areas should be within the boundary of the keyboard layout. Many of them are not good mappings, e.g., the centers of the candidate areas and the centers of the mapped keys are not well aligned. We formulate the fol-

lowing objective function to measure the goodness of the mapping/aligning:

$$\max_{r,\alpha} \quad \sum_{i=1}^{m} c(P_i), \tag{13}$$

s.t. $P_i (1 \leq i \leq m)$ is within the layout rectangular R,

where $c(P_i)$ measures the coverage quality of the candidate area P_i. Ideally each candidate area shall cover exactly one key. But in practice a candidate area may cover more than one key. Among all the covered keys, we expect the one with the maximum coverage to be maximized. With this heuristic, we denote the overlay area of a candidate area P_i and the key R_j as $C(P_i, R_j)$, and formally define $c(P_i)$ as follows:

$$c(P_i) = \max_j C(P_i, R_j).$$

The main challenge is to solve the optimization problem in Eq. 13. We propose an enumeration-based algorithm that optimizes the above objective function. At the first level, pairs of the candidate areas are enumerated. At the second level, two keys are enumerated to match the selected pair. With these two matchings between candidate areas and keys, the keyboard position can then be fixed on the 2D plane, or proved to be infeasible for a particular matching (e.g., the distance between two candidate areas is much larger or smaller than the distance between the two keys). However the computational cost of this scheme is too high because the number of possible matched candidate areas is as many as $\binom{m}{2} \times \binom{n}{2}$. We can reduce the search space by introducing heuristics to eliminate obviously wrong matchings. To reduce the enumeration space, we calculate *maximum intersections* of the candidate areas. A *maximum intersection* is defined to be a candidate area or a sub-part of a candidate area under the condition that all its neighbour areas have less density than it, as shown in Fig. 6. The number of maximum intersections is bounded by the number of keys in the keyboard, which is a small value compared to the number of keystrokes in a typing session. Instead of matching candidate ares to keys, we now match the maximum intersections to key regions, which are discrete regions of the keyboard. A key region is a smaller unit than a key. We divide a key into 4 equally sized key regions. This division allows the maximum intersections to match the boundary areas between keys. We use R'_i to denote a key region. The detail of the reconstruction algorithm is summarized in Algorithm 2.

Once r and α have been determined, all R_i are also determined. We locate each keystroke in relation to Key R_i which has maximum coverage by the polygon. Thus far, we have made a mapping from each keystroke to its possible key.

4. DISCUSSION

4.1 How Many Smartphones Should Be Used?

With more than one smartphone, more pairs of microphones are available. By overlaying half hyperbolas, the candidate region of the keystoke's position can be narrowed. Intuitively, the more smartphones used, the more half hyperbolas obtained, the smaller candidate set we get, and the more precise position is estimated. Of course, it also increases the computation time. In Fig. 7, the key 'a' is typed, and we plot three heatmaps generated by exploiting two, three and four microphones, respectively. In this comparison, when only two microphones are used, six keys are involved in the overlap region, the area of which occupies four keys. When three phones are used, the area of the overlap region occupies only 1.4 keys, and most of the part covers key 'a'. As we can see, the result with four phones is even better but still close to that obtained using three phones. Generally speaking, the candidate area is also influenced by the relative position and angle of phones (details in Section 5.5). We take a trade-off and exploit three smartphones to introduce our design.

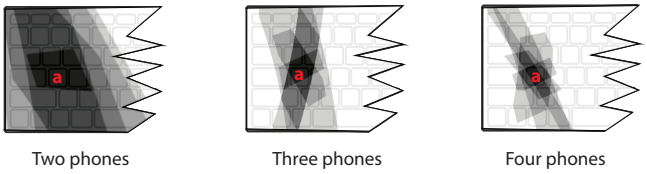


Figure 7: The candidate set of key position is well narrowed with more smartphones (i.e., microphones).

4.2 What If the Layout of the Keyboard Is Unknown?

In most attack approaches using emanation information [37, 6, 18], the layout of the keyboard is assumed to be known to the attackers. The authors in [18] define a key stroke as "far" or "near" to the accelerometer sensor, then utilize the correlation of distance to infer the most likely combination of characters. As we know, the relative distance varies with different keyboards. Such difference breaks the distance correlation, and thus undermines the inference accuracy.

How to adapt or even infer the exact layout of keyboard is beyond scope of this work. Prior knowledge of keyboard layout is critical to our approach, which is based on geometric

Algorithm 2 Algorithm for keyboard reconstruction

Input: Candidate areas $\{P_i\}_{i=1}^{m}$, an 2D plane defined by smartphones O, the keyboard layout R
Output: Keyboard center position r, the rotation angle α

```
Calculate the intersection of the candidate areas, and find the maximum intersections {I_1, I_2, ...}
bestObj = 0
foreach pair of key regions R'_{i_1}, R'_{i_2} do
    foreach pair of maximum intersections I_{j_1}, I_{j_2} do
        Match the centers of the segment I_{j_1} I_{j_2} and the segment R'_{i_1} R'_{i_2}
        Rotate the segment R'_{i_1} R'_{i_2} around its center until it coincides with I_{j_1} I_{j_2}
        Calculate the center of the keyboard r_0 and the rotation angle α_0
        Calculate the objective function value of Eq. 13 as obj
        if obj > bestObj then
            bestObj = obj
            r = r_0
            α = α_0
```

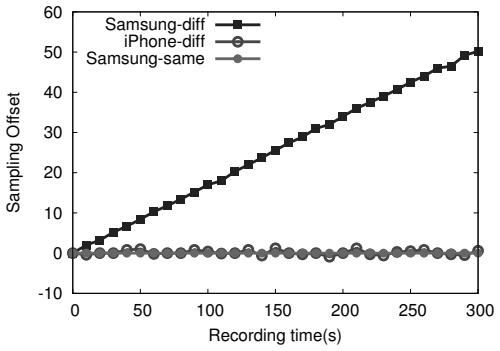


Figure 8: The sampling offset between different microphone pairs. Samsung-diff describes the sampling offset between two microphones in two Samsung Galaxy Note 2 Smartphones, while Samsung-same plots the sampling offset between two microphones in one Samsung Galaxy Note 2 Smartphone.

computation and does not rely on the content correlation. For example, there is a 'fn' button at the lower left position in some notebook keyboards such as Mac Pro, but not in HP notebooks. Our approach fails to distinguish these two cases, but can detect that the keystroke at that position is typed.

5. EVALUATION

In this section, we evaluate our approach through a series of experiments. To show our context-free attack is accurate and feasible in many scenarios, we test the detection accuracy for each key, and analyze the spatial correlations. In addition, considering more or fewer mistypes and some special characters exist in the text, we also try to identify three keys: 'space', 'enter', and 'backspace' (i.e., 'delete' on the Mac keyboard). Besides accuracy, robustness is also examined. Unlike the context-based attacks, our approach does not require any learning stage. Finally, we discuss the details in keyboard reconstruction.

5.1 Implementation

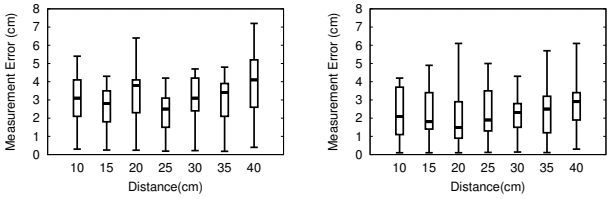
5.1.1 Overall Settings

We implement our primary acoustic localization program on Android v4.3 operating system. In the experiments, we use three Samsung Galaxy Note 2 smartphones, each of which equips with two microphones, one on the top and the other at the bottom. The sampling rate of microphone is 44.1kHz. By default, three smartphones are placed as shown like $1-3-5$ in Fig. 17, being parallel to the short edge of the keyboard. Our approach is evaluated for Apple keyboard MB869LL/A and a mechanical keyboard Filco-87.

5.1.2 Synchronization

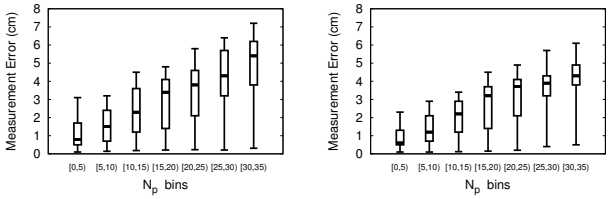
Accurate time synchronization between the microphones is a critical precondition of applying TDoA in our approach. To achieve this, we first synchronize the initial time differences, and then eliminate the clock drift of the microphones with linear fitting algorithm.

Initial time synchronization. We apply the method in [10] to synchronize the initial time among different microphones.



(a) Apple keyboard (b) Filco keyboard

Figure 9: Distance deviation vs. distance from microphone to key



(a) Apple keyboard (b) Filco keyboard

Figure 10: Distance deviation vs. N_p

The main idea is leveraging a reference signal and known ground truth of device locations to calculate the corresponding time difference. To be more specific, we type key 'a' and record the acoustic signals with three smartphones, respectively. The expected time difference can be easily obtained with the distance difference from microphones to key 'a'. By comparing this ground truth and measured result from GCC-PHAT, all the smartphones are synchronized to this reference acoustic signal. This method provides sample-level synchronization accuracy which guarantees not to incur an inconvertible error.

Clock drift elimination. After the initial timings are synchronized, we need to eliminate the clock drift of the microphones. Due to different oscillator frequencies, the build-in clocks may trigger to sample more or less than 44.1k per second. The synchronization issue imposed by the deviation in microphone sampling rate is a complex research problem studied by various researchers [30, 28, 15].

Figure 8 shows some observations on the sampling offset distributions. For Galaxy Note 2 smartphone, clock drift is not severe between two microphones in one smartphone. Even for the microphones in two smartphones, linearity relationship exists clearly and stably in 300s. In our experiments, we adopt simple linear fitting algorithm which provides an offset of only several sample points in a short duration. In Fig. 8, we can also find that, iPhone5 is accurately synchronized at sample-level even for two microphones not in one smartphone. Unfortunately, Apple does not provide the APIs to proactively access more than one microphone in iPhone 5 or 5S. Besides Samsung and iPhone, other smartphones like Nexus 5 are also tested to show a linear relationship between the elapsed time and the number of offset samples [30].

5.2 TDoA Deviation

In order to set a proper width for the hyperbola band, we analyze the distribution of TDoA deviation. Our first

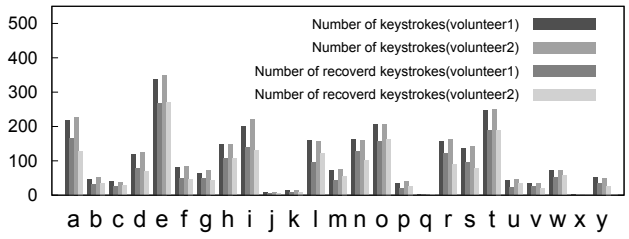


Figure 11: Detection accuracy for each letter involved in the text.

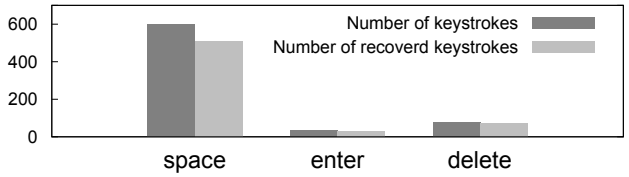


Figure 12: Detection accuracy for special characters in a natural way to type.

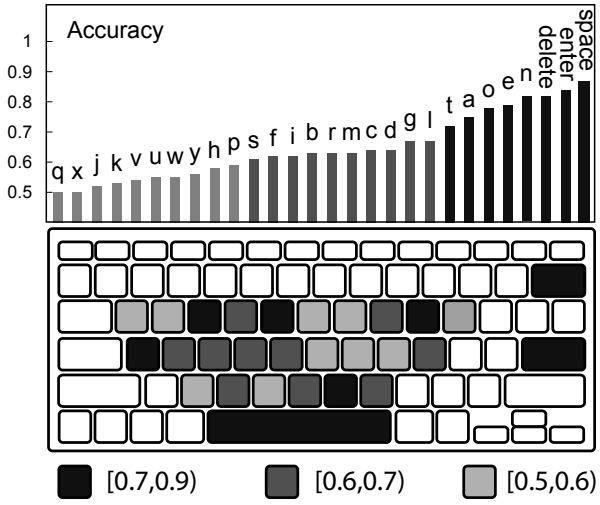


Figure 13: Character ranking according to detection accuracy.

hypothesis is that the deviation should be related with the distance from key to microphone. We conduct several groups of experiments on Mac MB869LL/A and Filco-87 keyboard to estimate the TDoA for keystroke 'a', by varying the distance. Each group considers 100 repeated cases. As shown in Fig. 9, the measured variance is irrelevant to the distance. For Mac MB869LL/A keyboard, the average measure error is the least (i.e. 2.2cm) when the smartphone is 25cm far from the key. For Filco-87 keyboard, the distance of 20cm achieves the minimum average measure error of 1.5cm.

We then explore the correlations between TDoA deviation and the number of excessive values N_p, the key factor in GCC-PHAT algorithm (detailed in Eq. 11, and β is set to 0.8). For each key, we repeatedly press for 10 times and count N_p as well as its measurement errors. As shown in Fig. 10, the deviation is strongly correlated to N_p. For experimental keyboards, a small value of N_p usually leads to little estimation deviation. This result confirms our intuition that the number of excessive values generated by GCC-PHAT algorithm influences the variance of estimated TDoA. The boxplots in Fig. 10 present the 5th percentile, the first quantiles, the median, the third quantiles and the 95th percentile for each range of N_p. To draw the half hyperbola band for a pair of microphones i and j, we set the lower and upper bound of the estimated distance difference Δd_{ij} according to the third quantiles. For example, if N_p is no more than 4, we set a width of $1.8cm$ for the generated hyperbola band.

5.3 Detection Accuracy

To evaluate the accuracy of our attack, we had two volunteers type text from Martin Luther King's "I Have a Dream" using only small letters, 'space', 'delete' and 'enter', then tried to recover the keystrokes, respectively. The key logger tool is used to record the ground truth of each keystrokes. In total there are 3372 keystrokes pressed by the first volunteer and 3452 keystrokes pressed by the second volunteer, with almost all the letters except letter 'z'. Specifically, letter 'e' is the most frequently typed. As illustrated in Fig. 11, in 337 and 350 actual keystrokes of 'e', our approach successfully recovers 268 and 269 of them in two traces. The average recovery accuracy for letter 'e' is 78.1%. Besides, as shown in Fig. 12, there are 597 instances of 'space' and 32 of 'enter' typed by the first volunteer, and our approach recovers 85.8% of them. Moreover, 70 of the keystrokes for 'delete' are also distinguished in a total of 78 keystrokes. Intuitively, 'delete' should not be contained in the text. In practice, however, people type 'delete' eliminate the inevitable mistypes, which are usually ignored in existing approaches. We also noted that, typing rate has been proved as a key factor in existing context-based approaches [18], because the displayed signatures may be greatly different. As recorded, the average typing speed is 577 and 321 characters per minute to complete the task for two volunteers. We know that top typists input around 600 - 700 characters per minute which is slightly faster than our first volunteer; hence the typing rate will not be a big challenge to our approach. In practice, our approach successfully recovers more than 72.2% of keystrokes for both of them.

5.3.1 Spatial Correlations

We rank the characters according to the detection accuracy, as illustrated in Fig. 13. More than 70% of 'space', 'enter', 'delete', 'n', 'e', 'o', 'a' and 't' are successfully distinguished. To find spatial correlations, we plot the detection accuracy on the keyboard with 3 grey levels. The darker the area, the more accurate the detection of this key. As we can see, most of those characters with high detection accuracy are located at the edges of the tested area (i.e., the white keys in Fig. 13 are not tested in our experiments). This is because when they are typed, the overlap area of generated hyperbola bands and tested keyboard is small. That is, we narrow the effective candidate areas by ignoring the part outside of the keyboard. Take key 'a' as an example; the generated candidate area often covers the key 'CapsLock', which is presupposed not to be tested. In addition, frequently typed characters are more likely to be exploited to

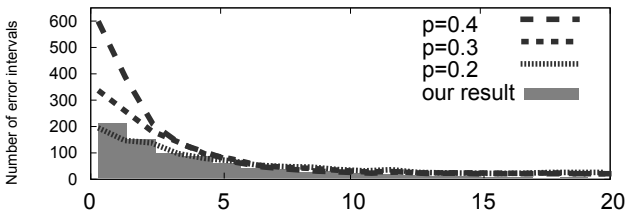


Figure 14: Distribution of interval between two consecutive faulty recognitions.

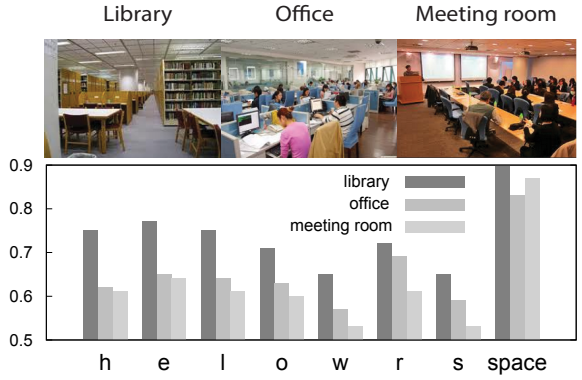


Figure 15: Detection accuracy in three situations: Library, Office, Meeting room.

reconstruct the keyboard, and thus their maximum intersections are narrowed.

5.3.2 *Misrecognition Interval*

In context-based approaches, context correlations are leveraged to help infer typed text, and they also increase the risk of successive false recognitions. For example, [37] presents an attack based on the Hidden Markov Model(HMM), which models the transition probability among the letters in a word, and the words in a sentence. Once a letter is misrecognized, a domino effect probably occurs, which largely decreases the detection accuracy. In our experiments, we examine the intervals between any pair of consecutive faulty recognitions in the second volunteer's trace. There are 944 keystrokes not identified. Figure 14 shows the distribution of intervals. Intuitively, if we consider that the identifications made in such a context-free way are independent, the interval should follow a geometric distribution. The average false negative rate (i.e., miss detection rate) in this trace is 28%. Therefore we plot three geometric distributions with parameter $p = 0.2, 0.3, 0.4$, respectively. As we can see, they show similar trends, which also verifies that the faulty recognitions in our approach are almost independent.

5.4 Robustness with Respect to Ambient Noise

Generally, emanation detection degrades when a noisy background is present, since the recorded signals may show greatly different distributions with a random noise. Some attacks recognize the pressed key based on fingerprint-based schemes [37, 2]. These approaches mainly rely on pre-built database or learning models, and identify the keystrokes by matching the extracted signals with existing signatures.

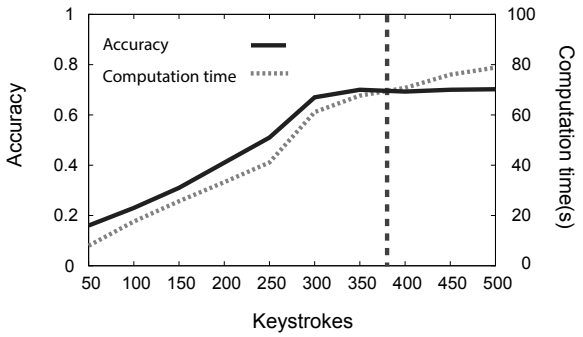


Figure 16: Efficiency of keyboard reconstruction using different number of keystrokes.

They may work well without any other background interference, but degrade when the sampled environment greatly differs from that of the learning stage. In contrast, our approach is designed based on geometric methods, ignoring any matching issues. In this group of experiments, we examine the accuracy in three situations: 1) the quiet library; 2) the office bustling with people and activity; 3) the noisy meeting room. We type "hello world" 100 times in each situation, and try to recover every keystroke. The results are shown in Fig. 15. In all situations, our approach achieves a detection accuracy of at least 64%. More than 83% of the character 'space' are recovered, even in a noisy meeting room.

5.5 Keyboard Reconstruction

In this section, we specifically discuss the stage of keyboard reconstruction, which is an essential part of keystroke recovery. In other words, if the position of the keyboard is incorrectly estimated, all the inferred text is totally meaningless.

5.5.1 *Efficiency of Keyboard Reconstruction*

To make our approach available in more general scenarios, the relative location and angle of the keyboard are supposed to be unknown to the attackers. Before recovering typed text, a number of keystrokes are exploited to reconstruct the keyboard. Intuitively, the more signal information collected, the higher the estimation achieved, but the more computation costs incurred. To find a trade-off, we estimate the keyboard location based on different numbers of keystrokes, and then examine how accurate our approach proves for text recovery. As illustrated in Fig. 16, processing time cost in keyboard reconstruction increases in a linear-like trend following the number of keystrokes. Varying with processing time, the recovery accuracy shows two obvious trends. Before we raise the number of keystrokes beyond 370, the accuracy monotonically increases from 18.1% to 72.2%. After that, it maintains around 72.2%, and even occasionally decreases when more keystrokes are involved. From this it can be understood that, the area of *maximal intersection* for a key may expand if the nearby keys are frequently typed during this period.

5.5.2 *Relative Position of Smartphones*

The location of the keys is narrowed down by overlaying several half hyperbola bands. The shape and location of the half hyperbola bands change as we put the detected device

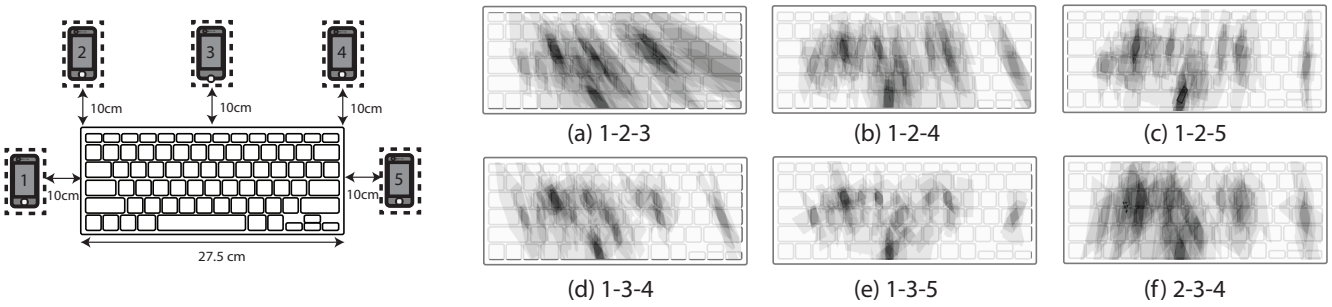


Figure 17: The heatmaps generated for keyboard reconstruction in different combinations of smartphone positions.

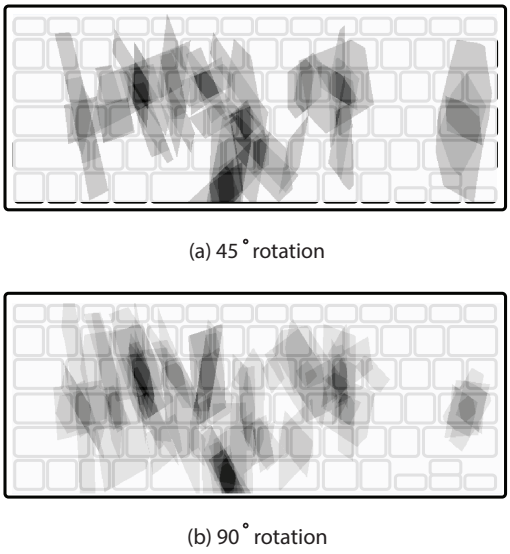


Figure 18: The heatmaps generated for keyboard reconstruction when the smartphone on position 5 rotates. (a) rotate 45° clockwise; (b) rotate 90° clockwise.

at different positions. We test the influence of the location of smartphones on the detection accuracy, by considering five possible positions as depicted in Fig. 17. These five positions are denoted by $1 \sim 5$ clockwise. Avoiding repetitive or symmetrical cases, we tested six combination possibilities for three smartphones (the three phones are assumed to be put at different positions). In the results, when the phones are positioned in $1-3-5$, almost all the *maximum intersections* tightly cover a single key or a part of one key. This prevents a large range of the keyboard rotation which greatly enhances the accuracy of keyboard reconstruction. In contrast, when the phones are positioned in $1-2-3$ and $2-3-4$, more than half of *maximum intersections* are close to each other or cover several keys, which degrades the keystroke recovery accuracy.

5.5.3 Relative Angle of Smartphones

Besides the relative position, the relative angle of the smartphones is also critical for keyboard reconstruction. It is easy to imagine that, the generated half hyperbola rotates while the smartphones spin around on the spot. In our experiments, three angles are considered: 0°, 45° and 90°. Assume that three smartphones are put at positions $1-3-5$. Then, we modify the relative angle of the smartphone on position 5. Fig. 18(a) and Fig. 18(b) show the heatmaps when it rotates 45° and 90° clockwise, respectively. As we can see, when we rotate the phone 90° clockwise, most of the intersection areas are more concentrated. That is, even if the error range is wide for the generated hyperbolas, their overlap area effectively covers the typed key.

6. RELATED WORK

6.1 Keyboard Emanation

Previous attacks have always been conducted with complicated devices and complex technical methods, which do not threaten the general public. However, with the growing popularity of personal electrical and mechanical devices [3], emanations of those devices create more personal information leakages, such as the keyboard emanations. Many efforts have been made to recover keystrokes through acoustic sources. The first concrete attack using keyboard acoustic emanations of which we are aware is made by Asonov and Agrawal [2]. They extract FFT values as the features of the keystrokes and use supervised learning with the labeled data including 100 clicks of each key. The drawback of supervised leaning is that it works well on labeled data, but fails on unlabeled data. That is, a model trained from one keyboard cannot apply to others, because the signatures generated from them are not ensured to be the same. or have likely distributions. To overcome this problem, [37] proposes a unsupervised learning method which cluster the cepstrum features of keys and recover text by HMM model. Though this method does not need pre-obtained data, they are based on a HMM model trained from some particular languages. Furthermore, some special characters are always pressed during typing. The existences of those characters will significantly reduce the performance of the HMM model. The dictionary-based method proposed by [6] exploits the similarity of keystrokes in a word and the constraints learned from dictionaries to narrow the range of possible words. Besides, the ability to recover typed text using the vibration of desktop has been studied by [18]. They correlate consecutive keys and usage of dictionary information. Recently, [33] exploits Amplitude Spectrum Density which measures multi-path fading to localize keystrokes on conventional surfaces. All of these methods are based on some preliminaries,

such as language knowledge or trained model. In contrast, our method is context-free and geometry-based.

6.2 Phone Localization

There have been tremendous efforts to resolve localization problems using different kinds of signals, such as acoustic signals, ultrasonic signals and radio signals [13, 17, 12, 29, 14]. Though some of them achieve high accuracy up to several centimeters [13, 17], their methods are based on complicated infrastructures and expensive equipment. A pairwise phone to phone acoustic distance measurement named BeepBeep is proposed by [22]. Due to the uncertainty of timestamping in the mobile phone, BeepBeep employs self-recording and sample counting to reduce errors introduced by local clock and receiving uncertainty. The local time is determined by the cross-correlation of two known acoustic signals. [24] and [36] aim to solve the problem of fast accurate localization between two moving phones. Both of them also are based on the traditional cross-correlation to determine the exact local time when receiving the signal. Different from previous work, our research is targeted at determining the local time with unknown signals. It significantly raises the uncertainty of local time estimation.

7. CONCLUSION

In this work, we present a new context-free and geometry-based attack to recover keystrokes using smartphones. Our geometry-based method utilizes TDoA techniques to calculate the relative positions among keystrokes and the microphones equipped in the smartphones. This approach is different from previous methods in two aspects. First, most previous methods are based on the correlations between the contexts of typing while our approach is a pure geometric method to support context-free attack. Our experiments show that the performance of our method does not degrade for random typed contents; furthermore, the success rate of correct key recognition is above 72.2%. Since our method is which independent of the previous context-based methods, an interesting direction of future research would be to combine these two methods together to achieve a better recognition performance. Second, we choose the commodity smartphones as our attaching hardware and show the viability of a low-cost device for such an attack. We hope our attack method will increase awareness of the need for mobile security in computing environments.

8. ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers for their valuable feedbacks. This research is supported in part by the NSFC under Grant No. 61402338, NSFC Distinguished Young Scholars Program under Grant No. 61125202 and NSFC under Grant No. 61472219.

9. REFERENCES

[1] P. Arena. Apple iPhone 5 has three microphones and HD voice support, what's in it for you. `http://www.phonearena.com/news/Apple-iPhone-5-has-three-microphones-and-HD-voice-support-whats-in-it-for-you_id34486`, 2012. [Online; accessed 30-July-2014].

[2] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 3–11, 2004.

[3] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder. Acoustic side-channel attacks on printers. In *Proceedings of USENIX Security Symposium*, pages 307–322, 2010.

[4] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of ACM CCS*, pages 73–84, 2010.

[5] A. O. Bauer. Some aspects of military line communicatoins as deployed by the german armed forces prior to 1945. In *Proceedings of 5th Aunnual Colloquium, The History of Military Comunications*, 1999.

[6] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of ACM CCS*, pages 245–254, 2006.

[7] D. Dagon, T. Martin, and T. Starner. Mobile phones as computing devices: The viruses are coming! *IEEE Pervasive Computing*, 3(4):11–15, 2004.

[8] A. Davis, M. Rubinstein, N. Wadhwa, G. Mysore, F. Durand, and W. T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics*, 33(4):79:1–79:10, 2014.

[9] W. V. Eck. Electromagnetic radiation from video display units: An eavesdropping risk? In *Computers and Security*, pages 4:269–286, 1985.

[10] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, 2002.

[11] W. Enck, M. Ongtang, and P. D. McDaniel. On lightweight mobile phone application certification. In *Proceedings of ACM CCS*, pages 235–245, 2009.

[12] L. Girod, M. Lukac, V. Trifa, and D. Estrin. A self-calibrating distributed acoustic sensing platform. In *Proceedings of ACM SenSys*, pages 335–336, 2006.

[13] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2-3):187–197, 2002.

[14] M. Hazas and A. Hopper. Broadband ultrasonic location systems for improved indoor positioning. *IEEE TMC*, 5(5):536–547, 2006.

[15] Y. Jia, Y. Luo, Y. Lin, and I. Kozintsev. Distributed microphone arrays for digital home and office. In *Proceedings of IEEE ICASSP*, pages 1065–1068, 2006.

[16] C. Knapp and G. C. Carter. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(4):320–327, 1976.

[17] M. Maróti, P. Völgyesi, S. Dóra, B. Kusy, A. Nádas, Á. Lédeczi, G. Balogh, and K. Molnár. Radio interferometric geolocation. In *Proceedings of ACM SenSys*, pages 1–12, 2005.

[18] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iphone: decoding vibrations from nearby

keyboards using mobile phone accelerometers. In *Proceedings of ACM CCS*, pages 551–562, 2011.

[19] R. Meng, J. Isenhower, C. Qin, and S. Nelakuditi. Can smartphone sensors enhance kinect experience? In *Proceedings of ACM MobiHoc*, pages 265–266, 2012.

[20] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: your finger taps have fingerprints. In *Proceedings of ACM MobiSys*, pages 323–336, 2012.

[21] E. Nordström, D. Aldman, F. Bjurefors, and C. Rohner. Search-based picture sharing with mobile phones. In *Proceedings of ACM MobiHoc*, pages 327–328, 2009.

[22] C. Peng, G. Shen, and Y. Zhang. Beepbeep: A high-accuracy acoustic-based system for ranging and localization using COTS devices. *ACM Trans. Embedded Comput. Syst.*, 11(1):4, 2012.

[23] C. Qin, X. Bao, R. R. Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *Proceedings of ACM MobiSys*, pages 1–14, 2011.

[24] J. Qiu, D. Chu, X. Meng, and T. Moscibroda. On the feasibility of real-time phone-to-phone 3d localization. In *Proceedings of ACM SenSys*, pages 190–203, 2011.

[25] Y. Rui and D. Florencio. Time delay estimation in the presence of correlated noise and reverberation. In *Proceedings of IEEE ICASSP*, pages ii–133, 2004.

[26] R. SINGEL. Declassified NSA Document Reveals the Secret History of TEMPEST. `http://www.wired.com/2008/04/nsa-release-se`, 2004. [Online; accessed 23-July-2014].

[27] S. Singh, S. Nelakuditi, R. R. Choudhury, and Y. Tong. Your smartphone can watch the road and you: mobile assistant for inattentive drivers. In *Proceedings of ACM MobiHoc*, pages 261–262, 2012.

[28] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network, IEEE*, 18(4):45–50, 2004.

[29] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of USENIX Security Symposium*, 2001.

[30] S. Sur, T. Wei, and X. Zhang. Autodirective audio capturing through a synchronized smartphone array. In *Proceedings of ACM MobiSys*, pages 28–41, 2014.

[31] T. Thomas. Malware on the move., 2008.

[32] H. Wang and P. Chu. Voice source localization for automatic camera pointing system in videoconferencing. In *Proceedings of IEEE ICASSP*, pages 187–190, 1997.

[33] J. Wang, K. Zhao, X. Zhang, and C. Peng. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of ACM MobiSys*, pages 14–27, 2014.

[34] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin. Sensing vehicle dynamics for determining driver phone use. In *MobiSys*, pages 41–54, 2013.

[35] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin. Detecting driver phone use leveraging car speakers. In *Proceedings of ACM MobiCom*, pages 97–108, 2011.

[36] Z. Zhang, D. Chu, X. Chen, and T. Moscibroda. Swordfight: enabling a new class of phone-to-phone action games on commodity phones. In *Proceedings of ACM MobiSys*, pages 1–14, 2012.

[37] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. In *Proceedings of ACM CCS*, pages 373–382, 2005.