2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) | 979-8-3503-4793-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/DSN58367.2023.00060

VoiceGuard: An Effective and Practical Approach for Detecting and Blocking Unauthorized Voice Commands to Smart Speakers

Xuening Xu[1], Chenglong Fu[2], Xiaojiang Du[1], and E. Paul Ratazzi[3]

[1]*Department of Electrical and Computer Engineering*, *Stevens Institute of Technology*, Hoboken, NJ, USA
[2]*Department of Software and Information Systems*, *UNC Charlotte*, Charlotte, NC, USA
[3]*Information Directorate*, *Air Force Research Laboratory*, Rome, NY, USA
Emails: xxu64@stevens.edu, chenglong.fu@uncc.edu, xdu16@stevens.edu, edward.ratazzi@us.af.mil

***Abstract*—Smart speakers bring convenience to people's daily lives. However, various attacks can be launched against smart speakers to execute malicious commands, which may cause serious safety or security issues. The existing solutions against sophisticated attacks such as voice replay attacks and voice synthesis attacks require intrusive modifications of the smart speaker hardware and/or software, which are impractical for general users. In this work, we present a novel security scheme - *VoiceGuard* that can effectively detect and block unauthorized voice commands to smart speakers. VoiceGuard does not require any modification to smart speakers' hardware or software. We implement a prototype of VoiceGuard on two popular smart speakers: Amazon Echo Dot and Google Home Mini, and evaluate the scheme in three real-world testbeds, which include both single-user and multi-user scenarios. The experimental results show that VoiceGuard achieves an accuracy of 97% in blocking malicious voice commands issued by illegitimate sources while having a negligible impact on the user experience.**

***Index Terms*—smart speakers, attacks, Bluetooth RSSI**

I. Introduction

Smart speakers have shown notable growth in market share. It is reported that in 2020 approximately 40 percent of U.S. households have smart speakers and this number is projected to surpass 75% by 2025 [6]. These speakers are powered by advanced speech recognition technologies and cloud backend servers, and have been widely adopted by common users to perform various tasks such as managing calendars, placing online orders, and controlling smart home IoT devices.

Despite their popularity, smart speakers also expose new attack surfaces for attackers to invade users' privacy and even issue malicious commands in smart home environments. For instance, attackers can play fabricated audio clips to a smart speaker to ask for the user's daily schedule or traveling plan [77]. Furthermore, smart speakers can be exploited by fake commands to execute malicious actions such as placing unwanted orders, turning off home security systems, or opening a smart lock when the owner is not at home.

DISTRIBUTION STATEMENT A. Approved for public release. Case Number: AFRL-2023-1864.

To cope with malicious commands from unauthorized sources, existing commercial smart speakers can be trained to recognize the users' voices during the setup phase. The learned features are then used to authenticate the identity of command issuers and block malicious commands if they are considered not to be from legitimate users. However, this voice-matching-based protection is still in its preliminary stage and can be easily circumvented. For instance, such protection can be evaded by replay attacks and impersonation attacks [31], [48], [72]. Although more recent works explore methods to enhance the Automatic Speech Recognition (ASR) systems used by smart voice assistants to differentiate playback from the live human voice [17], [39], [66], [74], [79], they are vulnerable to adversarial example attacks that generate synthetic audio mimicking the owner and successfully fool the voice assistants [23], [24], [27], [54], [86]. Voice liveness detection methods [14] employ machine learning models to help process audio signals, which adds another layer of vulnerability. An adaptive attacker can evade them with the knowledge of the detection methods. Besides, these works require the distance between the source and target to be very small. Most of the existing defense works focus on detecting malicious commands, while very few works [33] study practical methods that can block the execution of malicious commands after being detected.

Proximity-based methods [56], [58], [88] have been proven to be effective in designing security schemes for IoTs. For example, in [56], authors propose an access control scheme that only allows implantable medical devices to be accessed by devices that are in close proximity. For smart speakers, our observations are: (1) legitimate voice commands are usually issued by an owner who is physically close to the smart speaker; and (2) the network traffic of speaker activation has specific patterns (details given in Section IV-B). Based on the above observations, we design an effective and practical smart speaker protection scheme, VoiceGuard that not only detects but also blocks malicious voice commands. Unlike many existing defense methods [33], [65], [77] that either require modifications of smart speakers' firmware or hardware or need to update the service software on the voice command backend

2158-3927/23/$31.00 ©2023 IEEE
DOI 10.1109/DSN58367.2023.00060

servers, our VoiceGuard does not require any modification to smart speakers' hardware or software, and can be easily deployed for existing smart speakers.

Our contributions are summarized below:

- We propose an effective and practical scheme that utilizes the signal strength of a smart speaker's built-in Bluetooth to check if the owner is close to the smart speaker when a voice command is received by the speaker. Unlike some existing methods, our scheme does not require any modification or update to a smart speaker's hardware or software, nor on smart speakers' backend cloud servers.
- By leveraging a new finding on IoT event delays [34], [28], we design a Traffic Processing Module that can hold smart speaker's traffic for dozens of seconds without triggering any alarm or causing the connection to be terminated. Compared to methods such as firewalls and network filters that break the connection and require users to repeat a voice command, our method has minimal influence on the user experience.
- We implement VoiceGuard on two popular smart speakers: Amazon Echo Dot and Google Home Mini. We conduct comprehensive experiments in three real-world testbeds (a two-floor house, a two-bedroom apartment and a large office). The results show that VoiceGuard can identify legitimate and malicious voice commands with very high accuracy (above 97%). Also, VoiceGuard works well in multi-user scenarios.
- We evaluate VoiceGuard with both smartphone and smartwatch. The results are consistent and show the reliability and effectiveness of VoiceGuard. As long as a user has her phone or a wearable with her (or nearby), our scheme works well in detecting and blocking malicious voice commands.

The paper is organized as follows. In Section II, we describe the background of smart speakers and audio domain attacks. In Section III, we discuss the system and threat model. We present the system design and evaluation in Section IV and Section V, respectively. In Section VI, we describe related work. We discuss limitations and future works in Section VII. We conclude the paper in Section VIII.

II. Background

A. Smart Speakers

Voice assistant platforms such as Amazon Alexa, Google Assistant, and Apple Siri provide a new interface for users to access Internet services and control smart IoT devices at home. By providing IoT integration APIs such as skills [9] and intents [11], third-party IoT device vendors are able to integrate their products with voice control features, resulting in a trend that voice assistant platforms are taking the role of IoT management hubs. A typical voice assistant system consists of a backend cloud server and a client. The voice audio is captured by the client device and forwarded to the cloud server for speech-to-text transcribing and command execution. The voice assistant clients have been built into various smart home appliances such as smart speakers, smart TVs, and smart refrigerators. Among them, smart speakers play the most important role in popularizing voice assistants. Smart speakers are very common clients of voice assistant systems, which have been deployed in more than 40% of US households in 2020 [6]. Smart speakers combine internet connectivity with traditional speakers, and are activated by certain wake-up words (e.g., "Alexa"). After being woken up, they record the audio in the following short period and forward it to the backend servers. Smart speakers can be implemented using low-cost hardware by delegating most of the processing workload to cloud servers, and they are commonly used to control safety-sensitive IoT devices such as smart locks, home security systems, and electrical heaters. Also, most smart speakers provide Bluetooth connections for users to cast music streams directly from their smartphones.

Note that voice assistants on smart speakers and smartphones are independent instances. VoiceGuard is designed for smart speakers, but still allows users to issue voice commands on smartphones.

B. Audio Domain Attacks

The widely used smart speakers also open up surfaces for adversaries to launch attacks, such as fake voice commands. In recent years, automatic speech recognition (ASR) systems used by voice assistants have proven to be vulnerable to various attacks [23], [24], [27], [54], [86], [87].

1) Replay and Impersonation Attacks: The ASR systems used by the smart speakers are protected with voice biometric authentication approaches, which only rely on audio-domain features. An adversary can easily bypass such authentication by playing pre-recorded or synthesized user's voice commands [31], [48], [72].

2) Audio Adversarial Examples: In the past few years, many works [23], [24], [32], [43], [54], [63] on audio adversarial example (AE) attacks are proposed to attack different ASR systems, such as DeepSpeech [37], Lingvo [67], Kaldi [53]. However, the audio AEs generated in these works are directly fed into ASR systems, and they are not effective after the over-the-air transmission, which means these adversarial attacks do not impose real threats to the commercial voice assistant enabled smart devices, e.g., Amazon Echo and Google Home. Some other works [62], [82], [86] focus on open-source ASR systems where their models are available and have managed to generate audio AEs that can survive after being played through loudspeakers. Furthermore, Devil's Whisper [27] successfully attacks commercial ASR systems over the air, which puts commercial smart speakers at potential risk.

3) Inaudible Attacks: The non-linearity of the diaphragm of microphone and power amplifier of the receiver has been exploited to generate inaudible attacks in [59], [60], [84], [87]. An attacker can modulate the voice commands onto ultrasound carriers to generate inaudible voice commands, which are inaudible to humans but can be "heard" by microphones. Such an inaudible attack is also a real threat to smart speakers, but it needs a special ultrasonic speaker to play modulated ultrasonic

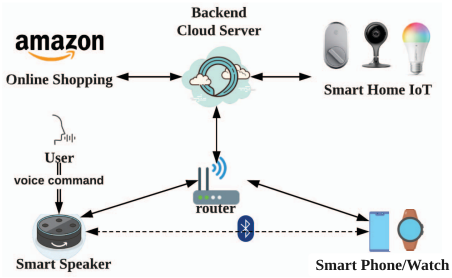


Fig. 1: The system model of a smart speaker.

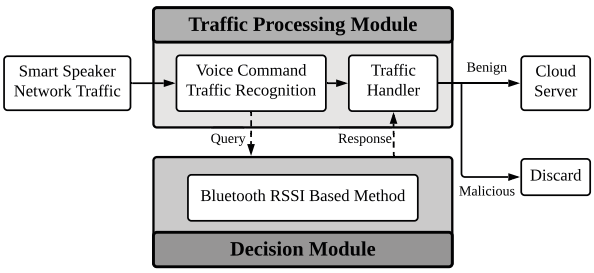


Fig. 2: Architecture of VoiceGuard.

signals. Moreover, non-linearity based inaudible attack can be successfully defended by the countermeasure proposed in [60].

III. System and Threat Model

A. System Model

In this work, we consider a typical smart speaker system as illustrated in Figure 1. The smart speaker is equipped with microphones and speakers to interact with users directly, and a WiFi function to exchange information with the backend cloud server. A smart speaker establishes a long-live network connection with the cloud server once it goes online. When activated by the activation word (e.g., "Alexa" for Echo speakers), it starts recording the sound and sends the audio to the cloud server for voice command transcribing and execution. Finally, the response of the voice command is returned to the smart speaker and played back to the user. In this service architecture, the cloud server processes and executes a voice command, which means it cannot take effect if the network packets containing the voice command do not reach the cloud server. Besides, most smart speakers are Bluetooth-enabled for casting audio streams from other smart devices.

B. Threat Model

We aim to protect smart speakers against fake/unauthorized voice commands from a guest or a compromised playback device (e.g., a smart TV) nearby. Existing smart speakers lack effective methods to handle sophisticated attacks such as replaying pre-recorded or synthetic owner's voices. A series of works shows that voice assistants (used by smart speakers) can be exploited by hidden voice commands that are embedded in music or video streams, or modulated on inaudible ultrasound to issue hazardous commands [59], [60], [84], [86], [87]. Although some voice assistants can be trained to recognize the owner's voice, attackers can still use pre-recorded or synthetic audio of the owner's voice to bypass the authentication [31] and execute malicious commands without owner's awareness.

In this work, we consider both on-scene and remote attackers whose goal is to execute malicious voice commands using a victim's smart speaker. We assume attackers have the following capabilities by following the assumptions of existing works [26], [86], [87]. First, attackers can acquire the victim's voice samples by searching published audio/video clips, making scam calls, and in-person spy recording. Then, attackers can utilize the acquired voice samples to synthesize attacking audio samples that can bypass voice match schemes of smart speakers [31]. For on-scene attackers, they can play attacking audio samples to the target smart speaker when they visit the victim's home as guests. For remote attackers, they can play malicious audio samples via a compromised playback device (e.g., a smart TV) for a specific victim or embed malicious commands in videos/audios that are published on popular media streaming platforms (e.g., YouTube and Spotify) for large-scale attacks. VoiceGuard monitors and analyzes real-time network traffic between a smart speaker and its cloud server. Note that the traffic between a commercial smart speaker (e.g., Amazon Echo, Google Home) and its cloud server is encrypted and mutually authenticated. Therefore, the attacker cannot evade the detection of VoiceGuard by modifying packets (e.g., changing packet sizes) between a smart speaker and its cloud server.

IV. VoiceGuard: The System Design

In this work, we present VoiceGuard as a defense scheme that can effectively detect and block malicious voice commands to smart speakers. VoiceGuard ensures that smart speakers only execute voice commands from legitimate users.

A. Overview of VoiceGuard

VoiceGuard is designed to detect and block fake voice commands targeting smart speakers. VoiceGuard can be deployed as software (consisting of several Python scripts with a total size of less than 100 KB) on any general-purpose computing device in a home, such as a desktop or a laptop. With VoiceGuard, users can still normally issue voice commands using the voice assistants on their smartphones. VoiceGuard consists of two modules as in Figure 2: the Traffic Processing Module and the Decision Module. The Traffic Processing Module has two parts: the Voice Command Traffic Recognition sub-Module determines if the traffic carries voice commands; the Traffic Handler sub-Module blocks a voice command if the Decision Module determines it is malicious, based on the smart speaker's current Bluetooth RSSI at the owner's smartphone or a smart wearable device (e.g., a smartwatch).

Most existing smart speakers show visual effects when they are activated and ready to accept voice commands. For example, Amazon Echo speakers light up their blue light

circle, and Google Home speakers blink their light dots when entering listening mode. Even if a smart speaker is activated by inaudible attacks, there will still be visual and/or audio (e.g., an acknowledgment chime or a voice statement) confirmations after the command is executed, which can be noticed by the owner. Based on the above observation, a voice command attack can be detected by the homeowner (the victim) if the victim is close (e.g., in the same room) to the smart speaker. Once a smart speaker is activated, the Traffic Processing Module determines if the network packets belong to a voice command. The details are given in Section IV-B. If the traffic carries a voice command, the Traffic Handler sub-Module temporarily holds the traffic, while the Decision Module determines if the voice command is legitimate. If so, the on hold network packets are sent to the backend cloud server. Otherwise, the network packets are considered to carry fake voice commands and are dropped. When queried by the Traffic Processing Module, the Decision Module makes a decision based on the current Bluetooth RSSI measurement at the owner's smartphone or smartwatch (detailed in Section IV-C).

B. The Traffic Processing Module

We employ the Traffic Processing Module to recognize voice command traffic instead of using keyword recognition services, because keyword recognition sensors (e.g., microphones) could be fooled by advanced attacks. For example, attackers can inject laser-based attacks [69] only to activate the speaker's microphone (the detection/security system is not aware of the activation). Besides, network traffic is inevitable for executing voice commands and has robust patterns [73].

The Traffic Processing Module is implemented as part of the VoiceGuard software and it has two functions: voice command traffic recognition and traffic handling.

1) Voice Command Traffic Recognition: The traffic between the Echo Dot and its cloud server differs from that of the Google Home Mini. Thus, the recognition procedures are slightly different and are presented separately. A survey shows that Amazon Echo dominates the US market with 70% market share followed by Google Home with 25% share [7]. In this paper, we present the voice command traffic recognition on these two popular smart speakers, which account for over 95% of smart speakers being used in the US.

The traffic flows originating from a smart speaker are complex and only some of them are related to voice commands. The Voice Command Traffic Recognition sub-Module is to identify voice commands and ignore all other irrelevant traffic, so that the Traffic Handler sub-Module can focus on the voice command traffic and process it accordingly. We run Wireshark [13] on a laptop that hosts the Traffic Processing Module to observe network traffic. Since our goal is to hold voice command traffic from smart speakers to clouds, below we only discuss network traffic originating from smart speakers.

Amazon Echo Dot. At the beginning, the Echo Dot sends several DNS queries to the home router asking for IP addresses of several different Amazon servers, and establishes connections with these servers. Then, the Echo Dot heartbeats every 30 seconds to keep the session alive with one of the Amazon servers, whose domain name is "avs-alexa-4-na.amazon.com". We refer to it as the AVS server for convenience. Besides the heartbeats, it establishes short-time connections with other servers occasionally. The Echo Dot starts sending a large volume of continuous traffic to the AVS server when it receives a voice command, and it continues to heartbeat periodically after the entire interaction with the server is done. Our experiments confirm that the voice command traffic is between the Echo Dot and the AVS server, which is what we need to monitor.

To filter out irrelevant traffic and only keep the traffic between the Echo Dot and the AVS server for further analysis, an intuitive way is to use IP address of the AVS server, which can be easily obtained from the DNS response during the beginning phase. Sometimes the current connection with the AVS server is disconnected and then the Echo Dot connects to an AVS server with the same domain name, but a different IP address. Given the situation, we must update IP address of the AVS server as soon as the new connection is established. After some preliminary works, we find simply retrieving IP address of the AVS server from DNS responses is not guaranteed to work for the Echo Dot, and sometimes we fail to acquire new IP address of the AVS server by tracking DNS responses.

Different connections may have different packet-level signatures [73], and we want to find the packet-level signature of establishing a connection with the AVS server. A packet-level signature is the length of a sequence of packets. Similar to [73], we only consider lengths of a subset of packets that are labeled as "Application Data" in the (unencrypted) TLS record header in TLS connections, because only such packets could contain voice commands. Though there are only a few times that the Echo Dot sends DNS queries asking for the IP address of the AVS server, we are still able to use them to find the packet-level signature, which can be used later to retrieve the new IP address of the AVS server when there is no DNS query. For each captured DNS query related to the AVS server, we retrieve the corresponding IP address from the DNS response and then start recording the length of each packet sent by the Echo Dot during the connection establishing phase. After many times of experiments, we have an interesting observation: the first part of the packet-length (in bytes) sequences are identical, and it is "63, 33, 653, 131, 73, 131, 188, 73, 131, 73, 131, 73, 131, 77, 33, 33". To further confirm this sequence is the packet-level signature of an Echo Dot connecting to the AVS server, we compare it with sequences of connecting to other six Amazon servers frequently used with the Echo Dot. We find that this sequence is different from other sequences, and hence it can be used to identify a new connection to the AVS server. Then we obtain the AVS server's new IP address from the packet header.

Now we focus on the voice command traffic between the Echo Dot and the AVS server. When the Echo Dot receives a voice command, it is important to immediately recognize the traffic that carries the voice command so that the Traffic Handler can hold the traffic and let the Decision Module make decisions. For an Echo Dot, when there is no voice command,

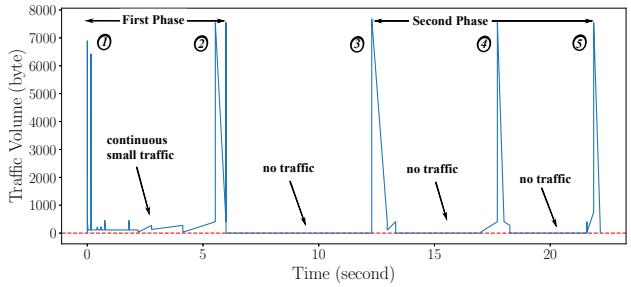


Fig. 3: Traffic spikes during a user-Echo interaction. Spikes ①, ③, ④, and ⑤ all occur after a no-traffic period. The naive method fails to distinguish ① from others and mistakenly considers ③④⑤ to be the first phase, which makes the Traffic Handler hold the traffic, incurring unnecessary delays.

the traffic only contains a heartbeat every 30 seconds with a length of 41 bytes. If we ignore the heartbeat traffic, then there is no traffic when there is no voice command. A naive method to detect a voice command would be: whenever there is a traffic spike after a no-traffic period (e.g., spikes ①, ③, ④ and ⑤ in Figure 3), then the Echo Dot receives a voice command. However, according to our experiments, this naive method is not accurate. We explain this below.

We use Figure 3 to present the details. A normal voice command invocation of an Echo Dot by a user has two phases. The first phase (referred to as the "command phase") is when the user gives a voice command to the Echo Dot. At the beginning of the command phase, the Echo Dot starts with a traffic spike ①, which is generated when it is activated by the voice "Alexa" and starts communication with the AVS server. After the spike ①, there are a number of small packets (time intervals are less than 1 second), followed by the spike ②, which is the voice command. The end of the first phase is indicated by no traffic for several seconds. After the no-traffic period, the second phase (referred to as the "response phase") starts, which includes one or more spikes. During the second phase, the Echo Dot speaks a response to the user.

For the example in Figure 3, a user asks for tonight's NBA schedule, and in the second phase, the response includes three NBA game schedules. A spike is caused at the end of each game schedule spoken by the Echo Dot, resulting in three spikes during the second phase. If we use the naive method, the response spikes ③, ④ and ⑤ will be recognized as voice command traffic, which will trigger the Traffic Handler to hold the traffic, incurring unnecessary delays.

We try to figure out the packet-level signatures for both phases. To this end, for each voice command in our experiments, we record the packet lengths of the first several packets of both phases. According to the collected length sequences, neither of the two phases has a fixed packet-level signature. However, each phase has its own frequently appearing traffic patterns, which can be used to differentiate the two phases. We present the details of differentiating two phases of the Echo Dot, and all the packet lengths are in the unit of bytes. For convenience, we only use the actual length and omit the unit.

For the first phase, packets of two lengths appear with the highest frequency, which are the packet of length 138 (p-138) and the packet of length 75 (p-75). Most of the time, at least one of p-138 and p-75 appears within the first 5 packets of the first phase, but not necessarily in the same place. When there is no p-138 or p-75 in the first five packets, the first phase has three fixed traffic patterns as follows: a) $[250-650, 131, 277, 131, 113]$; b) $[250-650, 131, 113, 113, 113]$; c) $[250-650, 131, 121, 277, 131]$. The length of the first packet is between 250 and 650, and the most common value is 277.

For the second phase, packets of two lengths appear frequently: packets of length 77 (p-77) and length 33 (p-33). p-77 and p-33 usually appear sequentially within the first five packets, but sometimes they can be the 6th and 7th packets. According to the collected sequences, they can always be found within the first seven packets.

Google Home Mini. The workflow of VoiceGuard on Google Home Mini is basically the same as that of the Echo Dot (Figure 2). Google Home Mini exchanges voice commands and responses with the cloud server with a domain name of "www.google.com" [52]. Instead of using packet-level signatures, we can accurately identify the connection that is used for transmitting voice commands between the Google Home Mini and the cloud server by tracking the smart speaker's DNS queries. Other than this, there are only two differences in voice command traffic recognition.

First, unlike Echo Dot, which exclusively uses TCP to communicate with its cloud server, Google Home Mini implements the QUIC [19] protocol that is built on top of UDP. It switches between QUIC and TCP according to the network conditions [22]. To accommodate the switching, we add a UDP forwarder in the Traffic Handler that runs together with the TCP proxy so that Google Home Mini's traffic can be held and forwarded no matter using TCP or UDP.

Second, rather than the long-live connection used by the Echo Dot, Google Home Mini's connection to the cloud server is on-demand. The TLS session is only established after the speaker is invoked by a voice command. In addition, Google Home Mini does not have the response spike as the Echo Dot (i.e., ③,④ and ⑤ in Figure 3), which makes Google Home Mini's traffic of voice commands easier to be recognized: any traffic spike occurring after an idle period is a voice command.

2) Traffic Handler: The Traffic Handler serves as an actuator to block traffic of malicious voice commands. As shown in Figure 2, after a traffic spike is recognized as a voice command, the Traffic Processing Module asks the Decision Module to check if the voice command is legitimate. The Traffic Handler temporarily holds the packets when waiting for the Decision Module's response, which should not break the established network connection so that the held packets can be accepted by the cloud server later if determined as legitimate. Otherwise, the user has to repeat the voice command.

We utilize the transparent proxy in [28] to achieve the goal of maintaining the network connection during a packet-holding period. The proxy is implemented on a laptop and sits in

(I) no delay

36	18:29:33.981571597	192.168.1.200	54.239.18.66	TLSv1.2	348 Application Data
38	18:29:33.981748657	192.168.1.200	54.239.18.66	TLSv1.2	202 Application Data
40	18:29:33.982207638	192.168.1.200	54.239.18.66	TCP	5858 56316 → 443 [ACK]
42	18:29:33.982207949	192.168.1.200	54.239.18.66	TLSv1.2	775 Application Data
44	18:29:33.982208064	192.168.1.200	54.239.18.66	TLSv1.2	146 Application Data
46	18:29:33.982208179	192.168.1.200	54.239.18.66	TLSv1.2	209 Application Data
48	18:29:33.982208284	192.168.1.200	54.239.18.66	TCP	1514 56316 → 443 [ACK]
52	18:29:33.983799664	192.168.1.200	54.239.18.66	TLSv1.2	1514 Application Data,
59	18:29:33.992532819	192.168.1.200	54.239.18.66	TLSv1.2	3321 Application Data,
66	18:29:34.013657735	54.239.18.66	192.168.1.200	TLSv1.2	108 Application Data

(II) delayed for 1.5 seconds and released

618	17:25:51.043178	192.168.1.200	52.46.158.181	TLSv1.2	348 Application Data
620	17:25:51.043501	192.168.1.200	52.46.158.181	TLSv1.2	202 Application Data
623	17:25:51.043995	192.168.1.200	52.46.158.181	TLSv1.2	477 Application Data
625	17:25:51.043995	192.168.1.200	52.46.158.181	TLSv1.2	146 Application Data
626	17:25:51.043995	192.168.1.200	52.46.158.181	TLSv1.2	209 Application Data
826	17:25:52.607958	52.46.158.181	192.168.1.200	TLSv1.2	108 Application Data
827	17:25:52.627486	192.168.1.200	52.46.158.181	TLSv1.2	184 Application Data
829	17:25:52.631281	192.168.1.200	52.46.158.181	TLSv1.2	184 Application Data
831	17:25:52.652357	192.168.1.200	52.46.158.181	TLSv1.2	184 Application Data

(III) delayed for 1.5 seconds and discarded

801	18:45:31.142286402	192.168.1.200	52.94.235.50	TLSv1.2	348 Application Data
803	18:45:31.142454110	192.168.1.200	52.94.235.50	TLSv1.2	202 Application Data
805	18:45:31.143036672	192.168.1.200	52.94.235.50	TLSv1.2	8754 Application Data
807	18:45:31.143036973	192.168.1.200	52.94.235.50	TLSv1.2	254 Application Data
1007	18:45:32.684398245	52.94.235.50	192.168.1.200	TLSv1.2	97 Encrypted Alert
1008	18:45:32.693698348	192.168.1.200	52.94.235.50	TCP	66 55818 → 443 [FIN, ACK]
1009	18:45:32.694620139	52.94.235.50	192.168.1.200	TCP	66 443 → 55818 [FIN, ACK]
1010	18:45:32.697262793	192.168.1.200	52.94.235.50	TCP	66 55818 → 443 [ACK] Seq=

Fig. 4: Three different cases of voice command traffic. "192.168.1.200" is the IP address of the Echo Dot. The other is the IP address of the Amazon AVS server.

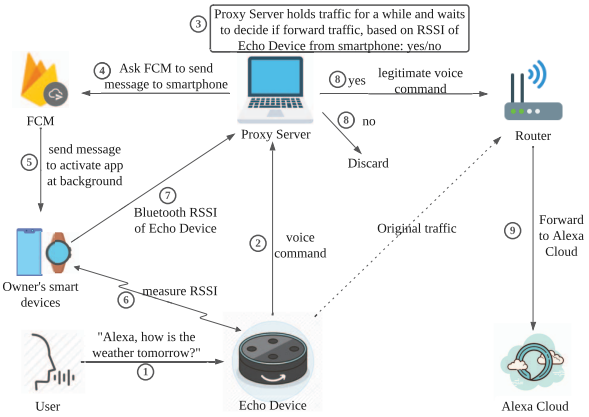


Fig. 5: The workflow of Bluetooth RSSI based method.

between the smart speaker and the home WiFi router. On receiving a TCP connection request from the smart speaker to the cloud server, the proxy accepts it and then launches another TCP connection with the cloud server. After both two TCP connections are established, the proxy forwards TCP payloads between them so that the proxy is completely transparent to upper layer network protocols. Every time a spike of packets is recognized as carrying voice commands, the Traffic Handler caches them in a queue and waits for the response from the Decision Module. During the waiting period, received TCP segments and keep-alive probes are acknowledged by the proxy to prevent the connection timeout.

Figure 4 presents the details of the Traffic Handler handling the voice command traffic of the Echo Dot in three different cases, where the numbers in the first column are the packet #. Case (I) is the traffic of a voice command without using the proxy. Case (I) sub-figure shows that: The first packet sent out from the Echo Dot is packet #36. Without holding the traffic, the Echo Dot receives a response (packet #66) from the cloud server in less than 0.04 seconds. In case (II), a proxy is used to hold and then release the packets of a voice command. The first five packets starting from #618 are recognized as carrying a voice command and cached in a queue for 1.5 seconds. The response from the server (packet #826) arrives 1.546 seconds later, which is right after the release of the held packets. The 1.5 seconds holding period gives the Decision Module enough time to determine if the voice command is legitimate (details in Section IV-C). In case (III), the packets are held but then discarded, rather than released to the cloud server. The TLS session between the Echo Dot and the cloud is closed (packet #1007) due to the unmatched TLS record sequence number.

C. The Decision Module

The Decision Module is designed to have a flexible framework that can utilize various methods to check the legitimacy of a voice command. In the current design, the Decision Module uses information from the smart speaker and the owner's smartphone or smartwatch to determine the legitimacy.

We understand that RSSI values are not very robust. However, based on our measurements (Figure 8), RSSI values of locations in a different room are quite different from those in the same room, which is sufficient for our purpose. Bluetooth RSSI based method utilizes the Bluetooth communication capability that is built into most smart speakers on the market. The owner's smartphone or smartwatch can estimate the distance from the speaker by measuring the speaker's Bluetooth signal strength (i.e., RSSI). In most cases, a user wears her smartwatch even at home and carries her smartphone or places it nearby. Based on this fact, the estimated distance is a good indicator to determine if the owner is close to the smart speaker. If the owner is far away from the speaker when it receives a voice command, then very likely this is an attack, which provides sufficient reason to block it and alert the owner.

There are many works on Bluetooth RSSI based indoor localization [2], [16], [61], [78]. However, they require extra devices to be anchor nodes or base stations. Our method does not require extra devices to be installed at home. Besides, we are interested in the RSSI between a smartphone/smartwatch and a smart speaker, which reflects the distance between them (not the distance between a device and an anchor node). Unlike works [46], [49], where users must perform specific finger gestures or mouth movements to conduct the identity verification process, the RSSI measurement and decision-making processes in VoiceGuard do not need user involvement, which means the entire process can be finished automatically, even if the smartphone is in owner's pocket all the time.

The workflow of the Bluetooth RSSI based method is illustrated in Figure 5. When the Decision Module is queried, it sends an RSSI measuring request to the owner's smartphone/smartwatch by pushing notification via the Firebase Cloud Messaging (FCM) API (step 4). On receiving the request (step 5), a dedicated mobile application is activated in the background to measure the current RSSI of the smart

TABLE I: Testing results of traffic pattern recognition.

		Predicted		
		Positive	Negative	Total
Actual	Positive	132	2	134
	Negative	0	149	149
	Total	132	151	283

Accuracy: 99.29% Precision: 100% Recall: 98.51%

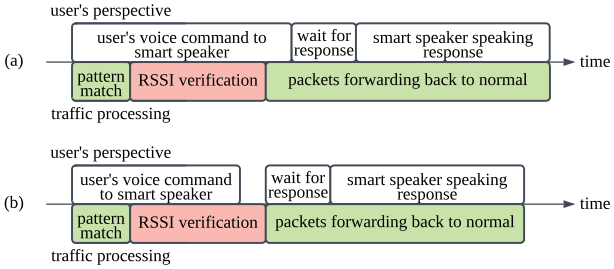


Fig. 6: The timelines of two different cases of delay.

speaker's Bluetooth signal (step 6) and sends the result back to the Decision Module (step 7). Finally, the Decision Module determines if the voice command is legitimate based on the measured RSSI and a threshold learned during the training.

To make it easy for a general user to choose a suitable RSSI threshold, we develop an easy-to-use Bluetooth RSSI threshold app that can be installed on the user's smartphone/smartwatch, whichever the user prefers. The user only needs to switch on the button on the screen and walk around the room (e.g., along the wall) where the smart speaker locates. The app periodically measures the RSSI of the smart speaker (e.g., every 0.5 seconds) and displays measured RSSI values in the middle of the screen. After the walking is done, the user switches off the button and the app calculates the minimum value of all the measured RSSI values as the RSSI threshold and shows it on the screen.

The Bluetooth RSSI based method has good scalability for **multi-user scenarios**. VoiceGuard maintains a list of devices belonging to (multiple) legitimate users of the smart speaker, along with an RSSI threshold for each device. VoiceGuard can ask FCM to push notifications to the group of devices simultaneously so that every device in that list reports RSSI value back to the Decision Module, which then compares each reported RSSI value with the corresponding RSSI threshold. The voice command is considered legitimate only if at least one device reports a larger RSSI value than its threshold, which means that at least one legitimate user is currently near the smart speaker. A new legitimate user can be easily added to the access of the smart speaker by registering her smart device (e.g., smartphone, smartwatch) in VoiceGuard. An attacker cannot register his device on VoiceGuard since the registration on VoiceGuard requires manual approval from the owner (e.g., entering login credentials) to access the device running VoiceGuard.

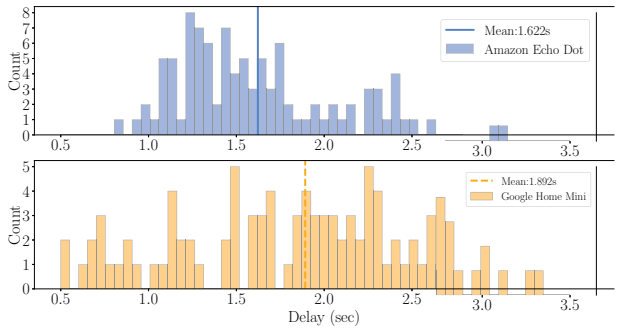


Fig. 7: RSSI query processing time for two smart speakers.

V. Performance Evaluation

We evaluate the performance of VoiceGuard using two popular smart speakers: Amazon Echo Dot and Google Home Mini. We present the case with one speaker as a prototype. VoiceGuard can also handle scenarios with multiple smart speakers: VoiceGuard identifies which smart speaker is being used based on the speaker's unique IP address, and then applies the same strategy as the one-speaker case to protect the smart speaker from unauthorized malicious voice commands.

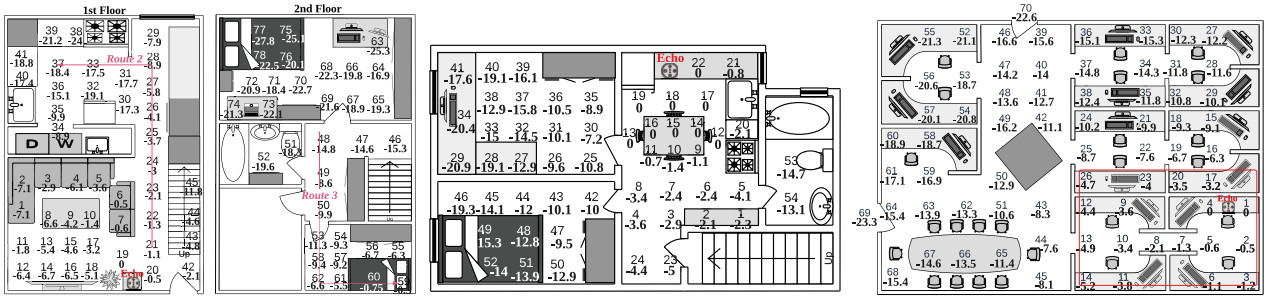
A. Evaluation of Traffic Processing Module

When a smart speaker receives a voice command, the Traffic Processing Module first recognizes voice traffic. Since the Google Home Mini's connection to the backend cloud server is on-demand. its activation can be easily recognized from spikes of its network traffic. In Section V-A1, we evaluate the traffic pattern recognition of the Echo Dot speaker's activation. In Section V-A2, we measure the delay of voice command traffic that is caused by the RSSI query.

1) Traffic Pattern Recognition: In our experiment, we activate the Echo Dot speaker 134 times with randomly generated voice commands, which trigger the actions of the Voice Command Traffic Recognition sub-Module 238 times. Recall that one speaker invocation could result in more than one traffic spike appearing after a no-traffic period, which triggers the recognition method more than once, as illustrated in Figure 3. For each of the spikes, we consider it as positive (i.e., a voice command) if it is in the first phase (see Figure 3). Otherwise, it is classified as negative (not a voice command). The testing results are summarized in Table I. Among the 134 speaker invocations, 132 are recognized correctly, which gives a recall of 98.51%. For the 149 spikes in the second phase, none of them are recognized as voice commands, which means the precision is 100%. The results show that the traffic pattern recognition method has a very good performance.

2) Delay Caused by RSSI query: The Traffic Handler holds the network traffic of a voice command while waiting for the response from the Decision Module. Below we present the measurements of the delays caused by the RSSI query, which shows that the delay has minimal effect on user experience.

Depending on the length of voice commands, delays caused by RSSI queries can be classified as two cases from the users'



(a) Echo Dot in the first testbed. (b) Echo Dot in the second testbed. (c) Echo Dot in the third testbed.

Fig. 8: RSSI measurements of the Echo Dot deployed at the first deployment location in the three testbeds: (a) two-floor house; (b) two-bedroom apartment; (c) office. Pixel 5 is used in the two smart home testbeds, and Samsung Galaxy Watch4 is used in the third testbed. Note that an owner does not need to measure the RSSI values for all locations at home. We develop a mobile app that can measure the RSSI values and determine the threshold for a particular home.

perspective as shown in Figure 6. In each sub-figure, the event sequences above and below the time axis stand for the view of the user and the Traffic Processing Module, respectively.

In case (a), the RSSI query process completes before the user finishes speaking the voice command. Hence, there is no delay in the voice command from the user's perspective. In case (b), the voice command is short and ends before the RSSI verification is done, which causes a short delay from the user's perspective. But the delay to the user is much shorter than the time of the RSSI verification because the RSSI verification is processed while the user is speaking the voice command.

To quantify the impact, we measure the delay of the entire workflow (smart speaker invocation, packet holding, and RSSI query) for 100 voice invocations on both Amazon Echo Dot and Google Home Mini. For all the invocations, the connection is never terminated due to the delay. The distributions of the RSSI verification time are given in Figure 7. For the Echo Dot, although two cases are slightly longer than 3 seconds, the average delay is only 1.622 seconds. 78% of the invocations have a delay of less than 2 seconds. For the Google Home Mini, the average delay is 1.892 seconds.

To determine if the delay affects the user experience, we develop a web crawler and collect 320 commonly used Alexa voice commands on [3], [4], and 443 Google Assistant voice commands on [8], [10], [5]. For the collected Alexa voice commands, the average length is 5.95 words and more than 86.8% of them have at least 4 words. For the collected Google Assistant voice commands, the average length is 7.39 words and more than 93.9% of them have at least 5 words. Considering the normal human speech pace of 2 words per second [1], there are 80% or higher chances that the RSSI query can be finished while the user is speaking the command, which means that in most cases there is no or little impact on the user experience. Even in extreme cases, the query only adds about 1 second of delay, which is very small.

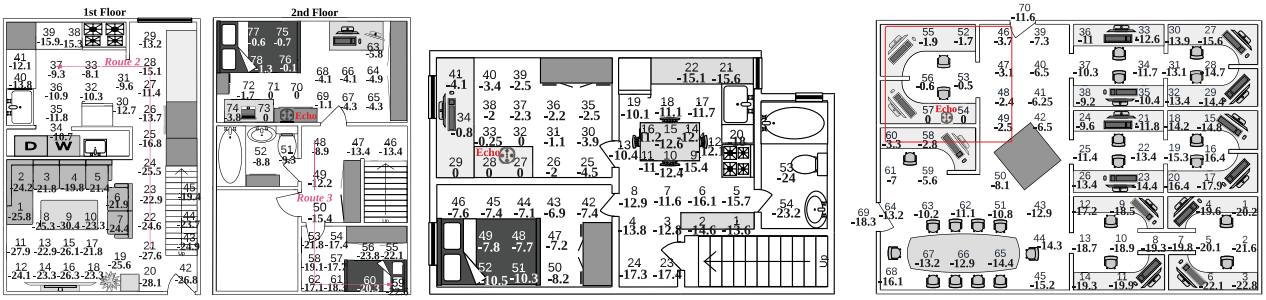
B. Evaluation of the RSSI Based Method

To evaluate the Bluetooth RSSI based method, we conduct extensive experiments in three real-world testbeds, including a two-floor house, a two-bedroom apartment, and a large office. In each testbed, both two smart speakers are evaluated at two different locations. Therefore, we have a total of 12 different evaluation scenarios. Due to the page limit, we only present the results of scenarios in which the Echo Dot was placed at two different locations in each of the three testbeds (Figure 8 and Figure 9). The distribution of RSSI values for the Google Home Mini at each speaker deployment location is similar to that of the Echo Dot, and hence the results are omitted.

We use two smartphones (a Google Pixel 5 and a Google Pixel 4a) that belong to two legitimate smart speaker users, to evaluate multi-user scenarios in the two smart home testbeds. Besides, a smartwatch (a Samsung Galaxy Watch4) is used to evaluate the performance of VoiceGuard when wearable devices (instead of smartphones) are used, which is conducted in the office testbed.

1) The Bluetooth RSSI Threshold: To make decisions based on the value of the smart speaker's Bluetooth RSSI, we need to figure out a threshold of RSSI values to differentiate scenarios of the owner being nearby the speaker and away from the speaker. We measure Bluetooth RSSI values using smartphones at different locations in the two smart home testbeds. Due to space limit, we only present the measurement results from the Pixel 5 when the Echo Dot is deployed at two different locations in each smart home testbed.

Figure 8a shows RSSI measurements at 78 different locations in the first testbed. At each measurement location, we measure the Bluetooth RSSI 4 times for each of the four orientations (up, down, left, right). Thus, we collect a total of 16 RSSI measurements at each location. There are 78 pairs of numbers in Figure 8a. For each pair, the positive integer on top is the location number (1 – 78), and the negative number (or 0) below is the average value of the Echo Dot's Bluetooth RSSI, in unit of dB. Similarly, we measure RSSI values at 54 different locations in the second testbed as shown



(a) Echo Dot in the first testbed. (b) Echo Dot in the second testbed. (c) Echo Dot in the third testbed.

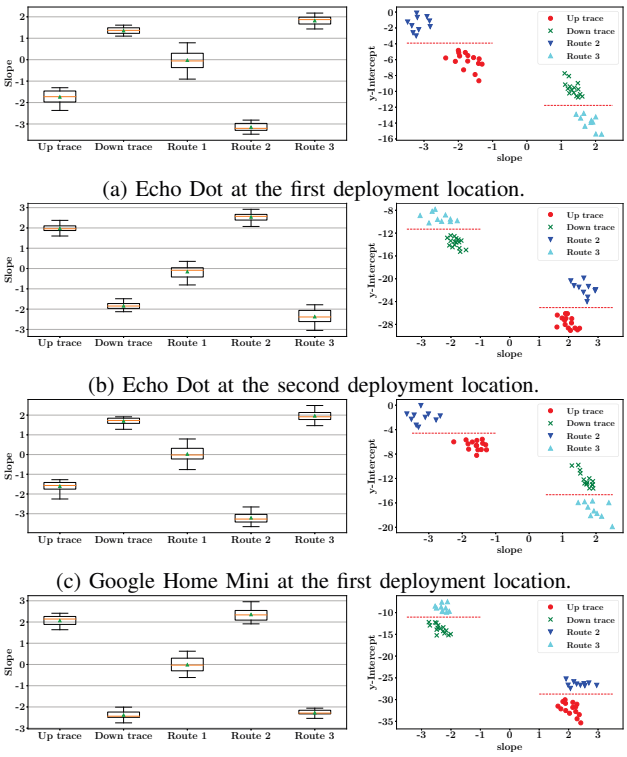
Fig. 9: RSSI measurements of the Echo Dot deployed at the second deployment location in the three testbeds. Pixel 5 is used in the two smart home testbeds to measure the RSSI values, and Samsung Galaxy Watch4 is used in the third testbed.

in Figure 8b. Besides, we use a Samsung Galaxy Watch4 to measure Bluetooth RSSI values at 70 different locations in the office testbed and the results are shown in Figure 8c, which is the case when the Echo Dot is deployed at the first location.

It is noteworthy that VoiceGuard does *not* require users to measure the RSSI values of different locations at home for the RSSI threshold. The purpose of Figure 8 and Figure 9 is to illustrate how Bluetooth RSSI varies as the distance changes, which shows that RSSI values vary a lot in different rooms. As mentioned in Section IV-C, we develop a mobile app for users to easily determine the RSSI threshold. Taking Figure 8a as an example, we can use the app to determine the RSSI threshold for the living room, which is set to -8. As we can see, all the locations within the living room (locations #1 to #24) have RSSI values higher than -8, which confirms that the threshold from the app is correct. Note that locations #25 to #27 have relatively high RSSI (larger than the threshold of -8) because these locations are within the line of sight of the Echo Dot. These locations are considered legitimate places to issue a voice command because the smart speaker's actions (e.g., voice response and light on) can be noticed by the owner if she is at one of these locations. For the first deployment location in the second testbed shown in Figure 8b, the app sets the threshold as -6. For the office testbed in Figure 8c, the red box indicates the legitimate area to issue voice commands to the smart speaker, and the thresholds are set to -6. For the second deployment location in the three testbeds shown in Figure 9, the thresholds are set to -7, -6 and -5, respectively.

2) Determining Floor Level via RSSI Traces: In Figure 8a, although most measurement locations on the 2nd floor have a much lower RSSI than the threshold, there are some exceptions: the room (on top of the smart speaker) in the right bottom corner in Figure 8a. If the owner stays at these locations #55, #56, and #59 to #62, she will be mistakenly considered as being in the speaker's room, which causes false negatives. This issue occurs in the layout with multiple floors, like the first testbed. To deal with it, we use a "floor level" to indicate which floor the owner is currently on. A voice command is always blocked if the owner is on the 2nd floor.

There are many previous works [18], [38], [76] studying recognizing daily activities using accelerometers built inside smartphones, including walking, jogging, going upstairs/downstairs, sitting, and standing. These works involve complicated pre-processing and feature extraction procedures for high accuracy of activity recognition. But we prefer a more



(a) Echo Dot at the first deployment location.

(b) Echo Dot at the second deployment location.

(c) Google Home Mini at the first deployment location.

(d) Google Home Mini at the second deployment location.

Fig. 10: The results are obtained using Pixel 5 to determine Up/Down traces in the first testbed with two floors. The left column shows that the slope of route traces' fitting lines can be used to differentiate Route 1 from the other 4 routes. The right column shows that the rest of 4 routes can be further distinguished with the help of both slope and y-intercept.

TABLE II: Results of the RSSI based methods in the first testbed (two-floor house).

	Echo Dot		Google Home Mini	
Correct / Total	1st Location	2nd Location	1st Location	2nd Location
legitimate (N)	89 / 91	100 / 103	90 / 94	82 / 86
malicious (P)	69 / 69	78 / 78	65 / 65	63 / 63
Accuracy	98.75%	98.34%	97.48%	97.32%
Precision	97.18%	96.30%	94.20%	94.03%
Recall	100%	100%	100%	100%

TABLE III: Results of the RSSI based method in the second testbed (two-bedroom apartment).

	Echo Dot		Google Home Mini	
Correct / Total	1st Location	2nd Location	1st Location	2nd Location
legitimate (N)	75 / 78	86 / 88	76 / 80	93 / 95
malicious (P)	59 / 59	64 / 65	57 / 57	50 / 50
Accuracy	97.81%	98.04%	97.08%	98.62%
Precision	95.16%	96.97%	93.44%	96.15%
Recall	100%	98.46%	100%	100%

TABLE IV: Results of the RSSI based method in the third testbed (office).

	Echo Dot		Google Home Mini	
Correct / Total	1st Location	2nd Location	1st Location	2nd Location
legitimate (N)	82 / 85	91 / 94	89 / 90	89 / 91
malicious (P)	47 / 47	52 / 52	50 / 50	51 / 51
Accuracy	97.73%	97.95%	99.29%	98.59%
Precision	94%	94.55%	98.04%	96.23%
Recall	100%	100%	100%	100%

light-weight approach to distinguish only users' up/down stair activity. In the following, we use the case in Figure 8a for the purpose of illustrating the method used to determine the floor level, which can be also applied to the case when the speaker is deployed at the second location as shown in Figure 9a, as well as cases when using Google Home Mini or Pixel 4a.

To obtain/update the floor level, we need to know when the owner is going upstairs or downstairs. This comprises two tasks: 1) knowing the time when someone is going upstairs or downstairs; 2) identifying whether the person going upstairs/downstairs is the owner. In our experiments, we use a Hue motion sensor [12] near the stairs to do the first task. When the owner goes upstairs, the speaker's Bluetooth RSSI value at the owner's phone becomes smaller and smaller (i.e., from location #42 to location #48). On the other hand, the RSSI value is getting larger when going downstairs. Based on this, an RSSI trace can be recorded whenever the motion sensor turns active, and then we can analyze the RSSI trace and determine if the owner goes upstairs (or downstairs). The above approach accomplishes the second task. Note that the motion sensor is not a must for our system. The usage of the motion sensor is to accommodate extreme cases and improve the performance of VoiceGuard. If the owner already has a motion sensor on/near the stair, the above motion sensor-assisted method can be used. If not, our system still works with a slightly increased false negative rate.

To evaluate the approach, we conduct experiments for a total of eight different cases (two smartphones, two smart speakers, and two different deployment locations). For each case, we collect 15 instances of going upstairs (Up) traces and 15 instances of going downstairs (Down) traces. There may be scenarios where the stairway motion is triggered by a guest and at the same time the owner is moving in a different route which generates a RSSI trace similar to the Up or Down traces. We list two example traces below (Route 2 and Route 3) and discuss how to differentiate them with the Up/Down traces. For Route 1, we collect 5 RSSI traces of random movements in each of the five rooms, i.e., kitchen, living room, restroom, and two bedrooms, and get 25 traces in total. For Route 2 and Route 3, we collect 10 traces for each of route. Then we apply linear regression and convert each trace into a tuple consisting of the fitting line's slope and y-intercept.

- **Route 1:** The owner moves within a room, which only makes the RSSI value fluctuate within a small range.
- **Route 2:** The owner moves from locations #21 to #37, producing RSSI traces similar to Up traces.
- **Route 3:** The owner moves from locations #48 to #59, producing RSSI traces similar to Down traces.

In our experiments, it takes about 8 seconds to walk from location #42 to location #48 or from location #46 to location #21. Based on this, we start recording the RSSI value every 0.2 seconds for 8 seconds when receiving active motion events, which generates a trace of 40 RSSI values. Then, we apply a linear regression to the 40 values and use the slope and the y-intercept of the fitting line as features to compare them with the values of the pre-recorded Up/Down traces.

Here we only present 4 results of the Pixel 5 in Figure 10. The results of the Pixel 4a are similar and consistent with those of the Pixel 5, which are omitted due to space limit.

Sub-figures on the left side show that the Up trace, Down trace, and Route 1 trace can be easily distinguished by their ranges of slopes, i.e., all Route 1 traces' slopes are between -1 and 1, while other routes' slopes are either smaller than -1 or larger than 1. However, traces of Routes 2 and 3 have ranges of slopes that overlap with those of Up and Down traces. To distinguish traces of Routes 2 (and 3) from Up (and Down) traces, we use the y-intercepts of the traces. The joint distribution of slope and y-intercept is given on the right column of Figure 10. For each case, Route 2 (and 3) traces can be easily separated from Up (and Down) traces, respectively. Based on these observations, we differentiate traces of different cases using the method that first checks the slope of a trace's fitting line and classifies it into three categories ($<$-1, (-1, 1), $>$1). If the slope is larger than 1 or smaller than -1, it further uses y-intercepts to distinguish them.

To sum up, when detecting motions, the Decision Module decides if the floor level needs to be updated, based on the RSSI trace fitting lines' slopes and y-intercepts. For instance, if the smart speaker is on the first floor and the current floor level is 'upstairs', the Decision Model blocks a voice command even if the measured RSSI is higher than the threshold of -8 dB.

3) Performance of the RSSI Based Method: To evaluate the RSSI based method, for each deployment location in the three testbeds, we conduct 7 days' real-world experiments using the Echo Dot and the Google Home Mini. During the experiments in the two smart home testbeds, two persons carrying the Pixel 5 and the Pixel 4a act as the legitimate users (owners) of the smart speaker, and another person acts as a malicious guest, who attempts to issue pre-recorded voice commands when the owners are not near the smart speaker. Malicious commands are issued only if no owner is in the room where the smart speaker locates. Two owners could be at any locations outside this specific room, or even outside the house. For the experiments in the office testbed, one person wearing the Galaxy Watch4 acts as a legitimate user and another person acts as a malicious user to issue pre-recorded voice commands when the legitimate user is not near the smart speaker.

Table II, Table III and Table IV list the experimental results of four different cases in the three testbeds, respectively. Taking "the Echo Dot at the 1st location" in Table II as an example, a total of 160 voice commands are issued during the 7-day experiment, and all are successfully recognized by the Traffic Processing Module. There are 91 voice commands issued by the owners (Negative) and the other 69 are issued by the malicious guest (Positive). According to the decisions made by VoiceGuard, all 69 malicious voice commands are classified as illegitimate, and 89 out of 91 voice commands issued by the owners are determined to be legitimate, i.e., only two of them are mistakenly classified. We calculate accuracy, precision and recall to evaluate the performance. The results show that the RSSI based method is very effective and reliable, with a recall of almost 100% and overall accuracy higher than 97%. Results in the three testbeds also imply VoiceGuard can be used in homes and offices with different layouts.

VI. Related Work

A. Countermeasures for Audio AEs

A series of defense schemes have been proposed to deal with audio AEs, which can be roughly classified into two approaches [41]. The first approach, as first proposed in [70], tries to enhance existing speech recognition models by training them with adversarial audio samples, making them capable of resisting AEs that are generated using the same AE generation method. But this approach fails to detect AEs that are generated by new AE generation methods. The second approach explores methods to differentiate adversarial audio samples from benign samples by applying audio preprocessing techniques [29], [55], [68], [82], [86]. The observation is that, imperceptible perturbations introduced by AE generation methods can be destroyed by audio signal processing methods, such as adding noise or downsampling used in [86], while benign samples are much more robust. After processing, audio AEs can be identified because they have different text transcripts before and after the transformation.

B. Protections for Voice-enabled Systems

Voice features extracted from voice commands have been applied on voice-enabled systems to authenticate users' identities [21], [30], [35], [40], [45], [50], [57]. However, these methods are vulnerable to replay attacks and impersonation attacks, where the user's voice is either pre-recorded or synthesized. Increasing noise and reverberation by recording and replaying is used in [74], [79] to differentiate a replay audio from a live human speech, but fail to have satisfying performance in practice. Advanced speaker recognition models, such as Gaussian Mixture Model and i-vector models [17], [39], have been proposed to defend voice authentication systems. Motivated by a series of ASVspoof challenges [44], [71], [80], several deep learning based methods [25], [75] are proposed to detect spoofing attacks. Some other works detect replay attacks by analyzing audio data, such as identifying sub-bass over-excitation phenomenon in [20] and extracting spectral power patterns in Void [15]. But our work uses side-channel information (e.g., owner locations), instead of analyzing audio signals. Besides, Void [15] requires a powerful server to perform machine learning tasks to achieve low latency and high accuracy, which greatly increases the cost of deployment, while VoiceGuard does not require powerful servers, and a general-purpose computing device is sufficient to have high performance. Furthermore, the working distance of VoiceGuard could easily exceed that of Void (260 centimeters).

C. Comparison with Liveness Detection

More recent works propose liveness detection methods to verify whether an audio clip is issued by a real person or replayed by a speaker. However, as mentioned in [14], current liveness detection methods employ machine learning models as part of audio processing pipeline, which adds another layer of vulnerability. An adaptive attacker can evade such detection with knowledge of the detection method. Furthermore, most of these works require a very small distance between the source and the target, which severely limits their applicable scenarios.

Many works [26], [64], [83], [89], [90] propose different approaches to detect liveness on smartphones. CaField [83] utilizes the physical field of acoustic energy. VoiceGesture [89] leverages the unique articulatory gesture of the user when speaking a passphrase. VoiceLive [90] uses time-difference-of-arrival changes to the two microphones of a phone. These approaches are designed for smartphones and require smartphones to be held and placed close to the user's mouth or loudspeaker, and hence do not work on smart speakers, which allow users to issue voice commands from meters away.

VoiceGuard is designed for smart speakers. Although VoiceGuard uses smartphones to help verify the legitimacy of voice commands, it does not require users to hold smartphones. Otherwise, users can directly use voice assistants on smartphones.

Some other works [33], [47], [49], [65], [77] attempt to apply speech liveness detection systems on smart speakers by using extra sensors. WiVo [49] utilizes the correlation between the mouth movement and the changes of the WiFi signals to detect liveness. But it only works when the user's mouth

is very close to the WiFi antenna (e.g., 20 cm). VAuth [33] uses an accelerometer that is embedded in some wearable devices to capture the user's body-surface vibrations, which are then used to verify the source of the voice command. However, VAuth needs an external Bluetooth transmitter, and it also requires the user to wear a dedicated device with an accelerometer. Speaker-Sonar [47] emits an inaudible sound and tracks the user's direction to compare it with the direction of the received voice command. However, this approach is sensitive to ambient audio noise. Air pressure sensors inside in-ear headphones are used in [65] to capture the noisy air pressure data caused by the mouth movement and then compare it with the voices to validate the liveness of the voice source. However, even the most comfortable earbuds can cause fatigue after being used for a long period of time. WearID [77] collects motion sensor data from the one inside an off-the-shelf wearable device and checks if the command sound is from a legitimate user. This approach requires the user to perform additional movements and put the wearable device close to the mouth when speaking the voice command. A dedicated obfuscator is deployed in [36] to constantly play a specially designed ultrasound signal to jam smart speakers, and it stops jamming if an authorized user is detected. But this approach requires a dedicated obfuscator that is equipped with an ultrasound speaker and an ultrasound microphone. Besides, the jamming signal emitted by the obfuscator may affect the normal use of the smartphone, such as making phone calls.

In contrast, VoiceGuard does not require the user to carry any extra sensors or devices, except the user's smartphone or smartwatch. Most people wear their smartwatches and carry their smartphones with them almost all the time. A legitimate user just talks to the smart speaker in the usual way, without the need to perform any specific gestures or mouth movements. Furthermore, VoiceGuard can not only detect malicious voice commands but also block them, which most of the aforementioned works cannot do.

VII. Discussion

In this section, we discuss the limitations of VoiceGuard and some future works that could improve the performance of VoiceGuard.

Non-applicable Scenarios: In the VoiceGuard design, it is assumed that a user has her smartphone with/near her or has a smart wearable device (e.g., a smartwatch) while talking to a smart speaker. Most of the time, the above is the case. In some rare cases, a malicious voice command attack could happen, e.g., when the following three conditions hold simultaneously: 1) the owner's smartphone is placed near the smart speaker for charging; 2) the owner is not near the smartphone; 3) an attacker is near the smartphone. However, the chance of all three conditions are true is very small.

Extensible Framework: The Decision Module of VoiceGuard is responsible for determining the legitimacy of voice commands. Although it currently only implements the Bluetooth RSSI-based method, it has an open and extensible framework so that other approaches such as user identification methods [42], [51], [81], [85] can be easily integrated into the Decision Module to improve the performance of VoiceGuard.

Potential Changes of Traffic Signature: Although VoiceGuard uses seldom-changed packet-level signatures (which have remained the same for over two years based on our experiments) to identify cloud servers with frequently changed IPs, there is still a small possibility that the packet-level signatures may change in future updates. To cope with this issue, in our future work, we plan to revise the Traffic Processing Module so that it can adaptively learn the packet-level signatures when they change.

VIII. Conclusion

In this work, we presented VoiceGuard, an effective and practical smart speaker protection scheme with two modules. The Traffic Processing Module can block malicious voice commands without affecting the user experience. The Decision Module checks the legitimacy of a voice command based on the RSSI value of a speaker's Bluetooth signal. VoiceGuard works transparently with the operation of smart speakers and does not require any modifications on smart speakers' hardware or software, nor on the backend cloud servers of smart speakers. VoiceGuard works well as long as a user has a smart wearable device (e.g., a smartwatch) or a smartphone with her, which is true in most cases. It also works well when the user's smartwatch or smartphone is near her. VoiceGuard does not require a user to carry any special/dedicated devices. We implemented VoiceGuard using two popular smart speakers: the Amazon Echo Dot and the Google Home Mini. We conducted comprehensive experiments in three real-world testbeds, including two smart homes and one office. Our experimental results show that VoiceGuard is very effective: it has an accuracy of more than 97% in differentiating legitimate from malicious voice commands, and a recall of almost 100% in blocking malicious voice commands. Furthermore, VoiceGuard can be easily applied to multi-user scenarios and still deliver great performance.

Acknowledgment

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-2204785 and CNS-2205868.

References

[1] "How fast does the average person speak?" 2016, https://wordcounter.net/blog/2016/06/02/101702_how-fast-average-person-speaks.html

[2] "Monitor: Reliable, multi-user, distributed bluetooth occupancy/presence detection," 2018, https://community.home-assistant.io/t/monitor-reliable-multi-user-distributed-bluetooth-occupancy-presence-detection/68505.

[3] "A comprehensive list of alexa voice commands you can use with your smart devices," 2020, https://www.businessinsider.com/alexa-voice-commands.

[4] "Every alexa command you can give your amazon echo smart speaker or display," 2021, https://www.cnet.com/home/smart-home/every-alexa-command-you-can-give-your-amazon-echo-smart-speaker-or-display/.

[5] "How to use google assistant, plus all the 'ok, google' commands you need to know," 2021, https://www.digitaltrends.com/mobile/ok-google-voice-commands-list/https://www.digitaltrends.com/mobile/ok-google-voice-commands-list/.

[6] "Smart speakers - statistics & facts," 2021, https://www.statista.com/topics/4748/smart-speakers/.

[7] "Smart speakers global market report 2021: Covid-19 growth and change to 2030," 2021, https://www.reportlinker.com/p06064491/Smart-Speakers-Global-Market-Report-COVID-19-Growth-And-Change-To.html?utm_source=GNW.

[8] "101 google assistant commands: The best things to ask your google home," 2022, https://www.the-ambient.com/guides/best-google-assistant-commands-382.

[9] "Alexa skills kit," 2022, https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html.

[10] "Google assistant commands: Here are the ones you need to know," 2022, https://www.androidauthority.com/google-assistant-commands-727911/.

[11] "Google assistant intents," 2022, https://developers.google.com/assistant/conversational/intents.

[12] "Hue motion sensor," 2022, https://www.philips-hue.com/en-us/p/hue-motion-sensor/046677570972.

[13] "Wireshark," 2022, https://www.wireshark.org/.

[14] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor, "Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems," *arXiv preprint arXiv:2007.06622*, 2020.

[15] M. E. Ahmed, I.-Y. Kwak, J. H. Huh, I. Kim, T. Oh, and H. Kim, "Void: A fast and light voice liveness detection system," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2685–2702.

[16] M. Altini, D. Brunelli, E. Farella, and L. Benini, "Bluetooth indoor localization with multiple neural networks," in *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*. IEEE, 2010, pp. 295–300.

[17] T. B. Amin, J. S. German, and P. Marziliano, "Detecting voice disguise from speech variability: Analysis of three glottal and vocal tract measures," in *Proceedings of Meetings on Acoustics 166ASA*, vol. 20, no. 1. Acoustical Society of America, 2013, p. 060005.

[18] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *Esann*, vol. 3, 2013, p. 3.

[19] P. Biswal and O. Gnawali, "Does quic make the web faster?" in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[20] L. Blue, L. Vargas, and P. Traynor, "Hello, is it me you're looking for? differentiating between human and electronic speakers for voice interface security," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018, pp. 123–133.

[21] J. P. Campbell, "Speaker recognition: A tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, 1997.

[22] D. Caputo, L. Verderame, A. Merlo, A. Ranieri, and L. Caviglione, "Are you (google) home? detecting users' presence through traffic analysis of smart speakers." in *ITASEC*, 2020, pp. 105–118.

[23] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 513–530.

[24] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[25] N. Chen, Y. Qian, H. Dinkel, B. Chen, and K. Yu, "Robust deep feature for spoofing detection—the sjtu system for asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[26] S. Chen, K. Ren, S. Piao, C. Wang, Q. Wang, J. Weng, L. Su, and A. Mohaisen, "You can hear but you cannot steal: Defending against voice impersonation attacks on smartphones," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 183–195.

[27] Y. Chen, X. Yuan, J. Zhang, Y. Zhao, S. Zhang, K. Chen, and X. Wang, "Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[28] H. Chi, C. Fu, Q. Zeng, and X. Du, "Delay wreaks havoc on your smart home," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[29] N. Das, M. Shanbhogue, S.-T. Chen, L. Chen, M. E. Kounavis, and D. H. Chau, "Adagio: Interactive experimentation with adversarial attack and defense for audio," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 677–681.

[30] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[31] P. L. De Leon, M. Pucher, and J. Yamagishi, "Evaluation of the vulnerability of speaker verification to synthetic speech," 2010.

[32] T. Du, S. Ji, J. Li, Q. Gu, T. Wang, and R. Beyah, "Sirenattack: Generating adversarial audio for end-to-end acoustic systems," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 357–369.

[33] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 343–355.

[34] C. Fu, Q. Zeng, H. Chi, X. Du, and S. L. Valluru, "Iot phantom-delay attacks: Demystifying and exploiting iot timeout behaviors," in *2022 52st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2022.

[35] B. Gajic and K. K. Paliwal, "Robust speech recognition in noisy environments based on subband spectral centroid histograms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 2, pp. 600–608, 2006.

[36] C. Gao, V. Chandrasekaran, K. Fawaz, and S. Banerjee, "Traversing the quagmire that is privacy in your smart home," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, 2018, pp. 22–28.

[37] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[38] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Generation Computer Systems*, vol. 81, pp. 307–313, 2018.

[39] R. G. Hautamäki, T. Kinnunen, V. Hautamäki, T. Leino, and A.-M. Laukkanen, "I-vectors meet imitators: on vulnerability of speaker verification systems against voice mimicry." in *Interspeech*, 2013, pp. 930–934.

[40] M. Hébert, "Text-dependent speaker recognition," in *Springer handbook of speech processing*. Springer, 2008, pp. 743–762.

[41] S. Hu, X. Shang, Z. Qin, M. Li, Q. Wang, and C. Wang, "Adversarial examples for automatic speech recognition: Attacks and countermeasures," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 120–126, 2019.

[42] A. Huang, D. Wang, R. Zhao, and Q. Zhang, "Au-id: Automatic user identification and authentication through the motions captured from sequential human activities using rfid," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, pp. 1–26, 2019.

[43] S. Khare, R. Aralikatte, and S. Mani, "Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization," *arXiv preprint arXiv:1811.01312*, 2018.

[44] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The asvspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," 2017.

[45] T. Kinnunen, B. Zhang, J. Zhu, and Y. Wang, "Speaker verification with adaptive spectral subband centroids," in *International Conference on Biometrics*. Springer, 2007, pp. 58–66.

[46] H. Kong, L. Lu, J. Yu, Y. Chen, L. Kong, and M. Li, "Fingerpass: Finger gesture-based continuous user authentication for smart homes using commodity wifi," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 201–210.

[47] Y. Lee, Y. Zhao, J. Zeng, K. Lee, N. Zhang, F. H. Shezan, Y. Tian, K. Chen, and X. Wang, "Using sonar for liveness detection to protect smart speakers against remote attackers," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–28, 2020.

[48] J. Lindberg and M. Blomberg, "Vulnerability in speaker verification-a study of technical impostor techniques," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[49] Y. Meng, Z. Wang, W. Zhang, P. Wu, H. Zhu, X. Liang, and Y. Liu, "Wivo: Enhancing the security of voice control system via wireless signal in iot environment," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 81–90.

[50] K. K. Paliwal, "Spectral subband centroid features for speech recognition," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 2. IEEE, 1998, pp. 617–620.

[51] S. Pan, N. Wang, Y. Qian, I. Velibeyoglu, H. Y. Noh, and P. Zhang, "Indoor person identification through footstep induced structural vibration," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 81–86.

[52] M. J. Park and J. I. James, "Preliminary study of a google home mini," *arXiv preprint arXiv:2001.04574*, 2020.

[53] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[54] Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5231–5240.

[55] K. Rajaratnam, K. Shah, and J. Kalita, "Isolated and ensemble audio preprocessing methods for detecting adversarial examples against automatic speech recognition," *arXiv preprint arXiv:1809.04397*, 2018.

[56] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun, "Proximity-based access control for implantable medical devices," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 410–419.

[57] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[58] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, "Rfid guardian: A battery-powered mobile device for rfid privacy management," in *Australasian Conference on Information Security and Privacy*. Springer, 2005, pp. 184–194.

[59] N. Roy, H. Hassanieh, and R. Roy Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017, pp. 2–14.

[60] N. Roy, S. Shen, H. Hassanieh, and R. R. Choudhury, "Inaudible voice commands: The long-range attack and defense," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 547–560.

[61] S. Sadowski and P. Spachos, "Rssi-based indoor localization with the internet of things," *IEEE Access*, vol. 6, pp. 30 149–30 161, 2018.

[62] L. Schönherr, T. Eisenhofer, S. Zeiler, T. Holz, and D. Kolossa, "Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems," in *Annual Computer Security Applications Conference*, 2020, pp. 843–855.

[63] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, "Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding," *arXiv preprint arXiv:1808.05665*, 2018.

[64] J. Shang, S. Chen, and J. Wu, "Defending against voice spoofing: A robust software-based liveness detection system," in *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2018, pp. 28–36.

[65] J. Shang and J. Wu, "Voice liveness detection for voice assistants using ear canal pressure," in *Proc. of the 17th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS 2020)*, 2020.

[66] W. Shang and M. Stevenson, "Score normalization in playback attack detection," in *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2010, pp. 1678–1681.

[67] J. Shen, P. Nguyen, Y. Wu, Z. Chen, M. X. Chen, Y. Jia, A. Kannan, T. Sainath, Y. Cao, C.-C. Chiu *et al.*, "Lingvo: a modular and scalable framework for sequence-to-sequence modeling," *arXiv preprint arXiv:1902.08295*, 2019.

[68] V. Subramanian, E. Benetos, and M. B. Sandler, "Robustness of adversarial attacks in sound event classification," 2019.

[69] T. Sugawara, B. Cyr, S. Rampazzi, D. Genkin, and K. Fu, "Light commands: laser-based audio injection attacks on voice-controllable systems," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2631–2648.

[70] S. Sun, C.-F. Yeh, M. Ostendorf, M.-Y. Hwang, and L. Xie, "Training augmentation with adversarial examples for robust speech recognition," *arXiv preprint arXiv:1806.02782*, 2018.

[71] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "Asvspoof 2019: Future horizons in spoofed and fake audio detection," *arXiv preprint arXiv:1904.05441*, 2019.

[72] R. Togneri and D. Pullella, "An overview of speaker identification: Accuracy and robustness issues," *IEEE circuits and systems magazine*, vol. 11, no. 2, pp. 23–61, 2011.

[73] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," in *Network and Distributed Systems Security (NDSS) Symposium*, vol. 2020, 2020.

[74] J. Villalba and E. Lleida, "Detecting replay attacks from far-field recordings on speaker verification systems," in *European Workshop on Biometrics and Identity Management*. Springer, 2011, pp. 274–285.

[75] J. Villalba, A. Miguel, A. Ortega, and E. Lleida, "Spoofing detection with dnn and one-class svm for the asvspoof 2015 challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[76] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, "Deep learning models for real-time human activity recognition with smartphones," *Mobile Networks and Applications*, vol. 25, no. 2, pp. 743–755, 2020.

[77] C. Wang, C. Shi, Y. Chen, Y. Wang, and N. Saxena, "Wearid: Wearable-assisted low-effort authentication to voice assistants using cross-domain speech similarity," *arXiv preprint arXiv:2003.09083*, 2020.

[78] Y. Wang, Q. Ye, J. Cheng, and L. Wang, "Rssi-based bluetooth indoor localization," in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 2015, pp. 165–171.

[79] Z.-F. Wang, G. Wei, and Q.-H. He, "Channel pattern noise based playback attack detection algorithm for speaker recognition," in *2011 International conference on machine learning and cybernetics*, vol. 4. IEEE, 2011, pp. 1708–1713.

[80] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[81] T. Xin, B. Guo, Z. Wang, M. Li, Z. Yu, and X. Zhou, "Freesense: Indoor human identification with wi-fi signals," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–7.

[82] H. Yakura and J. Sakuma, "Robust audio adversarial example for a physical attack," *arXiv preprint arXiv:1810.11793*, 2018.

[83] C. Yan, Y. Long, X. Ji, and W. Xu, "The catcher in the field: A fieldprint based spoofing detection for text-independent speaker verification," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1215–1229.

[84] Q. Yan, K. Liu, Q. Zhou, H. Guo, and N. Zhang, "Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves," in *Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[85] X. Yang, J. Liu, Y. Chen, X. Guo, and Y. Xie, "Mu-id: Multi-user identification through gaits using millimeter wave radios," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2589–2598.

[86] X. Yuan, Y. Chen, Y. Zhao, Y. Long, X. Liu, K. Chen, S. Zhang, H. Huang, X. Wang, and C. A. Gunter, "Commandersong: A systematic approach for practical adversarial voice recognition," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 49–64.

[87] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 103–117.

[88] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based iot device authentication," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[89] L. Zhang, S. Tan, and J. Yang, "Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 57–71.

[90] L. Zhang, S. Tan, J. Yang, and Y. Chen, "Voicelive: A phoneme localization based liveness detection for voice authentication on smartphones,"

in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1080–1091.