PIANO: Proximity-based User Authentication on Voice-Powered Internet-of-Things Devices

Neil Zhenqiang Gong*, Altay Ozen*, Yu Wu†, Xiaoyu Cao*, Richard Shin‡, Dawn Song‡, Hongxia Jin§, Xuan Bao¶
* Iowa State University, †UC Davis, ‡UC Berkeley, § Samsung Research America, ¶ Google Inc.

***Abstract*—Voice is envisioned to be a popular way for humans to interact with Internet-of-Things (IoT) devices. We propose a proximity-based user authentication method (called PIANO) for access control on such *voice-powered* IoT devices. PIANO leverages the built-in speaker, microphone, and Bluetooth that voice-powered IoT devices often already have. Specifically, we assume that a user carries a personal voice-powered device (e.g., smartphone, smartwatch, or smartglass), which serves as the user's identity. When another voice-powered IoT device of the user requires authentication, PIANO estimates the distance between the two devices by playing and detecting certain acoustic signals; PIANO grants access if the estimated distance is no larger than a user-selected threshold. We implemented a proof-of-concept prototype of PIANO. Through theoretical and empirical evaluations, we find that PIANO is secure, reliable, personalizable, and efficient.**

I. Introduction

IoT devices are ever on the rise and getting ubiquitous. According to Gartner, the IoT devices installed base will grow to 26 billion and generate incremental revenue exceeding $300 billion in 2020 [1]. An attacker can compromise a user's security and privacy via *unauthorized physical access* to the user's IoT devices. Specifically, many IoT devices store various private information of the devices' owners. For instance, a user's smartphone or smartwatch might store the user's sensitive emails (e.g., emails including social security number and credit card numbers), sensitive messages, and call history. Likewise, a user's Bee (a healthcare monitoring IoT device [2]) stores the user's insulin injection data and glucose levels. Imagine a user leaves his/her such IoT devices unattended (e.g., when the user goes to restroom or to have lunch) or loses them, then an attacker could have unauthorized physical access to these devices to access the user's private information on them, compromising the user's data security. Moreover, having physical access to one IoT device could enable an attacker to control other connected IoT devices. For instance, smartphone is used to remotely control garage door [3]; having access to a user's smartphone enables an attacker to remotely open the user's garage door via sending control commands to it, which could subsequently lead to house robbery.

Preventing unauthorized physical access often relies on *user authentication*, i.e., we authenticate the identity of the user before allowing the access. Existing user authentication methods such as *password* and *biometrics* are insufficient for IoT devices. Specifically, password and biometrics including fingerprint, face, and touch behaviors [4, 5] are inapplicable to IoT devices that do not have a keyboard, touchscreen, fingerprint reader, or camera. Moreover, password is tedious, e.g., the legitimate user needs to input the password every time the user uses the device. Voice-based speaker recognition (also a biometric authentication) is applicable to voice-powered IoT devices, but voice-based speaker recognition, like other biometrics, is vulnerable to forgery attacks [6]. For instance, an attacker can record the legitimate user's voice and replay it to bypass voice-based biometrics [7]. Detecting forgery attacks is a challenging unsolved problem, though we have seen some progress in the past decade [5, 6].

In this work, we focus on *voice-powered* IoT devices and propose a proximity-based user authentication method (PIANO) for voice-powered IoT devices. Traditional human-computer interfaces such as keyboard and touchscreen have limited application in resource-constrained IoT devices, and voice is envisioned to be a popular way for humans to interact with IoT devices [8, 9]. Therefore, we focus on voice-powered IoT devices. Specifically, humans control a voice-powered IoT device via *voice commands*, which are interpreted by the device via speech recognition techniques, and the device responds to humans via voice. Such IoT devices are often equipped with built-in microphone and speaker in order to support voice-based interactions; and they are often also equipped with Bluetooth, a pervasive wireless communication technology, to exchange data with other IoT devices.

Imagine a user Alice carries a smartwatch, which represents Alice's identity. When Alice uses her smartphone, the smartwatch and the smartphone are physically close; however, when an attacker tries to access Alice's smartphone while Alice is far away from her smartphone, the smartwatch and the smartphone are far away from each other. This scenario is not limited to smartwatch-smartphone pair. In fact, this is an emerging scenario in IoT: a user carries an IoT device which has already authenticated the user's identity. We call this device *vouching device*. The user has another IoT device, which requires authentication before being used. We call this device *authenticating device*. For instance, in our smartwatch-smartphone example, smartwatch is a vouching device and smartphone is an authenticating device. Likewise, the vouching device can be a user's smartphone and the authenticating device is the user's other voice-powered IoT device. We observe that when the legitimate user uses the authenticating device, the authenticating device and the vouching device are physically close. However, when an attacker tries to access the authenticating device while the legitimate user is away, the two devices are far away from each other. Therefore, in PIANO, access to the authenticating device is allowed if and only if the distance between the device and the vouching device is no larger than an *authentication threshold*, e.g., 1 meter.

A key component of PIANO is to estimate distance between the vouching device and the authenticating device accurately, efficiently, and securely. Existing distance estimation protocols [10–15] are insecure or inaccurate. Therefore, we propose a new distance estimation protocol called ACTION.

Our protocol leverages the speaker, microphone, and Bluetooth that voice-powered IoT devices often already have. Specifically, ACTION pairs the authenticating device and the vouching device via Bluetooth, which involves human interaction but only needs to be done once. When calculating distance between the authenticating device and the vouching device, the authenticating device first generates two randomized acoustic *reference signals* using a *signal-construction algorithm* and transmits them to the vouching device via a Bluetooth-based secure channel. Each device then uses a speaker to play a reference signal. Simultaneously, the two devices also record signals using microphones. Next, each device computes the times when the two reference signals arrived at it using a *signal-detection algorithm*. Finally, we estimate the distance by multiplying the *speed of sound* and the time which sound takes to travel from one device to the other.

We implemented a prototype of PIANO on two smartphones. Via theoretical and empirical evaluations, we find that PIANO has several promising features: *secure*, *reliable*, *personalizable*, *zero-interaction*, and *efficient*. Specifically, PIANO has a very low probability of falsely accepting an attacker (i.e., secure), even if the attacker leverages various *spoofing attacks* (we discuss them in Section III) to manipulate our distance estimation protocol. PIANO achieves a low probability of falsely rejecting the legitimate user (i.e., reliable). Users can tune the authentication threshold to meet their personalized needs (i.e., personalizable). For instance, they can set the authentication threshold to be 0.5 meter if they are in an environment where 1 meter is too long to be safe. PIANO requires no actions from users in the process of authentication (i.e., zero-interaction). In our prototype, authentication can be finished within 3 seconds (i.e., computationally efficient). Moreover, using PIANO 100 times only consumes 0.6% of the smartphone battery (i.e., energy efficient).

Our contributions can be summarized as follows:

- We propose PIANO, a novel proximity-based user authentication method for voice-powered IoT devices.
- We design a new acoustic signal based distance estimation protocol (ACTION) to estimate distance between two devices accurately, efficiently, and securely.
- We implemented a proof-of-concept prototype of PIANO. Via theoretical and empirical evaluations, we demonstrate that PIANO is secure, reliable, personalizable, and efficient.

II. Related Work

Distance estimation protocols: The key component of PIANO is a protocol that estimates the distance between the vouching device and authenticating device. Estimating distance between two devices has attracted much attention in various communities [10–15]. However, these protocols are *insecure* or *inaccurate*, making them insufficient for proximity-based authentication. We suspect the major reason why they are insecure is that they were not designed for security applications.

First, some protocols [10–13] leverage radio signals such as Wi-Fi, Bluetooth, and GSM. Radio signals can go through a wall, which has serious implications for security when using them in proximity-based authentication. For instance, if the authenticating device and the legitimate user who carries the vouching device are in two different rooms that are separated by a wall, or they are in two different floors next to each other in the same building, then the two devices could still have a small distance. Therefore, in such scenarios, an attacker can easily access the authenticating device.

Second, some protocols [14, 15] leverage acoustic signals. The key idea is that one device plays an acoustic signal; the other device detects when the acoustic signal arrives at it; and then the distance is the speed of sound multiples the time that the acoustic signal takes to travel from one device to the other. Acoustic signals are less likely to go through a wall. However, they are vulnerable to spoofing attacks. For instance, an attacker can simply replay the acoustic signals to "shorten" the distance for these protocols [14, 15]. Moreover, some protocols (e.g., Echo [14]) require accurate estimation of a device's processing delay, which is challenging for IoT devices given IoT devices often have limited computing power. In other words, they are inaccurate on IoT devices. Our proposed protocol also leverages acoustic signals, but they will be secure against various spoofing attacks and they do not rely on accurate estimation of processing delays.

Determining proximity using ambient signals: We note that some recent research [16, 17] proposed to perform proximity-based authentication via checking whether the authenticating device and the vouching device are physically close based on their ambient signals. The intuition is that two devices that are physically close share similar ambient radio signals, luminosity, and acoustic noise. These ambience-based approaches suffer from a few limitations. First, they aim to determine *relative* distances but not *absolute* distances between two devices. This limitation hurts the usability of the authentication system. For instance, some users might want to set the authentication threshold to be 0.5 meter while some other users might want to set it to be 1 meter. It is unknown how these ambience-based approaches can support such personalized user needs. On the contrary, our PIANO is *personalizable*, allowing users to tune the authentication threshold for their personalized needs. Second, these ambience-based approaches are insecure because attackers can modify the ambience around the two devices, e.g., attackers could play the same music around the two devices to modify their ambient acoustic signals.

III. Problem Definition and Threat Model

Proximity-based authentication: In our proximity-based authentication, a user who tries to access the authenticating device is authenticated if and only if the distance between the authenticating device and the vouching device is no larger than a user-selected authentication threshold. Intuitively, our proximity-based authentication propagates a user's identity from the vouching device to the authenticating device.

Threat model: We assume an attacker does not perform attacks to the authenticating device nor our authentication system when the legitimate user is physically close to the authenticating device. This is because an attacker faces the risks of exposing himself/herself in such scenarios. We consider the attacker's goal is to access the authenticating device when the legitimate user is away from the authenticating device (i.e., the

distance between the authenticating device and the vouching device is larger than the authentication threshold). Specifically, we consider the following two attacks.

- **Zero-effort attacks.** An attacker can directly try to use the authenticating device while the legitimate user is away. Due to distance estimation errors, the authenticating device would falsely authenticate the attacker with a certain probability. Since performing such attacks does not require much effort from the attacker, we call them *zero-effort attacks*. The success rates of zero-effort attacks are introduced by measurement errors from hardware and background noise.
- **Spoofing attacks.** In spoofing attacks, an attacker uses his/her own devices to play certain acoustic signals around the authenticating device and/or the vouching device, which aims to spoof the system to estimate the distance to be smaller than the authentication threshold. We will discuss specific spoofing attacks after we present our distance estimation protocol.

IV. Design of PIANO

Hardware requirements: PIANO requires the vouching device and authenticating device to be equipped with microphone, speaker, and Bluetooth. Voice-powered IoT devices often already have these hardware for functionality support.

Registration phase: In the registration phase, a user pairs the vouching device with the authenticating device using Bluetooth. This pairing process could involve human interactions, e.g., the user needs to manually confirm pairing between the two devices, but the pairing process only needs to be done once. After the two devices are paired, they can communicate securely via Bluetooth.

Authentication phase: When a user tries to use the authenticating device, the authenticating device uses PIANO to verify the user's identity. Specifically, PIANO first checks whether the vouching device is still paired with the authenticating device via Bluetooth. If not, which often means that distance between the two devices is larger than the authentication threshold (we denote the authentication threshold as τ), PIANO rejects the access; otherwise PIANO estimates the distance between the two devices using our distance estimation protocol called ACTION. If the estimated distance is no larger than the authentication threshold, the access is granted, otherwise it is rejected.

A. Overview of Our Distance Estimation Protocol

Our protocol has the following steps.

- **Step I:** The authenticating device constructs two snippets of acoustic signals, which we denote as S_A and S_V, respectively. We call these acoustic signals *reference signals*.
- **Step II:** The authenticating device securely transmits the two reference signals S_A and S_V to the vouching device via Bluetooth. The communication channel is secure so an attacker cannot eavesdrop the reference signals.
- **Step III:** The authenticating device and the vouching device record acoustic signals using a microphone. Moreover, the authenticating device uses a speaker to play the reference signal S_A and the vouching device plays S_V.
- **Step IV:** The authenticating device detects when the two reference signals were recorded, and we denote the timestamps as t_{AA} and t_{AV}, respectively. The vouching device also detects when the two reference signals were recorded, and we denote the timestamps as t_{VA} and t_{VV}, respectively.
- **Step V:** The vouching device securely transmits the local time difference $(t_{VA} - t_{VV})$ to the authenticating device via Bluetooth.
- **Step VI:** The authenticating device calculates the distance.

Next, we will elaborate Step I, Step IV, and Step VI.

B. Step I: Constructing Reference Signals

Fixed vs. randomized reference signals: Using fixed reference signals makes the distance estimation protocol vulnerable to a very basic spoofing attack, i.e., *replay attack*. Specifically, an attacker can obtain the fixed reference signals, e.g., via analyzing the implementation of PIANO. Then, in a replay attack, the attacker uses his/her own device to play the reference signals around the authenticating device or the vouching device such that the estimated distance is highly likely to be smaller than the authentication threshold. We note that existing acoustic signal based distance estimation protocols [14, 15] use fixed reference signals, and thus they are vulnerable to replay attacks. Therefore, we propose to randomize the reference signals every time authentication is required.

Frequency-domain randomized reference signals: It is challenging to randomize reference signals because how to randomize them also impacts the accuracy of detecting them. Specifically, we could construct randomized reference signals in either the time domain or the frequency domain. For instance, one way is to construct an array of random numbers and treat it as a reference signal in the time domain. However, such randomized reference signals include a wide range of frequency components with random powers. As a result, these reference signals will be easily interfered by background noise, which makes detecting them inaccurate. Therefore, we propose to construct randomized reference signals in the frequency domain. In particular, we discretize an appropriate frequency range (we will discuss the details of selecting the frequency range in Section VI-A) into N bins and take the central point of each bin as candidate frequencies. We denote the N candidate frequencies as a set F_R. To construct a reference signal, we first sample an integer n $(0 < n < N)$ and then select n frequencies from F_R uniformly at random. For each sampled frequency, we synthesize a sine wave with the frequency, and then we construct a reference signal by adding these sine waves.

C. Step IV: Detecting Reference Signals

In this step, both devices detect when the two reference signals arrived at them. Specifically, in Step III, each device has recorded a long audio sequence which includes the two reference signals. Then, in Step IV, each device detects the locations of the two reference signals in its recorded audio sequence, and translates the locations into timestamps. In the following, we take detecting one reference signal on the authenticating device as an example to illustrate our algorithm. Detecting the other reference signal is algorithmically the same, and the vouching device uses the same algorithm.

Algorithm 1: Our Signal Detection Algorithm

Input: X, S, F, R_f of frequency f in S, and $R = \{R_f | f \in F\}$.
Output: The location l of S in X.

1 **begin**
2 $P_{max} = -\infty$
3 **for** i = *1 to* $|X| - |S| + 1$ *with a step size* δ **do**
4 $P = NormPower(X[i \cdots i + |S| - 1], F, R)$
5 **if** $P > P_{max}$ **then**
6 $P_{max} = P$
7 $l = i$
8 **end**
9 **end**
10 //Reference signal S is not in X
11 $R_S = \sum_{f \in F} R_f$
12 **if** $P_{max} < \epsilon R_S$ **then**
13 $l = \perp$
14 **end**
15 **return** l
16 **end**

In signal processing, the *cross-correlation algorithm* is a popular method to detect the location of a reference signal in a long signal sequence. For instance, BeepBeep [15] used this algorithm. Detecting our frequency-domain randomized reference signals using the cross-correlation algorithm results in high errors (see our experimental results in Section VI-B). The key reason is a phenomena called *frequency smoothing*, in which the power of a frequency component in a reference signal is distributed to nearby frequencies after the reference signal is played by one device and recorded by the other device. Due to frequency smoothing, after a reference signal S is played by one device and recorded by the other device, its recorded version becomes S', which is significantly different from S. However, the cross-correlation algorithm tries to detect S in the recorded signal sequence, which results in high errors.

Therefore, we design a new algorithm to detect the locations of reference signals in the recorded signal sequence. Our algorithm leverages the frequency domain, and we call it *frequency-based signal detection algorithm.*

Overview of our algorithm: Our core idea is to move a window along the recorded signal sequence; for each window, we obtain the power spectrum of the signal in the window; the window whose power spectrum best matches the power spectrum of the reference signal is treated as the location of the reference signal. Specifically, Algorithm 1 shows our frequency-based signal detection algorithm. Suppose we want to detect the location of a reference signal S in a recorded signal sequence X. We denote the set of frequencies in S as F, and we denote the power at each frequency $f \in F$ in the reference signal as R_f. Our algorithm moves a window along the recorded signal with a step size δ. For each window, we compute the *normalized power* of frequencies in the reference signal. The index at which the normalized power reaches its maximum is treated as the location of the reference signal. When the reference signal is not in the recorded signal sequence, our algorithm outputs a special character $\perp$. Next, we discuss how we compute the normalized power.

Algorithm 2: $NormPower(W, F, R)$

Input: W, F, and R.
Output: Normalized power of frequencies F in W.

1 **begin**
2 $Y = PowerSpectrum(W)$
3 **for** $f \in F_R$ **do**
4 $i = \lfloor f / f_s \cdot |W| \rfloor$
5 $P_f = \sum_{k=i-\theta}^{i+\theta} Y[k]$
6 **end**
7 $P = -\infty$
8 //This sanity check enhances security
9 **if** $P_f > \alpha R_f$ *for all* $f \in F$ *and* $P_{f'} < \beta$ *for all* $f' \in F_R \setminus F$ **then**
10 $P = \sum_{f \in F} P_f - \sum_{f' \in F_R \setminus F} P_{f'}$
11 **end**
12 **return** P
13 **end**

Computing normalized power: Algorithm 2 shows how we compute the normalized power of frequencies F of the reference signal in a given signal window W. In a nutshell, a signal window has a large normalized power if 1) powers of the reference signal's frequencies in the window are comparable to those in the reference signal, and 2) the candidate frequencies that are not in the reference signal have small powers in the window. Specifically, we first get the power spectrum of the window via Fast Fourier transform (FFT). For each candidate frequency f in the reference signal, we locate the index of f in the power spectrum of the window (i.e., line 4); considering the frequency smoothing effect, we compute the power of f by aggregating the powers of the nearby 2θ frequencies (i.e., line 5), where θ is the width of frequency smoothing. Then, if the power of each candidate frequency f of the reference signal is larger than αR_f in the window, and the power of each remaining candidate frequency that is not in the reference signal is smaller than a threshold β, we compute the normalized power of the window as the sum of the powers of frequencies in the reference signal minus that of the remaining candidate frequencies (i.e., line 10), otherwise we treat the normalized power of the window as a very small number, implying that the reference signal is not in the window. Recall that R_f is the power of frequency f in the reference signal.

Next, we explain why we introduce the parameters α and β. Reference signals are often *attenuated* by hardware, i.e., after being played and recorded, a signal's powers become smaller. α is used to consider such attenuations. Suppose a background acoustic signal includes the frequencies in the reference signal and some other frequencies. If we do not perform the sanity check about the powers of frequencies that are not in the reference signal (i.e., the sanity check using the threshold β in line 9), then such a background noise could have a large normalized power, making detecting reference signals inaccurate. Likewise, an attacker can construct a spoofing reference signal via including all candidate frequencies and use it to perform replay attacks. If we do not perform sanity check in line 9, such spoofing reference signal will have a high normalized power, and our algorithm will detect it as the reference signal, making the corresponding replay attack succeed with a high probability.

Reference signal is not present in the recorded signal sequence: In some scenarios, the authenticating device and the vouching device are far away from each other so that they cannot record each other's reference signals, e.g., the user who carries the vouching device goes to have lunch while leaving the authenticating device in a shared office. Suppose an attacker tries to use the authenticating device via performing zero-effort attacks. During the authentication process, the reference signal played by one device is not present in the signal recorded by the other device, and thus the maximum normalized power is not a reliable indicator of the reference signal, making distance estimation unreliable. To consider such scenarios, our algorithm checks whether the maximum normalized power is smaller than ϵR_S (i.e., line 12 in Algorithm 1), where R_S is the power of the reference signal. If it is, our system outputs a special character and the authentication is denied.

Translating locations to timestamps: We denote by l_{AA} and l_{AV} respectively the detected locations of the reference signals S_A and S_V in the recorded signal sequence of the authenticating device. Moreover, we denote by l_{VA} and l_{VV} respectively the detected locations of the reference signals S_A and S_V in the recorded signal sequence of the vouching device. Suppose the location l_{AA} corresponds to a time point t_A on the authenticating device's time coordinate and the location l_{VA} corresponds to a time point t_V on the vouching device's time coordinate. Note that t_A and t_V are from two *different* time coordinates. As we will show in the next section, our distance estimation does not rely on the specific values of t_A and t_V. With these notations, we can transform the locations to timestamps as follows: $t_{AA} = t_A$, $t_{AV} = t_A + \frac{l_{AV}-l_{AA}}{f_A}$, $t_{VA} = t_V$, and $t_{VV} = t_V + \frac{l_{VV}-l_{VA}}{f_V}$, where f_A and f_V are the sampling frequencies that the authenticating device's microphone and the vouching device's microphone use to acquire acoustic signals, respectively. Again, t_{AA} and t_{AV} are in the time coordinate of the authenticating device; t_{VA} and t_{VV} are in the time coordinate of the vouching device; and the two time coordinates could be different.

D. Step VI: Estimating Distance

In this step, the authenticating device estimates distance. There are multiple ways to estimate distance between the two devices using the data we obtained in previous steps. Specifically, the distance between the two devices can be estimated by multiplying the speed of sound with the time that a reference signal takes to travel from one device to the other. In particular, given the timestamps t_{AA} and t_{VA}, we can estimate the distance as follows:

$$d_A = s \cdot (t_{VA} - t_{AA}), \tag{1}$$

where s is the speed of sound. Alternatively, we can also estimate the distance using the timestamps t_{AV} and t_{VV} when the reference signal S_V arrived at the authenticating device and the vouching device, respectively. Formally, we have:

$$d_V = s \cdot (t_{AV} - t_{VV}) \tag{2}$$

However, using either Equation 1 or Equation 2 requires the two devices to synchronize their time coordinate systems. For instance, Equation 1 requires that the timestamp t_{VA} on the vouching device and the timestamp t_{AA} on the authenticating device are measured from the same time coordinate system, which requires time synchronization on the two devices. However, time synchronization is generally a challenging task, and an error of 10 milliseconds in time synchronization could result in an error of more than 3 meters in distance estimation (speed of sound is around 340 m/s).

Therefore, we adopt a method developed in [15] to estimate distance, which avoids time synchronization. This method combines information about the two reference signals instead of a single one. Specifically, the method takes the average of the distances in Equation 1 and Equation 2. Formally, we have:

$$\begin{aligned} d_{AV} &= \frac{1}{2}(d_A + d_V) \\ &= \frac{1}{2} \cdot s \cdot (-\frac{l_{VV} - l_{VA}}{f_V} + \frac{l_{AV} - l_{AA}}{f_A}). \end{aligned} \tag{3}$$

where d_{AV} is the estimated distance between the two devices. Equation 3 means that computing distance reduces to computing the location differences $(l_{VV} - l_{VA})$ and $(l_{AV} - l_{AA})$, which can be estimated by the two devices locally without time synchronization.

V. Security Against Spoofing Attacks

We assume that the attacker knows the candidate frequencies in F_R, and we consider the following two spoofing attacks.

- **Guessing-based replay attacks.** An attacker could guess the reference signals and use them to perform replay attacks. Specifically, the attacker uses our signal construction algorithm to synthesize reference signals. Performing a successful replay attack requires the attacker to guess the two reference signals correctly.
- **All-frequency-based spoofing attacks.** An attacker can construct a spoofing reference signal that includes all candidate frequencies. Specifically, the attacker synthesizes a sine wave for each candidate frequency and constructs a spoofing reference signal by adding all these sine waves. Then, the attacker plays the spoofing reference signal to spoof our distance estimation protocol.

Mitigating guessing-based replay attacks: We can defend against guessing-based replay attacks via using a relatively large set of candidate frequencies. If the frequencies in a spoofing reference signal do not match those in the legitimate reference signal, then the spoofing reference signal has a very small normalized power, and eventually our algorithm will output that the reference signal is not present in the recorded signal sequence, which means that the attacker is denied. The probability that an attacker successfully guesses the candidate frequencies in one reference signal is $\frac{1}{2^N-2} \approx \frac{1}{2^N}$. The replay attacks require guessing two reference signals correctly. Therefore, the probability that the attacker successfully performs a replay attack is $\frac{1}{2^{N+1}}$. When we use a relatively large number (e.g., 30) of candidate frequencies, the probability of successful attack is negligible.

Mitigating all-frequency-based spoofing attacks: Suppose the attacker constructs a sine wave for each candidate frequency, and these sine waves have the same power P_a. Then, the attacker adds these sine waves to construct a long signal

and plays it in the entire authentication process. We can defend against such spoofing attacks via constructing the reference signals with large enough powers such that αR_f is larger than β (refer to the line 9 of Algorithm 2). When $\alpha R_f > \beta$, for all windows except those that include the reference signal, the sanity check in the line 9 of Algorithm 2 fails no matter how the attacker chooses P_a. Specifically, if $P_a \geq \alpha R_f$, the sanity check about the powers of the candidate frequencies that are not in the reference signal fails; if $P_a \leq \beta$, the sanity check about the powers of the frequencies that are in the reference signal fails; and if $\beta < P_a < \alpha R_f$, both sanity checks fail. As a result, our Algorithm 2 defines the normalized powers for these windows as negative infinity. Therefore, either the reference signal is still accurately detected or our algorithm reports that the reference signal is not present in the recorded signal. In either case, the attacker is denied access.

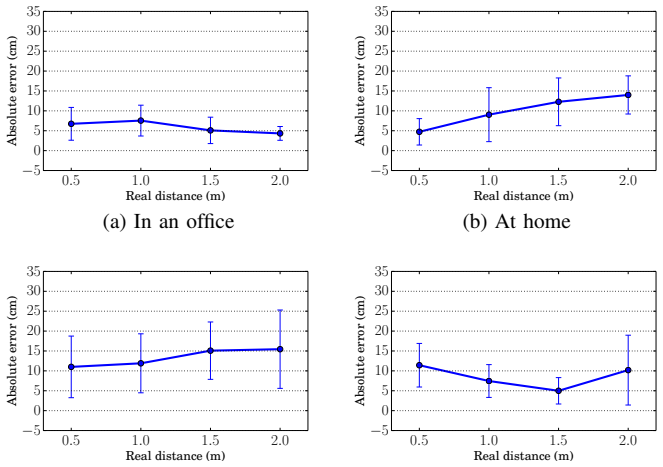
VI. Experiments

A. Experimental Setup

Proof-of-concept prototype: We implemented a prototype of PIANO as two Android apps and we run them on two Samsung Galaxy S4 smartphones; one is used as the authenticating device while the other is treated as the vouching device. The two smartphones are paired using Bluetooth. In our implementation of Algorithm 1, we use adapted step sizes instead of using a fixed step size to achieve a trade-off between efficiency and accuracy. Specifically, we first use a step size of 1,000 to locate the window where the normalized power reaches the maximum; then we use a step size of 10 to perform a more fine-grained search around the window to locate a more accurate maximum. Moreover, we detect the two reference signals simultaneously in one scan of the recorded signal, which is more efficient than detecting them in two scans.

Reducing impact of background noise: Background acoustic signals could impact the accuracy of distance estimation since our protocol uses acoustic signals. Specifically, if a certain background noise has a power spectrum that is close to that of a reference signal, our distance estimation protocol might detect the background noise as the reference signal, which makes distance estimation inaccurate. We collected background acoustic noises in various environments (office, home, street, etc.) and found that most powers of background noises concentrate on frequencies that are smaller than around 6K Hz. Therefore, when we select candidate frequencies, we do not consider frequencies that are less than 6K Hz.

Using inaudible sound frequency: We set the sampling frequency on both smartphones to be 44.1K Hz, which is the largest sampling frequency supported by the Android system. Given such sampling frequency and that background noises mainly concentrate on frequencies less than 6K Hz, the *aliasing frequencies* of background noise mainly concentrate in the range [38K Hz, 44K Hz]. Considering background noise, we use the frequency range [25K Hz, 35K Hz]. Specifically, we equally divide this frequency range to be 30 bins and take the center of each bin as a candidate frequency, i.e., we have 30 candidate frequencies. We note that our constructed reference signals are almost inaudible (they are not completely inaudible due to hardware imperfection).



(a) In an office (b) At home

(c) On the street (d) In a restaurant

Fig. 1: Distance estimation errors in different environments.

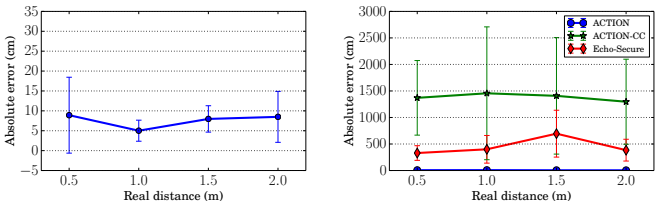
Setting parameters: When we construct a reference signal, we make its power the maximum value that can be represented by the Android system. Specifically, suppose we sampled n candidate frequencies when constructing a reference signal. For each frequency f, we synthesize a sine wave with a power $(\frac{32000}{n})^2$ (i.e., $R_f = (\frac{32000}{n})^2$ in Algorithm 1 and Algorithm 2), where we use 32000 because the Android system uses 16 bit integer to represent signals in the time domain. Since FFT requires the length of the signal to be a power of 2, we set the length of our reference signals to be 4,096, which lasts for 93 milliseconds given our sampling frequency is 44.1K Hz. We set $\epsilon = \alpha = 1\%$ and $\beta = 0.5\% \times R_f$ to be secure against various spoofing attacks. Moreover, we set $\theta = 5$ to consider frequency smoothing effects.

B. Accuracy of ACTION at Estimating Distance

With the current parameter setting of our prototype, we find that when the real distance between the two devices is larger than around 2.5 meters, ACTION determines that the reference signal is not present in the recorded signal and thus authentication on the authenticating device is denied. Therefore, we measure the accuracy of distance estimation when the real distance is smaller than 2.5 meters. Suppose the real distance between the authenticating device and the vouching device is d, and the distance estimated by ACTION is d_{AV}, we define the *absolute error* as $|d - d_{AV}|$. We report the absolute errors of distance estimation in different scenarios. For each real distance, we average the absolute errors over 10 trials.

1) Different Environments: We evaluate the accuracy of distance estimations in various environments.

In a shared office, at home, on the street, and in a restaurant: These environments represent places where a user could use PIANO in daily life, and they represent different levels of background noises. For instance, on the street, we have background noise introduced by cars and passersby. In a restaurant, people are chatting and having meals. Figure 1 shows the error bars of distance estimation in these environments when the real distance is 0.5, 1, 1.5, and 2 meters.



(a) (b)

Fig. 2: (a) Error bar of distance estimation when three users are using the authentication system on their devices simultaneously in a shared office. (b) Comparing three secure acoustic signal based distance estimation protocols.

We observe that distance estimation in ACTION is accurate in different environments. Specifically, in a shared office, the average absolute errors are between only 5 centimeters and 7 centimeters. Although the average absolute errors are larger on the street where the background noise is heavier, they vary between 10 and 15 centimeters.

Separated by a wall: We find that, when the two devices are close but are separated by a wall, one device detects that the reference signal played by the other device is not present, and thus the access to the authenticating device is denied. This is because the reference signals are significantly attenuated by the wall. This means that an attacker cannot access the authenticating device if the user, who carries the vouching device, is separated with the authenticating device by a wall.

2) Multiple Users: In a public place such as a shared office and a restaurant, multiple users that have adopted PIANO might launch the system on their devices at close times. We measure the accuracy of distance estimation in such scenarios in a shared office. In particular, we assume there are 3 such users. To simulate such scenarios, in each trial of our experiment, we generate 2 pairs of randomized reference signals, and use two other devices to play them when we launch our authentication system on the two smartphones. We performed such simulations at four real distances (i.e., 0.5, 1.0, 1.5, and 2.0 meters), and for each real distance, we repeat for 10 trials.

First, if two reference signals overlap significantly, ACTION will determine that the reference signals are not present in the recorded signal. This is because the overlapped reference signal will fail the sanity check at line 9 of Algorithm 2. However, the probability of such cases is very small. Indeed, in the 40 trials of our experiments, we only observe 3 trials that are such cases. Figure 2 shows the error bar of distance estimation in the remaining trials. We observe that, compared to Figure 1a where only one user is using PIANO in a shared office, the average errors are slightly larger. This is because reference signals played by different users could have partial overlaps, which decreases the accuracy slightly.

3) Comparison with Previous Methods: We compare three secure acoustic signal based distance estimation methods: ACTION, ACTION with our frequency-based signal detection algorithm replaced by the cross-correlation algorithm (denoted as ACTION-CC), and Echo [14] with randomized reference signals and our frequency-based signal detection algorithm (denoted as Echo-Secure). In contrast to Echo, Echo-Secure is secure against replay attacks. Echo was one of the first acoustic signal based distance bounding protocols. When using Echo in our proximity-based authentication, the authenticating device first sends a reference signal to the vouching device via Bluetooth; the vouching device immediately plays the reference signal after receiving it; the authenticating device records acoustic signals and detects when the reference signal arrived at the authenticating device. Then the distance is the speed of sound multiplies the elapsed time (subtracting the processing delay). We estimated the average processing delay via putting the two devices together (real distance is close to 0) and treating the elapsed time as the processing delay. The original Echo protocol uses fixed reference signal and does not discuss particular signal detection algorithm. In Echo-Secure, we use randomized reference signal, and our frequency-based algorithm to detect reference signals.

Figure 2b shows the results of the three methods. We performed the experiments in a shared office. We observe that ACTION is orders of magnitude more accurate than ACTION-CC and Echo-Secure. This implies that 1) our frequency-based algorithm is much more accurate than the cross-correlation algorithm at detecting our randomized reference signals, and 2) processing delays on the devices are unpredictable. Specifically, ACTION-CC is inaccurate because the reference signals change significantly in the time domain after they are played and recorded, due to frequency smoothing. As a result, cross-correlation algorithm tries to match the original reference signal with the changed reference signal, resulting in high errors. Echo-Secure is inaccurate because processing delay is very unpredictable on the devices. For instance, when the vouching device wants to play the reference signal, there is an unpredictable delay between the API to play acoustic signal is called and the signal is actually played.

C. FRRs and FARs of Authentication

In the above section, we studied the accuracy of estimating a given distance. In this section, we show the accuracy–*False Rejection Rate (FRR)* and *False Acceptance Rate (FAR)*– of authentication decisions made by PIANO. We denote by d_s the maximum distance at which the reference signal played by one device can reach to the other. With our current parameter setting, we have $d_s \approx 2.5$ meters. When the real distance between the two devices is no less than d_s, PIANO rejects the access without estimating the distance. Moreover, given a real distance d ($0 < d < d_s$) between the two devices, we assume the distance estimated by PIANO follows a Gaussian distribution whose mean is the real distance d and standard deviation is σ_d. We note that this assumption does not contradict with our results in the previous section because those distance estimation errors are *absolute errors*. Indeed, using our collected data, we verified that the average estimated distance is very close to the real distance. Furthermore, we consider σ_d to be constant and we estimate it by averaging the standard deviations at the four points (i.e., 0.5, 1.0, 1.5, and 2.0) obtained in our experiments. Under these settings, we compute FRR by averaging the FRRs at each legitimate distance (i.e., $\leq \tau$) and compute FAR by averaging the FARs at each illegitimate distance ($> \tau$).

Note that we use Bluetooth to pair the two devices. Therefore, FAR is 0 when the real distance between the two devices is larger than the communication range of Bluetooth.

TABLE I: FRRs in different scenarios.

	0.5m	1.0m	1.5m	2.0m
Office	5.6%	2.8%	1.9%	1.4%
Home	9.5%	4.8%	3.2%	2.4%
Street	12.6%	6.3%	4.2%	3.1%
Restaurant	8.5%	4.2%	2.8%	2.1%
Multiple users	7.9%	4.0%	2.6%	2.0%

TABLE II: FARs in different scenarios.

	0.5m	1.0m	1.5m	2.0m
Office	0.3%	0.3%	0.3%	0.4%
Home	0.5%	0.5%	0.6%	0.6%
Street	0.7%	0.7%	0.7%	0.8%
Restaurant	0.4%	0.5%	0.4%	0.4%
Multiple users	0.4%	0.4%	0.5%	0.5%

In other words, FAR is 0 when the real distance between the two devices is larger than 10 meters, which is roughly the communication range of Bluetooth on many commodity mobile devices [18].

Table I and Table II show the FRRs and FARs (when the two devices are within the communication range of Bluetooth) in different scenarios and for different authentication thresholds. First, PIANO achieves low FRRs and very low FARs. For instance, in a shared office, FRR is 2.8% and FAR is 0.3% when the authentication threshold is 1.0 meter. Second, we observe that FRRs decrease quickly while FARs slightly increase as authentication threshold increases. For instance, FRRs decrease by a half in all scenarios when the authentication threshold increases from 0.5 to 1.0 meter.

D. Efficiency

We measure the efficiency of our prototype in terms of both time and energy consumption. PIANO is fast. In our current implementation, one authentication can be finished within around 3 seconds. We stress that our prototype is just a proof-of-concept. There are various ways to optimize the time efficiency. For example, we can predict when a device will be used, e.g., when accelerometer and gyroscope data are available, we can detect a device is picked up. Therefore, we can perform authentication before the device is used. Moreover, we use a tool called PowerTutor [19] to measure the energy consumption of our prototype. PowerTutor measures the battery energy consumed by an Android app during a period of time. We find that performing 100 times of authentication only consumes 0.6% of the smartphone battery.

E. Security against Spoofing Attacks

We performed 100 trials of guessing-based replay attacks and all-frequency-based spoofing attacks that we discussed in Section V. In all of these trials, ACTION detects that the reference signals are not in the recorded signal because of the sanity check at line 9 in Algorithm 2 and line 12 in Algorithm 1. As a result, all these attack trials failed.

VII. Conclusion and Future Work

We propose PIANO, a proximity-based user authentication method for voice-powered IoT devices. PIANO propagates a user's identity from its vouching device to an authenticating device. The key component of PIANO is a new acoustic signal based protocol that can estimate distance between two devices accurately, efficiently, and securely. Via empirical evaluations, we find that our distance estimation protocol is accurate; PIANO achieves low FRRs and FARs at making authentication decisions; PIANO is fast and has low energy consumption; and PIANO is secure against various spoofing attacks. Interesting directions for future work include adapting PIANO to other application scenarios, e.g., web authentication.

Acknowledgement: We thank Grant Ho and anonymous reviewers for helpful comments. This material is supported by the National Science Foundation under Grants No. TWC-1409915, CNS-1238959, CNS-1238962, CNS-1239054, and CNS-1239166. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

[1] IoT Devices Statistics by Gatner Inc., April 2016.
[2] Bee., April 2016.
[3] Smart Garage Door., May 2016.
[4] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *TIFS*, 8(1), 2013.
[5] Neil Zhenqiang Gong, Mathias Payer, Reza Moazzezi, and Mario Frank. Forgery-resistant touch-based authentication on mobile devices. In *AsiaCCS*, 2016.
[6] Anil K Jain, Karthik Nandakumar, and Arun Ross. 50 years of biometric research: Accomplishments, challenges, and opportunities. *Pattern Recognition Letters*, 2016.
[7] Matthew P Aylett12 and Junichi Yamagishi. Combining statistical parameteric speech synthesis and unit-selection for automatic voice cloning. 2008.
[8] Examples of Voice-powered IoT, May 2016.
[9] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *SEC*, 2016.
[10] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6), 2007.
[11] Kasper Bonne Rasmussen, Claude Castelluccia, Thomas S Heydt-Benjamin, and Srdjan Capkun. Proximity-based access control for implantable medical devices. In *CCS*, 2009.
[12] Kasper Bonne Rasmussen and Srdjan Capkun. Realization of rf distance bounding. In *SEC*, 2010.
[13] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-level localization with a single wifi access point. In *NSDI*, 2016.
[14] Naveen Sastry, Umesh Shankar, and David Wagner. Secure verification of location claims. In *WiSec*, 2003.
[15] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *SenSys*, 2007.
[16] Alex Varshavsky, Adin Scannell, Anthony LaMarca, and Eyal de Lara. Amigo: Proximity-based authentication of mobile devices. In *UbiComp*, 2007.
[17] Hossein Shafagh and Anwar Hithnawi. Come closer - proximity-based authentication for the internet of things. In *MobiCom*, 2014.
[18] Bluetooth distance. http://goo.gl/aME7G.
[19] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *CODES/ISSS*, 2010.