2MA: Verifying Voice Commands via Two Microphone Authentication

Logan Blue
University of Florida
Gainesville, Florida
bluel@ufl.edu

Hadi Abdullah
University of Florida
Gainesville, Florida
hadi10102@ufl.edu

Luis Vargas
University of Florida
Gainesville, Florida
lfvargas14@ufl.edu

Patrick Traynor
University of Florida
Gainesville, Florida
traynor@ufl.edu

ABSTRACT

Voice controlled interfaces have vastly improved the usability of many devices (e.g., headless IoT systems). Unfortunately, the lack of authentication for these interfaces has also introduced command injection vulnerabilities - whether via compromised IoT devices, television ads or simply malicious nearby neighbors, causing such devices to perform unauthenticated sensitive commands is relatively easy. We address these weaknesses with Two Microphone Authentication (2MA), which takes advantage of the presence of multiple ambient and personal devices operating in the same area. We develop an embodiment of 2MA that combines approximate localization through Direction of Arrival (DOA) techniques with Robust Audio Hashes (RSHs). Our results show that our 2MA system can localize a source to within a narrow physical cone ($< 30^\circ$) with zero false positives, eliminate replay attacks and prevent the injection of inaudible/hidden commands. As such, we dramatically increase the difficulty for an adversary to carry out such attacks and demonstrate that 2MA is an effective means of authenticating and localizing voice commands.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**; *Access control*;

KEYWORDS

Internet of Things, authentication

ACM Reference Format:
Logan Blue, Hadi Abdullah, Luis Vargas, and Patrick Traynor. 2018. 2MA: Verifying Voice Commands via Two Microphone Authentication. In *ASIA CCS '18: 2018 ACM Asia Conference on Computer and Communications Security, June 4–8, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3196494.3196545

ASIA CCS '18, June 4–8, 2018, Incheon, Republic of Korea

ACM ISBN 978-1-4503-5576-6/18/06…$15.00
https://doi.org/10.1145/3196494.3196545

1 INTRODUCTION

One of the many promises of the Internet of Things (IoT) is convenience. From devices that automatically close curtains at sundown to connected door locks, IoT sensors, actuators and systems are expected to be deployed in virtually every environment in the coming decade. Because such devices often have extremely limited or simply lack traditional user interfaces, an increasing number are opting to integrate voice commands as their primary user interface. Voice interfaces are also widely lauded as a means of making computing inclusive by allowing young, those with disabilities, and the elderly alike to successfully interact with enabled systems [35].

Unfortunately, voice interfaces suffer from a number of security problems. First, most systems rely on ensuring that adversaries can not be within physical proximity of devices implementing such interfaces. Such devices need not be compromised by an attacker; recent news provides examples of nearby televisions and radios used to activate home assistant devices [3, 9]. This assumption renders systems vulnerable to any malicious neighbor (e.g., closely situated people in adjacent apartments, compromised IoT stereo, television playing commercials) and is unrealistic given the number of vulnerable IoT devices expected to be deployed in homes and business in the coming years. Second, multiple researchers have recently demonstrated that adversaries can inject commands without nearby users hearing them [20, 44, 45], thereby circumventing their ability to cancel such requests. Finally, even those systems that attempt to use biometrics do little to prevent replay attacks. Given the ease with which previously played audio can be subtly modified [36, 41], adversaries can easily generate previously unplayed commands in the absence of the user. Given that voice commands can cause the execution of sensitive operations (e.g., the purchase of goods; the actuation of physical systems including heating/air conditioning, door locks and window shades; perform online banking), these systems require stronger protections.

In this paper, we present Two Microphone Authentication (2MA). 2MA systems take advantage of the fact that multiple (at least two) microphones are likely to be present in settings where users deploy systems with voice interfaces. We aim to retain the utility of such systems while explicitly eliminating attacks from nearby sources, replayed legitimate requests, and hidden commands. We make the following specific contributions to achieve these goals:

- **Develop Two Microphone Authentication:** Applications ranging from reducing ambient noise to identifying the trajectory of bullets use unified arrays of microphones. To our knowledge, ours is the first work to propose the use of multiple microphones located on potentially mutually disinterested devices throughout an area (e.g., an apartment) to provide a stronger means of authenticating voice commands.
- **Design and Implement First Instance of 2MA:** 2MA systems can take many forms. We design a two-channel protocol (audio and network) and then implement a system using microphones located on an Amazon Echo/Google Home-like device and a mobile phone held by an authorized user. We demonstrate the ability to eliminate voice commands made when the user is not present and those injected from places outside of a narrow audio cone ($< 30^{\circ}$) with zero false positives.
- **Defend Against Inaudible Attacks:** We customize mechanisms including the Robust Sound Hash (RSH) and perform extensive tests to ensure that the recent collection of inaudible attacks [20, 44, 45][1] fail to inject sensitive commands without detection.

We spend significant time talking about our specific embodiment of a 2MA system; however, readers should see the idea of 2MA as a generic framework and by itself a contribution. It is our hope that readers build applications appropriate for other specific contexts (as we have done) in the future.

The remainder of the paper is organized as follows: Section 2 discusses related work; Section 3 provides background information on our underlying mechanisms; Section 4 specifies our security model and adversarial capabilities; Section 5 defines our 2MA protocol; Section 6 details the architecture of our system; Section 7 shows our experimental performance of our 2MA system, including against adversarial audio; Section 8 offers discussion on a number of important considerations; and Section 9 provides concluding remarks.

2 RELATED WORK

In recent years, speech has become a common command interface for many devices [6, 7, 10–12]. While convenient, this interface does not provide any means of authenticating an arbitrary speaker. This allows anyone within the device's audible proximity to issue commands. In April of 2017, this lack of authentication allowed Burger King to release a television advertisement that triggered nearby Google assistant enabled devices to read a promotional message posted online [9]. Similarly, a TV host in the UK caused Amazon Echo devices to purchase a children's dollhouse simply by reading text on the news [3]. The effects of these specific instances were eventually reversed, but only after millions of devices initiated sensitive operations. Using a similar attack, a malicious entity can unlock doors [8], make purchases [6, 13], or transfer money thousands of times before being stopped [15, 25]. This problem is compounded by the increasing number of IoT devices. A comprised device with a built-in speaker can be used by an adversary to play malicious audio commands. This is a realistic scenario given the recent discovery of the Mirai Botnet [32], which comprised millions of IoT devices. Audio commands can also be concealed so that they are imperceptible even if the victim is in the vicinity. These morphing techniques exploit weaknesses in speech recognition models employed by voice operated devices. The audio commands can be modified to sound like nonsensical audio [20, 44] or can be made completely inaudible [45] to the unsuspecting victim, while simultaneously being registered as legitimate commands by the voice operated device.

Many researchers have looked to incorporate speaker recognition into voice operated devices. This approach might ensure that such devices only accept commands from the their real owners. However, this approach has several limitations. First, the entropy in the human voice prevents speaker recognition systems from being used for large scale identification [17]. Second, researchers have shown it is possible to synthesize audio to effectively imitate the victim's voice, thereby defeating speaker recognition models [36, 41]. Tools making such attacks possible are widely available [5, 14].

The research community has spent significant effort in developing proximity based authentication systems. One of the best known, Zero-Interaction Authentication (ZIA), involves the use of an authentication token on a device (e.g., smart watch or phone) in the user's possession. The device monitors the user's proximity [21] or activity [33] and then performs a wireless handshake (e.g., via Bluetooth, WiFi) with the terminal device (e.g., a laptop) using the token. Once the handshake is successfully completed, the user is granted access to the terminal device. This approach has some limitations, most important of which are replay [27] and relay attacks [23, 26, 31, 43]. Similarly, zero-effort Two-factor authentication can leverage surrounding audio to establish proximity between the terminal and the user's mobile device [30], although some argue that this approach may be susceptible to adversarially injected noise [42]. However, none of these techniques tightly localize a device user located arbitrarily within the same room as the verifying device.

The use of the audio channel to perform these commands makes the weaknesses described above more acute. As such, none of the above solutions are appropriate to address the challenge identified in this paper. Our goal in developing an embodiment of a 2MA system is to overcome the limitations of these proposed systems in an ecosystem in which voice commands are increasingly common.

3 BACKGROUND

3.1 Direction of Arrival

Direction of Arrival (DOA) is a technique used to determine the direction a source (S) is located with respect to an array consisting of at least 2 receivers, R_1 and R_2. DOA assumes S, R_1, and R_2 are points on a 2D plane. The plane is defined such that R_1 and R_2 both lie on the x-axis, with R_1 at the origin. S transmits a signal (e.g., sound) which is subsequently registered by R_1 and R_2 at slightly different times. R_1 and R_2 are assumed to share a global time frame since they are apart of the same receiver array and both record the time when the signal arrived at them. The difference between when the signal arrives at both devices is used to compute the angle of incidence as follows:

[1] We thank the authors of these three papers for generously providing us with malicious samples against which we test our system. Their willingness to share these files not only makes our results valid, but also allowed us to independently validate their results (thereby furthering their science).

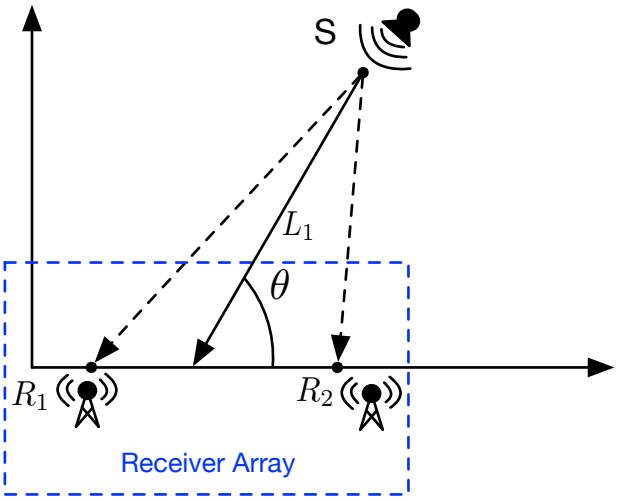


Figure 1: Direction of Arrival helps determine the direction of a sound source S relative to two receivers (R_1 and R_2).

$$\theta = \cos^{-1}(\frac{v_s \tau}{d}) \quad (1)$$

where v_s is the speed of sound, d is the distance between R1 and R2, and τ is the difference between the arrival times. Figure 1 provides a visual description of this technique and shows both the angle θ and its corresponding line L_1.

A single microphone array provides a direction of the source (with bounded uncertainty). However, the presence of a second microphone (or set of microphones) can assist in identifying the location of a source along (or nearby) L_1. Another 2MA system provides this second estimation. Unlike before, these microphones are not located on the same physical array, so achieving clock synchronization is necessary.

3.2 Clock Synchronization

Clock synchronization is the process whereby multiple independent clocks are made to adhere to a single time domain. Even if the clocks are initially set to the same time, their time values will gradually drift apart; a phenomenon known as clock skew. A large amount of research has been conducted in this area, especially with relation to distributed systems [22, 24]. The Network Time Protocol (NTP) has been shown to be accurate to tens of microseconds [4], more than sufficient for our system. NTP works on a client server architecture, where the client device sends a message contain the device's time when the first message is sent (T_a). The server records the "true" time (T_x) of when the first message is received and then it sends its response message. The response message contains T_a, T_x, and the "true" time when the server's response is sent, T_y. Finally, the client records their local time when the last message was received (T_b). Using the four recorded times, the client can now estimate the one way network latency (T_N) and the "true" (T_T) time using the following equations.

$$T_N = \frac{T_b - T_a - (T_y - T_x)}{2} \quad (2)$$

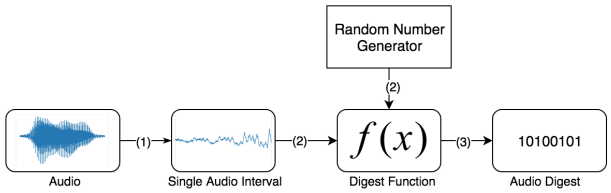
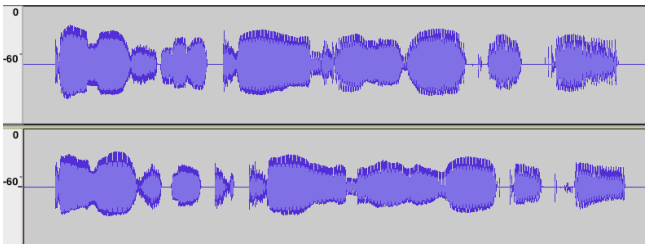
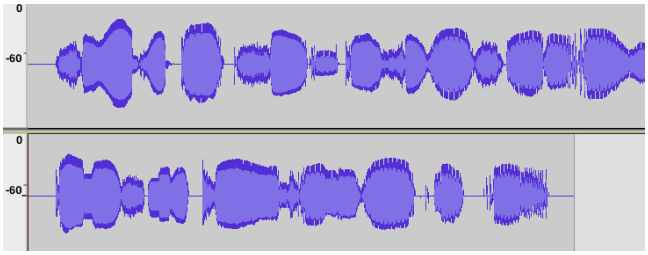


Figure 2: Robust Sound Hash steps: 1) The audio is split into intervals. 2) The interval and a random number are passed into a digest function. 3) The digest function generates a 512-bit audio digest.



(a) Same content, different speakers (BER: 0.299)



(b) Different content, same speaker (BER: 0.499)

Figure 3: RSH helps identify similar content in two audio streams, independent of the speaker. Cryptographic hashes can not perform this task because noise, timbre, and differences in microphones produce varying analog streams. Figure 3a shows waveforms for two different speakers from the TIMIT corpus saying the same content with a BER of 0.299. Figure 3b shows the same speaker saying two different sentences with a significantly higher average BER (0.499).

$$T_T = T_y + T_N \quad (3)$$

3.3 Robust Sound Hash

Once an audio command is given, 2MA needs a mechanism to determine if the commands heard on the mobile device and the voice operated device are the same. Intuitively, we could use a cryptographic hash function, compute the hash of each audio sample and then compare to see if there is a collision. However, cryptographic hash function are sensitive to minor variance in the input. In the

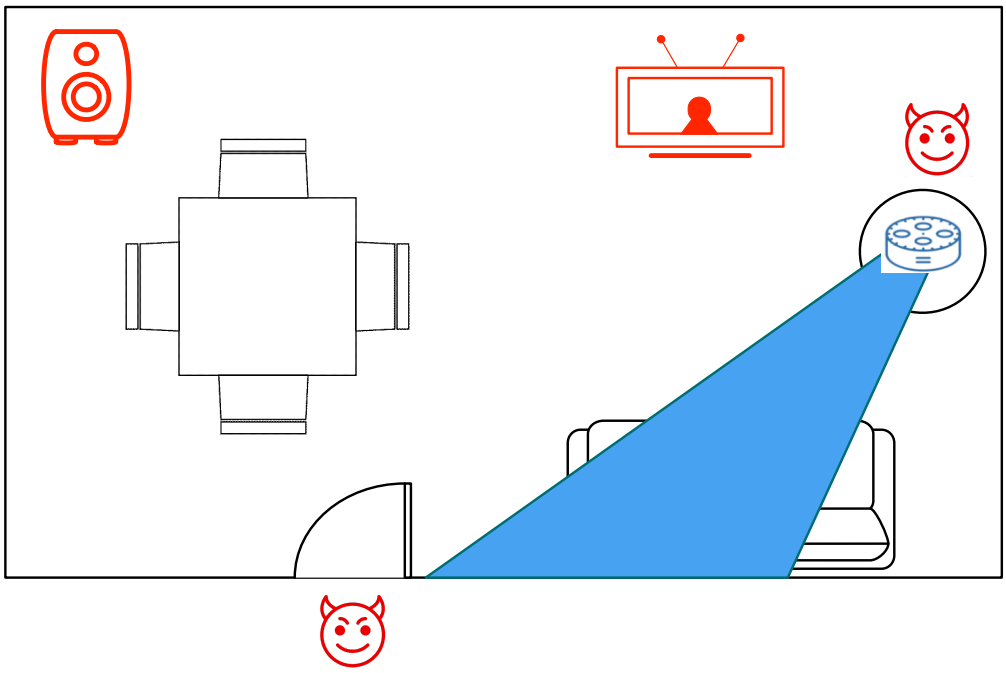


Figure 4: Any audio source in a room can broadcast malicious commands to the victim's home assistant. 2MA is designed to detect such commands by locating its source and only accepting the command if the source is closer to the user's mobile device.

case of audio, small changes (e.g., background noise) would compute two vastly different hash values for two audio samples even if they heard the same command. To determine if both recorded commands are the same, we use Robust Sound Hash (RSH): a digest function that is capable of summarizing the content of an audio signal [29]. Unlike cryptographic hashes that change drastically in response to minor variance in the input, RSH is designed to change slightly as noise is added. In other words, RSH will produce a similar speech digest[2] value for a signal even if the signal has been altered by background noises. This allows RSH to capture unique features of the input; in our case, semantics of a sentence or the words spoken. We use the Jiao et al. RSH construction [29] for our system, which is described at a high level in Figure 2. To make an RSH hash, an audio sample is first divided into one second intervals. These audio intervals are then passed to a function (alongside a secret key) to output a probabilistic 512-bit digest for each second in the input. The whole speech digest would be the concatenation of all the one second digests in the audio sample. To compare two different audio samples, the hamming distance between the two speech digests needs to be computed. This hamming distance would establish the bit error rate (BER) between both samples. A lower BER rate would mean that the two audio samples are similar to each other while a higher BER would mean the inverse. In Figure 3, we show that speech digests are robust in regards to two speakers speaking the same sentence. However, if the same speaker speaks different sentences, the speech digest error will be much higher. In our case, to determine the threshold of acceptable dissimilarity between two audio sample, we will use the same parameters derived by Reaves et al. [40] since they evaluated RSH in an adversarial setting.

We note that applying RSH in an adversarial environment is not a simple matter. Accordingly, properly parameterizing this algorithm to this specific context is a contribution of this work.

[2]To avoid confusion with cryptographic hashes, RSH will be called "speech digest" for the rest of the paper.

4 SECURITY MODEL

We now define our security model and adversarial abilities.

Assumptions: 2MA assumes that the user is in constant possession of their mobile device. This allows us to identify the user by the position of their mobile device. 2MA also needs a confidential channel during an initialization phase but does not assume a confidential channel while commands are being heard by the voice operated device. This simulates actual operating conditions (i.e., voice commands to such devices are said in the clear). Lastly, we also assume that the mobile device, voice controlled device, and cloud system are not compromised. Any other device (e.g., IoT systems) may be compromised. These assumption are similar to traditional 2FA models.

Adversaries: The goal of the adversary is to inject a malicious command to the voice operated device owned by the user. The embodiment of the adversary can vary from a friend or neighbor to a compromised device capable of emiting audio or a TV [3, 9]. Commands injected by the adversary can exercise the full range of commands available to authorized users, including but not limited to adding items to a shopping list, setting alarms, or make unauthorized purchase using the user's credentials that are stored

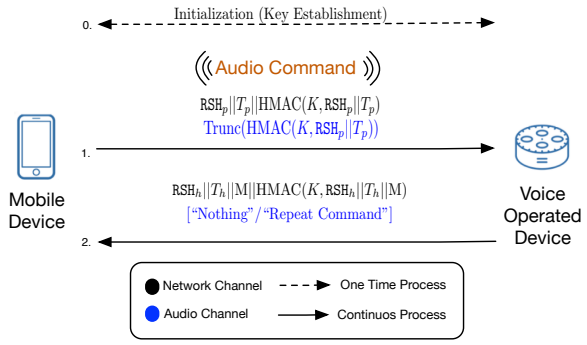


Figure 5: 2MA uses the audio and network channel to authenticate commands.

in the voice controlled device [3]. Moreover, voice commands can be both understandable to a nearby user or outside of the range of normal human comprehension [20, 44, 45]. As an example, an adversary outside the target environment may leverage the lack of authentication to inject commands that physically gain access to the home (e.g., unlock the front door). We show the possible adversaries in a home environment in Figure 4.

The above scenario assumes that the voice controlled device is in a location where the user has permanent control, such as a home. However, the voice controlled device could also be located in a *contested space* where the user only has temporary ownership of the device. Contested spaces include hotel rooms, rental cars, and public spaces. An adversary will have the same capabilities in both settings; however, contested spaces add new challenges to voice controlled devices. For example, a user may temporarily assume ownership of a built-in voice control device while staying in a hotel room. Here, the previous tenant of the room is considered an adversary as he/she could have left a wireless speaker in order to inject future malicious commands to the voice controlled device. Additionally, at the time of departure from the hotel, the voice controlled device is left unattended and may still contain user's credentials. In this case, the next tenant may inject malicious commands to the voice controlled device using the credentials left behind by the user. The problem in either case is that the user is only in possession of the voice controlled device for a limited time. Because users will not stay such environments for a long time, solutions such as biometrics are not appropriate (and remain vulnerable to replay attacks).

Security Goals: Given the above assumptions and adversaries, the goal of any 2MA system is to tie the source of a voice command to the holder of a mobile device, thereby stopping the above adversaries from being able to inject malicious commands. Legitimate commands should not be replayable, nor should hidden/inaudible commands be accepted.

5 AUDIO AUTHENTICATION PROTOCOL

Figure 5 details our audio authentication protocol. While 2MA may be more broadly embodied, this specific protocol is designed to be used between a mobile device and a voice operated home assistant (e.g., Amazon Echo/Google Home). The goal of the protocol is to authenticate the command to the user and to determine if the command recorded at the home assistant is similar to the command heard on the mobile device. We make use of an audio channel and a network channel to communicate between both devices.

Our protocol consist of two phases: During the first phase, the mobile device and the home assistant perform a one-time initialization where they derive a shared secret K. There are multiple means by which K can be derived; we point the reader to the literature on authenticated key exchange for more information as this is not a contribution of this work.

Phase Two occurs when the user speaks a command. At this time (denoted by Step 1), both the mobile device and home assistant locally compute a speech digest (RSH_p and RSH_h, respectively). The mobile device then computes $\text{HMAC}(K, \text{RSH}_p||T_p)$, where RSH_p is the speech digest at the mobile device, T_p is the time when the mobile device first receives the audio command, and K is the shared secret between the two devices. Note that both the mobile device and home assistant have tightly synchronized clocks via that mechanisms discussed earlier in the paper. In compliance to NIST standards [37], the mobile device then truncates the HMAC to 32-bits and uses this value to create a tone that will be sent to the home assistant via the audio channel.[3] This truncated value is necessary because the audio channel has limited bandwidth for robust acoustic signaling [40]. In parallel, the mobile device also constructs a string of form $\text{RSH}_p||T_p||\text{HMAC}(K, \text{RSH}_p||T_p)$ and sends it to the home assistant via the network channel.

In Step 2, the home assistant first validates if the string received from the mobile device is authentic by parsing out the values RSH_p and T_p and recomputing the HMAC as above. Validating the HMAC tells the home assistant that the string is authentic to the mobile device of the user and that it was received at time T_p. However, this does not tell the home assistant whether or not the command heard at the mobile device is the same as the command heard locally. Thus, to authenticate the command, the home assistant passes the locally recorded audio and the speech digest RSH_p from the mobile device through an audio similarity filter (we give more details on its construction in Section 6.2) that will determine if both commands are similar. Once the filter outputs a result, the home assistant will then send through the network a string of form $\text{RSH}_h||T_h||M||\text{HMAC}(K, \text{RSH}_h||T_h||M)$ where RSH_h is the speech digest of the command heard at the mobile device, T_h is the time when the command was first recognized, and M indicates if the commands were a match with each other. If so, the home assistant will execute the actual command and output the results through the audio channel. Otherwise, the home assistant will alert the user of the failed command. Alerting the mobile device of this operation allows it to log all commands executed by the home assistant under its authorization.

6 SYSTEM ARCHITECTURE

We designed and constructed this emboddiment of a 2MA system based on two main mechanisms; Command Location Bounding

[3] NIST 800-107, an HMAC of a strong algorithm can securely be truncated as needed by applications. This standard specifically identifies truncation to as short as 32-bits for audio applications because the real-time nature of audio makes it unlikely that an adversary can successfully attack such a system. Accordingly, our approach is compliant with best-practices.

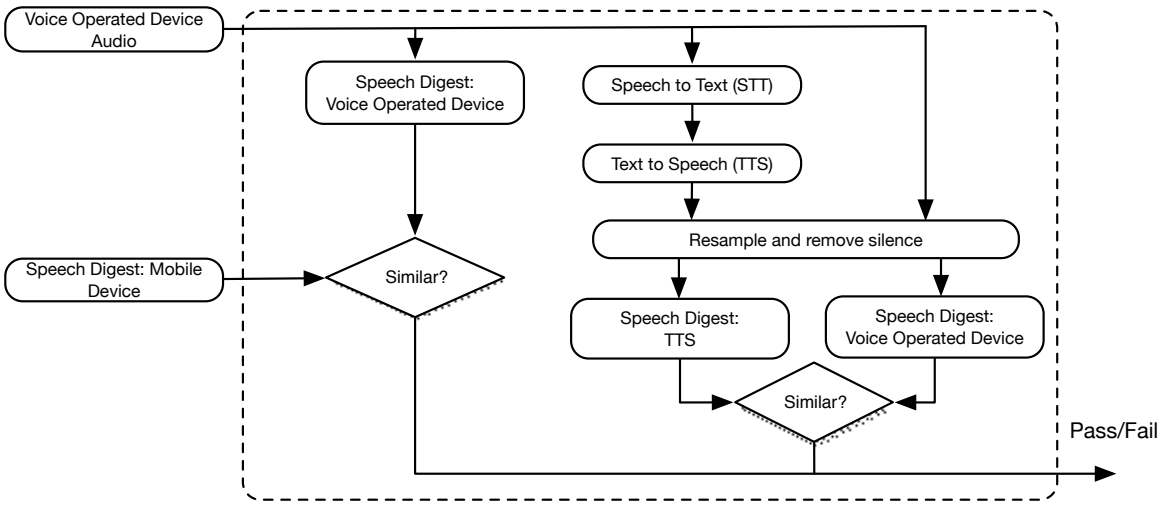


Figure 6: Our command filter first ensures that the audio heard in the mobile device is similar to that of the voice operated device. Then, to stop hidden commands, the filter generates a new audio from the extracted command heard at the voice operated device and compares this audio to the original for similarity.

and Audio Similarity Filtering. Command Location Bounding is used to define an area from which non-malicious commands should originate from. Audio Similarity Filtering ensures that the audio detected at both the mobile device and the home assistant share similar characteristic to avoid inaudible/malicious command injection. Through the protocl described in the previous section and its instantiation as a system in this section, we achieve the security goals established in Section 4.

6.1 Command Location Bounding

Home assistants execute all commands that they hear. By accepting all commands, home assistants rely solely on their physical security to prevent malicious command injection. Since not all potential deployment locations for home assistants are physically secure, we need a more robust security measure. This technique attests to the authenticity of a command by colocating the source of the audio and the user's mobile device. By forcing this constraint onto a command, we limit the physical area from which an attack could originate.

We calculate the DOA of the audio command as our location metric. Although DOA is not able to derive the exact location of either the mobile device or the command's origin, DOA can be used to derive the direction from and bounds for the origin of a sound. By using directionality, we can remove the majority of the physical locations in a region. While highly precise distance bounding protocols would provide greater localization [18, 19, 28, 38, 39], the specialized radios necessary to implement such protocols are not available on any mass-marketed smartphone or home assistant. Instead, we select components that are already widely deployed on all systems and could be activated via software update. The inclusion of such radios could be used as part of another embodiment of a 2MA system, and we leave that exercise to future work.

We construct the chirp emitted by the mobile device as described in Step 1 of our protocol from Section 4. The chirp is generated in the 20 KHz range to make it inaudible to human hearing (thereby minimizing any negative impact on user experience). We use this chirp to calculate DOA_P (DOA of the mobile device), and then add the system tolerance for deviation from that direction. If DOA_V (DOA of the voice) falls within that tolerance, it is passed on to the next phase of processing.

We note that an adversary operating in the absence of a user would not be able to execute an attack. Specifically, if the user's mobile device is not in the room, the command will be rejected because no chrip (yet alone a correct one) will be generated.

6.2 Audio Similarity Filter

Location bounding allows us to eliminate commands (audible to the user or otherwise) that originate outside of the directional cone. However, an adversary within the cone may still be able to inject commands. All systems allow a present user to verbally cancel commands they can hear; accordingly, we need to ensure that inaudible commands [20, 44, 45] originating within the cone can not activate the home assistant.

The first goal of our filter is *to ensure that the same command is heard at both the mobile device and the home assistant.* Without such a protection, an adversary may attempt a relay attack or to inject garbled (but audible) audio [44]. Figure 6 shows the construction of our audio similarity filter. As inputs, we get the audio recorded at the home assistant and the speech digest RSH_p that was extracted from the string sent by the mobile device. The home assistant then computes the speech digest RSH_h of the locally stored audio and the hamming distance between RSH_p and RSH_h. We use a simple majority rule to determine if RSH_p and RSH_h are similar to each other. We set our BER threshold for this comparison at 0.384, which was

previously derived by Reaves et al., [40]; however, we independently validated this setting.

However, the home assistant is still vulnerable to hidden commands, or commands that are unintelligible to human hearing but still register as an actual command to the home assistant [20, 45]. To stop this, our system uses the audio recorded at the home assistant and passes it through a speech to text (STT) module[4]. If the module does not transcribe the audio sample, the input is rejected. The home assistant then passes correctly transcribed commands through a text to speech (TTS) module. We can then compare this new audio sample to the original audio using RSH.

The new audio signal may not have the same characteristics or duration as the original audio. This is a common occurrence because the original audio may contain silence at the beginning or end of the audio or during long pauses between words, as is common in a normal conversation. Additionally, the original audio may have been sampled at a different rate than the new audio. To fix these problems, we first trim silence. We then resample the new audio for the same duration as the original audio. We do this so that words heard in each audio match as close a possible with each other in regards to when the command was said. We then recompute the RSH value of trimmed original audio and compute the RSH value of the new audio.

Second, since we are actively changing the speaker of the new audio by using the TTS module (i.e., the speaker is no longer the user), comparing the newly generated digests would have a higher probability of failing if we were to use the original BER threshold of 0.384. As such, we used the TIMIT corpus [2][5] to derive a new BER that would be receptive to the subtle changes of speakers (i.e., the user and a machine generated voice command). We used the corpus to represent audio that would be recorded on the home assistant and the text file in the corpus to derive new audio by passing it through our TTS module. In total we used 2,310 different audio clips that contain 7,453 seconds of sample date. We calculated an average BER of 0.4105 with a median of 0.4238. From those, we used 0.4105 as our new BER for our second speech digest comparison in our audio filter. These parameters allow us to accurately determine if the audio heard matches the command that was derived.

Finally, with our new threshold rate of 0.4105, we now compare the speech digests of the trimmed original file and the new audio generated by the TTS module to determine if both are similar. In the case of a benign command, the second comparison would pass. However, in the case of a hidden command, the distorted audio input would not.

7 EXPERIMENTS

We perform a number of experiments to characterize our system and design choices. Our first three experiments find minimum audio levels and test seemingly equivalent (but flawed) alternative designs. We perform these tests using a Music Angel JH-MD5BT Bluetooth speaker [1] and 2 Huawei Nexus 6Ps, 2 Google Pixels, and 2 Samsung Galaxy S8s as our mobile devices. We then focus on our proposed design and measure the accuracy of the DOA and RSH. Through these experiments, we demonstrate that our proposed approach dramatically increases the difficulty of launching command injection attacks and is practical on currently deployed hardware.

7.1 Volume and Phone Command Recognition

We tested the mobile device's ability to detect audio commands at various volumes. Our aim was to determine the minimum volume at which a home assistant device is activated by a command. We assume that the microphones in both mobile phones and home assistants are similar; therefore, the results are applicable to both kinds of devices.

This experiment was conducted in a quiet environment with a background noise between 20dB-25dB. This level of background noise is similar to that of a quiet home. We used the Music Angel Speaker to play a single "OK Google", command at various volumes. Our two Nexus 6ps mobile devices were used as receivers during this test. One mobile device was used to measure the volume at a set distance from the speaker while the other device was used to see if the Google Assistant could detect the command at the same distance as the other mobile device.

Starting at the initial volume of 25dB, we incrementally increased the volume of our speaker by 5dB, until we reach a volume of 45dB. We found that the minimum volume that a home assistant could reliability detect a command (100% detection rate) was 40dB. At it quietest, the assistant could properly detect a command around 30dB (i.e., noise levels of a quiet conversation) with a reliability of 40%. Given that all these decibel levels are audible to the human hearing, a user could stop any maliciously injected command if they are near the home assistant. For 2MA systems, this implication allows us to leverage the user as an additional security check against maliciously injected commands targeted at the home assistant.

7.2 Localization Based on Audio Degradation

Our second and third experiments were designed to address a seemingly simple solution - using audio degradation over distance. Because the intensity of audio decreases with distance, the volume of an audio command should be greater at a closer point than further away. This information can be used to determine the relative location of the mobile device and home assistant. However, our experiments demonstrate that such an approach is unreliable.

We performed the experiment in the same quiet environment mentioned above. The speaker was placed at a fixed location while mobile devices were placed at various distances from the speaker. The speaker played a 400Hz tone at a constant volume for 10 seconds. Both mobile devices were placed between 1 and 20 feet (0.3-6.1 meters) from the speaker at 1 foot (0.3 meter) increments. Our experimental data is shown is Figure 7.

Figure 7 demonstrates that there is a weak relationship between the distance and the volume ($r^2 = 0.513$). However, even with a constant audio source, the variation in the volume at any given distance is too great to tightly bound distance to volume. We believe that this variation can be attributed to hardware limitations and environmental effects such as reflection and constructive/destructive interference. Without a tight relationship, volume based distance

[4]We treat the speech to text module as a black box and make no assumption on the underlying algorithms.

[5]The TIMIT Audio Corpus is viewed as the "gold standard" for audio testing and is therefore the most appropriate audio for calibration and testing.

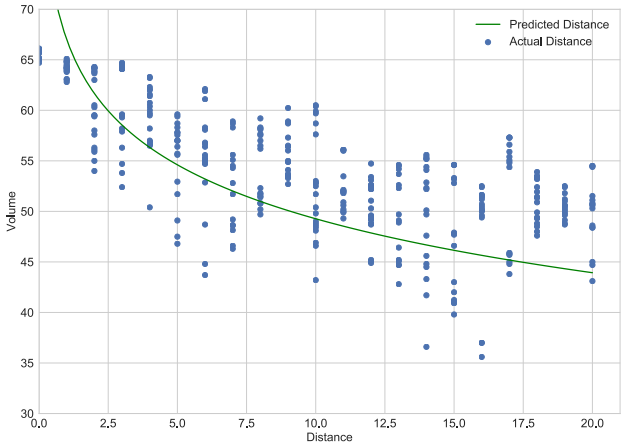


Figure 7: Our experiments reveal that the relationship between the volume and the distance is not tight. For example, when the difference in volume at two receivers is 10dB, the difference in distance the sound has to travel between the source and the two receiver can be from 43ft to 62ft. The existence of this high variance disqualifies volume as an effective distance bounding metric.

bounding appears unfit for 2MA systems in general because it cannot reliably determine which device is closer.

7.3 Frequency Ratios and Audio Attenuation

We also tested localization based on the frequency of an audio command. Naturally, higher frequencies attenuate faster than lower ones [34]. We used this observation to detect which of the two devices is closer to the audio source.

The experiment setup is similar to that of Section 7.2, except for the addition of two different types of mobile devices, the Google Pixel and Samsung Galaxy S8. We use two phones of each type, one phone acting as the home assistant and the other as the user's mobile device. During the experiment, we use all five different combinations of the home assistants and phones by replacing the phone types as needed.

Each test consisted of playing a piece of audio from the TIMIT database [2] over the speaker at a fixed volume. The two mobile devices were placed at unequal distances from the audio source. Each device recorded the audio[6] at their respective locations. We then divided the audio sample into eight frequency bands (150Hz-500Hz, 500Hz-1kHz, 1kHz-1.5kHz, 1.5kHz-2kHz, 2kHz-2.5kHz, 2.5kHz-3kHz, 3kHz-3.5kHz, 3.5kHz-4kHz) and found the peak volume within each band.

Based on these peak volumes, we created an attenuation ratio for every measured audio sample using:

$$ratio = peak_{high}/peak_{low} \quad (4)$$

where $peak_{high}$ is the peak volume from a higher frequency band and $peak_{low}$ is the peak volume from a lower frequency band. We then compared these ratios to determine which device is closer to the source. A higher attenuation ratio means that a device is closer to the audio source.

[6] All recordings were done at 8kHz sampling rates.

Devices		Correct Percentage	
Hub	Phone	Home	Hallway
Nexus	Galaxy	89.49%	23.77%
Nexus	Pixel	85.64%	21.38%
Nexus	Nexus	87.18%	31.43%
Galaxy	Galaxy	93.34%	65.18%
Pixel	Pixel	68.46%	65.95%

Table 1: Using attenuation to distance bound two devices works well in open environments, such as homes. However, it does not work as well in high reverb environments like hallways .

In Table 1, we show how often we were able to correctly predict whether an audio source was closer to the home assistant or a mobile device. In the case of our quiet room experiments, all of the scenarios (except for the pixel/pixel) achieved a success rate of at least 85%. However, in a reverberant environment, accuracy for all scenarios dropped to between 21%-65%. We believe this is due to the high amounts of reverberation within the space. Specifically, higher frequencies reflect cleaner (with less loss) than lower frequencies. It stands to reason that the reverberation in the environment is artificially increasing the volume of the high frequencies significantly more than the lower frequencies within the audio samples. By unevenly amplifying the frequency domain, reverberation will cause this type of distance bounding to fail.

We have shown that frequency based localization is not suitable for authentication in our scenario. The frequency attenuation ratios are not robust in environments with high reverberation and thus unfit for a general 2MA system.

7.4 2MA Direction Bounding

We then tested the DOA techniques proposed for our embodiment of a 2MA system. We use a Raspberry Pi B+ (2014) running Raspbian Stretch as a stand in for a home assistant. We connected our Raspberry Pi to a "Respeaker 4 Microphone Array" as our home assistant device. The Respeaker array has 4 microphones spaced 5.9 cm apart in a square and is able to make 4 channel recordings at sample rates up to 44100 Hz. It is important to note that the Respeaker 4 Microphone Array is connected directly to the Raspberry Pi via its 40 GPIO pin connection. We placed our Pi in a quiet environment for testing.

The command data set consisted of 4 channel recordings that were generated at a fixed distance and angle from the home assistant. After recording the audio, we calculated the time the waveform reached each channel by locating the first point where the amplitude of the waveform broke a given threshold. This was repeated four times, once for each microphone. Then these times were used to calculate the DOA for the command. The chirp data set was generated using high frequency tones played from our Music Angel speaker. For each data set we generated 20 samples at 6 different locations around our microphone array. Specifically these points are 0°, 10°, 20°, 30°, 90°, and 180° around the device from a set 0° point.

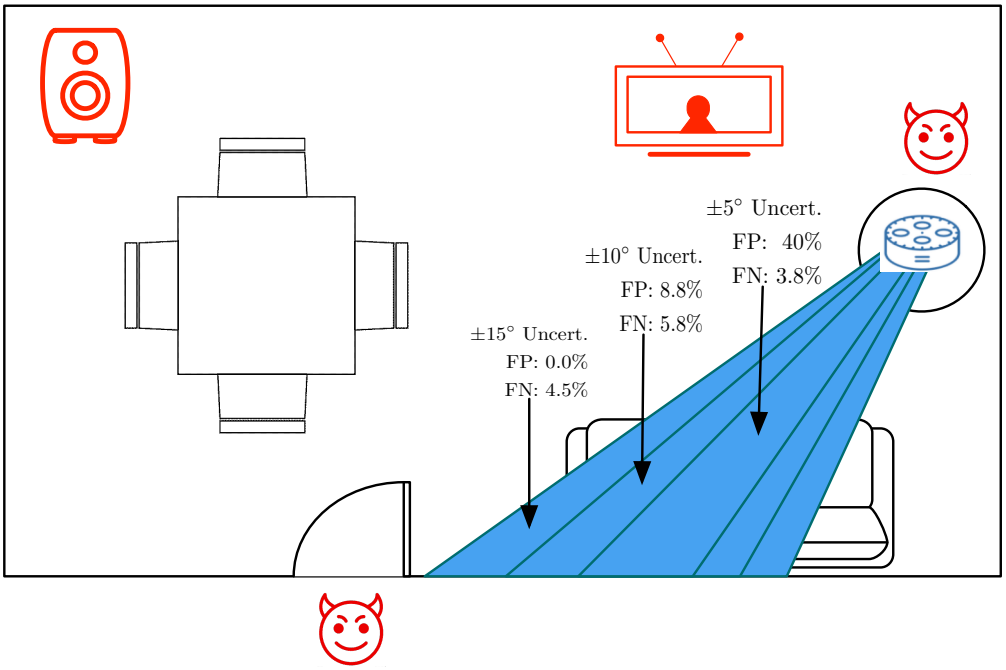


Figure 8: While the false positive rate of our DOA system seems to vary with the uncertainty cone, the false negative rate seems to remain consistent.

Next we cleaned our two data sets by examining the relative time of arrival between each of the four microphones located on the array. That is, due to the physical properties of the device, the command's arrival at each microphone can differ from the other microphones on the array by at most 11 samples. This number was determined by calculating the distance an audio wave travels within $2.2 * 10^{-5}$ seconds, or one sample at our 44100 Hz sampling rate. We then took the maximum distance between any of the two microphones on the array (5.9 cm for adjacent microphones, 8.3 cm for diagonally microphones) and divided that by the distance traveled by the wave during a single sample. Recordings beyond 11 samples apart represented some other action (e.g., context switching) in the Pi and were therefore discarded.

Using our cleaned data sets, we then constructed a comparison to simulate a benign setting. In this comparison we derived how precisely we can calculate the DOA using our current hardware and signal processing. This consisted of matching runs originating from the same location from both our command data set and our chirp data set. Effectively this places a mobile device and the speaker of the command at the same approximate location. We found an average difference between our calculated DOA angles of 4.6° with a standard deviation of 3.4°. From these values we estimate that our uncertainty cone should be approximately ±15° degrees from the chirp DOA to ensure a low false positive rate. In fact, within our experimental set of 80 runs, 0 of them were misidentified as malicious command with an uncertainty cone of ±15°. By tightening our uncertainty cone to ±5° and ±10° we found that our DOA mechanism would have misidentified 32 and 7 pairs respectively.

We then constructed a comparison to mimic an adversarial setting. Similar to before, we compared runs from our command data set that originated from 0°. However, in contrast we compared it to all other runs in the chirp data set that originated from a location that was greater that our uncertainty away. After running all 440 adversarial comparisons, we found that at our preferred uncertainty cone of ±15° had a false negative rate of 4.5%, misidentifying only 13 of our tests. Unlike in our benign testing, the uncertainty values of ±5° and ±10° degrees did not perform considerably worse. Respectively, ±5° and ±10° achieve false negative rates of 3.9% and 5.8% only misidentifying 14 and 21 of our tests. Our adversarial test data can be seen in the context of a typical room in Figure 8

It is important to assess false negatives in context. Were these messages attempted with a user outside of the room, none would be successful. If the user is in the room, they would need to verbally cancel approximately 4 out of every 100 messages the attacker injected (as opposed to all 100 without a 2MA system). Accordingly, our proposed techniques *dramatically increase the difficulty of successfuly launching such an attack while minimizing burden on a user to respond.*

7.5 2MA Audio Similarity Filter

We tested the reliability of the filter against adversarial command injections. We used Google's Speech Recognition API as our speech to text (STT) module and Google's Text to Speech python library as our text to speech (TTS) module.

One of the metrics we tested for was the reliability of our audio similarity filter. In other words, we wanted to know the number of

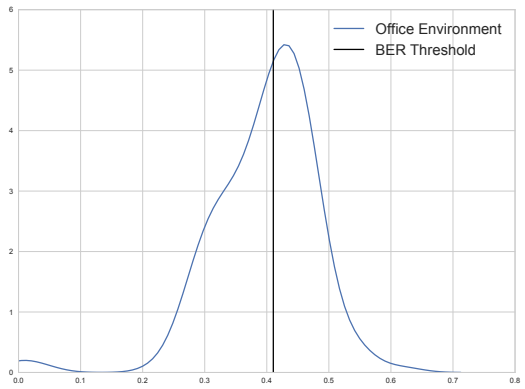


Figure 9: Most of the speech digest error can be attributed to the first second of recording, which includes the activation phrase "OK Google".

times a benign command would successfully pass the audio similarity filter. To determine this, we first performed a control experiment to test the reliability of the TTS and STT modules of our similarity filter in regards to the second speech digest comparison[7]. Using the microphone built into a Macbook Air 2015, we recorded ourselves speaking four different command (e.g.,"OK Google, call 911", "OK Google, set an alarm", "OK Google, how's the weather?", and "OK Google, transfer money") 10 times each at the same tone and pace as the output of our text to speech module. Our results showed that 39 out of 40 (97.5%) commands passed our audio similarity filter as valid commands. We investigated why the last command did not pass and concluded that there was audio clipping happening on our recorded commands (i.e., the first part of "O" in "OK" was not caught by the microphone).

Since our control test demonstrated reliability, we expanded our experiment to include a real world office scenario with white noise generators in the background. We recorded the same commands as before but this time the commands were spoken at the same tone and pace as a regular conversation. We set up two microphones three meters apart and recorded the commands. In this case, each microphone played the role of either the user's mobile device or the voice operated home assistant. We used two different configuration for testing; audio originating at an equidistant location from both microphones and audio originating closer to one microphone than the other. In total, we obtained 80 different samples.

Of the 80 samples, all passed the first speech digest comparison, were correctly transcribed by the STT module and then converted to new audio samples by our TTS module. After removing silence and resampling the audio as explained in Section 6.2, 62/80 (77.5%) passed the final speech digest comparison. In Figure 9, we show the BER rate of each second from the 80 audio samples we gathered. A closer inspection showed that many of the seconds with a BER greater than the threshold came from the first second in the audio sample. We believe this drop in reliability can be accounted to the late detection of activation phrase in the first second.

Adversarial Audio. An adversary can give the home assistant a malicious command in three different ways: audible, inaudible and hidden. Once a command reaches the audio similarity filter, our 2MA system has already indicated that the user and the source of the command are co-located. In this case the adversary can only broadcast the audio command from the same general direction as the user, we expect that the user will over hear the malicious command and take necessary action. However, that is not true in the case of inaudible or hidden commands.

A hidden command is an audio command that has been obfuscated or "hidden" by noise [20]. This means that the user will not detect the audio, but merely hear a sound. The authors of the attack provide us with ten different hidden command audio samples. We used these to test the audio similarity filter. Of the ten samples provided, eight samples were not transcribed by the STT module[8]. Therefore, the audio similarity filter automatically rejected these command injections. One of the two adversarial audio clips that was in fact transcribed by the STT module, the phrase "OK Google", was only an activation phrase and not an actual command. The other command was partially transcribed from the original audio, which was later rejected as an actual command by the second speech digest comparison of our filter.

Finally, we also tested our audio similarity filter against inaudible commands [45]. An inaudible command is one that can not be heard by the normal ear as it exists in the ultrasound range. The authors provided us with four adversarial raw audio samples, with commands encoded within frequencies of 24kHz and 32kHz. These samples were passed through the audio similarity filter. Interestingly, the STT module did not transcribe any of the samples and rejected all of them as actual commands. Although none of the attack audios samples passed our audio similarity filter, we do not claim our 2MA system to necessarily address these unknown attacks.

8 DISCUSSION

8.1 Limitations

No system is perfect, especially those that operate in an analog environment. Accordingly, our embodiment may fail to detect some malicious commands under certain conditions. This includes an adversary who is located within our system's uncertainty cone. We assume that in such a scenario, the owner will hear the malicious audio commands and take the necessary action. Concretely, 2MA treats the owner as an additional layer of security. Therefore, 2MA will not be able to detect a malicious command if the adversary generates it after first subduing the real owner or if the real owner has difficulty hearing. We believe that our system could easily be extended for the hearing impared to include confirmation displays on the mobile device; however, we leave such extensions to future work.

[7] We are testing the second speech digest comparison because we are actively changing the speaker for one of the audio samples.

[8] We believe that Google's Speech Recognition has improved since the publishing of the attack paper

8.2 Improving Accuracy

The ability to provide a tighter uncertainty cone would further reduce false negatives. Better hardware in the home assistant would help make this possible. For instance, our Respeaker 4 Microphone Array is representative of less provisioned home assistants. The Google Home with only 2 microphones is the most similar to our device. However, better equipped home assistants such as the Amazon Alexa and Apple Homepod with 7 and 6 microphones respectively are also available. Having additional microphones would allow us to further limit this cone-however, we believe that our results represent the average hardware available on the market

8.3 Application Considerations

Sensitive Commands. 2MA systems authenticate any command that is heard by the home assistant before it is executed. However, not all commands made are sensitive or require an authentication process. Commands such as "How's the weather today?" or "What time is it?" should be readily available to be queried by anyone without having to go through the authentication steps set by 2MA. However, banking transaction are more sensitive and would benefit from the extra security. By setting an access control scheme or a user defined blacklist and whitelist for commands, 2MA systems could be deployed without removing the convenience of voice operated devices.

Smart TV Authentication. Voice commands can be used to remotely control Smart TVs. Unfortunately, these commands can also come from various, potentially malicious, sources. Although our 2MA embodiment focused on authenticating voice commands for home assistants, we can further expand our command authentication to include smart TVs. By simply replacing the home assistant's microphones for the microphones found in the smart TV, a 2MA embodiment could then colocote the user's mobile device with the voice command's origin.

Purchase Authorization. The use of a voice interface for online shopping is becoming increasingly popular. From Amazon [6] to Apple [7] and, more recently, Google Home's shopping integration with Walmart [16]. While convenient, the voice interface's lack of command authorization enables an adversary to make purchases without the consent of the owner. A possible solution to this lack of authorization can be to deploy a 2MA system. For any command to successfully be executed, 2MA would require the user's mobile device (and therefore, the user) to be in the same vicinity as the voice interface. In this case, the colocation of the user and the command would implicitly authorize any online purchase made. However, if the user's mobile device is not near the home assistant, then the command would be treated as malicious and automatically rejected.

8.4 Change of Ownership

IoT devices are becoming more prevalent in today's world. For example, hotel rooms are now equipped with devices (e.g., home assistants, window blinds, thermostat) that allow occupants to seamlessly "connect" to the room by logging in to their personal cloud profile. One problem that arises from the ubiquity of these devices is the possible exposure of personal information when the ownership is transferred from one user to the other (e.g., the next guest that uses the same hotel room). From the perspective of the original user, once they have checked out of the room, the IoT device can be considered dead. Accordingly, the IoT device should remove all of their credentials from the device. However, this may not the case. Once a new guest checks-in to the room, they will have access to these same device. If the devices never removed the previous user's credentials, the new user will have access to them. This exposure could potentially allow the new user to make commands on behalf of the previous one.

Similarly, built in IoT devices could change ownership in real estate transactions. Much like hotel rooms, houses are either built with IoT devices permanently fixed to the structure or the owner has to augment their home throughout their stay. When the house is eventually sold to a new owner, many of these devices will likely change ownership as well. Unfortunately, these devices have no way in verifying that the ownership of the home has changed. If the original owner did not actively remove their credentials, then the new tenant could give commands to the IoT devices with the previously stored credentials.

In both scenarios described above, the underlying issue comes from devices being unaware of the change of ownership. A possible solution to this problem is to implement a 2MA system that generates mobile tokens to temporarily authenticate sensitive commands for a set amount of time. For example, if a new tenant were to make a sensitive command after the mobile token expires, then the 2MA system would prevent the command from being placed. Such system would have a similar concept as continuous authentication [33, 43]. However, unlike continuous authentication, the goal of this system would not be to completely stop the new tenant from using these built-in devices. Rather, the goal is to prevent the new tenant from making sensitive commands before setting up their own home cloud environment. Developing and parameterizing a system as described is an interesting area that we leave for future work.

8.5 2MA Systems With Additional Hardware

Our 2MA embodiment was made to be easily deployable. We made concessions to avoid adding extra hardware other than the user's mobile device and the home assistant itself. However, many IoT device (e.g., smart security cameras, smart baby monitors, and bluetooth speakers) contain microphones that could be potentially used as additional inputs. A 2MA system could leverage these additional microphones with DOA, or alternative localization technique, to find the exact location of a voice command. In a space containing additional microphones, a 2MA system could construct separate uncertainty cones that could then be used to pinpoint the internal location of the voice command. Rather than only showing directionality of the command, we could determine a small region from which the command originated. Having a tighter bound on the location of the command would serve to increase the security of a 2MA system.

It is our hope that readers see 2MA as a generic framework, and build applications appropriate for specific contexts (as we have done) going forward.

9 CONCLUSION

Voice commands dramatically simplify many user interfaces, and are critical to the operation of many IoT devices. Unfortunately, such commands are only protected by the assumption that only authorized users are capable of speaking to these devices. As has repeatedly been demonstrated recently, this is no longer a realistic assumption [3, 9]. In this work, we propose Two Microphone Authentication (2MA). 2MA systems take advantage of the presence of multiple microphones being present in an ecosystem to localize and authenticate the source of a command. We demonstrate that such a construction works using independent devices (e.g., a mobile phone and a home assistant) in both benign and malicious settings, and in so doing dramatically increase the effort required by an attacker to inject such commands successfully. To this end, we show that the increased deployment of microphones in many settings can be used to improve authentication.

10 ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grant number CNS-1702879. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] [n. d.]. Music Angel JH-MD5BT Bluetooth Speaker. https://www.amazon.com/imiss-music-angel-jhmd05bt-mini-bluetooth-wireless-portable-speaker/dp/B00F86RRNY/?tag=napcardnao-20. ([n. d.]). 2017-12-18.
[2] [n. d.]. TIMIT: Acoustic-Phonetic Continuous Speech Corpus. https://catalog.ldc.upenn.edu/ldc93s1. ([n. d.]). 2017-12-18.
[3] [n. d.]. TV anchor says live on-air 'Alexa, order me a dollhouse' âĂŞ guess what happens next. ([n. d.]).
[4] 2010. Network Time Protocol Version 4: Protocol and Algorithms Specification. https://tools.ietf.org/html/rfc5905. (2010).
[5] 2017. Adobe demos "photoshop for audio," lets you edit speech as easily as text. https://arstechnica.com/information-technology/2016/11/adobe-voco-photoshop-for-audio-speech-editing/. (2017).
[6] 2017. Amazon Alexa Line. https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011. (2017).
[7] 2017. Apple Siri. https://www.apple.com/ios/siri/. (2017).
[8] 2017. August Home Supports the Google Assistant. http://august.com/2017/03/28/google-assistant/. (2017).
[9] 2017. Burger King 'O.K. Google' Ad Doesn't Seem O.K. With Google. https://www.nytimes.com/2017/04/12/business/burger-king-tv-ad-google-home.html. (2017).
[10] 2017. Cortana. https://www.microsoft.com/en-us/windows/cortana. (2017).
[11] 2017. Google Assistant. https://assistant.google.com/. (2017).
[12] 2017. Google Home. https://madeby.google.com/home/. (2017).
[13] 2017. Google Home now lets you shop by voice just like Amazon's Alexa. https://techcrunch.com/2017/02/16/google-home-now-lets-you-shop-by-voice-just-like-amazons-alexa/. (2017).
[14] 2017. LyreBird. https://github.com/logant/Lyrebird. (2017).
[15] 2017. Starling Bank Integrates API into Google Home. http://bankinnovation.net/2017/02/starling-bank-integrates-api-into-google-home-video/. (2017).
[16] 2017. Walmart Makes Voice Shopping Even More Affordable with New Google Device. https://blog.walmart.com/innovation/20171004/walmart-makes-voice-shopping-even-more-affordable-with-new-google-device. (2017).
[17] Salil Prabhakar Antil K. Jain, Arun Ross. 2004. Information Fusion in Biometrics. *IEEE Transactions on Circuits and Systems for Video Technology* (2004).
[18] C. Cremers, K.B. Rasmussen, and S. Capkun. 2012. Distance hijacking attacks on distance bounding protocols. Proceedings of the IEEE Symposium on Research in Security and Privacy.
[19] C. Meadows, P. Syverson, and L. Chang. 2013. Towards more efficient distance bounding protocols for use in sensor networks. Proceedings of the Conference on Security and Privacy for Emerging Areas in Communication Networks.
[20] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden Voice Commands. In *25th USENIX Security Symposium.*
[21] Mark D. Corner and Brain D. Noble. 2002. Zero-Interaction Authentication. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom '02).* ACM, New York, NY, USA, 11.
[22] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 147–163.
[23] Aurélien Francillon, Boris Danev, and Srdjan Capkun. 2011. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In *In Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS).*
[24] Sukumar Ghosh. 2014. *Distributed systems: an algorithmic approach.* CRC press.
[25] Google. 2017. Transactions Developer Preview. https://developers.google.com/actions/transactions/. (2017).
[26] Tzipora Halevi, Di Ma, Nitesh Saxena, and Tuo Xiang. 2012. *Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data.* Springer Berlin Heidelberg, Berlin, Heidelberg.
[27] Otto Huhta, Prakash Shrestha, Swapnil Udar, Mika Juuti, Nitesh Saxena, and N Asokan. 2015. Pitfalls in Designing Zero-Effort Deauthentication: Opportunistic Human Observation Attacks. *arXiv preprint arXiv:1505.05779* (2015).
[28] J. Clulow, G.P. Hancke, M.G. Kuhn, and T. Moore. 2006. So near and yet so far: Distance-bounding attacks in wireless networks. Proceedings of European Conference on Security and Priacy in ad-hoc and sensor networks (ESAS).
[29] Yuhua Jiao, Liping Ji, and Xiamu Niu. 2009. Robust Speech Hashing for Content Authentication. *IEEE Signal Processing Letters* (2009).
[30] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srfjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. Proceedings of the 24th USENIX Security Symposium.
[31] Z. Kfir and A. Wool. 2005. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05).*
[32] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and other botnets. *IEEE Computer Society* (2017).
[33] Shrirang Mare, Andrés Molina Markham, Cory Cornelius, Ronald Peterson, and David Kotz. 2014. Zebra: Zero-effort Bilateral Recurring Authentication. In *IEEE Symposium on Security and Privacy (S&P).*
[34] Peter R Marler and Hans Slabbekoorn. 2004. *Nature's music: the science of birdsong.* Academic Press.
[35] Chase Martin. 2017. 72% Want Voice Control In Smart-Home Products. Media Post – https://www.mediapost.com/publications/article/292253/72-want-voice-control-in-smart-home-products.html?edition=99353. (2017).
[36] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. 2015. *All Your Voices are Belong to Us: Stealing Voices to Fool Humans and Machines.* 20th European Symposium on Research in Computer Security.
[37] National Institute of Standards and Technology. 2012. Recommendation for Applications Using Approved Hash Algorithms. NIST Special Publication 800-107 - Revision 1. (2012).
[38] N.O. Tippenhauer, C. Popper, K.B. Rasmussen, and S. Capkun. 2011. On the requirements for successful gps spoofing attacks. Proceedings of the ACMConfrence on Computer and Communication Security (CCS).
[39] N.O. Tippenhauer, H. Luecken, M. Kuhn, and S. Capkun. 2015. UWB Rapid-Bit-Exchange system for distance bounding. Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks.
[40] Bradley Reaves, Logan Blue, Hadi Abdullah, Luis Vargas, Patrick Traynor, and Thomas Shrimpton. 2017. AuthentiCall: Efficient Identity and Content Authentication for Phone Calls. In *26th USENIX Security Symposium (USENIX Security 17).*
[41] Maliheh Shirvanian and Nitesh Saxena. 2014. Wiretapping via Mimicry: Short Voice Imitation Man-in-the-Middle attacks on Crypto Phones. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.*
[42] Babins Shrestha, Maliheh Shirvanian, Prakash Shrestha, and Nitesh Saxena. 2016. The Sounds of the Phones: Dangers of Zero-Effort Second Factor Login based on Ambient Audio. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*
[43] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. 2014. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In *the Proceedings of 2014 IEEE International Conference on Pervasive Computing and Communications (PerCom).* 163–171.
[44] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine Noodles: Exploiting the Gap Between Human and Machine Speech Recognition. *11th USENIX Workshop on Offensive Technologies* (2015).
[45] Guoming Zhang, Chen Yan, Xiaoyu Ji, Taimin Zhang, Tianchen Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible Voice Commands. *Computer and Communications Security (CCS)* (2017).