CommanderGabble: A Universal Attack Against ASR Systems Leveraging Fast Speech

Zhaohe Zhang
University of Oklahoma
Norman, OK, USA
zxz003@ou.edu

Edwin Yang
University of Oklahoma
Norman, OK, USA
edwiny@ou.edu

Song Fang
University of Oklahoma
Norman, OK, USA
songf@ou.edu

ABSTRACT

Automatic Speech Recognition (ASR) systems are widely used in various online transcription services and personal digital assistants. Emerging lines of research have demonstrated that ASR systems are vulnerable to hidden voice commands, i.e., audio that can be recognized by ASRs but not by humans. Such attacks, however, often either highly depend on white-box knowledge of a specific machine learning model or require special hardware to construct the adversarial audio. This paper proposes a new model-agnostic and easily-constructed attack, called *CommanderGabble*, which uses fast speech to camouflage voice commands. Both humans and ASR systems often misinterpret fast speech, and such misinterpretation can be exploited to launch hidden voice command attacks. Specifically, by carefully manipulating the phonetic structure of a target voice command, ASRs can be caused to derive a hidden meaning from the manipulated, high-speed version. We implement the discovered attacks both over-the-wire and over-the-air, and conduct a suite of experiments to demonstrate their efficacy against 7 practical ASR systems. Our experimental results show that the over-the-wire attacks can disguise as many as 96 out of 100 tested voice commands into adversarial ones, and that the over-the-air attacks are consistently successful for all 18 chosen commands in multiple real-world scenarios.

CCS CONCEPTS

• **Security and privacy**; • **Human-centered computing** → **Ubiquitous and mobile computing**;

KEYWORDS

ASR misinterpretation, syllabification, adversarial audio

ACM Reference Format:
Zhaohe Zhang, Edwin Yang, and Song Fang. 2021. CommanderGabble: A Universal Attack Against ASR Systems Leveraging Fast Speech. In *Annual Computer Security Applications Conference (ACSAC '21), December 6–10, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3485832.3485892

1 INTRODUCTION

Automatic Speech Recognition (ASR) systems have proliferated, so also have adversarial attacks (e.g., [4, 13, 15, 32, 47, 55, 57]) against them. Deep neural networks and efficient training methods have greatly improved the accuracy of ASR systems over time, including Google's Speech recognition technology, whose word error rate fell to 4.9% in 2017 from 23% in 2013 [44]. However, ASR misinterpretations are non-negligible and still happen frequently in practice. Recent research efforts have demonstrated that dialects or accents variations [57], words that have the same or similar pronunciation but different spellings (homophones) [32, 56], and even nonsensical sounds [13] may lead to speech misinterpretations. Such systematic errors can be utilized to develop malicious attacks, such as vApp squatting attacks [32].

Rate of speech (i.e., speed at which a user speaks) is a well-known factor that may cause both ASR and human misinterpretations [11, 12, 39, 49], but little effort has been made to exploit it for constructing adversary audio attacks. In particular, due in part to the limitations of the articulatory machinery, timing and acoustic realization of syllables are affected. Pronunciation may be thus altered in several ways, such as phoneme reduction, time compression or expansion, and changes in the temporal patterns [11, 12]. In this context, a *syllable* is regarded as a pronunciation unit, i.e., a single, unbroken sound of a spoken or written word; and a *phoneme*, as the core spoken language component, is a unit of sound.

If a person hyperarticulates and makes significantly long pauses among syllables, ASR misinterpretations could occur, but overall, slow rate of speech usually does not hinder human understanding nor the performance of ASR speech recognition. In this paper, we thus focus on exploiting the misinterpretation errors brought by fast speech rate as well as investigating their security implications.

A fast speech rate usually results in distorted pronunciation that humans cannot distinguish and meanwhile ASR systems fail to correctly interpret [39]. We observe that ASR experiences the following three phoneme misinterpretation patterns while parsing rapidly spoken commands: (1) **reduction**: some phonemes are directly omitted; (2) **replacement**: a specific phoneme is replaced with another phonetically similar phoneme; (3) **coalescence**: the ASR cannot accurately discriminate between inter-word silence and intra-word silence, and phonemes in neighboring syllables merge together to form a new syllable. If an attacker wants the ASR to execute a malicious command, she can manipulate that command such that the resulting sped-up audio will become incomprehensible to humans while still being interpreted in the same way.

All three phoneme misinterpretations are illustrated in Figure 1a, where a user quickly speaks the command, "*Open the door*", which is erroneously parsed by the Google assistant as "*Oh panda our*". The

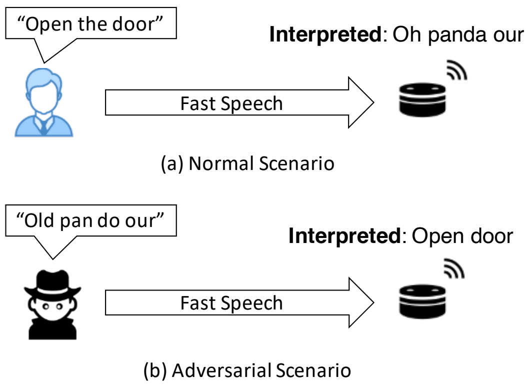


Figure 1: An example to show impact of fast speech on performance of ASR speech recognition.

original phonetic representations of the three words can be denoted as [OW P AH N], [DH AH], and [D AO R], respectively. For the interpreted phrase, the corresponding phonetic representations become [OW], [P AE N D AH], [AW ER]. We have the following observations. First, the phoneme reduction effect is visible in the phoneme D disappearing from the third word. Second, the phoneme replacement effect is evident in phonemes AO and R being replaced by AW and ER, respectively. Third, the original word "*open*" is disassembled while other phonemes merge to form new words such as "*panda*", demonstrating the phoneme coalescence effect.

Now consider the scenario in Figure 1b: the attacker aims to stealthily make ASR execute the command "open door". To this end, the attacker carefully crafts a meaningless phrase "Old pan do our" by manipulating the phonemic structure of the original command "Open door", and broadcasts it at high speed. Since the phonemic structure has been significantly changed and the fast speech further alters the pronunciation, the legitimate user usually cannot realize that somebody is executing a command (i.e., the injected command is incomprehensible to humans). However, due to the aforementioned three distortion effects induced by fast speech, the ASR then misinterprets the spoken words into the command of "Open door" and then executes that command. Consequently, the attacker successfully injects a command into the ASR system. Though the concept of the attack is straightforward, it relies on addressing two main technical challenges.

First, before constructing adversarial commands, we need to understand the distortion effect of fast speech on different syllables. To address this, we compare the phonetic structure of the 5,000 most frequently used words [19] and 100 commonly used voice commands in Amazon Alexa or Google Assistant [36, 43] with that of corresponding misinterpretations induced by fast speech. As a result, we can identify the syllable structures that can be stably captured by ASRs regardless of the playback speed, which are suitable for use in constructing adversarial commands.

Second, after identifying these stable syllable structures, we then need to account for the unstable structures in the malicious commands that we wish to produce. We develop a method to manipulate the syllables in the original phonetic structure so that after we feed the sped-up, adversarial audio into the ASR, the original command is executed. Meanwhile, to avoid arousing suspicion, the created adversarial audio should not be intelligible to humans. For each extracted phonetic structure, we find an adversarial word to map to it. If no such a word is found, we utilize a customized phoneme morphing technique to change phonemes in the syllable until finding a matching word. The combination of those adversarial words forms an adversarial command candidate, which will be synthesized into fast speech. Once the resultant audio can be recognized by the ASR but not by humans, it can be then used to conduct the actual attack.

We conduct a real-world evaluation by launching the discovered attack over-the-wire and over-the-air against 7 practical ASR systems (including 4 online transcription services and 3 voice-controllable digital assistants). Our experimental results show that the over-the-wire attacks can transform 96 of 100 selected widely used voice commands into adversarial ones. We performed the over-the-air attacks in three different typical scenarios, including household, teleconference, and in-vehicle. We find that such attacks are viable in practice. The average success rates for Amazon Alexa, Google Assistant, and Microsoft Cortana can reach as high as 95%, 97%, and 93%, respectively.

Our contributions are summarized as follows.

- We systematically explore the misinterpretations introduced by fast speech of voice commands for popular ASRs, and analyze the phonetic structure variation between the original command and the one that ASRs recognize.
- We discover that manipulating the phonetic structure of a command and increasing its speed can cause it to be incomprehensible to humans but recognized as desired by the ASR. We thus create a universal attack of crafting adversarial voice commands against ASRs.
- We implement the proposed technique and conduct extensive real-world experiments to verify its feasibility, robustness, and suspiciousness.
- We describe, implement, and test a defense mechanism for this attack.

Roadmap: The rest of the paper is organized as follows. We introduce the background knowledge about ASR and linguistics in Section 2. Section 3 discusses the adversary model. The detailed attack design and countermeasures are then presented in Section 4 and 5, respectively. Section 6 shows experimental results. Section 7 summarizes related work. At last, Section 8 concludes the paper.

2 BACKGROUND

2.1 ASR Principle

ASR is a technique that allows machines to translate audio streams of speech into written text. A typical ASR system has two major components, feature extraction and decoding [54]. The former distills the raw speech signal into acoustic features with relevant information only [47], which include Mel-Frequency Cepstral Coefficients (MFCC) [40], Perceptual Linear Prediction (PLP) [29], etc. The extracted acoustic features are then decoded using a pre-trained acoustic model to predict the probability of each phoneme being spoken, as well as a pre-trained language model to predict the probability of which word comes next.

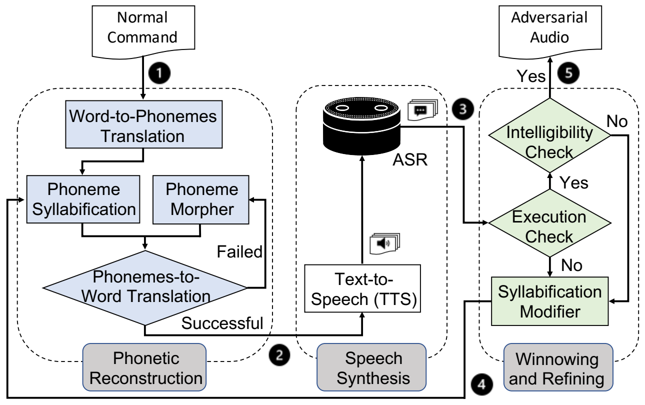


Figure 2: Flowchart of constructing adversarial audio.

2.2 Syllable Structure and Phonemes

In phonology, the phonetic pattern is referred to as *syllable structure*, consisting of three elements: the onset, the nucleus, and the coda [23]. According to the *Sonority Sequencing Principle* [18], which stipulates that onsets rise in sonority towards the nucleus while codas fall in sonority, the nucleus is usually (but not always) a vowel, as vowels are high in sonority. In general, the rankings in sonority for these three elements are middle, high, and low, respectively. The onset-nucleus-coda combination often can be abstracted as a consonant-vowel-consonant (abbreviated CVC) syllable. Each CVC-structured syllable is prominent as the vowel in such a syllable is always different in quality from neighboring consonants.

As mentioned earlier, phoneme is the unit of sound distinguishing one word from another. Advanced Research Projects Agency (ARPA) developed a set of phonetic transcription codes (called "ARPABET") in the 1970s, which represents phonemes of General American English with distinct sequences of ASCII characters. There are 39 phonemes in ARPABET in total. A modified form of the ARPABET system is employed by the CMU Pronouncing Dictionary, a widely used open-source pronunciation dictionary that contains over 134,000 English words and their pronunciations [1]. This work utilizes the CMU pronunciation dictionary to convert words into phonemes.

3 ADVERSARY MODEL

We consider a black-box attack, where the attacker does not know the ASR mechanism of the attacked system. Our adversary model aligns with existing black-box adversarial audio attacks (e.g., [4, 5, 13, 31, 52]). The goal of the attacker is to generate adversarial audio that can be executed by the victim ASR and is incomprehensible to human listeners. Towards this goal, she manipulates the phonetic structure of the original command and synthesizes the manipulated command into fast speech.

We assume that the attacker generates malicious audio samples offline before the actual attack takes place. Also, unlike some existing adversarial audio attacks, where the audio needs to be directly fed to ASR systems (e.g., [15, 47]) or the effectiveness of over-the-air attacks highly depends on the utilized speakers and recording devices (e.g., [54]), we consider both over-the-wire and over-the-air attacks, and make no specific constraint for the type of the speaker or victim microphone.

For an over-the-wire attack, we assume that the adversary is able to directly feed the adversarial audio into the victim ASR, which accepts an input file with the format such as WAV or FLAC. An adversary can launch an over-the-air attack remotely or with an attacking speaker in the vicinity of the target ASR device. For example, an attacker can embed the generated adversarial audio into a video and tempt the victim to play it (e.g. through YouTube), or secretly play the adversarial audio during an online meeting (e.g., by inserting it into a shared presentation slide as sound effect). Alternatively, she may stealthily leave a remotely controllable speaker around the victim ASR device in the victim's office, vehicle, or home.

4 ADVERSARIAL ATTACK DESIGN

We propose a generic technique to make any command inconspicuous to human ears, meanwhile allow it to be executed by ASR systems. Unlike previous adversarial attacks against ASRs, which require adding artificial noise to the audio (e.g., [6]), the proposed attack does not need to craft any noise.

4.1 Design Overview

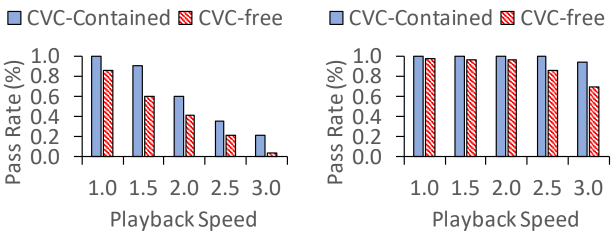
We utilize four important phases to generate audio adversarial examples, including phoneme reconstruction, speech synthesis, audio transmission, and finally, winnowing and refining. Figure 2 plots the flowchart of the proposed attack.

The attacker first selects a target command, and inputs it into the phase of *phonetic reconstruction* (Figure 2, step ❶), which aims to convert the command into a different text for a potential audio adversarial example. This phase translates the command into its phonemic representation and then extracts the syllables according to a pre-defined syllabification rule. Next, each separate syllable will be re-mapped into a word. If no such matching word is found, the phoneme morpher will alter the phonemes within the syllable until the modified syllable has a suitable match. The sequence of translated words generated in the first phase composes an adversarial command candidate.

The attacker then performs *speech synthesis* (step ❷) to convert it into adversarial audio candidates in fast speech with a speech synthesizer (e.g., Google TTS [27]). After that, each audio candidate is sent to an ASR. The ASR response then inputs to the last module, *winnowing and updating* (step ❸), to remove ineffective adversarial audio candidates, those which cannot be interpreted as the original command or can be discerned as the command by humans. If all candidates fail, the attacker employs the *syllabification modifier* to update the pre-defined syllabification rule, and provides it to the first phase (step ❹) to re-generate the adversarial command candidate. If a candidate passes both effectiveness checks, it can be regarded (step ❺) as an adversarial audio to attack the target ASR.

4.2 Phonetic Reconstruction

The phase of phonetic reconstruction first extracts the syllables with pre-defined structures in the phonetic representation of a target command, and then maps each of them to a new word to generate a potential adversarial command.



(a) Amazon Alexa **(b) Google STT**

Figure 3: Pass rates for CVC-contained and CVC-free words.

W2P Translation: This process translates each word in a selected command into its phonetic representation. It can be achieved by the CMU pronunciation dictionary, which covers most words used in commands of various ASRs. Also, for words not in the dictionary, we employ the LOGIOS Lexicon Tool [3] to convert them into phonemes.

Phoneme Syllabification: The goal of this step is to divide the obtained phoneme sequence into syllables based on the specified rules, which the attacker can use to build an adversarial command. We begin by selecting a syllable structure suitable for generating a successful adversarial command and then develop two syllabification rules.

Selection of CVC: After phoneme syllabification, we should be able to convert each of the resultant syllables into a word to include in the adversarial command. The most frequent syllable type in English is CVC with three segments [21], which consists of a vowel flanked by exactly one consonant on each side. We thus expect that more words have that structure than any other. We empirically verify that CVC words take the highest percentage (599/5,000=11.98%) in the 5,000 most frequently used words [19]. We also expect that words with this structure will be the most consistently recognized by the ASR.

To verify that hypothesis, we collect and analyze the latest list of Amazon Alexa commands (consisting of 281 commands) [43] as well as the latest list of voice commands for Google Assistant (consisting of 275 commands) [36]. These two lists have 456 and 441 unique words, respectively. Among each group of unique words, we find that CVC words are the most frequent in both groups (23.68% and 26.08%, respectively). We also count the number of words whose syllables contain CVC. If the phoneme representation of a word contains at least a CVC sub-syllable, we refer to such a word as a *CVC-contained* word, otherwise, we call it as a *CVC-free* word. For Amazon Alexa and Google Assistant, 83.80% and 82.10% of the unique words are CVC-contained.

Furthermore, we explore the distribution of recognized syllables for the ASR under different speech rates. Specifically, we randomly select 100 words (with half CVC-contained and half CVC-free) from the unique words used for constructing Amazon Alexa commands and input each to Google TTS to generate audio files with different playback speeds. We then transmit audio files over-the-wire to the Alexa ASR API [7] for transcription. We vary the playback speed from 1.0× to 3.0× in increments of 0.5×, where 1.0× denotes the normal playback speed. When the ASR fails, it generates empty output, so for each speed, we compute the *pass rate* as the ratio of the non-empty ASR recognition results to the total number of tested words. Figure 3a plots corresponding pass rates of CVC-contained and CVC-free words. We can see that under different fast speech rates, the pass rates of CVC-contained words are consistently larger than that of CVC-free ones. We repeat the above experiments with Google Assistant, and obtain similar results shown in Figure 3b. These results demonstrate convincingly that the ASRs tend to better recognize CVC-contained than CVC-free words.

Algorithm 1 Default Syllabification Rule

```
Require: A sequence of syllables, i.e., $[p_{1,1}, \cdots, p_{1,l_1}], \cdots, [p_{m,1}, \cdots, p_{m,l_m}]$; a function $State(p)$, which returns C if the phoneme $p$ is consonant, otherwise V.
Ensure: A sequence of sub-syllables, i.e., $[S_1, \cdots, S_k]$
1: $k \leftarrow 1$
2: for $i = 1$ to $m$ do
3:     for $j = 1$ to $(l_i - 2)$ do
4:         $S_{temp} = [State(p_{i,j}), State(p_{i,j+1}), State(p_{i,j+2})]$
5:         if $S_{temp} = [CVC]$ then
6:             $S_k \leftarrow S_{temp}$ {CVC is found}
7:             $k \leftarrow j + 1$
8:             $j \leftarrow i + 1$
9:         end if
10:    end for
11: end for
```

As there are more words to match a CVC-structured syllable and CVC-contained words are easier to be recognized by ASRs, we select CVC as a foremost feature to syllabify the initial phoneme sequence of the command.

Syllabification Rules: We develop the following two syllabification rules:

(1) CVC-based Rule: We extract sub-sequences of phonemes with the pattern of CVC in the phoneme representation of each word. We assume that the command has m words. After W2P, we obtain pronunciation for each word. Let $[p_{i,1}, p_{i,2}, \cdots, p_{i,l_i}]$ denote the i^{th} ($i \in \{1, 2, \cdots, m\}$) pronunciation, where l_i is the total number of phonemes in this pronunciation, and $p_{i,j}$ is the j^{th} ($j \in \{1, 2, \cdots, l_i\}$) phoneme in this pronunciation. Correspondingly, we can extract $l_i - 2$ syllables, each with three consecutive phonemes. We then check each syllable in turn to determine whether its phonetic structure is CVC. Pseudocode for this rule is shown in Algorithm 1. For example, the word "broadcast" has a sequence of 8 phonemes – [B R AO D K AE S T] with pattern "CCVCCVCC". Accordingly, we obtain CVC-structured syllables [R AO D] and [K AE S].

(2) CVC-absent Rule: The phoneme sequences of some words may not contain CVC, so an alternative rule is needed to handle such cases. Specifically, for each vowel in the phoneme sequence of a word, if there is a consonant before it, we extract this CV-structured phoneme sub-sequence. Otherwise, if there is a consonant after it, we extract the VC-structured phoneme sub-sequence. When the previous two checks both fail, we simply extract this single vowel. The reasoning for the CVC-absent rule design is twofold: (i) each word's syllable structure must contain one of the three

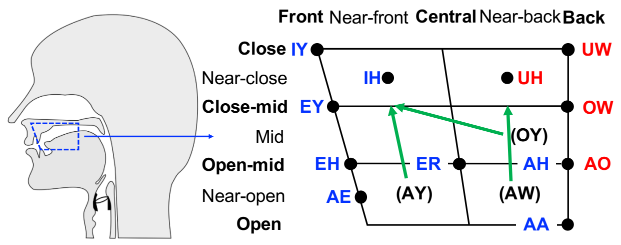


Figure 4: ARPABET vowels in IPA: (1) the vertical axis is mapped by vowel height; (2) the horizontal axis determines vowel backness; (3) where a dot appears, the ones to the right and left represent rounded and unrounded (spread) vowels.

patterns, i.e., CV, VC, and V, so this rule retains much important information about the original phoneme sequence; (ii) if we only search for a phonetic pattern longer than CVC, and extract phoneme sequences based on such patterns, we may lose much information in the command level. For example, the phonetic pattern of the word "answer" ([AE N S ER]) is VCCV, so the CVC-based rule does not work. The CVC-absent rule is therefore initiated, enabling us to extract two syllables, [AE N] (with VC) and [S ER] (with CV).

Finally, the phoneme syllabification phase can take an input to modify the default syllabification rules in the event they cannot generate an adversarial audio. We discuss this feedback/refinement loop in Section 4.4.

P2W Translation: P2W, the inverse of W2P, converts each qualified syllable into a word. The criteria for a qualified syllable include: (1) it matches with a word's syllable; (2) it is different from the syllable of the original command word, from which it is extracted. The first requirement makes sure that we can construct an adversarial command, and the second aims to increase the difference between the generated adversarial command and the original so that the synthesized audio file is less identifiable. If neither of the two requirements is satisfied, this syllable is then inputted to the module of phoneme morpher.

Phoneme Morpher: Vowels are the most sonorous sounds, as discussed earlier. They have been shown to contribute more toward intelligibility than consonants for humans [22, 30]. Thus, in order to reduce human perception of the synthesized command, the phoneme morpher displaces the vowel in the syllable with another vowel, similar enough that the ASR recognizes the adversarial audio as the original command. American English contains 12 spoken vowels (15 including diphthongs). In the following, we discuss how to measure the similarity of two vowels, and develop methods to perform vowel displacement.

Vowel Similarity: Different vowels have different features, and we thus obtain the similarity of vowels by comparing their features. We explore the features of vowels using the International Phonetic Alphabet (IPA) vowel quadrilateral [10], which roughly describes tongue position and lip roundness when we use the mouth to pronounce vowels, as shown in Figure 4. Each diphthong, as a gliding vowel, is a combination of two adjacent vowel sounds within the same syllable. The IPA vowel quadrilateral describes each vowel's articulatory feature in three dimensions, tongue height, tongue backness, and lip rounding. We use an additional feature, tenseness, that is also widely used to classify vowels [20] and which

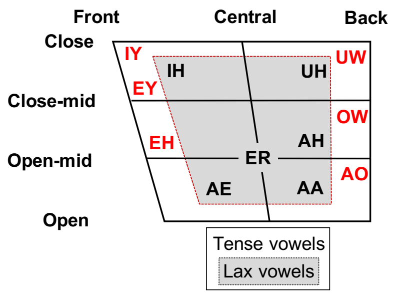


Figure 5: Tenseness of vowels (monophthongs).

describes the muscle effort or constriction during the pronunciation of a vowel [37]. Tenseness divides vowels (monophthongs) into categories of tense and lax vowels, as shown in Figure 5.

Thus, each vowel has a *feature set* of these four articulatory features. We employ the *Jaccard index*, which is commonly used to measure the similarity of two finite data sets [9], to measure the similarity between the articulatory feature sets for a pair of vowels. We compute the Jaccard index $J(V_1, V_2)$ between the two vowels V_1 and V_2 as

$$J(V_1, V_2) = \frac{|F_{V_1} \cap F_{V_2}|}{|F_{V_1} \cup F_{V_2}|} = \frac{|F_{V_1} \cap F_{V_2}|}{|F_{V_1}| + |F_{V_2}| - |F_{V_1} \cap F_{V_2}|}, \qquad (1)$$

where F_{V_i} is the feature set of vowel V_i ($i \in \{1, 2\}$), and $|A|$ represents the number of elements in set A. The Jaccard index ranges from 0 to 1, with 0 denoting that both vowels have the exactly same features and 1 indicating that the two vowels do not share any feature. Computing the Jaccard index for every pair of vowels among the 15 different vowels with Equation 1, we obtain a symmetric 15×15 similarity matrix. Each diagonal entry of the matrix denotes the Jaccard index of the corresponding vowel with itself.

Vowel Displacement Strategy: After determining vowel similarities, we can then select a substitution for a given vowel phoneme to finish phoneme morphing. Specifically, we try all the other vowels from smallest to largest Jaccard index until the resultant syllable is qualified and can be successfully converted into a word.

4.3 Adversarial Audio Synthesis

A speech synthesizer turns normal language text into speech, and enables users to define the playback rate. Such text-to-speech (TTS) systems have been widely used in various scenarios, including automotive, voice assistants, audio books, etc. We utilize Google TTS with WaveNet voices [41] to convert the adversarial command candidate into adversarial audio candidates with different fast playback rates. Next, the synthesized adversarial audios are transmitted to the testing ASR to identify the effective one, which can be used to launch attacks against the target ASR.

Audio Transmission: Each generated audio can be transmitted *over-the-wire* or *over-the-air* into the target ASR. In an over-the-wire adversarial attack, the audio with a format such as WAV or FLAC is directly passed to the ASR. Such attacks focus on making online transcription services (e.g., Google STT [26]) to generate target texts. Conversely, an over-the-air adversarial attack requires the audio to be played via a speaker towards the target voice-controlled platform (e.g., Amazon Alexa).

Playback Speed Set Up: We can adjust the parameter of speaking rate in the range $[0.25\times, 4.0\times]$ with Google Cloud TTS [25], where $1.0\times$ denotes the normal native speed, $2.0\times$ is twice as fast, and $0.5\times$ is half as fast. In this work, we select the range $[2.0\times,\ 3.0\times]$ of the playback speed to investigate generating adversarial commands. In particular, we take advantage of the unintelligibility of fast speech and also utilize phonetic structure manipulation to compensate for the distortion effect of fast speed rate. Generally, if the playback speed rate is too close to $1.0\times$ (e.g., in the range of $[1.0\times, 2.0\times]$), the resultant audio may be easily understood by humans, while if it is too fast (e.g., above $3.0\times$), the introduced distortion effect of the audio would be too dramatic to be compensated for, causing the failure of ASR recognization.

4.4 Winnowing and Updating

The phase has two goals: (1) winnowing out adversarial audio candidates that either cannot be correctly executed by the target ASR or can be understood by humans; (2) updating the syllabification rule (applied in the phase of phonetic reconstruction) if there is no adversarial audio candidate left after winnowing.

Execution Check: We test the machine recognizability of each adversarial audio candidate based on audio transmission methods: (1) for the over-the-wire mode, the maliciously crafted audio passes the execution check if the target ASR completely transcribes the desired text, and it fails the check when the ASR transcribes one or more words incorrectly; (2) for the over-the-air mode, if the audio successfully triggers the ASR to perform a desired action (e.g., setting an alarm or sending a message) as specified by the adversarial voice command, it then passes the execution check.

Intelligibility Check: Once the adversarial audio candidate can be recognized by the ASR, the following question is whether the audio can also be recognized by humans - an adversarial audio attack is not practical if so.

As mentioned earlier, fast speech and phonetic structure manipulation can both hinder human intelligibility of the audio. To verify this intelligibility degradation effect, we compare human with ASR intelligibility using *word error rate (WER)*, the ratio of incorrectly recognized words (including substitutions, deletions, and insertions) to the total number of words, quantifying the similarity between the selected command and the recognized one [35, 51]. Let N denote the total number of words in the command. WER can be then computed by

$$\mathrm{WER} = \frac{S + D + I}{N}, \tag{2}$$

where S, D, and I represent the respective numbers of word substitutions, deletions, and insertions. A higher WER indicates a command less comprehensible to humans. For example, in the case that the command "enable toll" is interpreted as "in a bot", we have $N = 2$, $S = 2$, $D = 0$, $I = 1$, and thus WER $= 3/2 = 1.5$.

WER provides word-level similarity between the recognized command and the target one. To complement the intelligibility test, we also calculate the *phoneme error rate (PER)* as the ratio of phonological distance between the recognized and target commands to the length of the phoneme sequence of the target command. The phonological distance can be denoted with the Levenshtein distance [33], which is defined as the sum of phoneme substitutions, deletions, and insertions between the phoneme sequence of the recognized command and that of the target command. Unlike WER, PER quantifies the phoneme-level similarity between two commands.

We observe both WER and PER between the target command and the one recognized by listeners. If both are larger than a predefined threshold, γ_0, the intelligibility check is passed and the candidate can be used to conduct the actual attack, otherwise, this check fails. Empirically, we set $\gamma_0 = 0.5$.

Syllabification Modifier: If the adversarial audio candidate fails either check, the adversary modifies the syllabification rules correspondingly.

When Execution Check Fails: We compare failed transcription with corresponding target commands to identify the reason of execution failure. Empirically, we find the leading phoneme of a word is often ignored when performing the initial phoneme syllabification. For example, the phonetic representation of the word "open" is [OW P AH N], and after phoneme syllabification, we extract the subsequence [P AH N] with pattern CVC; consequently, the ASR often translates the generated adversarial audio candidate into "pen" rather than "open". Accordingly, we modify the syllabification rules based on whether the leading phoneme of a misinterpreted word is a vowel or consonant,

- If a word's syllable begins with pattern "VCVC", we extract the sub-syllable consisting of the first two phonemes, and then begin the CVC-based rule;
- If a word's syllable begins with pattern "CCV" or "CCCV", we first remove all consonant phonemes between the first consonant and the first vowel, and then initiate the CVC-based or CVC-absent rule.

When Intelligibility Check Fails: To increase the pronunciation difference, we propose to expand the phonetic structure of the original command before phoneme syllabification. Specifically, within a phoneme sequence of a word, we change each single vowel (those with no neighboring vowels) into two of the same vowel, and leave any instances of two or more consecutive vowels unchanged. The CVC-absent rule is then applied. The newly generated adversarial command candidate is longer (i.e., has more words), and its pronunciation is expected to be further deviated from that of the original command.

4.5 Discussions

4.5.1 Extension to Other Languages. We emphasize that the proposed attack in this paper targets ASRs recognizing American English. Considering the similarity between American English and other western languages (e.g., Spanish and German) which also utilize vowel and consonant sounds, as long as the speed-induced ASR misinterpretation can be identified, the discovered attack can be customized for those languages.

4.5.2 Algorithm Scalability. To guarantee the unintelligibility of the generated audio, we employ manual human evaluation, as do some other hidden voice command generation techniques (e.g., [14]). The attacker can use crowdsourced evaluation of intelligibility. For example, we can use Amazon Mechanical Turk (Turk) [2], which pays human workers to complete online tasks. This approach has been adopted by extensive existing work (e.g., [14, 17, 45]) to increase the efficiency of intelligibility evaluation. We can generate a

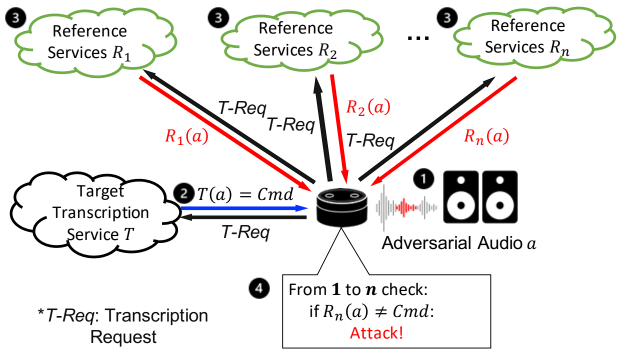


Figure 6: Flow chart of *xRef.*

batch of adversarial audio candidates and post all tasks simultaneously, speeding up the process and ensuring its scalability.

5 COUNTERMEASURES

The proposed adversarial voice attack crafts adversarial voice commands by exploiting the misinterpretation made by ASR systems when translating fast speech. Intuitively, to defend against such an attack, we can increase the ASR's robustness against fast speech by training with more fast speech audio samples. However, this approach is impractical, because it is difficult to obtain sufficient fast speech samples, considering that a wide range of factors (e.g., speech rate, pitch, and tone) may affect the fast speech. We here introduce two more practical defense methods.

5.1 Cross-reference Transcription

Empirically, we find that most commercial ASR systems can recognize voice commands at regular speeds (i.e., 1.0× - 1.9×) with high accuracy, while once the playback speed exceeds 1.9×, their recognition both degrades and diverges from each other. These differences in high-speed audio transcriptions appear due to variances in architecture and training datasets among ASR systems. Based on this observation, we propose *cross-reference transcription (xRef)* to detect the proposed adversarial voice attack.

Figure 6 presents the flow chart of *xRef.* Suppose that an attacker constructs an adversarial audio a. When the ASR receives the adversarial audio (Figure 6, step ❶), it sends the transcription request to its ASR server R. The audio is interpreted as a voice command, i.e., $T(a)$. If the ASR figures out that $T(a)$ is equal to a valid command *Cmd* (step ❷), *xRef* is initiated: the adversarial audio is then sent to another one or multiple ASR transcription services $R_1, \cdots,$ R_n; each reference service transcribes the adversarial audio and returns the result to the local ASR device (step ❸); the ASR can then compare the transcription $R_1(a), \cdots, R_n(a)$ from n reference services with *Cmd* to detect the adversarial audio (step ❹). When the transcription from any reference service does not match with that recognized by the target transcription service, i.e., $R_i(a) \neq Cmd$ ($i \in \{1, \cdots, n\}$), the received audio a is regarded as malicious and would not be executed.

Usually, when two reference services are introduced, *xRef* can achieve a success rate of nearly 100% for detecting the adversarial audios, as verified in Section 6.4. However, it requires extra time to obtain reference transcription, and more reference services may cause a longer delay for the ASR to execute a benign voice command.

5.2 Silence Detection

Another defense technique is to build a machine learning model to recognize fast speech through its intrinsic property of containing less silence (both intra- and inter-word) for each command than normal speech. We can take advantage of this observation to build a CNN (Convolutional Neural network) to detect such differences in silence and then filter out dangerous fast speech. If the detected silence in a voice command is below a certain threshold, the ASR would alert the user first and ask for the command to be repeated at a slower speed.

6 EXPERIMENTAL RESULTS

We implement *CommanderGabble* to generate adversarial audios for arbitrary selected commands in American English. We evaluate both over-the-wire and over-the-air attacks against different off-the-shelf ASR platforms. To test the human comprehensibility of generated adversarial audios, we conduct a survey asking participants to listen to adversarial audios and indicate the meaning they hear. We also implement the proposed countermeasure - *xRef* and test its effectiveness.

6.1 Evaluation Setup

We randomly select 100 commands from a pool of commonly used voice commands in Amazon Alexa or Google Assistant [36, 43]. To understand the impact of command length on the attack performance, these commands have varying lengths. Specifically, we consider short (one word), medium (two or three words), and long (more than three words) commands.

We test four popular online transcription services, Amazon Transcribe, Google STT, IBM Watson STT, and MS Azure STT for over-the-wire attacks, as well as three widely used voice-controlled digital assistants, Amazon Alexa, Google Assistant, and Microsoft Cortana for over-the-air attacks. In our experiments, we use an Amazon Echo Dot (3rd Gen) for Amazon Alexa, a Google Pixel 4 phone for Google Assistant, and a Lenovo ThinkPad X1 Carbon laptop for MS Cortana. For over-the-wire attacks, we generate adversarial audio files for the selected 100 commands; for over-the-air attacks, we test under three different practical environments:

- *Household:* The victim ASR is on the table, and nearby is the speaker (Logitech Z200) playing adversarial audio (with a noise level between 15-20 dB).
- *Teleconference:* The adversarial audio is embedded in a PowerPoint slide, played by the presenter during an online meeting (via Zoom [58]), and broadcasted to all attendees. The victim ASR is beside one attendee (with a desktop) and can hear the meeting audio via the computer speaker (Logitech Z200). The noise level is similar with that in the household scenario.
- *In-vehicle:* The victim ASR is put near the car center console, and the adversarial audio is played via the car speaker (Kenwood KFC-1666S), when the vehicle engine starts with a noise level of around 60 dB. Specifically, we use Android Auto [24] which integrates Google Assistant, and Amazon

Table 1: Tested commands for over-the-air attacks.

Environment	ID	Command
Household	C1	Stop
	C2	Continue
	C3	Unlock the door
	C4	Call my phone
	C5	Show me the back door camera
	C6	Turn off the light in living room
Teleconference	C7	Bluetooth
	C8	Location
	C9	Call my phone
	C10	Recent messages
	C11	Turn on the light
	C12	Set the alarm at 3am
In-vehicle	C13	News
	C14	Home
	C15	Enable Tollway
	C16	Cancel Route
	C17	How long will it take to drive to library
	C18	What is my current location

Alexa with Alexa Auto [8] mode. Microsoft currently has no specified design for this scenario so we continue to use regular MS Cortana.

We test 6 commands (including two short, two medium, and two long commands) under each environment, as shown in Table 1. Note that adversaries may have different goals in each environment and thus utilize different commands. We choose and test approximate commands to make the attacks practical.

6.2 Attack Effectiveness

6.2.1 Over-the-wire (OTW) Attack. We first utilize examples to demonstrate the effectiveness. Specifically, we randomly select 3 commands from each of the three groups (short, medium, and long). Next, we generate adversarial audio for each command. For comparison, we also utilize Google TTS to directly convert each original command into a normal audio file with the same playback speed as the corresponding adversarial audio uses.

Table 2 presents the number of successful transcriptions in all three trials of each group for different ASRs. We have the following observations. First, *CommanderGabble* is successful for all nine commands against different ASR systems, convincingly demonstrating the attack effectiveness. Second, most normal audio files fail to generate correct translation (especially for long commands) when they are played at high speed, verifying the speed-induced distortion effect. Also, different ASR systems may have different performance of recognizing fast speech.

Large-scale Tests: To examine the overall impact of the OTW attacks, we use *CommanderGabble* to generate adversarial audio files for the selected 100 commands. By checking whether each transcription by the tested ASR is correct or not, we compute the translation accuracy as the ratio of successful transcriptions to the total number of transcription attempts.

Figure 7 shows the translation accuracy of adversarial audio and normal audio files for each group of commands, as well as that for the total commands. We can observe two major tendencies. First, compared with normal audio, the crafted adversarial audio always achieves much higher translation accuracy. The average translation accuracy of the adversarial audio across four different ASRs for all commands is as high as 90%, while that of the normal audio is just 28%. Second, *CommanderGabble* achieves the best attack performance overall for medium commands with an average translation accuracy of 95%, while for short and long commands, the attack performance slightly decreases to 85% and 87%, respectively.

6.2.2 Over-the-air (OTA) Attack. A voice-controllable digital assistant can be activated by either the wake-up word or pressing a special button. An adversary thus has to cause the ASR to recognize the corresponding wake-up word before injecting any voice commands. With the proposed technique, we successfully construct adversarial audio files for different wake-up words, as demonstrated

Table 2: OTW case studies (N and A refer to normal and adversarial audio).

	Short		Medium		Long	
ASR	N	A	N	A	N	A
Amazon Transcribe	0/3	3/3	0/3	3/3	0/3	3/3
Google STT	1/3	3/3	1/3	3/3	0/3	3/3
IBM Watson STT	0/3	3/3	0/3	3/3	0/3	3/3
MS Azure STT	2/3	3/3	1/3	3/3	0/3	3/3

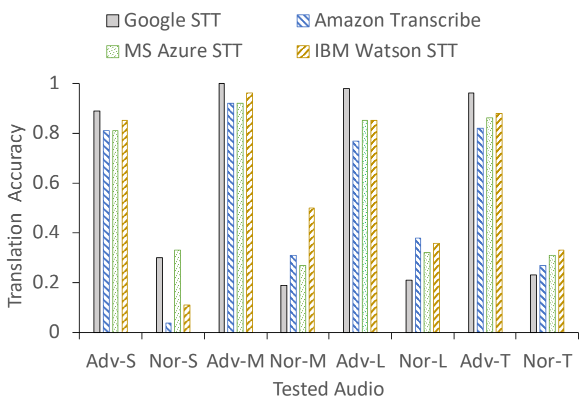


Figure 7: Translation accuracy for adversarial audio (Adv) and normal audio (Nor) files generated with different groups of commands (S, M, L denote short, medium, and long commands respectively, and T represents all commands).

Table 3: Wake-up words and their adversarial commands.

Wake-up Word	Adversarial Command	Speed	Successful?
Okay Google	kaye go oh	2.0× - 2.1×	✓
Alexa	A leh sa	2.0× - 2.2×	✓
Hey Cortana	hye core ta	2.0× - 2.5×	✓

Table 4: Example of over-the-air evaluation.

Scenario	Adversarial Command	Speed	Successful?		
			Alexa	Google	Cortana
Household (C5)	sho me bad ack dawe orr cam murr	2.0×	✓	✓	✓
Teleconferencing (C7)	Lu Two	2.0× - 2.5×	✓	✓	✓
Vehicle (C15)	eh en a abe bah ul tow ole	2.0× - 2.3×	✓	✓	✓

Table 5: Overall performance for OTA attacks.

Command ID	Success Rate		
	Amazon Alexa	Google Assistant	Microsoft Cortana
C1	10/10	10/10	10/10
C2	10/10	10/10	10/10
C3	7/10	8/10	8/10
C4	10/10	10/10	9/10
C5	10/10	10/10	9/10
C6	10/10	10/10	10/10
C7	8/10	9/10	7/10
C8	9/10	8/10	8/10
C9	10/10	10/10	10/10
C10	8/10	9/10	9/10
C11	10/10	10/10	10/10
C12	10/10	10/10	10/10
C13	5/10	6/10	5/10
C14	6/10	6/10	5/10
C15	6/10	8/10	4/10
C16	8/10	8/10	-*
C17	8/10	8/10	6/10
C18	9/10	9/10	7/10

* C16 is not supported by Cortana and thus triggers no action.

in Table 3. Also, Table 4 presents three over-the-air attack examples under three different scenarios. Each attack can successfully trigger the intended action against different voice-controllable systems. Note that an adversarial command may generate multiple adversarial audio files with different playback speeds.

Overall Attack Impact: We repeat the above experiments for all 18 commands against the three voice-controllable systems. To test attack stability, we perform 10 trials of attacks with the adversarial audio for each command. Thus, we have $18 \times 3 \times 10 = 540$ tests in total. We define *success rate* as the ratio of attack attempts that successfully trigger the ASR to perform a desired action to the total number of attack trials. Table 5 presents the success rates of all commands, prompting three key observations.

First of all, all commands with different lengths in different scenarios can be converted into successful adversarial commands to stealthily trigger an ASR to execute corresponding functions, except C16 ("Cancel Route") with MS Cortana, which does not support that command (despite successfully interpreting it). These results verify the effectiveness of the proposed attack under different command lengths, scenarios, and ASR platforms.

Table 6: Percentage of adversarial audios that *xRef* successfully detects when one reference service is introduced (we use Amazon, Google, MS, and IBM to refer to Amazon Transcribe, Google STT, Microsoft Azure STT, and IBM Watson STT, respectively).

	Amazon	Google	MS	IBM
Amazon	N/A	99%	98%	98%
Google	99%	N/A	100%	100%
MS	98%	100%	N/A	95%
IBM	98%	100%	95%	N/A

Second, in both household and teleconference environments, the achieved success rates are consistently high. For household (C1-C6), the average success rates against Amazon Alexa, Google Assistant, and MS Cortana are 95%, 97%, and 93%, respectively, and 92%, 93%, and 90%, respectively, for teleconference (C7-C12). The success rates for in-vehicle (C13-C18) are decent but comparably lower, because there is much higher environmental noise (coming from the running engine and outside the car) here than in the other scenarios.

Lastly, a longer command achieves a slightly higher success rate in most cases. For example, for teleconference, both Amazon Alexa and Google Assistant achieve average success rates of 85%, 95%, and 100% for short (C7 and C8), medium (C9 and C10), and long (C11 and C12) commands, respectively. This is caused by the fact that longer commands provide richer context information, facilitating ASRs to recognize them with the pre-trained language models.

6.3 Human Comprehensibility Tests

In this section, we test the human comprehensibility of adversarial audios, which can successfully trigger target actions by the ASRs. We recruited 28 volunteers (aged 25-40 years old; fluent in English; 11 self-identified as females and 17 as males), including 18 native American-English speakers. [1] To mitigate any subjective effects (such as priming effects [38]) that may potentially influence the participants' understanding of the adversarial audio files, no voice command was disclosed to the participants before the tests.

We prepared an adversarial audio file for each command and each of the three wake-up words. Meanwhile, to demonstrate the impact of phonetic manipulation on command intelligibility, we also synthesized each original wake-up word or command with the same playback speed that the corresponding adversarial version used for comparison. Thus, we had 21 adversarial audio files and 21

[1]The survey does not cause any potential risks to the participants, such as psychological, physical, etc. The study has been reviewed and approved by our institution's IRB.

Table 7: Average detection rate for adversarial and normal audios when the number of reference services varies.

Number of references	One	Two	Three
Adversarial	98.3%	100%	100%
Normal	100%	100%	100%

normal audio files in total. The subjects were asked to listen to 18 audio files (3 and 6 adversarial audio files for wake-up words and scenario-associated voice commands, plus 9 corresponding normal audio files) under different scenarios (i.e., household, teleconference, and in-vehicle). The order in which audio files were presented to the participants was randomized. Each participant was asked to indicate whether she or he had identified any meaning in the audio. If the listener had identified any meaning, she/he was also asked to indicate what meaning she/he heard. We recorded each interpretation and calculated the WER and PER for each audio file.

The results demonstrate that none could comprehend any adversarial audio file. The corresponding WERs and PERs are consistently above 0.5, and more than half are greater than or equal to 1, which means that the corresponding adversarial audio files are completely unrecognizable. Also, the WER and PER for each adversarial audio are larger than that for the corresponding normal audio file, demonstrating that the phonetic manipulation degrades the comprehensibility of the audio.

6.4 Evaluation of *xRef*

We demonstrate the performance of *xRef* to defend against the proposed adversarial attack. We test all 100 adversarial audio files generated in Section 6.2.2.

One Reference Service: We first set one reference service for *xRef*. Thus, the ASR would send the transcription request to two transcription services. By comparing the results from both, the ASR can then determine whether the input audio is malicious or not. Table 6 presents the detection results. We see that *xRef* can always achieve a high detection rate of no less than 95%.

More Reference Services: To further increase the detection rate of adversarial audios, we can increase the number of reference services, denoted with T. We choose T ($T \in \{2, 3, 4\}$) references from the selected four transcription services, and have $\binom{4}{T} = \frac{24}{T!(4-T)!}$ possibilities. We calculate the detection rate for every possibility, and then compute the average detection rate for all possibilities. To verify that *xRef* does not misrecognize a voice command spoken at a normal speed as an adversarial one, we generate normal audio files of all 100 commands spoken at a normal speed, which varies from 1.0× to 1.9×, with increments of 0.1×. We then send generated normal audios to different transcription services and compare the corresponding results. If recognition results of different transcription services for a same normal audio file are consistent, we consider that the normal audio is successfully detected.

Table 7 compares the average detection rates from adversarial and normal audios when *xRef* utilizes different numbers of reference services. We observe that for normal audios, the detection rate is always 100% regardless of the number of utilized transcription services, and for adversarial audio, when *xRef* employs two or three reference services, the detection rate reaches 100%.

7 RELATED WORK

Due to their increased affordability and convenience, ASR systems are becoming integral components of our smart home/vehicle/office environment. Meanwhile, there are emerging adversarial attacks aiming to compromise ASRs, mainly including audio attacks and misinterpretation-based attacks.

Audio Attacks: In general, existing adversarial audio attacks can be divided into the following types:

White-box Audio Attacks: Attacks in this category require complete white-box knowledge (i.e., internal structures or workings) of the target attack system (e.g., [15, 16, 34, 47, 54]). For example, by exploiting the characteristics of DNN-based ASR systems, [47] proposes an adversarial attack with psychoacoustic modeling to reduce the perceptible noise. Those attacks usually adopt an open-source speech-to-text engine such as DeepSpeech [28], Kaldi [42], or Lingvo [48], as a concrete attack target, and embed hidden voice commands inside audio samples such as adversarial noise [15, 47] or music [54]. The audio adversarial examples in initial studies (e.g., [15, 47]) only work over-the-wire and do not remain adversarial being played over-the-air, and a recent study [16] successfully generates audio adversarial examples that are effective over-the-air, but as commercial ASRs are proprietary, it is unlike in practice for an adversary to know all details of the attacked system. Such white-box attacks are thus limited to practical systems such as Amazon Alexa or Google Assistant.

Black-box Audio Attacks: The adversary does not need to know the specifics about the victim system under a black-box setting (e.g., [4, 5, 13, 31, 52]). Instead, the adversary just knows that the target system employs a transcription model of certain language (e.g., American English) and such other information that is generally available for any public ASRs [5]. For example, [31] proposes a framework to perform black-box attacks leveraging evolutionary multi-objective optimization, while it only tests the framework on two ASRs (Deepspeech and Kaldi-ASR); [13] proposes adversarial audio attacks against Google Assistant using nonsensical word sequences which have some phonetic similarity with a relevant target command; [4] exploits knowledge of the signal processing algorithms commonly used by ASRs and attacks the feature extraction module to generate audio samples, which are correctly transcribed by various practical ASRs while sounding like unintelligible noise. Besides, [52] focuses on the security of speaker recognition systems and designs adversarial audio examples to obtain access to ASR, while the generated audio is not necessary to be imperceptible. Different from existing black-box audio attacks, our work combines phoneme manipulation with fast speech in an innovative manner, and crafts adversarial audio that is incomprehensible to humans but can be recognized by off-the-shelf ASRs.

Hardware-assisted Attacks: With the help of a dedicated signal generator and ultrasound speaker hardware, recent studies (e.g., [46, 50, 55]) demonstrate success in generating inaudible attack by leveraging the nonlinearity of the microphone circuits, which makes high frequency signals arriving at a microphone get shifted to lower frequencies. Another work [53] creates inaudible attack against ASRs, while it requires a waveform generator and a Piezoelectric (PZT) transducer designed for exciting ultrasonic guided wave. Hardware dependence on launching these inaudible attacks, however, incurs the extra hardware requirement. Meanwhile, as injected ultrasonic audio is not within the range of human hearing, it can easily filtered out. Also, by finding indelible traces of non-linearity in recorded voice signals, the inaudible ultrasonic attack can be detected [46]. Our work, on the contrary, is easily-constructed and has no extra hardware requirement.

Misinterpretation-based Attacks: [32, 57] also exploit ASR misinterpretations to construct adversarial black-box attacks, which can surreptitiously cause users to trigger malicious applications. Except for the attack goals, the misinterpretation sources of these attacks also differ from our attack. [32, 57] both utilize empirical speech errors, and thus the effectiveness of attacks built on such misinterpretations highly depends on the size of utilized training datasets. [57] also leverages other types of linguistic issues (such as regional vocabulary) to craft vApp (voice assistant applicaton) squatting attacks. Our work, on the contrary, is the first to explore the misinterpretations introduced by fast speech, and to successfully create adversarial audio attacks with manipulated fast speech.

8 CONCLUSION

We propose *CommanderGabble* for a model-agnostic and easily-constructed adversarial attack against off-the-shelf ASR systems. It is the first to point out the vulnerability of current ASR systems in dealing with fast speech, and it successfully crafts adversarial audio by combining phoneme manipulation with fast speech. *CommanderGabble* is able to manipulate the phonetic structure of a voice command and synthesize it into a sped-up, human-unintelligible, adversarial audio version that ASR interprets as the original. Extensive experiments show the effectiveness and robustness of *CommanderGabble* under different practical scenarios. We further suggest realistic options for manufacturers to defend against this attack.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant No.1948547.

REFERENCES

[1] 2014. The CMU Pronunciation Dictionary (version 0.7b)[Online. http://www.speech.cs.cmu.edu/

[2] 2021. *Amazon Mechanical Turk.* https://www.mturk.com

[3] 2021. *LOGIOS Lexicon Tool.* http://www.speech.cs.cmu.edu/tools/lextool.html

[4] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. 2019. Practical Hidden Voice Attacks against Speech and Speaker Recognition Systems. In *Proceedings 2019 Network and Distributed System Security Symposium (NDSS).* Internet Society.

[5] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. 2020. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. *arXiv e-prints* (2020), arXiv–2007.

[6] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. 2017. Did you hear that? Adversarial examples against automatic speech recognition. In *31st Conference on Neural Information Processing Systems (NIPS 2017).*

[7] Amazon. 2020. *About the Automatic Speech Recognition (ASR) Evaluation tool.* https://developer.amazon.com/en-US/docs/alexa/asr/about-asr.html

[8] Amazon. 2020. New Alexa Features for the Vehicle. https://www.amazon.com/b?ie=UTF8&node=21439303011

[9] Karm Veer Arya and Robin Singh Bhadoria. 2019. *The Biometric Computing: Recognition and Registration.* CRC Press.

[10] International Phonetic Association. 2020. IPA charts and subcharts in four fonts. https://www.internationalphoneticassociation.org/

[11] Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, et al. 2007. Automatic speech recognition and speech variability: A review. *Speech communication* 49, 10-11 (2007), 763–786.

[12] Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, Richard Rose, et al. 2006. Impact of variabilities on speech recognition. In *Proceedings of the 11th International Conference on Speech and Computer (SPECOM).* 3–16.

[13] Mary K. Bispham, Ioannis Agrafiotis, and Michael Goldsmith. 2019. Nonsense Attacks on Google Assistant and Missense Attacks on Amazon Alexa. In *Proceedings of the 5th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*. INSTICC, SciTePress, 75–87.

[14] 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 (USENIX Security 16).* USENIX Association, Austin, TX, 513–530.

[15] N. Carlini and D. Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops (SPW).* 1–7.

[16] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. 2020. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *Proceedings 2020 Network and Distributed System Security Symposium (NDSS).*

[17] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. 2020. 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).* 2667–2684.

[18] Robert Daland, Bruce Hayes, James White, Marc Garellek, Andrea Davis, and Ingrid Norrmann. 2011. Explaining sonority projection effects. *Phonology* (2011).

[19] Mark Davies and Dee Gardner. 2009. *A Frequency Dictionary of Contemporary American English.* Taylor & Francis.

[20] Hope Dawson, Michael Phelan, et al. 2016. *Language files: Materials for an introduction to language and linguistics.* The Ohio State University Press.

[21] Pierre Delattre and Carroll Olsen. 1969. Syllabic features and phonic impression in English, German, French and Spanish. *Lingua* 22 (1969), 160–175.

[22] Daniel Fogerty and Diane Kewley-Port. 2009. Perceptual contributions of the consonant-vowel boundary to sentence intelligibility. *The Journal of the Acoustical Society of America* 126, 2 (2009), 847–857.

[23] E. C. Fudge. 1969. Syllables. *Journal of Linguistics* 5, 2 (1969).

[24] Google. 2020. Android Auto. https://www.android.com/auto/

[25] Google Cloud. 2020. *Method: text.synthesize.* https://cloud.google.com/text-to-speech/docs/reference/rest/v1/text/synthesize

[26] Google Cloud. 2020. *Speech-to-Text.* https://cloud.google.com/speech-to-text

[27] Google Cloud. 2020. *Text-to-Speech.* https://cloud.google.com/text-to-speech/

[28] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[29] Florian Hönig, Georg Stemmer, Christian Hacker, and Fabio Brugnara. 2005. Revising perceptual linear prediction (PLP). In *Ninth European Conference on Speech Communication and Technology.*

[30] Diane Kewley-Port, T Zachary Burkle, and Jae Hee Lee. 2007. Contribution of consonant versus vowel information to sentence intelligibility for young normal-hearing and elderly hearing-impaired listeners. *The Journal of the Acoustical Society of America* 122, 4 (2007), 2365–2375.

[31] Shreya Khare, Rahul Aralikatte, and Senthil Mani. 2018. Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization. *arXiv preprint arXiv:1811.01312* (2018).

[32] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill Squatting Attacks on Amazon Alexa. In *Proceedings of the 27th USENIX Conference on Security Symposium* (Baltimore, MD, USA) *(SEC'18).* USENIX Association, USA, 33–47.

[33] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[34] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. 2020. AdvPulse: Universal, Synchronization-Free, and Targeted Audio Adversarial Attacks via Subsecond Perturbations. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, USA) *(CCS '20).* Association for Computing Machinery, New York, NY, USA, 1121–1134.

[35] Richard P Lippmann. 1997. Speech recognition by machines and humans. *Speech communication* 22, 1 (1997), 1–15.

[36] Taylor Martin, David Priest, Dale Smith, and Andrew Gebhart. 2020. *Every Google Assistant command for your Nest speaker or display.* https://www.cnet.com/how-to/every-google-assistant-command-for-your-nest-speaker-or-display/

[37] Peter Hugoe Matthews and Peter Hugoe Matthews. 2014. *The concise Oxford dictionary of linguistics.* Oxford University Press.

[38] David E Meyer and Roger W Schvaneveldt. 1971. Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal*

of experimental psychology 90, 2 (1971), 227.

[39] N. Mirghafori, E. Fosler, and N. Morgan. 1996. Towards robustness to fast speech in ASR. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, Vol. 1. 335–338.

[40] Lindasalwa Muda, Mumtaj Begam, and I Elamvazuthi. 2010. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *Journal of Computing* 2, 3 (2010), 138–143.

[41] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[42] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society.

[43] David Priest, Tauren Dyson, and Taylor Martin. 2020. *Every Alexa command you can give your Amazon Echo smart speaker.* https://www.cnet.com/how-to/every-alexa-command-you-can-give-your-amazon-echo-smart-speaker/

[44] Emil Protalinski. 2019. *ProBeat: Has Google's word error rate progress stalled?* https://venturebeat.com/2019/05/10/probeat-has-googles-word-error-rate-progress-stalled/

[45] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. 2019. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International conference on machine learning*. PMLR, 5231–5240.

[46] Nirupam Roy, Sheng Shen, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Inaudible voice commands: The long-range attack and defense. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 547–560.

[47] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2019. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. In *Proceedings 2019 Network and Distributed System Security Symposium (NDSS)*. Internet Society.

[48] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295* (2019).

[49] M. A. Siegler and R. M. Stern. 1995. On the effects of speech rate in large vocabulary speech recognition systems. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1. 612–615 vol.1.

[50] Liwei Song and Prateek Mittal. 2017. POSTER: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2583–2585.

[51] Constantin Spille, Birger Kollmeier, and Bernd T Meyer. 2018. Comparing human and automatic speech recognition in simple and complex acoustic scenes. *Computer Speech & Language* 52 (2018), 123–140.

[52] Henry Turner, Giulio Lovisotto, and Ivan Martinovic. 2019. Attacking Speaker Recognition Systems with Phoneme Morphing. In *European Symposium on Research in Computer Security*. Springer, 471–492.

[53] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. 2020. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In *Network and Distributed Systems Security (NDSS) Symposium.*

[54] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, XiaoFeng Wang, and Carl A. Gunter. 2018. CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 49–64.

[55] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible Voice Commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 103–117.

[56] N. Zhang, X. Mi, X. Feng, X. Wang, Y. Tian, and F. Qian. 2019. Dangerous Skills: Understanding and Mitigating Security Risks of Voice-Controlled Third-Party Functions on Virtual Personal Assistant Systems. In *2019 IEEE Symposium on Security and Privacy (SP)*. 1381–1396.

[57] Yangyong Zhang, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, and Guofei Gu. 2019. Life after Speech Recognition: Fuzzing Semantic Misinterpretation for Voice Assistant Applications. In *Proceedings 2019 Network and Distributed System Security Symposium (NDSS)*. Internet Society.

[58] Zoom. 2021. Video Conferencing, Web Conferencing, Webinars, Screen Sharing - Zoom. https://zoom.us/