UltraSnoop: Placement-agnostic Keystroke Snooping via Smartphone-based Ultrasonic Sonar

YANCHAO ZHAO, YIMING ZHAO, SI LI, and HAO HAN, Nanjing University of Aeronautics and Astronautics, China
LEI XIE, Nanjing University, China

Keystroke snooping is an effective way to steal sensitive information from the victims. Recent research on acoustic emanation-based techniques has greatly improved the accessibility by non-professional adversaries. However, these approaches either require multiple smartphones or require specific placement of the smartphone relative to the keyboards, which tremendously restricts the application scenarios. In this article, we propose UltraSnoop, a training-free, transferable, and placement-agnostic scheme that manages to infer user's input using a single smartphone placed within the range covered by a microphone and speaker. The innovation of Ultrasnoop is that we propose an ultrasonic anchor-keystroke positioning method and a Mel Frequency Cepstrum Coefficients clustering algorithm, synthesis of which could infer the relative position between the smartphone and the keyboard. Along with the keystroke time difference of arrival, our method could infer the keystrokes and even gradually improve the accuracy as the snooping proceeds. Our real-world experiments show that UltraSnoop could achieve more than 85% top-3 snooping accuracy when the smartphone is placed within the range of 30–60 cm from the keyboard.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing;**

Additional Key Words and Phrases: Keystroke snooping, ultrasonic tracking and localization, system implementation, smartphone

ACM Reference format:
Yanchao Zhao, Yiming Zhao, Si Li, Hao Han, and Lei Xie. 2023. UltraSnoop: Placement-agnostic Keystroke Snooping via Smartphone-based Ultrasonic Sonar. *ACM Trans. Internet Things* 4, 4, Article 22 (November 2023), 24 pages.
https://doi.org/10.1145/3614440

1 INTRODUCTION

Snooping keystrokes (a.k.a., keystroke inference attacks) seriously threatens information security and privacy. By launching such an attack, an adversary has an opportunity to steal sensitive information such as accounts, passwords, credit card numbers, SSNs, and confidential documents [1, 15, 29, 30] from the victims when they are typing on a keyboard. Smartphone-based snooping

This work was supported in part by the National Natural Science Foundation of China under Grant (No. 6217221, 61972200) and in part by the Natural Science Foundation of Jiangsu Province (No. BK20200067), in part by the A3 Foresight Program of NSFC (Grant No. 62061146002).
Authors' addresses: Y. Zhao, Y. Zhao, S. Li, and H. Han, Nanjing University of Aeronautics and Astronautics, China; yczhao@nuaa.edu.cn, 736531683@nuaa.edu.cn, 386510072@qq.com, hhan@nuaa.edu.cn; L. Xie, Nanjing University, China; email: lxie@nju.edu.cn.

2577-6207/2023/11-ART22 $15.00
https://doi.org/10.1145/3614440

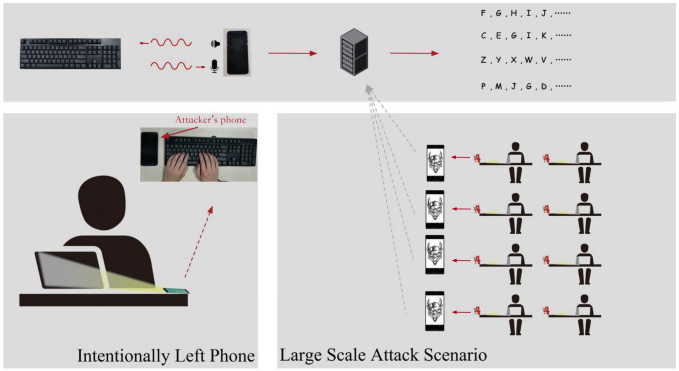

Fig. 1. Attack scenarios of intentionally leaving phone to snoop at public (left), and installing malicious software to snoop in a large scale (right).

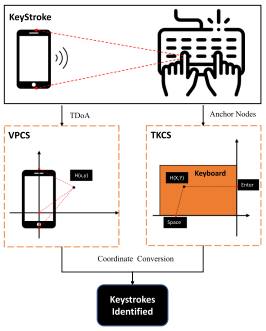

Fig. 2. Keystroke inference process.

[15, 18, 24] further eases the launching when an adversary could intentionally leave his or her own smartphone near the victim's keyboard. Furthermore, an attacker could spread a malicious mobile app (e.g., in app markets) that pretends to be a normal audio playing and recording application but stealthily collects user's keystroke data over the Internet. He or she may affect a large volume of smartphones and enable large-scale keystroke inference attacks as shown in Figure 1.

Existing work on keystroke snooping using smartphones could be divided into three categories: voiceprint-based methods [1], language-model-based methods [2], and ranging-based methods [15, 29, 30]. The voiceprint-based approaches assume that each key has a unique "fingerprint" that could be identified through received keystroke emanations. These solutions typically rely on supervised learning to derive an identification model from the target keyboard [1]. Since the voiceprint of a keyboard may change over time during the usage, such solutions are not reliable and need retraining repeatedly. Linguistic approaches assume that the sequence of typing follows some language models, so keystrokes could be derived from the context information [2]. However, this type of methods could not be used for password inference and may fail with typos, since the password typically does not follow any language model. As a result, linguistic approaches are usually applied along with other techniques to improve the accuracy [7, 23].

The up-to-date work on keystroke inference mostly focuses on ranging [15, 29], which basically uses microphones on a single or multiple smartphones to measure the **time difference of arrival (TDoA)** of the acoustic emanation from different keys. Two microphones receive the same keystroke sound. Due to the different distances, there will be TDoA. Based on TDoA we make a single curve with two microphones as the focus. More microphones can have more single curves. The intersection of multiple single curves is the exact key position.

However, these solutions have a very strong assumption on the prior knowledge of the keyboard as well as the relative position/direction of measuring smartphones to build a keyboard layout. Eavesdrop [7] has proved that these schemes have low keystroke snooping accuracy when the phone–keyboard relative position is unknown. To address this problem, Zhu et al. [29] proposes a keyboard localization and reconstruction method, but the assumption on multiple smartphones makes their method impractical. Eavesdrop exploits the language limits to identify several pilot keys as the anchors to estimate the phone–keyboard relative position further. However, the system depends on the known language context, which significantly limits the attack scenario. Eavesdrop [7] leverages linguistic statistics to identify a set of pilot keys that serve as anchors for estimating the relative position between a phone and keyboard. Nonetheless, the effectiveness of this system

is contingent upon prior knowledge of the language context, thereby constraining its potential attack scenarios. Yu et al. [26] eavesdropped on smartphone keyboard inputs by tracking the attenuation of acoustic signals, phase changes of keystrokes, and Doppler effects received through the smartphone's microphone. However, the work is applicable to smartphone keyboards and relies on the directional changes in the movement of the user's fingers. Other researchers endeavored to use WiFi [6, 13] or eye movements [4] to snoop.

However, shortcomings are still inevitable for existing work. Acoustic-emanation-based approaches suffer from environmental noises and the interference from the other keyboards. Linguistic approaches require linguistic context that is not reliable for an accurate inference. Current ranging-based solutions only work with the prior knowledge of the keyboard position and rely on the use of multiple recording devices and specific placements of those devices. This fact greatly limits the application scenarios. After all, an attacker could not control the target to place the smartphone to the presumed position. Thus, none of existing techniques can be easily applied to launch the large-scale attack scenario in Figure 1.

In this article, we propose a placement-agnostic keystroke inference method and develop a smartphone rooted snooping system named UltraSnoop. Motivated by recent mobile phone acoustic sensing techniques, the system actively emits acoustic signals and passively records echoes as well as acoustic emanations to infer keystrokes on **commercial-off-the-shelf (COTS)** smartphones. To achieve placement-agnostic keystroke inference, we introduce **Virtual Phone Coordinate System (VPCS)** and **Theoretical Keyboard Coordinate System (TKCS)** to convert the TDoA into placement-agnostic locations and infer keystrokes based on them. Initially, the TDoA of each keystroke is extracted from recorded echoes. Subsequently, the TDoA of keystrokes are translated into VPCS coordinates by VPCS. Finally, the VPCS coordinates undergo further processing utilizing TKCS to obtain precise estimates of the keystrokes. A schematic representation of this keystroke inference process can be observed in Figure 2.

The advantages of our method are in three aspects. First, the actively emitted ultrasound signal enables distance measuring [3, 25] and finger movement/tapping tracing [21] on a single smartphone so as to localize a keystroke. The higher the sampling rate, the more accurate the keystroke positioning we can achieve. Second, our method is context free and training free. Thus it is robust for password snooping and keyboard variance. Last, we adopt ultrasound-based signals that are not perceivable. It is worth noting that the ultrasound signal will not interfere with keystroke emanations; hence it is complementary to existing voiceprint-based approaches.

However, to achieve this, we have to overcome the following technical challenges.

- **Keystroke detection and position:** The active sonar suffers from multiple echoes from all surrounding objects. How to effectively extract the keystroke and localize its position is the first challenge.
- **Positioning Accuracy:** Both the hardware limits (the chips can support up to 192 kHz, while the android system only support up to 48 kHz) and the environmental changes could affect the localization accuracy. With these non-negligible errors, how to accurately infer the key identity becomes the second challenge.
- **Keyboard position and layout construction:** Now, we consider the large scale keystroke snooping attack, where we cannot presume the relative position of the phone and keyboard or even the layout of the keyboard. It is another challenge to localize the keyboard and construct the layout of the keyboard.
- **Signal occlusion from hand:** If the phone is placed on the right or left of the keyboard, then we can hardly estimate the position of keystroke only with the echo due to the hand occlusion. How to estimate the keystrokes of occluded hands becomes the last challenge.

To solve the first challenge, we use the echo of a short-term pulse signal that is actively sent to make a rough measurement of the keystroke. The specially modulated signal can effectively measure the position where the keystroke occurs by an echo. Combined with the TDoA between keystrokes reaching different microphones, we can further determine the single curve where the keystrokes are located. Then we combine rough positioning and single curve to further determine where the keystrokes occur.

We then managed to solve the second challenge. Along with the TDoA of keystroke sound, we are able to identify the sound of different keystrokes. By utilizing **Mel Frequency Cepstrum Coefficients (MFCCs)** [30] and the stable voiceprint, we propose a **MFCC-Position Mapping and Clustering (MPMC)** algorithm to accumulatively improve the inference accuracy over the snooping process. Furthermore, this algorithm could trace back all the previous input and thus rectify the snooping errors.

Next, to solve the third challenge, we propose an "anchor point"–based solution to build the coordinate system without knowing the position of the phone. We mainly chose the anchor points as "Enter" and the "Space," which is mainly based on the observation that these keys have distinguished "voiceprint" [1] that could be told apart from other keys even across different keyboards. Then the *VPCS* model and *TKCS* model are proposed to build both the coordinate systems of the phone and the keyboard, the combination of which enables us to obtain the relative position and the keyboard layout.

To solve the last challenge, we observe that keystrokes could be classified into the groups by TDoA of the same keystroke recorded with two microphones. Along with the input sequence of unoccluded hand, we can get TOP-k result. Altogether, we can accurately snoop the keyboard input independent of positions and keyboards.

The contributions can be summarized as follows:

- We present a transferable, training-free, and placement-agnostic keystrokes snooping system based on a single smartphone. Our system greatly eases keyboard snooping via a pure software method and enables large-scale keyboard snooping attacks.
- We propose two virtual coordinate systems, VPCS and TKCS. Combined with active acoustic sensing, the system implements placement-agnostic keystrokes snooping without any training.
- We further develop an MFCC-Position mapping and clustering algorithm, which greatly improve the snooping accuracy against the localization error caused by the hardware limits and environmental interference.
- We conduct extensive real-world experiments that show our system could achieve more than 85% inference accuracy even when the smartphone's placement is unknown in advance.

2 RELATED WORK

2.1 Keyboard Snooping Attack

The keyboard snooping attacks has long been noticed and researched. Initially, Asonov et al. [1] notice that each button has its own distinctive voiceprint. Thus, they developed a voiceprint-based supervised learning algorithm to identify a single key. An improved algorithm was proposed by Berger et al. [2]. Different from the first algorithm, in the training stage, not only is the single button considered, but also the front and rear keystrokes are all used as the training input. Zhuang et al. [30] present a improved algorithm based on the previous method by considering that the candidate range is too large in the previous algorithm. Zhu et al. [29] adopts the TDoA of the keystroke from double microphones on the same mobile phone and combines the position of the three hyperbolas of the mobile phone to locate of the keystroke. Another TDoA-based method

is proposed by Liu et al. [15]. It assumes that the relative distance between the keyboard and the smartphone is known and classifies the similar TDoAs into the same key cluster. Among the candidate keys in the same key cluster, it distinguishes them with feature of voiceprint. Eavesdrop [7] combined the linguistic model and TDoA to estimate the phone-keyboard relative position and propose a Position free Keystroke Monitoring scheme based on MFCC and k-means clustering. LOL [20] presents an angle-based sampling method to collect keystroke acoustic signals for training and enables small angular position-independent keystroke inference. Mixed CNN and RNN [8] classifies keystrokes from audio and motion sensor data of multiple phones based on Mixed Convolutional and Recurrent Neural Networks. Yu et al. [26] used the attenuation of acoustic signals received by the microphone of a smartphone to locate the user's keystroke fingers and tracked finger movements through phase shifts and Doppler effects to reduce errors in keystroke location based on acoustic signal attenuation.

2.2 Acoustic Signal-based Applications

Acoustic signal recently has been widely applied in developing new mobile applications. Wang et al. developed a *UbiK* [24] based on acoustic signals that enables the keystroke input on a printed keyboard on a paper. It achieves 95% of localization accuracy across a variety of settings. Wang et al. [25] developed a phase-based gesture tracking system and achieved millimeter-level accuracy and less than 15.5-ms latency. Chen et al. [3] developed an acoustic device-free hand tracking system. Their error of tracking the movement can be within 0 to 4 cm. Tan et al. [22] implement a lip-reading system through ultrasonic sensing using the reflection of ultrasonic from mouth. Their average accuracy around 74.8% with the number of words ranging from 1 to 6. Zhou et al. [28] utilized acoustic signal to do user identification and authentication. Zhang et al. [27] developed a system called *SoundTrak* to turn the phone into an active sound signal to track three-dimensional (3D) gestures, and Ling et al. [14] identify some gestures by acoustic signals. Mao et al. [16] developed a high-precision acoustic motion tracking system, CAT. Huang et al. [9] used the acoustic signal to get the current position. Lee et al. [12] used chirp for communication between smartphones.

2.3 Summary of the Related Work and Our Difference

Inspired by the predecessors, our work has major differences from these works and makes up for some of their drawbacks. The accuracy of voiceprint-based research is not practical, as the voiceprint could vary from different keyboards. Context-based research is unable to recognize random characters due to language restrictions, while the current ranging-based research relies on the relative position of smartphones and keyboards. Inspired by recent progress of active acoustic signal ranging and tracing technology [3, 25], we explore the active ultrasonic ranging-based method to solve the keyboard positioning problem and infer the keystroke accurately. Compared with existing techniques, our proposed UltraSnoop achieves placement-agnostic keyboard snooping with a single device, thereby addressing the issue of decreased performance in distance-based approaches due to changes in relative positioning of phone and keyboards. Our method is training free and does not necessitate pre-training, rendering it appropriate for large-scale attack scenarios. Furthermore, our approach is transferable and more facilely applicable to a variety of devices than voiceprint-based solutions.

3 MODELS AND PRELIMINARIES

In this section, we introduce the attack models followed by technical preliminaries.

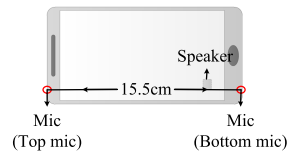

Fig. 3. The structure of a smartphone equipped with top and bottom microphones.

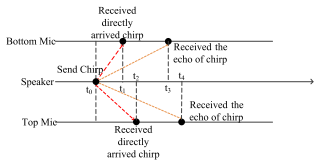

Fig. 4. Time sequence of events within a cycle.

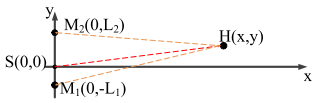

Fig. 5. Geometric model of *VPCS*.

3.1 Adversary Model

We assume a scenario where an attacker can approach the victim's computer and be able to place his or her own mobile device (e.g., smartphone or tablet) near the keyboard used by the victim or install a malicious app on victim's phone. To launch a successful keystroke snooping attack, the device should be able to play and record acoustic signals in the sampling rate of 48 kHz and beyond, which is available on most modern smartphones. There are two representative scenarios: (1) The attacker places the mobile device on a public desk shared with the victim, for example in a library, and (2) the attacker accesses the microphones of the victim phone through a malicious software. The software has been working in the background when the victim places the phone next to a keyboard, so that the attacker can retrieve the recorded data over the Internet.

In addition, we assume that the attacker knows all the possible keyboard layouts. Then the attacker can build a set of theoretical models that contain the spacing and direction information of each key for victim's keyboard. After snooping two or more keys including the anchor nodes, the attacker can calculate the keyboard position and layout.

3.2 Keystroke Localization Model

A large volume of the COTS smartphones support stereo recording and playback. Most of them are equipped with two unfiltered microphones and a speaker, e.g., 360 N6 Pro, Samsung GALAXY S6. Based on such COTS smartphones, we propose ultrasound-based keystroke detection and localization algorithm.

The basic idea is as follows. As is shown in Figure 3, a smartphone is equipped with two unfiltered microphones at the top and bottom of the phone as well as a speaker for playing preset acoustic signals. We choose to use unfiltered microphones, because a filtered microphone is designed to eliminate unnecessary signals. In that case, we cannot receive samples that include both the short acoustic pulse and its echoes. The distance between two microphones is a constant, for the phone we used is 15.5 cm.

Figure 4 shows the timeline graph of our system. We send a short acoustic pulse at t_0. The nearest microphone receives the direct arrived short acoustic pulse at t_1, and the other microphone receives the directly pulse at t_2. After a short period of time two microphones will receive the echo of the pulse at t_3 and t_4. We can use the time interval between the pulse and its received echoes to measure the distance from the speaker to the place where the echo is generated. As we can see in Figure 5, the length of this distance is from H to M_1 or M_2. Figure 6 shows the relative position between the smartphone and the location of the pressed key. H represents the finger pressing the key, and S represents the speaker emitting the pulse. M_1 represents the bottom microphone. M_2 represents the top microphone, which is often used as a noise-canceling microphone. Both microphones will record the pulse and the echo simultaneously. It should be noticed that this model is affected by multi-path reflections, and we have designed a mechanism to address this problem (see Section 4.2).

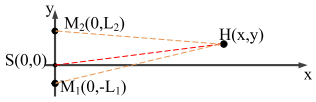

Fig. 6. Localization scenario.

Fig. 7. The relative position "Anchor point".

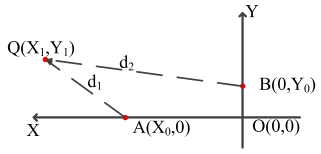

Fig. 8. Geometric model of TKCS.

3.2.1 Virtual Phone Coordination System. Figure 5 shows the geometric model of the relative position between the smartphone and the location of a keystroke. We put the four objects into a virtual coordinate system called *VPCS* and assign each object a virtual coordinate. The origin is the speaker at $S(0,0)$. To simplify our discussion, we align the Y axis with the line between two microphones M_1 and M_2. X axis is a line across the point S and is perpendicular to Y axis. The bottom microphone $M_1(0,-L_1)$ and the top microphone $M_2(0,L_2)$. $L1$ and L_2 are the distances between the two microphones and the speaker, respectively. The keystroke hand $H(x,y)$. The paths of the echo from the hand to two microphones equal to $d_{SHM_1} = d_{SH} + d_{HM_1}$ and $d_{SHM_2} = d_{SH} + d_{HM_2}$, which can be computed as follows:

$$d_{SHM_1} = (t_3 - t_0)c = (t_3 - t_1)c + d_{SM_1}, \tag{1}$$

$$d_{SHM_2} = (t_4 - t_0)c = (t_4 - t_2)c + d_{SM_2}, \tag{2}$$

where the distance between two microphones (M_1 and M_2) and speaker S are represented by d_{SM_1} and d_{SM_2}, while c stands for the speed of sound in the air.

We can solve these equations in the virtual coordinate system shown in Figure 5,

$$\frac{4x^2}{d_1^2 - L_1^2} + \frac{4(y + L_1/2)^2}{d_1^2} = 1, \tag{3}$$

$$\frac{4x^2}{d_2^2 - L_2^2} + \frac{4(y - L_2/2)^2}{d_2^2} = 1. \tag{4}$$

Then we can get following equation:

$$x = \frac{\sqrt{(d_2^2 - L_2^2)(d_1^2 - L_1^2)((L_1 + L_2)^2 - (d_2 - d_1)^2)}}{2(d_2 L_1 + d_1 L_2)}, \tag{5}$$

$$y = \frac{d_1 L_2^2 - d_2 L1^2 - d_2^2 d_1 + d_1^2 d_2}{2(d_2 L_1 + d_1 L_2)}, \tag{6}$$

where d_1 means d_{SHM_1} and d_2 means d_{SHM_2}.

3.2.2 Theoretical Keyboard Coordinate System. If our positioning method has a high precision, then we can recover the keystroke by the method of fitting the coordinates of the keystroke in the *TKCS* with the keyboard model, which we used to steal. However, with the support of current hardware, the error of our positioning method has an error of 3 to 7 points, and each point represents approximately 7 mm. So for each of our measured keystroke positions, the actual position of the keystroke is the intersection of the two circles centered on two microphones. The inner radius of the circle is the measured distance minus the error, and the outer radius is the measurement distance plus the error.

ALGORITHM 1: Anchor Detection

Input: A set of keystrokes clustering in VPCS.

Output: Index of Anchor point

Put the keystrokes into a high-pass filter and split it into equal segments. Extract the tapping peak of ever keystrokes *sample*.
for $i = 1; i <= length(sample); i++$ **do**
  $x \leftarrow sample(i)$
  The energy accumulation for each sample is calculated through a sliding window with a window value of W($0.002 * FS$). And normalize it.
end for
Calculate the energy of each peak. Select the $top - k$ samples *Loc* while having a maximum total energy and a long duration of the maximum peak of the samples.
Divid *Loc* into multiple groups (Loc_Anchor_1 & Loc_Anchor_2) using the $K - mean$ algorithm and related position.
for $i = 1; i <= length(sample); i++$ **do**
  Calculate the Euclidean distance (ED) between the current sample and the average of Loc_Anchor_1(*or* Loc_Anchor_2).
  $a \leftarrow average(sample(Loc\_Anchor_1))$
  $ED_S \leftarrow \left(\sum_{j=1}^{n} \left|a_j - sample_{i,j}\right|^p\right)^{\frac{1}{p}}$
  if $ED_S \leq Threhold$ **then**
    Add $sample_i$ into Loc_Anchor_1.
  end if
  Also do the same for anchor "$Anchor_2$".
end for
return Loc_Anchor_1 & Loc_Anchor_2

Thus, we propose another way to find the positioning coordinates to the keyboard. We noticed that the keys "Enter,""Space," and "Shift" have special locations, unique stroke energy features, and especially "voiceprints," which allow us to first distinguish them from other keystrokes. In such a way, we build a coordinate system of the theoretical keyboard that provides every key's theoretical location by using echo of short acoustic pulse and MFCCs. These anchor nodes are placed at the edge of the keyboard and have special sound profiles, since these keystrokes are generated by the two or three key axes. The keys "Enter" and "Space" are frequently presented in a sentence or at the end of a sentence. For the "Shift key," we usually press it one or more times, when switching input methods or at the beginning of the input. As such, we follow Algorithm 1 to find the features of specific keystroke as "anchor points" through their sound contours, energy, and voiceprints and get the location by using echo. The notation description of Algorithm 1 is shown in Table 1. Note that we do not need to identify all anchor points in keystrokes but to pick some of them with high confidence.

Figure 7 is the schematic diagram of the mutual positional relationship between each key and two anchor points on the keyboard. A represents one anchor point, i.e., "Space," and B represents the other anchor point, i.e., "Enter." Q represents a pressed key on the keyboard. We can put them into a new virtual coordinate system called $TKCS$ shown in Figure 8. The X axis is aligned to the "Space" bar. The Y axis is aligned to the "Enter" and is perpendicular to X axis. The "Space" is on the X axis located at $A(x_0, 0)$. The key "Enter" is on the Y axis located at $B(0, y_0)$. Then we can see that the key is located at $Q(x_1, y_1)$. All the parameters of x_0, y_0, x_1, y_1 can be obtained when we get the model of the keyboard.

Table 1. Notation Related to Algorithm 1

Notation	Description
sample	A set of keystrokes
W	The step size of the sliding window
FS	Sampling rate of keystrokes
Loc	Mapping position of keystrokes
Loc_Anchor_1	The first divided anchor group
Loc_Anchor_2	The Second divided anchor group
ED_S	Euclidean distance from sample to an anchor group
$Threhold$	Empirically set thresholds

3.2.3 Coordinate System Transformation. After calculating the coordinate system of $VPCS$ via Algorithm 1, we get two clusters with each associated to an anchor point. Based on these clusters, we then convert $VPCS$ into $TKCS$. First, we know that the key "Space" is a long bar. Since we assume the model of the keyboard is known in advance, we can compute the vertical distance L of the "Enter" to the line that is passing through the "Space" in the following formula:

$$\begin{aligned} y_S &= k \times x_S + b, \\ \frac{|k \times x_E + (-1) \times y_E + b|}{\sqrt{1+k^2}} &= L. \end{aligned} \tag{7}$$

Here (x_E, y_E) is the center point of the "Enter" cluster, and (x_S, y_S) is the center point of the "Space" cluster. k is the ramp of the straight line passing through the "Space" in $VPCS$, and b is the intercept.

By solving this equation, we find two lines that are symmetric about the line passing through the center of the two clusters. We select a line that does not split the key cluster and designate it as the X axis of $TKCS$. Then we go through the center of the "Enter" cluster to make a Y axis for the vertical X axis.

Next, coordinate X_0 and coordinate Y_0 in $TKCS$ are calculated by the vertical distance of the remaining keystrokes to each axes. We make all coordinates in the first quadrant. At this point, if we mistakenly used another candidate line as the X axis. Some coordinate points may be in the $TKCS$, making the distance to the $Enter$ not equal to the distance calculated in the $VPCS$.

We can calculate its distance, d_1 and d_2 between two "Anchor point," when we calculate a keystroke's coordinates of VPCS. Then we convert it into the TKCS,

$$\begin{aligned} (x_1 - x_0)^2 + (y_1)^2 &= d_1^2, \\ (x_1)^2 + (y_1 - y_0)^2 &= d_2^2. \end{aligned} \tag{8}$$

By solving these formula, we can get two solutions and we select the solution in first quadrant.

3.3 Factors of Affecting Accuracy

The accuracy of our system is affected by some main factors that we will analyze below.

3.3.1 Sampling Rate. In our system, we need to play the pre-edited acoustic signal and record its echo and the keystrokes of the keyboard. When recording the acoustic signal of keystroke, the acoustic signal digitized by an analog-to-digital converter with a fixed sampling. When playing the acoustic signal, the acoustic signal is converted from a digital signal to an analog signal by a digital-to-analog converter. The sampling rate limits the quality of the acoustic signal. Higher sampling rate enables higher-quality acoustic signals. The higher-quality acoustic signals means that we can extract the information we need from a larger amount of acoustic signals.

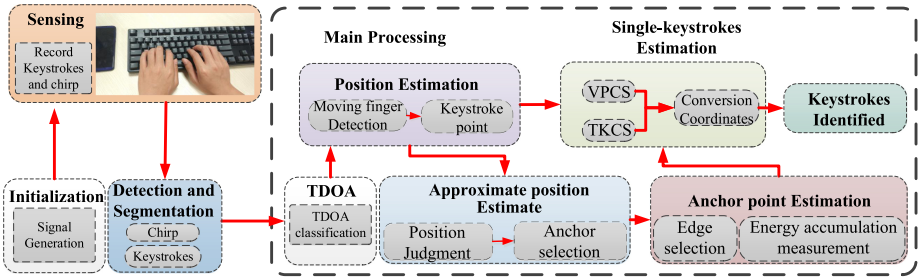

Fig. 9. System overview of UltraSnoop.

Some current smart devices can already support a sampling rate of up to 192 kHz with the development of technology and hardware. However due to the system or hardware limitations, most smart devices generally only support a sampling rate of 48 kHz. So we use the sampling rate used by common mobile devices, 48 kHz. At a speed of 343 m/s, the distance between two acoustic points $\Delta d \approx 7.15$ mm. But if using a sampling rate of 192 kHz, then the distance between the two points can be accurate to 1. 79 mm. The distance between two point can be calculated by $\Delta r = \frac{c}{f_s}$, which f_s is the sampling rate.

3.3.2 Microphone. If the acoustic signal passes through a microphone containing a low-pass filter before the signal is used by the application, then we may not get the real acoustic signal we need. There are a large number of mobile phones with multiple microphones on the market. But these phones are all noise-reducing microphones except the bottom microphone. The system requires at least two microphones without low-pass filters.

3.3.3 Structure of Phone. To calculate the distance between the microphone and the location of keystrokes, we need to solute the triangle equation in the coordinate system, VPCS. In our hypothesis, we put the speaker in the middle of the two microphones to reduce the difficulty of the hypothesis calculation. Furthermore, our assumption is predicated on the presence of two microphones located at the top and bottom of the phone. While eavesdropping requires certain structural specifications, this configuration is commonly found in mainstream commercial smartphones.

The distance between the two microphones also affects the accuracy of the measurement. The distance of Samsung Galaxy 5s smartphone $d = 15.5$ cm, and the speaker is on the back of the phone, in the middle of the two microphones. To eavesdrop on a phone, the system needs to determine the distance between the two microphones on the device. Thankfully, obtaining structural data for all phones from the network is a straightforward process.

3.3.4 Distance between Devices and Keyboard. Any microphone and speaker have a power range. Within this range, we are able to receive significant echoes from an object. When acoustic waves propagate in the medium, the acoustic energy is reduced in propagation due to beam divergence, absorption, reflection, and scattering. The attenuation is equal to the product of the attenuation coefficient and the length of the path. The attenuation coefficient is dependent upon the type of material and the energy of the radiation. So we should keep the keyboard within the ultrasonic coverage.

4 SYSTEM DESIGN

In this section, we will introduce the model and implementation details of the system in this article.

4.1 System Overview

The basic idea of our system is to find the location of the place where the keystroke is produced by using the short acoustic pulse signal. Our system mainly includes the following steps that are illustrated in Figure 9:

4.1.1 Initialization. First, the system generates a pre-edited ultrasonic short acoustic pulse and initializes all microphones and the speaker.

4.1.2 Sensing. After initializing the system, we play the pre-edited signal. Then we keep the microphone in the recording state. From the recorded audio stream, we split the signal into two upper and lower signal channels.

4.1.3 Detection and Segmentation. At this phase, we separate keystrokes and echoes from the audio stream. We use filters to separate the signals into different frequency bands. We use a high-pass filter to obtain short-time pulses from the signal and a band-pass filter to get keystrokes while filtering out low-frequency noise.

4.1.4 TDoA Classification. When we get a piece of data containing keystrokes, we get the TDoA of the key by comparing the signal of two channels of microphones. The keystrokes are classified by the TDoA.

4.1.5 Position Estimation. The filtered signals contain a direct pulse from the speaker to the microphone and several echoes reflected from other objects. We identify the desired one from the keystroke by comparing changes in different windows.

4.1.6 Anchor Point Estimation. To find the relative location between keyboard and phone, we introduced the "Anchor point." Therefore, we need to extract the feature of the keystrokes from a series of keystrokes as an "anchor point."

4.1.7 Single-keystrokes Estimation. The *TKCS* is established by the relative relationship between the "anchor point" and the key. Then two coordinate systems are combined to get the mark of the keystroke cluster.

4.1.8 Keystrokes Identified. After getting the mark of the keystroke cluster, we use the cluster centering method to identify the keystrokes.

4.2 Signal Design and Localization

The core part of our system is the short acoustic pulse-based localization. In this subsection, we will introduce how to use it to localize the keystrokes.

4.2.1 Acoustic Signal Generation. In our system, we design the pulse in a fixed frequency of inaudible band (ultrasonic). At a sampling rate of 48 kHz, one period of the signal consists of 480 sampling points. The signal consists of the short acoustic pulse signals of 10 samples in every 480 samples, which is illustrated in the top graph of Figure 10. The periodic frequency of the transmitted acoustic signal is $F = \frac{F_S}{S}$, where F represents the frequency of the system, F_S represents the sampling rate of recording, and S represents the number of samples of a periodic signal. Therefore, the frequency F of our system is 100 Hz, while the detection range of our system is $R \approx \frac{c \times S}{2F_S}$, $R = 343/200 = 1.715$ m. We need to pass the acoustic signal through a high-pass filter before sending it; otherwise, the signal will be perceived by the human ear.

After a careful selection, we finally chose 21.5 kHz as our signal carrier. This acoustic signal can be emitted by the COTS smartphone's speaker whose sampling rate is 48 kHz and beyond. We chose this signal becuase of its characteristics of being detected easily, as shown in Figure 10.

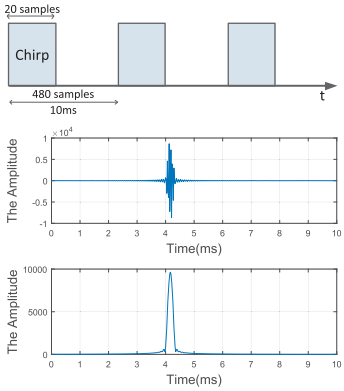

Fig. 10. The signal we transmit (top). Original signal (middle) and signal profile (bottom). The short acoustic pulse will be emitted from the speaker. The chirp and its echo will be record by the two microphones.

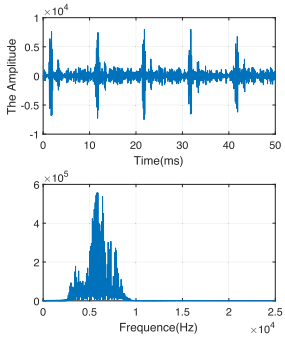

Fig. 11. The signal processed after the high-pass filter. (a) The output of high-pass filter and (b) its frequency response diagram.

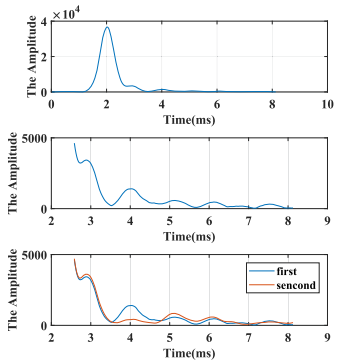

Fig. 12. Hilbert transform output. Signal envelope (top). Signal (middle) and comparison of two envelopes (bottom).

4.2.2 Noise Cancellation. After receiving the acoustic signal stream from two microphones, we adopt a high-pass filter to remove the background noises produced by speaking and tapping on the keyboard. We noticed that the upper limit of the frequency of the sound produced by tapping the mechanical keyboard is about 11 kHz. Thus, we need to design the pass frequency of the high-pass filter to be greater than 11 kHz. We set it to 15 kHz.

The signal and its echo after the filter are illustrated in Figure 11. The top graph of Figure 11 shows a mixed signal that contains the original signal and its echo.

Then we multiply the received signal by the orthogonal signal of the short pulse. According to the following trigonometric function, we simplify one of the paths,

$$cos(\alpha) \times sin(\alpha) = \frac{sin(\alpha + \alpha)}{2}. \tag{9}$$

Since we are using high-frequency signals, the high-frequency part $2 \times \alpha$ will become a 5 kHz (48 kHz−21.5 × 2) [5] signal at a sampling rate of 48 kHz. Then we pass the signal through a band-pass filter centered at a frequency of 5 kHz. This operation will remove the noise interference.

The bottom of the Figure 11 shows the frequency response of the mixed signal after the received signal is multiplied by the quadrature signal.

4.2.3 Envelope Detection. After the filtering step, although we got the short acoustic pulse and its echo, we still had difficulty in discerning the specific arrival time of a single echo. So we first split the signal into multiple windows. Each window contains only one short pulse period. Then we process the acoustic signal to obtain its envelope. To simplify our step of detecting the peaks of the echo, we use Hilbert transform to do the enveloping. Hilbert transform is commonly used in signal processing and can be seen as passing the original signal through a filter. It can be expressed as $\hat{x} = H[x(t)] = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau$, where $\hat{x} = H[x(t)]$ is the output of Hilbert transform. We can see that the expression of the Hilbert transform is actually the result of convolving the original signal. This signal used to convolve is $h(t) = \frac{1}{\pi t}$. The result of the acoustic signal we generate is shown in Figure 10, where the Hilbert transform shows the outline of the signal.

4.2.4 Keystroke Detection. To detect the keystroke, first, we need to detect the peaks that correspond to the echo of an object reflection except for the maximum direct arrival peak, as there is just one peak in our signal after the envelope. In Figure 12, the top graph shows a window data after the envelope, and the middle graph is a partial enlargement of it. Obviously, we can see that there is a nearest peak at 3 ms, which represents a object at a distance of about 17 cm($\approx \frac{Sv \times \Delta t}{2}$), and then we add the distance from speaker to the microphone to calculate the complete distance. However, the nearest can be anything, such as the edge of the keyboard or the hand of the victim, and so on. To find where the keystrokes occur, we choose to compare the changes in the reflected waveform before and after the keystroke to find the position, which represents where the keystroke occurred. As shown in the third graph of Figure 12, the peak of the first wave and the weak reflection of the second wave occur at 4 ms, which corresponds to the keystroke.

4.2.5 Distance Calculation and Transformation of Coordinates. Once we obtain the moment when keystrokes occurred, we calculate the coordinates of the keystrokes in *TKCS* according to Equations (5) and (6).

4.3 Keyboard Localization and Layout Construction

Notice that our system is not dependent on the relative position of the device and the keyboard, which can be self-located in following steps.

4.3.1 Approximate Position Estimate. In the first phase of self-positioning, we need to estimate the approximate position of the phone. Since we are using active signals for position measurement, it is unavoidably limited by hand occlusion. So we roughly divide the position into the following three types: the mobile phone is on the left, on the top, and on the right. Basically, when placing it on the left side, our system can only measure most of the clicks in the left hand. Then, for the right hand, we cannot use the active pulse to measure the position of the keystroke. We mainly infer right-hand keystrokes by the TDoA and MFCC voiceprint. When the phone is on the right side of the keyboard, the situation is similar. The difference is that in these two cases, we choose different "anchor points." Specifically, on the left side, we choose the "left shift key" and the "ESC." When we click on "ESC," we can detect a very obvious movement. On the other side, we choose "Space" and "Enter" as anchor points. Finally, when the phone is located above the keyboard, it is an ideal environment for our system. In this case, we use the "Space" and the "Enter" as anchor points. We use the system to measure the keystrokes of both hands.

4.3.2 Anchor Point Detection. After estimating the approximate location of the phone. In recorded microphone samples containing anchor points such as "Enter" and "Space" (or the "left shift key" and the "ESC"), through comparing the TDoA and MFCCs profile of these keystrokes, we can find the location of these points so as to determine the location of the keyboard and smartphone. After converting the position of all keystrokes into the *VPCS*, we can rebuild the *TKCS* with the anchor points we measured.

4.3.3 Relative Position Estimation. After finding anchor points, we use them as the reference to establish *TKCS*. Then we convert the coordinates from the coordinate system of *VPCS* to *TKCS* by the method described above.

4.3.4 Interference. The interference mainly comes from the following aspects: (1) the periodic frequency, (2) the power range, and (3) the width of the peak in the envelope. The periodic frequency is determined by the interval at which we send the acoustic signal. If the interval is too large, then the recognition accuracy and real-time performance of our system decrease. By contrast, if the interval is too small, then the echo of the acoustic signal of the previous segment will

easily interfere with the recognition of the latter acoustic signal. The location of the phone will also affect the accuracy, due to the range of transmitting. We need to place the keyboard within the range of microphone coverage and place it within the detection range R.

For these reasons, we cannot merely rely on the ultrasonic localization to identify the keystrokes. According to our experiments, the error can be within the range of three to seven samples (2.1–5 cm).

4.4 MFCC-Position Mapping and Clustering Algorithm

Beyond the echo-based localization, we further improve the performance by proposing an error correction algorithm called MPMC, whose intuition is based on the observation (as shown in Figure 13) that the position variation is distributed around the real position. Thus, we can cluster and map the key with same MFCCs to the position cluster head. As is known to all, the most common method for calculating voiceprints today is MFCCs [10, 11, 17, 19]. We basically combine the MFCCs with the position estimated from the active acoustic sensing and derive the accurate position by mapping their clustering heads. This algorithm consists of flowing steps.

4.4.1 Detect the Keystrokes. Once we receive the acoustic stream from microphone, we put it into a band-pass filter to get the signal including the keystrokes. The first step is to detect the keystrokes and find the begin of the keystrokes. It is much easier for us to slice and gather the keystrokes using the acoustic emanation. As is shown in Figure 14, we detect the energy level of the signal based on the accumulated signal as

$$A(t) = \sum_{i=t}^{t+W} x^2(i), \tag{10}$$

where A is the signal energy. W represents the length of the windows of signal, and $x(t)$ is the amplitude of the acoustic signal. Since one keystroke can be divided into three peaks, which are touch peak, hit peak, and release peak. The whole process is over 200 ms. Among them, the energy of the hit peak is the largest, lasting about 10 ms. Then we use this as the basis for detecting the keystroke. Here we select $W = 5$ ms as the length of the windows.

After getting a slice acoustic signal with keystrokes, we get the position of the moving object by comparing the front and back windows of the keystrokes echo.

4.4.2 Voiceprint Calculation. When we gather a small part of location of keystrokes together, we need to translate them into corresponding content. We calculate the voiceprint by the MFCCs [30]. The MFCCs stand for the sound feature of the keystrokes generated by the same key, which can be used to separate different keys from the same set of theoretical keys.

When we get location of keystrokes, we transform the coordinate of $VPCS$ to $TKCS$ through the aforementioned method. We put a certain range of keystroke points near a theoretical key X into a theoretical group of X. Then, according to the match-filter, the keys in a theoretical key group are distinguished according to the MFCCs process into different key sets. We use a method similar to the weighted average for classification. Make these points closer to the theoretical key, the greater the weight. In the same theoretical group, we pick out the set of keys that will maximize the weight of this theoretical group and then maximize all the weights of key groups by the following formula:

$$\arg\max \sum_{k=1}^{K} \sum_{i=1}^{N_k} W_k^2(i), \tag{11}$$

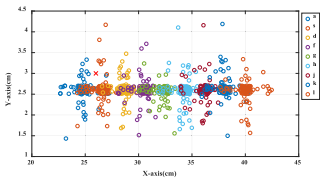

Fig. 13. The point of the keystrokes located.

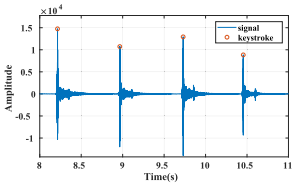

Fig. 14. The result of keystrokes detection.

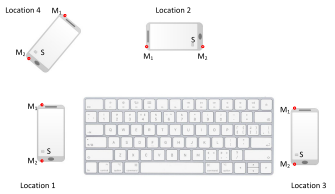

Fig. 15. Placement of phone and keyboard.

ALGORITHM 2: MPMC Algorithm

Input: The MFCCs m_1 and coordinate A of the new keystroke, Average MFCCs M of clusters and cluster center coordinates a, Fixed value K, k.
Output: $Top-k$ probability of the inference

for $i = 1; i <= k; i++$ **do**
  $R_1[i] \leftarrow \sqrt{A^2 - a[i]^2}$
  $P_1[i] \leftarrow K/R_1[i]$
end for
for $i = 1; i <= k; i++$ **do**
  $P_m[i] \leftarrow$ **Similarity**$(M[i], m_1)$
end for
for $i = 1; i <= k; i++$ **do**
  $P[i] \leftarrow 1 - (1 - P_1[i]) \times (1 - P_m[i])$
end for
$Top-k = sort(P)$
return $Top-k$

where N_k represents the number of keystrokes in kth key set and W_k represents the weight of the ith keystrokes in the Kth key set.

4.4.3 Accumulative Error Correction. Specifically, when the input keystrokes are accumulated to a certain amount, we will get some clusters based on MFCCs, and calculate the center of each cluster. Note that we can traceback the inference as snooping proceed. Given a new input, we first calculate the distance between that input and the center of clusters to get a distance probability sequence. Then we calculate the similarity of its MFCC to the average MFCCs of the clusters to obtain a MFCC probability sequence. Then we get the probability that the input belongs to the clusters by combining two probability sequences. The process is shown in Algorithm 2. The notation description of Algorithm 2 is shown in Table 2.

We can see in Figure 13 that there is a point X at coordinates $(26, 3)$. The center of clusters A is at $(25, 2.6)$ and the center of clusters S is at $(27.2, 2.7)$. The distance between X and center of clusters A or center of clusters S is almost the same. The distance from X to A is about 1.08 cm, and the distance from X to S is 1.14 cm. We get a distance probability sequence P_1 for two clusters. Then we calculate the similarity between the MFCC of X and the average MFCCs of the two clusters and get a MFCC probability sequence P_m for two clusters. We get a probability sequence P that the keystroke belongs to the two clusters by combining P_1 and P_m. This algorithm can obtain a candidate sequence, $Top-k$, or get one output, $Top-1$, by adjusting the input parameters.

Finally, we label all keystrokes through the cluster based on the coordinates of the different keystrokes according to the the output of Algorithm 2.

Table 2. Notation Related to Algorithm 2

Notation	Description
A	The coordinate of the new keystroke
m_1	The MFCC of the new keystroke
M	The average value of MFCC clusters
a	The coordinates of clusters center
K	Empirically set fixed value
k	The number of clusters center
R_1	Distance sequence between new keystrokes and clusters
P_1	Probability sequence of new keystrokes belonging to clusters in coordinate
P_m	Probability sequence of new keystrokes belonging to clusters in MFCC
P	Mixed probability sequence of new keystrokes belonging to clusters

4.5 Keystroke Inference in Case of Hand Occlusion

When the phone is placed to the left (or right) side of the keyboard, one of the hands will obscure the reflective surface of the other hand. The proposed method is only capable of measuring the keystrokes of one hand that can reflect. With the keystroke of the other hand, we can only recover the keystrokes by inferring.

The basic idea is that we first find different anchor nodes for different hands. Specifically, for the right hand we use "Space" and "enter" while for the left hand we use "ESC" and "Space." The position of the anchor points are measured following the aforementioned method.

For those keystroke from the occluded hand, they are inferred by the following methods:

- We separate the keystrokes into different groups according to TDoA.
- In different TDoA groups, we further grouped the keystrokes according to the MFCC voiceprint.
- Based on $VPCS$ and $TKCS$, we plot the TDoA curve and determine the set of candidate keys for each group.
- The set of candidate keys is given as the result of $TOP - k$ and are selected for further inference.

5 EVALUATION

In this section, we present experiment setups and results.

5.1 Experiment Setups

5.1.1 Hardware Settings. Our hardware mainly includes smartphones and keyboards The smart device needs two high-quality microphones and a 48-kHz sampling rate speaker to emit ultrasonic waves. In our experiments, we chose $360N6pro$ as the smart device. The operating system is Android 7.1.1, and the processor is Qualcomm Snapdragon 660. The phone has two microphones on a straight line at the top and bottom edges. The distance between the two microphones is 15.5 cm. We used its built-in speaker to play pre-defined acoustic signals. Two microphones record the keystrokes, short acoustic pulse, and their echoes. The playback sampling rate we used is 48 kHz. The sampling rate of the recording is also 48 kHz. Due to the limitations of the current hardware of mobile phones, we only chose 48 kHz. Although the system supports a sampling rate of 192 kHz, the upper limit of the response frequency of the microphone and speaker is 22 kHz. Regarding the keyboard, we chose a regular Dell keyboard and ikbc's C104 blue switch mechanical keyboard. The keyboard size is $443 * 138$ mm, with a numeric keypad.

5.1.2 Environment Setting. We selected an open experimental environment within an area of 5×6 m^2, full of multipath effects and background noise from air condition as well as people's speech.

5.2 Evaluation Criteria

Our main consideration is the accuracy of recognition. Consider whether the evaluation criteria can help classify keystrokes of a certain key into the same category and correctly identify the key:

- **Precision:** Precision represents that the probability of being correctly identified the key x. It is defined as

$$P_x = \frac{N_x^T}{N_x^T + N_x^F},$$

  where the N_k^T means the number of being correctly identified as key X. N_k^F is the number of being wrong to be judged to be key x.
- **Recall:** Recall of the key x is defined as

$$R_x = \frac{N_x^T}{N_x},$$

  where N_x^T is the same as above and N_x is the number of all the keystrokes of x.
- **Top-k:** This criterion is based on the fact that k candidates are given in our system, and the correct probability of the candidate. It is defined as

$$T = \frac{\sum_{k=1}^{K} P_k^{T,w}}{\sum_{k=1}^{K} N_k},$$

  where P_k^T represents the number of the keystrokes that there is correct one in all the candidate.

5.3 Experiment Design and Results

We designed several experiments to verify the feasibility and statistical performance of our system.

In the first experiment, we successfully validated the feasibility of hand perception. In the second and fourth experiments, we respectively demonstrated the system's ability to locate a single keystroke and distinguish between different keystroke positions, which also implies finger position estimation. The accuracy of anchor node detection was evaluated in the third experiment. Last, in the last three experiments, we assessed the keystroke accuracy after coordinate system conversion under varying scenarios.

5.3.1 Performance of Hand Tracking. First, we verified the feasibility of ultrasound-based gesture localization using smartphone. We conduct an experiment to track the hand movement when a hand was moving around the microphones. In the experiment, we first turned on the system with no obstacles in front of the phone. After a while, a hand was placed 20 cm away, then gradually moved away from the phone, and finally moved toward the phone.

As shown in the result of first experiment, when tracking the trajectory of the hand, the experimental results correspond with our design from 20 to 40 cm. In Figure 16, we can easily see that there was a significant echo at 20 cm in the range of the microphone at 1 s, and it began to diminish at 2 s. Finally, the farthest reaches 4 s, while the echo gradually weakens. Then, as the palm approached the phone, the echo reappeared at the 20cm position at 6 s.

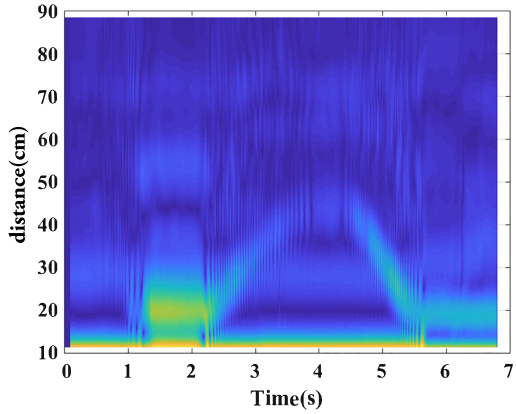

Fig. 16. The trace of hand in the first experiment.

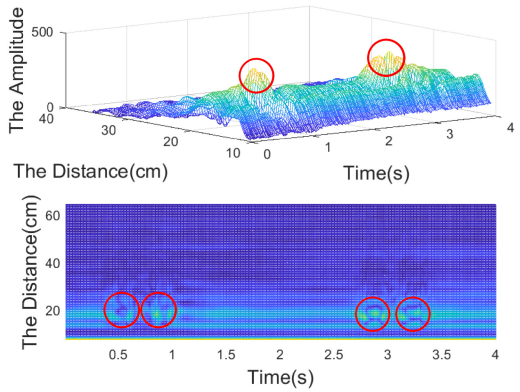

Fig. 17. The 3D echo waveform of click and top view.

5.3.2 Localization of Finger Clicks. We need to verify whether we can measure where the finger clicks on the keyboard, because we need to determine the position of the key by recognizing the position where the action of the finger click occurs. In the experiment, after playing the signal, we clicked four times near the microphone and speaker about 20 cm. Figure 17 shows the results of the click experiment. In the top view of Figure 17, there are bright spots at about 20 cm at 0.5 s, 1 s, 2.5 s, and 3 s. We observed a significant peak change of the waveform in the waveform diagram. This is due to the change in the size of the echo reflecting surface at this distance caused by the click action.

From this, we can see that the keystroke snooping method based on short acoustic pulse positioning is feasible. We are able to detect finger clicks and hand movements, but the echo gradually weakens as the distance increases due to the attenuation.

5.3.3 Anchor Node Detect. The third experiment we set on the keyboard is as follows: According to the steps of our system, we need to pick out the anchor points first. We typed a long piece of text on the keyboard with one hand, including a "Space" and an "Enter" key. The reason for using one hand is mainly to avoid the situation where the hand is occluded, and the case where the hand is occluded will be verified below. According to the algorithm we proposed above, some of the "anchor points" will be identified.

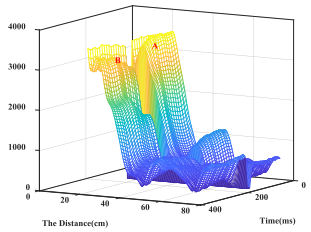

Fig. 18. Key *A*'s 3D image of echo.

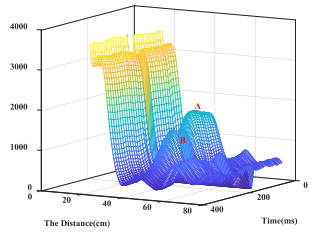

Fig. 19. Key *L*'s 3D image of echo.

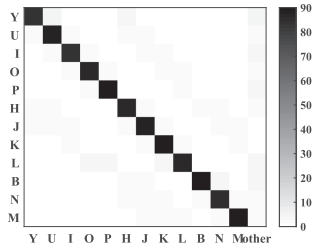

Fig. 20. The confusion matrix of Keystroke Inference experiment.

Table 3. The Detection Accuracy of "Anchor Point"

Accuracy (%)	"Space"		"Enter"		"Shift"	
	top-5	top-3	top-5	top-3	top-5	top-3
volunteer 1	100	100	100	96	98	93
volunteer 2	100	100	100	98	99	95
volunteer 3	100	100	100	100	100	98
volunteer 4	100	94	98	92	98	94
volunteer 5	100	100	100	100	99	97

We verified detection accuracy of anchor points in a series of keystrokes containing "Space" and "Enter."

The experiments involved five volunteers using our equipment on two keyboards, a regular Dell keyboard and ikbc's C104 blue switch mechanical keyboard. Each volunteer typed a piece of text of 100 characters including anchor points each time, repeating 10 times. The input contains "Space" keystrokes, "Enter" keystrokes, and several keystrokes physically close to the select anchor points. We give results of "Anchor point" for top-3 and top-5 by algorithm. The results are shown in Table 3. We aggregated all the experimental results of the same volunteer to get the accuracy rate. It is seen that the results of top-3 are almost completely correct. It is relatively easy to select 3 of 10 keys from 100 keystrokes. We first picked those objects that are at the edges of the coordinate system and then extracted them by algorithm. However, the text we designed includes consecutive "Space" and "Enter." When the volunteers typed too fast, the keystrokes may overlap, the difficulty of recognition increases, and the accuracy decreases slightly.

5.3.4 Single Keystroke Inference. Then we tested with two easy-to-distinguish keys, namely *A* and *L*—one at the left edge of the keyboard and the other at the right edge—to further verify the feasibility of our system. We placed the phone and keyboard on the table, with the bottom microphone of the phone 20 cm away from the left edge of the keyboard. The 'A' key is 24 cm away from the bottom microphone, and the 'L' key is approximately 40 cm away from the bottom microphone. The size of those keys is a square with a side length of 12.5 mm, and the spacing between two keys in a row is about 6.5 mm. We typed each key three times in this experiment.

According to result of the single keystroke experiment (see Figure 18) that contains the windows of short acoustic pulse echo before and after the keystrokes, we can easily see that in the window of key *A* there is a significant change of the echo near 24 cm. Similarly, there is an obvious amplitude change of echo at the location around 40 cm, as shown in Figure 19. We also aligned the signals of each slice and then plotted such a 3D echo map. As we can see in Figure 18 and Figure 19, there is a gap in the graph because of the keystroke. Point *A* of the graph represents the value before the

Table 4. The Accuracy of Our Experiment 4 without MPMC Correction

Accuracy (%)	Y	U	I	O	P	H	J	K	L	B	N	M
Experiment 1	69	59	60	52	60	66	50	44	46	64	56	54
Experiment 2	60	57	63	62	64	53	59	59	64	67	47	50
Experiment 3	54	55	51	57	56	51	51	49	52	43	53	43
Experiment 4	37	38	31	32	31	39	26	32	30	39	27	24

keystroke occurs. Similarly, *B* represents the value after the keystroke. There is an obvious change between point *A* and point *B*, because the finger is clicked here at the location of about 24 cm. In Figure 19, there is also an obvious change between point *A* and point *B* at the location of about 40 cm. Because of the obvious distance difference, we can easily distinguish between the two keys without utilizing voiceprint verification.

5.3.5 Keystroke-set Inference. Next, we set up an experiment to further test the feasibility of our system in a more realistic setting. In the experiment, we placed the phone about 20 cm from the top left of the keyboard. We only use the right hand, click on the part of the right-hand key (*YUIOHJKLNM*), and "anchor." Each key is repeated multiple times.

In this experiment, we present the results of the confusion matrix in Figure 20. In this matrix, we can see that each key is judged with only a small number of misjudgments, and the misjudgment only appears in the nearby keys. So our top-3 or top-5 will be more accurate.

5.3.6 Relative Keyboard Localization. To verify the feasibility of our self-localization method, we set up another set of experiments in which we changed the relative position between the smartphone and the keyboard, as shown in Figure 15. Position 4 is set in about 70 cm from the top left of the keyboard.

In this experiment, we set four positions of phone. From Table 4, in the first three experiments at 20 cm, we observed that position 1 is more accurate for the button on the left side. In the same way, position 3 has a higher accuracy in positioning the button on the right side. For position 2, the action of tapping the previous row(*YUIOP*) of buttons produces a change in position relative to the phone. Therefore, the positioning accuracy of the buttons in the previous row is high. For position 4, when the tap is pressed, the finger is only pressed down, and there is no position change, so the positioning effect is poor.

In position 4, because the phone is too far away from keyboard, there is no interference between the phone and the keyboard. However, due to the attenuation of the acoustic signal, the echo becomes weak, and the peak position cannot be accurately identified. So we got a poor result, although we can increase the identifiable range of our phone by increasing the power of the phone's speakers. However, the result is that in order for the signal not to interfere with the next cycle, we need to lengthen the cycle. Longer cycles can affect our recognition efficiency. Phone–keyboard distance and whether keystrokes require finger movement influence the accuracy of keystrokes snooping.

The results of our method with MPMC correction is shown in Figure 21. The accuracy results of position 2 and position 3 rise to about 80%, the accuracy of position 1 rises to about 70%, and that for position 4 is about 50%. Moreover, the distance between the keyboard keys and the phone varies. The system can effectively recognize keystrokes when the phone is 30–60 cm from the keyboard.

To understand the effect of the number of keystroke on the snooping performance, we conduct experiments with different numbers of keystrokes under positions 1 and 4. The results are shown in Figure 22. The detection accuracy of position 1 rises with accumulation of keystrokes and

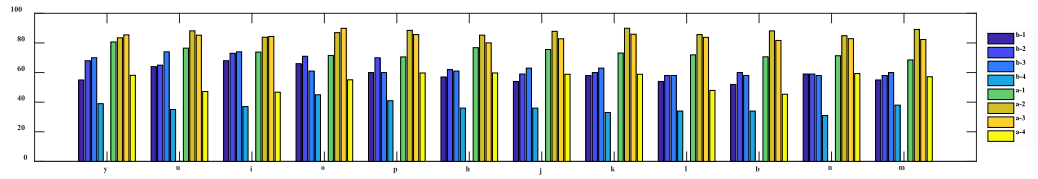

Fig. 21. Comparison of our system with/without MFCCs. $b-1$ to 4 represents the result without MFCCs, while $a-1$ to 4 represents the result with MFCCs.

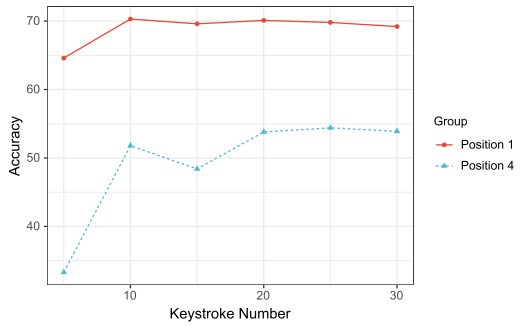

Fig. 22. Accuracy with different number of keystrokes for positions 1 and 4.

eventually reaches 70% when the number of keystroke reaches 10. The accuracy of position 4 reaches 53% when the number of keystroke reaches 20. The system has a low recognition accuracy under two positions when keystrokes are less than 10, since the center of MFCC clusters are far from the ground truth. With the accumulation of keystrokes, the center of MFCC clusters gradually approaches the ground truth, and the MPMC algorithm helps to correctly infer keystrokes. The constant change of center of MFCC clusters brings fluctuating of accuracy.

5.3.7 Two-handed Inference Experiment. In the experiment, we used both hands to input the following text multiple times: "this is an experiment," of course, including "anchor points," "Space," "Enter," "L shift," and "ESC." The phone is placed on the top, left, and right. The occluded keystrokes are inferred by TDoA and voiceprints.

Table 5 shows the results of the two-handed experiment. We see that the mostly occluded keystrokes have lower inference accuracy. For example, (*HMN*) has a lower inference accuracy when the phone is placed on the left side of the keyboard. From Tables 4 and 5, two-handed keystroke inference without MPMC is less accurate than one-handed for overlapping keys (*HIMNP*). After MPMC calibration, the inference accuracy of the two-handed experiment improved significantly.

We also evaluate the accuracy of the overall system, as shown in Table 6.

6 DISCUSSION

In this section, we discuss several limitations of UltraSnoop.

Accuracy of Location. Essentially, the accuracy of typing inference is heavily reliant on the precision of echolocation, which in turn is constrained by the level of accuracy in signal transmission and collection. Furthermore, we employed a "voiceprint" clustering technique due to insufficient positional accuracy. By increasing the sampling rate for instance to 192 kHz, the distance

Table 5. The Top-3 Accuracy of Two-hand Keystroke Inference

Accuracy (%)	a	e	h	i	m	n	p	r	s	t	x
Top side	58	69	49	69	42	45	64	65	54	57	47
Left side	52	64	44	49	47	48	63	56	51	50	53
Right side	48	60	43	55	51	60	57	65	47	52	50
Top side with MPMC	64	91	64	77	49	53	73	90	65	74	70
Left side with MPMC	70	80	57	70	68	54	70	68	72	68	78
Right side with MPMC	66	72	50	78	68	72	66	77	54	71	60

Table 6. The Accuracy of Overall System

Anchor Point Detect	Keystroke-set	After MPMC	word		
			top-1	top-3	top-5
95%	55%	74.7%	74.7%	92.8%	97.3%

represented by a sample point would shrink to approximately 0.18 cm. The use of a wider frequency also ensures minimal distortion in high-frequency signal waveforms.

Occlusion. One issue to address is that of occlusion. In real-world usage scenarios, the smartphone and keyboard may be separated by various complex obstructions that could significantly impede UltraSnoop's performance. To achieve localization, it is crucial to detect the direct wave, as calculating the distance based on multipath echo measurements would be excessively cumbersome and unfeasible. Hence, in future research, we plan to explore alternative methods for accurately detecting the position of an object that is obstructed from view.

Limited Distance. Ultrasound attenuation and transmission power restrictions of smartphone signals impose limitations on the operational range of UltraSnoop, which is restricted to within 60 cm of the keyboard. This constraint reduces its efficacy in actual attack scenarios. Our Relative Keyboard Localization experiment demonstrated suboptimal performance at distances beyond approximately 70 cm, as ultrasound attenuation in the air hinders the effective recognition of keystroke peak positions. Previous research has leveraged microphone arrays for far-field sound recording, offering a promising avenue for enhancing the applicability of our work in the future.

7 CONCLUSION

In this article, we propose a single-smartphone-based training-free keyboard snooping attack system named UltraSnoop. The system has no strict smartphone placement requirements, which will greatly ease large-scale keystroke snooping attacks. Basically, UltraSnoop uses a sonarlike method that emits inaudible ultrasonic short-acoustic-pulse waveforms to perform both the detection and the localization of the keystrokes. With only ultrasonic signals, the proposed keystrokes inference system can achieve an accuracy of over 70%. We further propose a MFCC-position mapping and cluster algorithm to accumulatively increase the snooping accuracy. Extensive experiments show that our method could achieve an average accuracy of 85% for top-3 candidates. Furthermore, our method is compatible with previous acoustic emanation-based approaches and could improve the accuracy when working collaboratively.

REFERENCES

[1] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'04)*. IEEE, 3.

[2] Yigael Berger, Avishai Wool, and Arie Yeredor. 2006. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'06)*. ACM, 245–254.

[3] Huijie Chen, Fan Li, and Yu Wang. 2017. EchoTrack: Acoustic device-free hand tracking on smart phones. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'17)*. IEEE, 1–9.
[4] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. 2018. EyeTell: Video-assisted touchscreen keystroke inference from eye movements. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'18)*. IEEE, 144–160.
[5] Yuchi Chen, Gong Wei, Jiangchuan Liu, and Cui Yong. 2017. Fine-grained ultrasound range finding for mobile devices: Sensing way beyond the 24 kHz limit of built-in microphones. In *Computer Communications Workshops*.
[6] Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. 2018. No training hurdles: Fast training-agnostic attacks to infer your typing. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'18)*. ACM, 1747–1760.
[7] Yuyi Fang, Zhiwei Zhao, Zi Wang, Geyong Min, Yue Cao, Haojun Huang, and Hao Yin. 2018. Eavesdrop with PoKeMon: Position free keystroke monitoring using acoustic data. *Fut. Gener. Comput. Syst.* 87 (2018), 704–711. https://www.sciencedirect.com/science/article/abs/pii/S0167739X17300493
[8] Tyler Giallanza, Travis Siems, Elena Smith, Erik Gabrielsen, Ian Johnson, Mitchell A. Thornton, and Eric C. Larson. 2019. Keyboard snooping from mobile phone arrays with mixed convolutional and recurrent neural networks. *Proc. ACM Interact. Mob. Wear. Ubiq. Technol.* 3, 2 (2019), 1–22.
[9] Wenchao Huang, Xiang-Yang Li, Yan Xiong, Panlong Yang, Yiqing Hu, Xufei Mao, Fuyou Miao, Baohua Zhao, and Jumin Zhao. 2016. WalkieLokie: Sensing relative positions of surrounding presenters by acoustic signals. In *Proceedings of the the ACM Conference on Ubiquitous Computing (Ubicomp'16)*. ACM, 439–450.
[10] Jesper Højvang Jensen, Daniel P. W. Ellis, Mads Græsbøll Christensen, and Søren Holdt Jensen. 2007. Evaluation of distance measures between gaussian mixture models of MFCCs. In *Proceedings of the International Society for Music Information Retrieval (ISMIR'07)*. 107–108.
[11] Tomi Kinnunen, Rahim Saeidi, Johan Sandberg, and Maria Hansson-Sandsten. 2010. What else is new than the Hamming window? Robust MFCCs for speaker recognition via multitapering. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*.
[12] Hyewon Lee, Tae Hyun Kim, Jun Won Choi, and Sunghyun Choi. 2015. Chirp signal-based aerial acoustic communication for smart devices. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'15)*. IEEE, 2407–2415.
[13] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. 2016. When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'16)*. ACM, 1068–1079.
[14] Kang Ling, Haipeng Dai, Yuntang Liu, and Alex X. Liu. 2018. UltraGesture: Fine-grained gesture sensing and recognition. In *Proceedings of the IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'18)*. IEEE.
[15] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom'15)*. ACM, 142–154.
[16] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: High-precision acoustic motion tracking. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom'16)*. ACM, 69–81.
[17] K. Sri Rama Murty and Bayya Yegnanarayana. 2006. Combining evidence from residual phase and MFCC features for speaker recognition. *IEEE Sign. Process. Lett.* 13, 1 (2006), 52–55.
[18] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'16)*. ACM, 1515–1525.
[19] Yasunari Obuchi. 2005. Mixture weight optimization for dual-microphone MFCC combination. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 325–330.
[20] Zhenquan Qin, Jiaxin Du, Guangjie Han, Gaopeng Yong, Linlin Guo, and Lei Wang. 2019. LOL: Localization-free online keystroke tracking using acoustic signals. *Soft Comput.* 23, 21 (2019), 11063–11075.
[21] Ke Sun, Ting Zhao, Wei Wang, and Lei Xie. 2018. VSkin: Sensing touch gestures on surfaces of mobile devices using acoustic signals. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom'18)*. ACM, 591–605.
[22] Jiayao Tan, Cam-Tu Nguyen, and Xiaoliang Wang. 2017. SilentTalk: Lip reading through ultrasonic sensing on mobile phones. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'17)*. 1–9.
[23] Jian Wang, Rukhsana Ruby, Lu Wang, and Kaishun Wu. 2016. Accurate combined keystrokes detection using acoustic signals. In *Proceedings of the IEEE Mobile Ad-Hoc and Sensor Networks (MSN'16)*. IEEE, 9–14.
[24] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous keyboard for small mobile devices: Harnessing multipath fading for fine-grained keystroke localization. In *Proceedings of the ACM ACM International Conference on Mobile Systems, Applications, and Services (Mobisys'14)*. 14–27.

[25] Wei Wang, Alex X. Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (Mobicom'16)*. ACM, 82–94.
[26] Jiadi Yu, Li Lu, Yingying Chen, Yanmin Zhu, and Linghe Kong. 2019. An indirect eavesdropping attack of keystrokes on touch screen through acoustic sensing. *IEEE Trans. Mobile Comput.* 20, 2 (2019), 337–351.
[27] Cheng Zhang, Qiuyue Xue, Anandghan Waghmare, Sumeet Jain, Yiming Pu, Sinan Hersek, Kent Lyons, Kenneth A. Cunefare, Omer T. Inan, and Gregory D. Abowd. 2017. Soundtrak: Continuous 3d tracking of a finger using active acoustics. *Proceedings of the ACM Conference on Ubiquitous Computing (Ubicomp'17)*, 30.
[28] Qianqian Zhou, Yanni Yang, Feng Hong, Yuan Feng, and Zhongwen Guo. 2016. User identification and authentication using keystroke dynamics with acoustic signal. In *Proceedings of the 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'16)*. IEEE, 445–449.
[29] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'14)*. ACM, 453–464.
[30] Li Zhuang, Feng Zhou, and J. Doug Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.* 13, 1 (2009), 3.

Received 28 October 2022; revised 29 March 2023; accepted 24 July 2023