AdvCAPTCHA: Creating Usable and Secure Audio CAPTCHA with Adversarial Machine Learning

Hao-Ping (Hank) Lee†, Wei-Lun Kao‡, Hung-Jui Wang‡, Ruei-Che Chang*, Yi-Hao Peng†, Fu-Yin Cherng§, Shang-Tse Chen‡
Carnegie Mellon University†, National Taiwan University‡,
University of Michigan, National Chung Cheng University§*
haopingl@cs.cmu.edu, {b07902027, r10922061}@csie.ntu.edu.tw, rueiche@umich.edu, yihaop@cs.cmu.edu, fuyincherng@cs.ccu.edu.tw, stchen@csie.ntu.edu.tw

***Abstract*—Audio CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is an accessible alternative to the traditional CAPTCHA for people with visual impairments. However, the literature has found that audio CAPTCHA suffers from both lower usability and security than its visual counterpart. In this paper, we propose AdvCAPTCHA, a novel audio CAPTCHA generated by using adversarial machine learning techniques. By conducting studies with people with and without visual impairments, we show that AdvCAPTCHA can outperform the *status quo* audio CAPTCHA in security but not usability. We demonstrate AdvCAPTCHA's feasibility of providing detection of malicious attacks. We also present an evaluation metric, *thresholding*, to quantify the trade-off between usability and security for audio CAPTCHA design. Finally, we discuss approaches to the real-world adoption of AdvCAPTCHA.**

I. Introduction

CAPTCHA is a technique to determine whether the user is human or not to protect online systems from bots. The typical CAPTCHA consists of a series of visual-oriented tests that are approachable for humans but challenging for machines to solve (e.g., recognizing distorted hand-written characters). However, such visual-based tasks are not accessible to people with visual impairments (PVIs). As a result, audio CAPTCHA — which prompts users with continuous audio cues — has been developed as the alternative to support the access of PVIs.

The *status quo* audio CAPTCHA tasks are designed with a similar concept as its visual-based counterpart: users will be asked to type in heard audio digits or words that are intentionally designed to be challenging for machines to recognize. Specifically, these tasks are usually composed of several alphanumeric or words spoken at variant speeds and pitches, combined with ambient background noise. These elements are brought in to create randomness, preventing the system from automated speech-to-text attacks.

Though audio CAPTCHA makes the online security mechanism more accessible, prior studies found that audio CAPTCHAs are challenging and time-consuming to solve for users, especially for PVIs [9], [12]. Moreover, several potential security issues have been discovered in audio CAPTCHA such that the verification tasks can be solved by using a combination of machine learning techniques [10], [28], [30], [32].

To strike the balance between the security and the usability of the audio CAPTCHA, we present AdvCAPTCHA, a novel computational method that leverages adversarial machine learning techniques to generate audio CAPTCHAs to defend against speech-to-text models while preserving the CAPTCHAs' usability under perturbations. We draw our design from adversarial machine learning techniques in the audio domain, which have been a popular methodology to produce adversarial examples via computed perturbations that break machine learning-based speech-to-text applications [2], [14]. Additionally, such perturbations could potentially be generated to be imperceptible by humans [27], making the perturbed audio sound close to the original audio. We design a total of three AdvCAPTCHA variants that are based on different audio perturbation mechanisms, including Kenan, Devil, and Volcano. To summarize, Kenan works by using the Discrete Fourier Transform to break down audio into components, then selectively removes those below a certain intensity threshold to create speech that is still intelligible to humans but causes machines to mistranscribe it; Devil is generated through an adaptive audio perturbation approach via a substitute model trained on data from speech-to-text applications, to create human-imperceptible background noises that cause machines to mistranscribe, while being less disturbing for humans; and finally, Volcano combines audio digits that are only recognizable for machines and audio digits only recognizable for humans to target text-to-speech applications to output predefined digits different from the real CAPTCHA task answers.

We examine a comprehensive evaluation of AdvCAPTCHAs (Section V) with locally trained machine learning models and commercial automatic speech recognition (ASR) services. We use the currently common audio CAPTCHA design scheme as a baseline for the comparison. We found that the three AdvCAPTCHAs could yield an estimated attack success rate (i.e., a malicious attacker can computationally solve the audio CAPTCHA task with commercial ASR services) of 0.22%, 31.91%, and 61.19%, respectively, outperforming the baseline's 70.10%. Furthermore, we demonstrate AdvCAPTHAs' capability to perform detection for attacks — e.g., honeypot[1] — and can identify around 99% of the commercial ASR speech-to-text attempts. Such a property can further enhance AdvCAPTHAs' defense capability in real-world settings.

Symposium on Usable Security and Privacy (USEC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-5-2
https://dx.doi.org/10.14722/usec.2024.23035
www.ndss-symposium.org, https://www.usablesecurity.net/USEC/

[1] https://en.wikipedia.org/wiki/Honeypot_(computing)

We also conducted a two-fold evaluation with 121 sighted users and 11 PVIs to better understand the usability of the AdvCAPTCHAs (Section VI). Our findings reveal that AdvCAPTCHAs' usability is, overall, lower or, at most, comparable to the baseline method. Finally, we propose *thresholding* (Section VII) — an evaluation metric that quantifies the trade-off between AdvCAPTCHAs' security and usability — to find the optimal balance between security and usability based on design priorities.

In summary, the contributions of this work are three-fold:

- AdvCAPTCHAs, compared with the *status quo* audio CAPTCHA design, provide better security against both locally trained machine learning models and commercial ASR models with the potential to preserve comparable usability.
- AdvCAPTCHAs demonstrate the feasibility of providing detection of attacks. To our knowledge, this is the first work to implement honeypot-like attack detection in the context of audio CAPTCHA.
- We proposed an evaluation metric — *thresholding* — that quantifies the trade-off between usability and security in CAPTCHA design.

II. Related Work

A. Adversarial Machine Learning in Audio

Adversarial machine learning techniques have been shown effective in deceiving machine learning models [31]. When applied to the audio domain, the adversarial attacks can lead machines to perceive audio differently from what humans hear [14]. These methods were first used to attack voice-controlled devices. *Hidden voice command* [13] generates sounds that are unintelligible to humans but are understandable as a specific command to mobile devices. *DolphinAttack* [35] uses ultrasonic carriers to produce inaudible attacks that can trigger several smart home and mobile devices while remaining unnoticeable by humans.

Instead of producing unintelligible or inaudible audio, [14] adds a small perturbation to an existing audio clip. The perturbation has little effect on human understanding but can make the perturbed audio be recognized as a completely different target sentence. Subsequent works have extended these attacks in several aspects. For example, [27] leverages the psychoacoustic principle to make the perturbation almost imperceptible to the human ear. [25] perturbs a music clip instead of speech audio and thus can be used in a different attack scenario. [1] extends the *hidden voice command* attack to the more practical black-box attack setting — i.e., attackers have no access to model parameters.

The same "attack" methods can be used as defenses in the CAPTCHA design. Shekhar et al. [29] proposed to use adversarial examples to enhance the robustness of audio CAPTCHA. However, they only use simple classifiers instead of the more powerful ASR models as the attacker. In this paper, we identify and leverage two existing audio perturbations helpful in designing a secure and usable audio CAPTCHA: *Hear "No Evil", See "Kenansville"* [2] and *Devil's whisper* [15]. Note that when applying adversarial machine learning in the CAPTCHA designs, we need to select a model as the target victim model for adversarial example generation. Such model will be referred to as *victim model* in this paper.

B. Security and Usability of Audio CAPTCHAs

The security of audio CAPTCHA [16] has been long studied in the literature. Researchers have tried to "break" audio CAPTCHA challenges using machine learning models trained by annotated audio CAPTCHA corpus. For example, Tam et al. [32] were able to use SVM to break the older version of Google audio reCAPTCHA with 45% success rate; Bursztein and Bethard [11], on the other hand, used a supervised learning algorithm to break 75% of audio CAPTCHAs on eBay. In a more recent study, Sano et al. [28] leveraged hidden Markov models and built a reCAPTCHA solver with a 52% success rate. In another research thread, instead of training their own speech recognition algorithms, researchers consider threat models in which attackers may utilize accessible commercial speech-to-text services to attack audio CAPTCHAs and achieve high success rates. For example, Solanki et al. [30] tested a set of state-of-the-art commercial ASR on breaking common audio CAPTCHA services, and demonstrated that it is both applicable and profitable ($485.3 per day by breaking Google's reCAPTCHA v2.0) for attackers to use such an attack system. Bock et al. [10] proposed *unCaptcha*, which also leveraged from a set of ASR services, and achieved an 85.15% success rate on breaking reCAPTCHA challenges. Following prior work, we consider both machine learning models and commercial ASR models as threat models to evaluate the security of our proposed AdvCAPTCHAs, which will be elaborated in Section III.

While audio CAPTCHAs are designed for PVIs as accessible alternatives, they have also been long condemned for their poor usability [9], [12] or privacy [4], [7]. Specifically, a prior study found that audio CAPTCHAs on common websites were harder and required more time to solve than the visual-based counterparts [12]. Thus, researchers have proposed alternative designs to improve the usability, either combining modalities beyond audio [20], looping in external human assistants [36], or introducing additional puzzles or mechanisms on top of the audio representations [5], [7]. Depending on the type of tasks that users are required to complete, audio CAPTCHAs could be classified into content-based and rule-based [18]. For the former, users are normally asked to perform "speech-to-text" tasks, which is the most common type of design; for the latter, users will be asked to solve quizzes such as simple math calculation [18]. Such audio CAPTCHAs, while mitigating the need for short-term memory (i.e., memorizing the words you heard) when solving CAPTCHA tasks, their security were compensated as they could be easily solved by attacks with naive threat models, and could be even more vulnerable to attackers with prior knowledge of these audio CAPTCHAs (e.g., easily performing white-box attack) [18]. Considering the low security of rule-based audio CAPTCHAs currently achieve, it may not be surprising that while content-based CAPTCHAs may be less usable, they are still the *status quo* audio CAPTCHAs. In this paper, we focus on improving content-based audio CAPTCHAs by enhancing their usability and security using adversarial machine learning.

III. Threat Model and Hypothesis

A. Low-Resourced and Well-Resourced Attackers

We considered two common threat models in the literature when designing AdvCAPTCHAs:

1) Low-resourced attackers: One common threat model, which is also introduced in prior work (e.g., [10]), is an assumption that attackers, instead of training machine learning models from scratch to break audio CAPTCHAs, would take a more cost-effective approach to leverage from models that are ready at hand. Specifically, prior work demonstrates promising attack success rates to extant audio CAPTCHA by leveraging commercial ASR services [10], [30]. In this work, we consider the circumstances where attackers use commercial ASR services — e.g., Google Cloud Speech, Microsoft Azure — to implement solvers of AdvCAPTCHAs.

2) Well-resourced attackers: We also considered well-resourced attackers with sufficient computing resources to self-train deep models (e.g., access to large RAM and GPU, multiple computers, and unlimited IP addresses). Additionally, we consider such well-resourced attacks with both *black-box* and *white-box* scenarios. Here, we borrow the concepts of black-box attack and white-box attack from the adversarial machine learning literature, of which the former refers to the attacker having no access to the information of the "targets" (e.g., CAPTCHA tasks) when building the attack mechanism, and the latter refers to the attacker having access to such information when preparing their attack.

a) Black-box Attack: We consider scenarios when attackers have no prior knowledge of AdvCAPTCHA, nor do they obtain task samples of AdvCAPTCHA. Given such an assumption, attackers will fine-tune publicly available pre-trained speech models with audio digit datasets to improve attack success rates to audio CAPTCHAs and speed up model training time [3].

b) White-box Attack: We also consider scenarios when attackers are able to get prior access to AdvCAPTCHA examples (e.g., storing AdvCAPTCHA tasks given any occurrence) before training their models. These AdvCAPTCHA examples could then be further used to fine-tune attackers' models to improve attack success rates to AdvCAPTCHA [3].

B. Targeted and Untargeted Defense of AdvCAPTCHA

We also incorporate *attack detection* in the design of AdvCAPTCHA. Specifically, we create AdvCAPTCHA tasks as targeted adversarial examples, which can mislead attacker models to output our pre-defined digits. For example, a targeted-defense AdvCAPTCHA can mislead machine learning models to output "5" when humans perceive it as "3." Such manipulated human-machine mismatches could distinguish automated adversaries from actual human users. It is also noteworthy that, most of the time, the implementation of attacker models remains unknown, and thus targeted-defense AdvCAPTCHA tasks are also required to be transferable across different machine learning models that could be encountered in the real world. Yet, transferable and targeted adversarial examples in the audio domain have been shown to be extremely challenging [3]. This work proposes a simple yet effective methodology to produce transferable and targeted defense (see Section IV). On the other hand, for AdvCAPTCHA tasks that solely aim to cause ASR models to mistranscribe the correct answers, we refer to them as *untargeted-defense AdvCAPTCHA*.

IV. Designing Adversarial ML-Infused Audio CAPTCHA

The design objective of AdvCAPTCHA is to create audio CAPTCHAs that better resist attacks from both commercial ASR services and machine learning-based approaches while maintaining their comprehensibility to human users. In total, we created two untargeted-defense AdvCAPTCHAs: *Kenan* and *Devil*, and one targeted-defense AdvCAPTCHA: *Volcano*.

A. Kenan AdvCAPTCHA Design

This audio CAPTCHA was created on top of the *"Kenansville" Perturbation* proposed by Abdullah et al. [2], in which the authors successfully created speech audio that made the state-of-the-art commercial ASR models mistranscribe and misidentify. In short, the perturbation builds upon the assumption that commercial ASR models recognize speech audio from its specific characteristics. The authors proposed the approach to remove the least amount of speech components from the audio associated with such characteristics that generate speech audio, causing intentional mistranscriptions to commercial ASR models. Modifying speech audio will also impact a human's ability to recognize the audio. Thus, we engineered and created a data pipeline to create Kenansville Perturbation speech audio, while fine-tuning the pipeline to ensure the audio is still perceptible by humans.

We first decompose the given audio into individual audio components and retrieve their corresponding intensities via the Discrete Fourier Transform. Then, we iteratively select a threshold that makes a "victim" commercial ASR model (e.g., Google Cloud Speech) mistranscribe the audio. We then remove all components whose intensity falls below the threshold. The same process is applied to all of the other victim ASR models. We refer to the AdvCAPTCHA design as *Kenan* in the rest of the paper.

B. Devil AdvCAPTCHA Design

The design of this audio CAPTCHA is inspired by the *Devil's Whisper Perturbation* proposed by Chen et al. [15], in which the authors aimed to generate human-imperceptible background noises that make the victim commercial ASR models mistranscribe. We create similar background noises to replace the ones used in the *status quo* audio CAPTCHA designs, which are disturbing and require higher cognitive loads to solve the CAPTCHA tasks [12], [34]. By using the background noises created via Devil's Whisper Perturbation, we can create background noises with stronger mistranscribing capability, yet potentially less disturbing for humans to solve CAPTCHA tasks.

We first train a local substitute model with data obtained from querying the victim commercial ASR model (e.g., Google Cloud Speech). Using the substitute model, we generate adversarial examples that make the victim ASR model mistranscribe. Additionally, to ensure the effects of the adversarial example

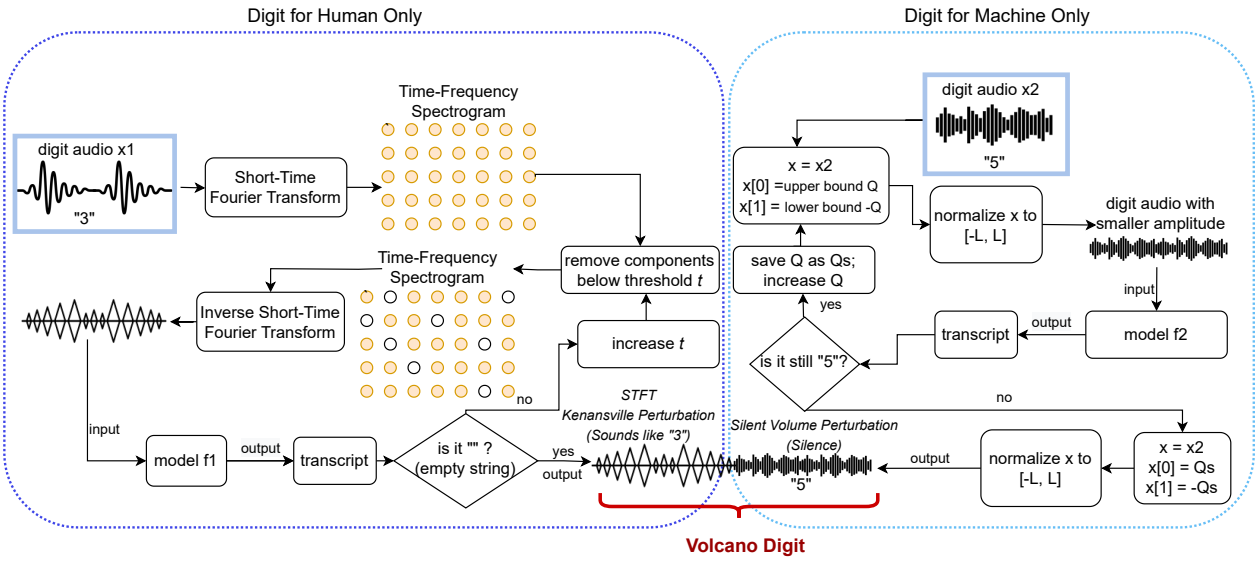

Fig. 1. Pipeline to generate Volcano AdvCAPTCHA. Volcano AdvCAPTCHA combines the Digit-For-Human-Only (purple box on the left) and the Digit-For-Machine-Only (blue box on the right). The two navy blue rectangles represent the two input clean digits, digit audio x1 and x2; x[0] and x[1] represent the first and second elements of the digit audio x2, respectively.

are transferable to the actual victim commercial ASR model, we iteratively increase the intensity of audio perturbation until the resulting adversarial examples are also mistranscribed by the victim ASR model. The same process is applied to all the other victim ASR services. To enhance the imperceptibility of the resulting perturbed audio, we incorporate the gradient-based, white-box adversarial attack [27] into the AdvCAPTCHA tasks generation pipeline. In the rest of the paper, we refer to this AdvCAPTCHA design as *Devil*.

C. Volcano AdvCAPTCHA Design

We also create a new speech-to-text adversarial perturbation — *Silent Volume-Kenansville Perturbation (Volcano Perturbation)* — to enable attack detection by targeting text-to-speech applications to output pre-defined digits different from the real CAPTCHA task answers. The perturbation combines audio digits that are only recognizable for machines (Digit-For-Machines-Only) and those only recognizable for humans (Digit-For-Humans-Only) (see Figure 1).

1) Digits-For-Humans-Only Module — Short Time Fourier Transform Kenansville Perturbation: To create Digits-For-Humans-Only, we adapt the data pipeline from the Kenansville Perturbation (described in Section IV-A) and use Short-Time Fourier Transform (STFT) as the audio decomposition algorithm, as shown in Figure 1. Specifically, STFT splits the audio into short time windows before conducting a Discrete Fourier Transform (DFT) on each window segment and decomposes the input speech audio into finer components. It is noteworthy that the goal is to create speech audio that is undetectable — i.e., return 'void' — by speech-to-text applications, different from the original Kenansville Perturbation proposed by Abdullah et al. [2] which uses DFT instead of STFT, and aims only to make speech-to-text applications mistranscribe.

2) Digits-For-Machine-Only Module — Silent Volume Perturbation: We generate Digits-For-Machines-Only by decreasing the volume of clean-digit audio. Prior work has shown that with intended distortions, speech audio could be generated as machine-transcribable but sounds like noises to humans [1]. After some experiments, however, we found that distorted audio digits were much less transferable across speech-to-text applications than merely reducing the audio volume. Based on this observation, we minimize the volume of clean digits to the point where text-to-speech applications can still transcribe them correctly while being hardly comprehensible to humans (see Figure 1). For each victim commercial ASR model, we iteratively search for the minimal volume of clean digit audio that is still transcribable by the model. We found that the generated digits may sometimes result in audio volumes that are still perceptible to humans. To further ensure the human imperceptibility of these digits, we utilized Otsu's method [26] and further removed digits with a higher volume — for all generated digits, we calculated an Otsu's threshold and filtered all the digits with volume higher than such a threshold.

Finally, after generating Digits-For-Humans-Only and Digits-For-Machines-Only, we concatenate the two types of digits generated from the two modules to create Volcano.

D. Set Up

While various sets of characters (or words) have been used as the corpus of audio CAPTCHA tasks [9], [12], in this preliminary exploration, we select one of a common subset — numeric digits (i.e., 0-9) — to simplify security evaluation, and to minimize the negative impacts caused by the similarity of characters' phonemic (i.e., 'q' and 'u', '8' and 'a') on users' performance and perceived usability [18], [34]. We generate all of our CAPTCHA tasks from numeric digits created by the Google Cloud text-to-speech API[2], which provides a clean audio dataset containing twenty speakers with various pitches and tones.

[2]https://cloud.google.com/text-to-speech/

To provide a control group for AdvCAPTCHA, we include a baseline design that emulates the industry-wide standard design, of which we refer to the design of Securimage[3], an open-source CAPTCHA project. The resulting baseline CAPTCHA tasks contain audio digits of various pitches and speeds and background noises, which are common audio CAPTCHA techniques to enhance the difficulty of examining speech-to-text attacks [9], [12], [34]. We refer to the baseline audio CAPTCHA design as *Baseline* in the rest of the paper.

When conducting security evaluation for audio CAPTCHA tasks, we follow prior work to segment each audio CAPTCHA task into digits before examining speech-to-text transcription/recognition [10]. Specifically, we generate our audio CAPTCHA digits directly from digit audio clips in our audio dataset created via the Google Cloud text-to-speech API described above. For usability evaluation, we follow the format of audio CAPTCHA tasks of prior work [18], with each audio CAPTCHA task consisting of one-second silence at the beginning and a 1.25-second timespan between each digit. The resulting CAPTCHA tasks are 10-12 seconds.

V. Security Evaluation

Following the threat models described in Section III, the security of AdvCAPTCHA will be evaluated with two types of speech-to-text adversary attacks that utilize: 1) commercial ASR services and 2) machine learning models trained in black-box attack and white-box attack settings, respectively.

A. Experiment Setup

1) Attack Using Commercial Automatic Speech Recognition Models: We select five representative commercial ASR models: *Google Cloud Speech (Default Model)*, *Google Cloud Speech (Phone Call Model)*, *Microsoft Azure*, *IBM*, and *Facebook Wit*. These models have been used to represent the state-of-the-art commercial ASR services in prior studies [2], [10], [30], and are easily accessible for attackers.

We feed each generated audio CAPTCHA digit clip (as described in Section IV-D) into the aforementioned commercial ASR models and record the transcription results. We further utilize optimization techniques that can potentially boost the attack success rates. Specifically, we follow prior works to 1) map raw transcriptions to digits that share similar/same phonemes [10], [30], and 2) build an ensemble model that produces transcriptions based on the outputs of all the five selected commercial ASR models [10], which we refer to as *Ensemble* in the following sections.

We follow the *unCaptcha* approach [10] to attack audio CAPTCHA tasks and conduct an *exact-homophone mapping*, in which we map transcriptions results to digits that share the same phonemes (e.g., 'for' and 'four'). For the *Ensemble* model, we use majority voting to produce the final digit output based on all five commercial ASR model outputs. Finally, whenever ASR models do not provide a valid digit output or when draws occur in the *Ensemble* model (e.g., 'one' and 'five' receive the same votes), we randomly select among all digits (i.e., 0-9) or the ones with the highest votes. The experiment was conducted in May 2021.

[3]https://www.phpcaptcha.org/

2) Attack Using Self-trained Machine Learning Models: We consider both the black-box and white-box scenarios when examining our machine learning-based attacks on AdvCAPTCHA. To recap, we assume attackers performing black-box attacks have no prior knowledge of how our AdvCAPTCHA is generated, except that AdvCAPTCHA is made from audio digits created from the Google Cloud text-to-speech API. Thus, we self-train a local ASR model with numeric digits created by Google Cloud text-to-speech API. To speed up our model training, we fine-tune a DeepSpeech2.0 [6] [4] model pre-trained on the Librispeech [5] dataset. Since the output space of the DeepSpeech model is character-based, for each input audio digit, we chose the digit with the minimum Levenshtein Distance [6] from the output string. We refer to the model as *Black-box Self-trained Model* in the rest of the paper. We feed audio digit clips of the three types of AdvCAPTCHA into the Black-box Self-trained Model and record the results.

On the other hand, we assume attackers who perform white-box attacks collect AdvCAPTCHA examples and use them to fine-tune their local DeeepSpeech models. To avoid over-fitting, we add supplemental corpus from *Fluent Speech Command* [7] to the training data. Following prior work [15], we use seven hours of audio corpus from *Fluent Speech Command* in addition to the AdvCAPTCHA examples of around one hour. We refer to those models as *White-box Self-trained Models*. We report the average accuracy of five-fold cross-validation for White-box Self-trained Models.

3) Evaluation Metrics: For each type of audio CAPTCHA (i.e., Baseline, Kenan, Devil, Volcano), we measure its security of the digit-level Average Digit Error Rate (DigitErrRate), and the task-level Estimated Attack Success Rate (AtkSucRate). The Average Digit Error Rate represents the errors in digit classification of a CAPTCHA scheme against speech-to-text applications — e.g., the higher the error rate, the higher the chances an ASR model fails to generate a correct corresponding digit transcription of an audio CAPTCHA. The Estimated Attack Success Rate, on the other hand, is a security measurement estimated based on the Average Digit Error Rate of an audio CAPTCHA scheme against a speech-to-text application (e.g., ASR model) to represent the probability of an attacker using that speech-to-text application to break a six-digit audio CAPTCHA task.

We set up our experiment based on our threat models (see Section III): we assume attackers will first clip the six digits in an audio CAPTCHA task and recognize each digit separately. It is also noteworthy that we quantify the trade-off between human errors (as a proxy of usability) and security by setting a parameter T for each CAPTCHA design — by correctly answering at least T digits in a six-digit audio CAPTCHA task, the task is considered passed (the client is a human). In practice, the optimal T for each CAPTCHA design should maximize the chances of humans being classified as humans while minimizing the chances of machines being classified as humans. The task-level Estimated Attack Success Rate of

[4]https://github.com/SeanNaren/deepspeech.pytorch

[5]https://www.openslr.org/12/

[6]https://en.wikipedia.org/wiki/Levenshtein_distance

[7]https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/

TABLE I. THE DIGIT ERROR RATES AND ESTIMATED SUCCESS RATES FOR LOW-RESOURCED ATTACKERS ATTACKING EACH CAPTCHA DESIGN WITH COMMERCIAL ASR MODELS. THE MINIMUM SQUARE ERROR (MSE) FOR EACH CAPTCHA INDICATES THE LEVEL OF AUDIO PERTURBATION.

	Avg. MSE	Google Cloud Speech (Default Model)		Google Cloud Speech (Phone Call Model)		Microsoft Azure		IBM		Facebook Wit		Ensemble	
		Avg. digit error rate	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*
Baseline	2.81e-2	32.50%	9.46%	9.88%	53.57%	27.25%	32.39%	67.50%	0.12%	52.38%	1.17%	5.75%	70.10%
Kenan	7.45e-4	88.75%	0.01%	85.62%	0.03%	35.00%	31.91%	77.50%	0.28%	74.38%	0.52%	35.00%	31.91%
Devil	5.33e-4	23.57%	56.82% (19.93%, T=6)	15.71%	75.97% (35.86%, T=6)	25.71%	51.72% (16.81%, T=6)	65.00%	0.22% (0.18%, T=6)	52.14%	9.1% (1.20%, T=6)	7.86%	92.51% (61.19%, T=6)
Volcano	1.01e-1	93.00%	0.58%	91.50%	1.01%	91.00%	1.18%	90.00%	1.59%	88.50%	2.32%	95.00%	0.22%

each CAPTCHA design, thus, can then be calculated from the digit-level Average Digit Error Rate as: $AtkSucRate = \sum_{t=T}^{6} \binom{6}{t} (1-DigitErrRate)^t (DigitErrRate)^{6-t}$. The binomial probability formula represents the probability that an ASR model correctly outputs at least T digits in a six-digit audio CAPTCHA task ($1 \leq T \leq 6$). We provide a detailed discussion on the selection of T for each CAPTCHA design in Section VII. To briefly summarize, we found different optimal values of T for different CAPTCHA designs: $T = 6$ for Baseline, $T = 5$ for Kenan and Devil, and $T = 3$ for Volcano. For each six-digit CAPTCHA task, a client will be considered as "human" as long as they output any five of the six digits correctly in Kenan, three for Volcano, and five for Devil. The T used in each CAPTCHA design remains the same, unless specified, for the rest of the paper because we see it as a possible parameter used when deployed in the real world.

Note that the actual Attack Success Rate is likely to be lower than our estimation. This is because we assume perfect clipping from audio CAPTCHA tasks to digits in our experiment, which is unlikely in a real-world setting and will unavoidably decrease attackers' success rate.

B. Results

As a reminder, the creation of AdvCAPTCHA requires a victim model as an anchor to ensure the capability against speech-to-text attacks. Indeed, such defensive capability varies based on which victim model is selected. For each AdvCAPTCHA design, to select the best victim model, we run a comprehensive analysis in which we iterate each commercial ASR model as a victim model to create AdvCAPTCHA tasks and pick the one that yields the AdvCAPTCHA with the lowest estimated attack success rate averaged against all the commercial ASR models. For Kenan and Devil, we select Google Could Speech (Phone Call Model) as the final victim model that yields the best security results. For Volcano, we select Google Could Speech (Phone Call Model) as the victim model for Digit-For-Machine-Only, and Microsoft Azure as the victim model for Digit-For-Human-Only. The full analysis for selecting the victim models for each AdvCAPTCHA design is reported in Appendix A.

1) Results of Attacks from Commercial ASR Models: Considering the scenario for attackers to use only a single commercial ASR model, Google Cloud Speech (Phone Call Model) yields the best estimated attack success rate of 53.57% against Baseline. We also found that the Ensemble Model achieves the highest estimated attack success rate (70.10% for Baseline) compared to all commercial ASR models, which aligns with the findings in prior work regarding the attack improvement via ensemble [30]. To rank AdvCAPTCHAs designs by the estimated attack success rate against the Ensemble Model: Volcano is the best with 0.22%, followed by Kenan's 31.91%, and Devil's 92.51%. Given the high estimated attack success rate of Devil against the Ensemble Model, we additionally explored a more secure alternative version of Devil by raising T from 5 to 6, and yielded an estimated attack success rate of 61.19%. Our results suggest the technical feasibility of AdvCAPTCHA's capability of achieving overall better security against commercial ASR models — i.e., lower estimated success rate — than Baseline(T=6)'s 70.10% (see Table I).

TABLE II. THE DIGIT ERROR RATES AND ESTIMATED SUCCESS RATES FOR WELL-RESOURCED ATTACKERS ATTACKING EACH CAPTCHA DESIGN WITH SELF-TRAINED MACHINE LEARNING MODELS.

	Black-box Self-trained		White-box Self-trained	
	Avg. digit error rate	*Est. attack success rate*	*Avg. digit error rate*	*Est. attack success rate*
Baseline	66.00%	0.15%	67.80%	0.11%
Kenan	63.50%	2.7%	58.00%	5.10%
Devil	89.50%	<0.01%	81.00%	0.13%
Volcano	85.54%	4.30%	84.00%	5.60%

2) Results of Attacks from Self-trained Machine Learning Model: Our results show that attacks from self-trained models yield estimated attack success rates around or below 5% (see Table II). Thus, it can be challenging for attackers to break AdvCAPTCHA designs using a self-trained model in both black-box and white-box scenarios. As expected, all of the white-box self-trained models achieve an estimated attack success rate that is at least no worse than black-box self-trained models do (e.g., the attack success rate improves from 2.7% to 5.10% for Kenan, from 4.3% to 5.6% for Volcano), except for Baseline. Baseline is the only audio CAPTCHA design which attackers gain no benefits from attacking using white-box self-trained models compared to their black-box counterparts (i.e., black-box: 0.15%; white-box: 0.11%). A possible explanation is that the environmental background noise used in Baseline contains randomness and that the white-box models fail to learn the features properly.

Our results also reveal that AdvCAPTCHAs are, overall, more vulnerable under commercial ASR models than the self-trained models. The results hold true even with a white-box self-trained model, with the attack success rate below 6% across the board. In contrast, attackers can leverage ensemble commercial ASR models to achieve a higher success rate (e.g., 61.19% for Devil (T=6)). A possible explanation is that the randomness in AdvCAPTCHA makes it extremely hard for an attacker to train effective speech-to-text models with a limited dataset and can be prone to overfitting on the training set. Thus, the self-trained models can be less robust than the generic

TABLE III. THE PROBABILITY FOR HUMANS AND ATTACKER MODELS PASSING DIFFERENT NUMBERS FOR DIGIT-FOR-MACHINE-ONLY IN A SIX-DIGIT VOLCANO TASK. DUE TO THE SILENT VOLUME PERTURBATION IN VOLCANO, HUMANS WOULD HARDLY PERCEIVE DIGIT-FOR-MACHINE-ONLY COMPARED TO ATTACKER MODELS. THE PROBABILITIES OF ATTACK-DETECTION DIGITS TRIGGERED BY HUMANS AND ATTACKER MODELS ARE PROVIDED IN THE LAST THREE COLUMNS.

Number of Digit-For-Machine-Only Entered	***1***	***2***	***3***	***4***	***5***	***6***	**Passed**	**Failed w/ Detection**	**Failed w/o Detection**
Human	23.14%	8.82%	6.06%	5.23%	4.96%	4.13%	90.98%	5.23%	3.79%
Google Cloud Speech (Default)	99.70%	96.75%	84.73%	58.57%	26.57%	5.68%	0.58%	**58.57%**	40.93%
Google Cloud Speech (Phone Call)	98.24%	88.10%	63.73%	32.52%	10.13%	1.38%	1.01%	**32.52%**	66.47%
Microsoft Azure	99.78%	97.46%	87.14%	62.68%	30.06%	6.87%	1.18%	**62.68%**	36.14%
IBM	92.10%	67.15%	34.29%	11.21%	2.09%	0.17%	1.59%	11.21%	87.20%
Facebook Wit	69.60%	29.56%	7.59%	1.16%	0.10%	<0.01%	2.32%	1.16%	96.52%
Ensemble	100%	100%	99.92%	98.82%	90.48%	56.79%	0.22%	**98.82%**	0.96%
Black-box Self-trained Model	75.28%	36.38%	10.87%	1.95%	0.19%	<0.01%	4.30%	1.95%	93.75%
White-box Self-trained Model	73.79%	34.46%	9.89%	1.70%	0.16%	<0.01%	5.60%	1.70%	92.70%

commercial ASR models, which are trained on a more diverse dataset, against adversarial audio examples.

3) Results for Targeted Defense of Volcano: Volcano offers the capability of attack detection of whether a failed CAPTCHA attempt is a machine or not by using the targeted adversarial perturbation technique (see Section IV-C). Table III summarizes the probability for each ASR model recognizing Digits-For-Machines-Only by the number of digits in a six-digit AdvCAPTCHA task. Our security evaluation considers a simplified scenario in which we only examine such attack detection when clients fail the CAPTCHA task. Note that clients pass Volcano when they get three out of the six digits correct (see Section V-A3), and only the clients who recognize four or more Digits-For-Machines-Only are regarded as speech-to-text attackers.

Table III also summarizes the results of Volcano's overall targeted defense against each ASR model (in the last three columns). We found that Microsoft Azure has a probability near 63% to be captured in attack detection, followed by Google Cloud Speech (Default Model)'s near 59%. IBM and Facebook Wit have lower probabilities of about 11% and 1%, respectively. For the well-resourced self-trained models, Black-box and White-box Self-trained Models are captured with the probability of 1.95% and 1.70%. It is noteworthy that the Ensemble Model, which is the strongest among all attacker models, is captured around 99% in attack detection. The results suggest that attacker models with stronger speech-to-text recognition capability also tend to be captured by Volcano's targeted defense. In other words, the targeted defense of Volcano provides an additional shield against popular commercial ASR attack solutions to audio CAPTCHA.

VI. USABILITY EVALUATION

To provide a comprehensive usability evaluation, we ran two online, controlled, within-subject user studies with 1) 121 non-PVI participants and 2) 11 PVIs, respectively. For the former, we invited a larger scale population to demonstrate the general applicability of our novel designs as typically done in prior audio CAPTCHA studies [21]. For the latter, we examined quantitative and qualitative insights on the effectiveness of AdvCAPTCHA for PVIs, who were day-to-day users of audio CAPTCHA. Thus, the two-fold user study design provides both breadth and depth aspects regarding the applicability of AdvCAPTCHA.

A. Study I: Usability Evaluation with Non-PVIs

1) Procedure: We built an online test bed to conduct the anonymous usability evaluation. Each participant solved and evaluated six-digit audio CAPTCHA tasks that were created with the three AdvCAPTCHA designs (i.e., Kenan, Devil, Volcano) and the baseline design (i.e., Baseline) described in Section IV. We balanced the order of the CAPTCHA designs between participants using a Latin Square design, and we randomly picked six digits for each audio CAPTCHA task.

Upon arrival, participants were asked to read through the study's informed consent on the welcoming page. Then, they reported information about their demographics and their prior knowledge of audio CAPTCHA. Participants were also instructed on how to solve audio CAPTCHA tasks on the test bed and were asked to adjust their devices' volume until they could hear an example six-digit audio clip clearly.

We follow the prior study on evaluating the usability of audio CAPTCHAs [18]: for each audio CAPTCHA design, participants completed three different six-digit tasks, and then they answered questions about their perceived satisfaction, easiness, and open-ended feedback about the design. For both satisfaction and easiness, participants ranked the designs on a five-Likert scale with "5" coded as "very satisfied" and "very easy", and "1" coded as "very unsatisfied" and "very difficult." After completing all the audio CAPTCHA tasks, participants were asked to rank their preferences among the audio CAPTCHA designs and to share their underlying reasons for choosing the most and the least favorite design, respectively. Besides their responses for usability evaluation, the test bed also logged participants' answers and the time spent on each audio CAPTCHA task. The study protocol was approved by our institution's IRB.

2) Recruitment and Participants: We posted our anonymous online study recruitment on our university's online student forums and a subject-recruitment Facebook group. We collected 121 responses from participants with normal or corrected-to-normal vision and hearing. Among 120 partici-

TABLE IV. RESULTS OF THE USABILITY TESTING IN STUDY I.

	Accuracy	Completion Time (seconds)	Satisfaction (1-5)	Easiness (1-5)
Baseline	89.81%	16.13	3.55	3.83
Kenan	89.53%	19.93	3.01	2.99
Devil	95.32% (79.06%, T=6)	17.03	3.44	3.60
Volcano	90.08%	22.71	2.12	2.16

pants [8] that provided valid responses, 57 were male and 63 were female, with a mean age of 25.9 (SD = 8.87). On a scale from 1 (unfamiliar) to 5 (familiar), participants self-reported their familiarity with audio CAPTCHA with a mean rating of 3.05 (SD = 1.20). On average, participants took 12.06 minutes (SD = 6.11) to complete the study, and we compensated each participant with a gift card worth $75 NTD ($2.5 USD).

B. Results of Study I

After manually checking the collected responses, all 121 responses were valid and we collected a total of 1,452 audio CAPTCHA task attempts.

We evaluate the usability of AdvCAPTCHAs and Baseline via the three metrics following the best practices in prior studies: *accuracy*, *completion time*, and *perceived satisfaction and easiness* [18], [34], as shown in Table IV. To assess if the AdvCAPTCHAs (i.e., Devil, Kenan, Volcano) differ from Baseline with statistical significance with respect to these metrics, we ran a mixed-effects regression analysis on each metric. In short, AdvCAPTCHAs achieved better or at least similar accuracy than Baseline. Overall, participants spent more time solving AdvCAPTCHAs than Baseline. Participants perceived Baseline and Devil as more satisfying and easier to answer than Kenan and Volcano. More details of the statistical analysis results can be found in Appendix B.

1) Accuracy: To make our accuracy evaluation consistent between machines and humans, when examining users' accuracy in getting through each audio CAPTCHA task, we matched their answer keys with the same mechanism as described in the security evaluation (see Section V-A). To recap, we quantitatively determine an optimal threshold T for each audio CAPTCHA design that we see as a possible parameter used when deployed to the real world. A task is considered passed (the client is a human) if a participant's answer key contains T or more of the correct digits. We set $T = 6$ for Baseline, $T = 5$ for Kenan and Devil, and $T = 3$ for Volcano.

In general, AdvCAPTCHA achieved similar or better accuracy performance than Baseline (89.81%): 95.32% for Devil, 89.53% for Kenan, and 90.08% for Volcano. However, we also found that creating a "stronger" version of AdvCAPTCHA — e.g., by raising Devil's T from 5 to 6 — also impacts the accuracy significantly (accuracy of Devil dropped from 95% to 79%). Since Volcano is a targeted defense mechanism, we additionally examined the probability that a user was misidentified as a "machine attacker," as described in Section V-B3. As shown in the first row of Table III, 90.98% of the participant attempts successfully passed tasks of Volcano, with around 5% (5.23%) of them misidentified as attackers. Indeed, there is room for improvement in Volcano to lower the rate of misidentifying humans as machines. Nevertheless, our results reveal the potential of Volcano's applicability of attack detection in a near real-world setting. As a comparison, commercial ASR models such as Google Cloud Speech (Default Model) and Microsoft Azure have a probability of around 60% falling for Volcano's attack-detection digits.

2) Completion Time, Perceived Satisfaction and Easiness: On average, our participants spent more time on solving AdvCAPTCHA than Baseline, with Baseline in 16.13 seconds (SD=7.47), Devil in 17.03 seconds (SD=9.95), Kenan in 19.93 seconds (SD=15.52), and Volcano in 22.71 seconds (SD=18.32). In addition, Baseline and Devil were generally perceived as more satisfying and easy to answer, while Volcano was the least satisfying and the most difficult.

3) Preferences: We examined our participants' preferences (i.e., preference ranking) for the four audio CAPTCHA designs. 47.32% of the participants ranked Baseline as their favorite design, citing reasons such as "the noise is continuous and expectable" (11 out of 120) and "human's noise is easy to be distinguished from environmental noise in the background" (14 out of 120). This was followed by Kenan's 27.68% and Devil's 23.21%. Volcano was the least favorite, with only 1.78% of our participants ranking it as the favorite design, with reasons such as "noise is ear-piercing and uncomfortable" (18 out of 120) or "mixing digits with two different voice volumes is confusing" (12 out of 120).

C. Study II: Usability Evaluation with PVIs

To evaluate the usability of AdvCAPTCHA, we conducted another study with PVIs, who were familiar with and relied on audio CAPTCHA in their everyday life.

1) Participants: Eleven participants (nine males, and two females) with a mean age of 28.82 (SD=4.17) were recruited through public recruitment posts on social media. Seven of them are congenitally blind while the other four are adventitiously blind. All of them were familiar with and relied on screen readers to navigate computer screens. Participants were confident in recognizing audio English number digits with a self-reported confidence average of 4.64 (SD=0.5, on a scale of 1-5) and familiar with a baseline audio CAPTCHA design with a self-reported score averaged 4.27 (SD=0.79, on a scale of 1-5). They were compensated with $450 NTD ($15 USD) for their participation.

2) Procedure: We conducted an online study with each of the participants with visual impairments using the conferencing application Google Meet. Note that our test bed was implemented to be accessible and easy-to-use using a screen reader, and none of our participants reported any technical issues throughout the study.

After being informed of our study procedures and requirements, each participant was asked to solve three different tasks for each of the three AdvCAPTCHA designs plus the Baseline (3×4=12 tasks for each participant), the same as Study I. The tasks for each participant were generated and assigned randomly to mitigate potential systematic biases.

[8] Due to a server transmitting error, we lost one participant's self-report demographic information and preference ranking.

Similar to Study I, after completing the three tasks of one audio CAPTCHA design, participants were asked to rate (on a scale of 1-5) satisfaction and easiness, and make open remarks on the tasks. The interviews were held in a semi-structured manner to elicit more insights on the influential factors of usability. After completing the twelve assigned tasks, participants were asked to rank the four audio CAPTCHA designs (i.e., three AdvCAPTCHA designs and Baseline), and to provide reasons for their ranking. Finally, participants were asked to comment on the comparison between AdvCAPTCHAs and audio CAPTCHAs they encountered in their life, and to provide overall feedback and suggestions for improving AdvCAPTCHAs. The study took an average of 50 minutes to complete. The study protocol was approved by our institution's IRB.

D. Results of Study II

All audio CAPTCHA designs shared a similar accuracy (over 90%), and that Baseline was solved the fastest and Volcano the slowest, similar to the trend found in Study I. However, PVIs reported slightly different than non-PVIs on satisfaction and easiness measurements among AdvCAPTCHA designs (see Table V). Specifically, Devil was the most satisfying, easy-to-use, and favorite design for PVIs (six out of 11 ranked it in first place) rather than Baseline. A possible explanation is that Baseline contains large background noise, such as car driving, children, and horns, which are particularly distracting for PVIs to solve audio CAPTCHA tasks. In contrast, Devil contains quieter background noises. Nevertheless, considering our relatively small sample size (N=11) and potential novel experience that may result in more positive feedback of Devil [33], as well as the fact that Devil yields lower accuracy when having a higher security standard (i.e., by raising T from 5 to 6, the accuracy drops from 97% to 81%), we could not simply conclude Devil is a better alternative for PVI users. Further longitude studies in the wild with more PVIs are necessary to validate our findings. Still, both our studies conducted with non-PVI and PVI users reveal that Devil has the potential to achieve at least a similar usability to Baseline.

Echoing our findings in Study I, Volcano was also perceived as the least favorable type (eight out of 11 put it as their least favorite design) by PVIs. Our participants spent the longest time solving the tasks and felt them the most difficult.

It is noteworthy that our PVI participants preferred Devil over Baseline, which was different from the results of non-PVI participants. Specifically, PVI participants rated satisfaction and easiness for Devil averaged 3.58 and 4.00, respectively, and 3.44 and 3.60 for Baseline. In addition, slightly more PVI participants ranked Devil in the first place (N=6) than Baseline (N=5). Finally, PVI participants, overall, spent more time solving audio CAPTCHA tasks than non-PVI participants. This is because PVI would need more time to navigate and interact with websites using screen readers.

When our participants elaborated their considerations on ranking AdvCAPTCHAs, "digits clarity" (ten out of 11) and "background noises" (five out of 11) were the two most frequent reasons cited, which aligned with our findings in Study I. Indeed, both characteristics of audio CAPTCHA are associated with the dimension of *Distortion* in Yan et al.'s CAPTCHA usability assessment framework [34]. Thus, it may not be surprising that Volcano was the least favorite design as it was criticized for its "sudden peaks with high frequency in the background" (five out of 11) and "confusing characters due to sound distortion (e.g., '6' and '3')" (eight out of 11). On the contrary, Devil was the most favorite design and received more positive feedback on digit clarity (six out of 11), and its background noises were praised to be either subtle or consistent enough not to impact the clarity of the digits (eight out of 11).

TABLE V. RESULTS OF THE USABILITY TESTING IN STUDY II.

	Accuracy	**Completion Time (seconds)**	**Satisfaction (1-5)**	**Easiness (1-5)**
Baseline	91.67%	19.72	3.42	3.83
Kenan	97.22%	30.47	2.33	2.75
Devil	97.22% (80.56%, T=6)	22.41	3.58	4.00
Volcano	97.22%	33.04	1.92	2.25

In the same vein, Kenan was less favorite than Baseline with the disadvantages of "unclear" (ten out of 11) and "inconsistent distortion mechanism across digits (i.e., sometimes the volume is distorted, sometimes the tone)" (three out of 11), while digits in Baseline were praised for being clear (seven out of 11). Still, participants perceived Kenan to have fewer background noises (two out of 11).

VII. THRESHOLD SELECTION

In practice, there is a trade-off between security and usability in CAPTCHA design [17], [23], [24]. Thus, when designing AdvCAPTCHAs, we model this trade-off by considering a threshold parameter for each CAPTCHA design. The design rationale is that, for an inherently easier CAPTCHA (e.g., smaller perturbations), setting a higher bar (i.e., passing a task if and only if all the digits were answered correctly) for users to pass the task may have less impact on the usability of CAPTCHA. Similarly, an inherently more difficult-to-solve CAPTCHA may benefit from, regarding usability, lowering the bar to decrease task difficulty (i.e., passing a task when only partial digits were answered correctly). In AdvCAPTCHAs, the threshold serves as an adaptive criterion for passing a CAPTCHA task: for the six-digit tasks in each CAPTCHA design c, we empirically selected the optimal criterion threshold T_c to pass the CAPTCHA — i.e., the number of digits in a six-digit task that should be correctly answered to pass the task ($1 \leq T_c \leq 6$).

We approached the threshold selection between security and task difficulty with the receiver operating characteristic (ROC) curve [19], a well-known method for model evaluation and selecting threshold. In our analysis, CAPTCHA could be seen as a binary classifier that classifies a user into human or non-human (i.e., machine). For example, when we take humans as 'true' instances and machines as 'false' instances, the attack success rate could be seen as a false-positive rate as it represents that a CAPTCHA misclassifies machines into humans; while human accuracy resembles a true positive rate, indicating the ratio for humans to pass the test. This correspondence enables us to construct an ROC curve yielded

from true positive and false positive rates under different thresholds.

A. ROC Curve of AdvCAPTCHAs against each Attacker Model

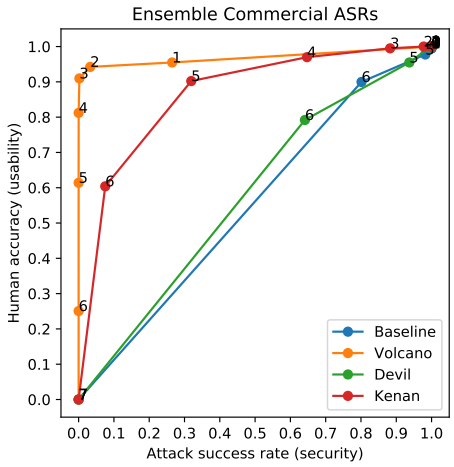

Fig. 2. ROC Curves of AdvCAPTCHAs when encountering Ensemble commercial ASRs. For each CAPTCHA type c, we plot points with attack success rate and human accuracy (a_c^T, h_c^T) as coordinates under each threshold $T = 0, 1, ..., 7$. Note that our CAPTCHA tasks contain only six digits. This means that if $T = 0$, defenders will accept any clients, and thus, both attack success rate and human accuracy will be 1.0. In contrast, with $T = 7$, defenders will reject all the clients and make both quantities 0.0.

Now, we detail the analysis of a CAPTCHA's security-usability trade-off using the ROC curve (see Figure 2). Given an attacker model, for each type of CAPTCHA c, we first enumerated six possible thresholds $T = 1, 2, ..., 6$. For each threshold T, we yielded a pair of attack success rate a_c^T and human accuracy h_c^T $((a_c^T, h_c^T))$. The "Overall Human Accuracy" rows in Table VI determined the human accuracy h_c^T, and the attack success rate a_c^T was calculated using the binomial probability as mentioned in Section V-A with $DigitErrRate$ of CAPTCHA type c with the threshold T.

As the core objective of CAPTCHA is to differentiate humans from machines, we desire higher *gap* [24] between human and machine, which we refer to as *human-machine gap*. Given an attacker model, the human-machine gap g_c^T of CAPTCHA type c with the threshold value T could be calculated as follows:

$$g_c^T = h_c^T - a_c^T \quad \text{for} \quad T = 1, 2, ..., 6$$

The criterion threshold value T_c would then be the one maximizing the *human-machine gap*.

$$T_c = \underset{T}{\text{argmax}}\ g_c^T$$

For our low-resourced threat model, we use Ensemble Commercial ASRs (Ensemble) as an example (see Figure 2). Among the four CAPTCHA designs with their criterion threshold parameter $T_{Baseline}, T_{Volcano}, T_{Devil}, T_{Kenan}$, Volcano achieved the best performance with $g^2_{Volcano} = 0.91$ $(T_{Volcano} = 2)$, with Baseline the lowest $g^6_{Baseline} = 0.10$ $(T_{Baseline} = 6)$. Similarly, Kenan $(g^5_{Kenan} = 0.58)$ and Devil

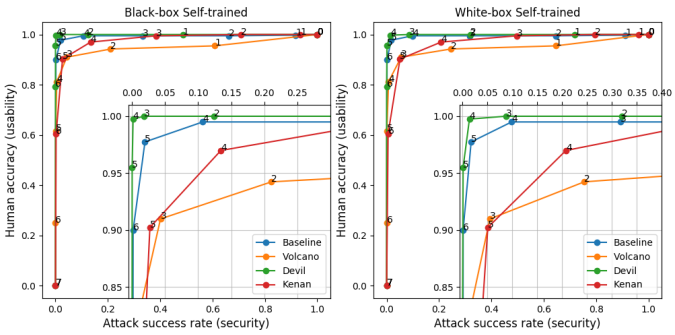

Fig. 3. ROC Curves of AdvCAPTCHAs when encountering White-box & Black-box Self-trained Models. Note that the upper-left point represents a perfect CAPTCHA as described in Section VII-A. We zoom in the upper left corner of each figure for a detailed comparison.

$(g^6_{Devil} = 0.15)$ also achieved a higher human-machine gap compared to Baseline against Ensemble.

Note that the diagonal line from $(0.0, 0.0)$ to $(1.0, 1.0)$ of the ROC curve is known as no discrimination [22], and in the context of CAPTCHA, it indicates that the CAPTCHA is no better than randomly deciding whether a user is human or not. The upper left point $(0.0, 1.0)$ represents a perfect CAPTCHA scheme with no attack success rate and perfect human accuracy. Figure 2 shows that the ROC curves of Baseline and Devil are similar to the diagonal, indicating that when encountering a strong attacker model such as Ensemble, Baseline and Devil would fail to distinguish humans from it. On the other hand, the ROC curves of Kenan and Volcano are closer to the upper left point, indicating better performance on the defense. We provide all ROC curves against all commercial ASRs in Appendix C.

For our well-resourced threat model, we summarize the results in Figure 3 and the human-machine gaps in the last two columns of Table VII. When encountering the black-box self-trained models, Devil outperforms all the other CAPTCHA with $g^4_{Devil} = 0.99$, larger than Baseline's $g^5_{Baseline} = 0.96$. As for the white-box self-trained model, Devil achieves $g^4_{Devil} = 0.98$ while Baseline has $g^5_{Baseline} = 0.96$. Note that a slight decrease in the human-machine gap is expected since the white-box self-trained model has been fine-tuned with samples of CAPTCHAs, resulting in higher attack success rates. These results demonstrate that Devil is close to a perfect CAPTCHA against the well-resourced threat model. The other two Adv-CAPTCHA designs, Kenan and Volcano, achieve about 86% on the human-machine gaps against black-box and white-box self-trained models with approximately 10% lower than Devil and Baseline. Such an accuracy difference between humans and machines, however, may still be acceptable to distinguish humans from attacker models in some circumstances when the need for security is lower and can be traded for human accuracy and usability.

B. Threshold Selection Across Attacker Models

As mentioned, for each CAPTCHA design against each attacker model, we select a threshold T_c to optimize the CAPTCHA's security-usability trade-off. In reality, however, a global threshold will be necessary because we cannot always have preconceived information about attackers' models. In this

TABLE VI. THE OVERALL HUMAN ACCURACY, ATTACK SUCCESS RATE, AND HUMAN-MACHINE GAP UNDER EACH THRESHOLD. NOTE THAT WE SELECT OUR GLOBAL THRESHOLD WITH THE ONE THAT YIELDS THE LARGEST HUMAN-MACHINE GAP.

CAPTCHA	Participants	Fixed Threshold						Global Threshold
		1	*2*	*3*	*4*	*5*	*6*	
Baseline	Overall Human Accuracy	1	1	1	1	0.97	0.90	6
	Overall Success Rate	0.94	0.74	0.49	0.30	0.19	0.11	
	Human-Machine Gap	0.06	0.26	0.51	0.70	0.78	**0.79**	
Kenan	Human Accuracy	1	1	1	0.97	0.90	0.60	5
	Overall Success Rate	0.89	0.68	0.41	0.19	0.06	0.01	
	Human-Machine Gap	0.11	0.32	0.59	0.78	**0.84**	0.59	
Devil	Overall Human Accuracy	1	1	1	1	0.95	0.79	5
	Overall Success Rate	0.73	0.46	0.31	0.23	0.16	0.08	
	Human-Machine Gap	0.27	0.54	0.69	0.77	**0.79**	0.71	
Volcano	Overall Human Accuracy	0.95	0.94	0.91	0.81	0.61	0.25	3
	Overall Success Rate	0.55	0.18	0.04	0.004	0.0003	9e-6	
	Human-Machine Gap	0.40	0.76	**0.87**	0.81	0.61	0.25	

TABLE VII. LOCAL THRESHOLDS AND THE CORRESPONDING HUMAN-MACHINE GAPS FOR ALL CAPTCHA DESIGNS AGAINST EACH ATTACKER MODEL.

	Microsoft Azure		Google Cloud Speech (Phone Call)		Google Cloud Speech (Default)		IBM		Facebook Wit		Ensemble Commercial ASRs		Black-box Self-trained		White-box Self-trained	
	local thold	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*	*local thold*	*hu-mach gap*
Baseline	6	0.58	6	0.26	6	0.81	5	**0.96**	6	0.85	6	0.10	5	0.96	5	0.96
Kenan	5	0.58	4	**0.96**	3	**0.97**	4	0.94	4	**0.93**	5	0.58	5	0.88	5	0.85
Devil	6	0.62	6	0.43	6	0.59	5	0.93	5	0.86	6	0.15	4	**0.99**	4	**0.98**
Volcano	3	**0.9**	3	0.9	3	0.9	3	0.89	3	0.89	2	**0.91**	3	0.87	3	0.85

section, we discuss how we select a global threshold based on each model's local results across all the commercial ASRs, black-box self-trained, and white-box self-trained models.

A global threshold for a CAPTCHA design is obtained through a similar procedure to its local thresholds. The difference is that instead of depending on a specific model, we calculate an overall attack success rate. Specifically, we consider all of our threat models: low-resource (i.e., commercial ASRs), well-resourced (i.e., black-box and white-box self-trained models) as described in Section III. We define the overall attack success rate as the mean of the success rates under these three threat models (The "Overall Success Rate" rows in Table VI). Then, with the overall attack success rate and the overall human accuracy, we could calculate the overall human-machine gap under each threshold following the same procedure mentioned in Section VII-A. As shown in the "Human-Machine Gap" rows in Table VI, we yield the best human-machine gap for the usability-security trade-off with the global thresholds of $T_{Baseline} = 6$, $T_{Volcano} = 3$, $T_{Devil} = 5$, $T_{Kenan} = 5$.

As shown in Table VII and Table VI, almost all of these global thresholds differ from our local thresholds by no more than one, with the only exception of Kenan against Google Cloud Speech (Default Model) — i.e., a local threshold of three and a global threshold of five. Nevertheless, the threshold of either three or five can result in Kenan against Google Cloud Speech (Default Model) close to the upper left perfect point on the ROC curve (see Appendix Figure 4). According to the "Human-Machine Gap" rows of Table VI, we could see that Kenan and Volcano achieve 87.32% and 84.03% on the human-machine gaps, which outperform Baseline's 79.28%. Devil achieves 79.36%, similar to Baseline. Still, as shown in Table V, Devil has more positive feedback than Baseline from PVIs due to less annoying background noises. Our results suggest that AdvCAPTCHAs provide more capability and flexibility in designing for the audio CAPTCHA security-usability trade-off.

Finally, we took a closer look at the vulnerability from random guess attacks when decreasing the CAPTCHA task passing bar. Even with the lowest threshold of our AdvCAPTCHAs — i.e., $T_{Volcano} = 3$ —, the attack success rate of random guess attack in a six-digit CAPTCHA task is still lower than 2% ($\sum_{t=3}^{6} \binom{6}{t}(0.1)^t(0.9)^{6-t} = 0.01585$). We also explored the effectiveness of our global thresholds when encountering unseen models — i.e., humans and attacker models never seen during the threshold selection process. To that end, we conducted K-Fold and Leave-One-Out cross-validation on humans and attacker models, respectively, to simulate scenarios of unseen clients. The detailed procedure is described in Appendix Section D. In summary, even against hidden attacker models and humans, AdvCAPTCHAs could still achieve 91%, 80%, and 87% on the average human-machine gap with Volcano, Devil, and Kenan, respectively, and outperform Baseline's 78%.

VIII. DISCUSSION AND FUTURE WORK

In this section, we discuss the considerations when deploying AdvCAPTCHA in real-world scenarios.

Our findings show that, in the context of *untargeted-defense* audio CAPTCHA, our proposed AdvCAPTCHAs can provide an overall stronger defense against adversarial audio CAPTCHA attacks. While they still suffer from lower usability than Baseline, overall, we found the potential of Devil to

provide similar usability. To further improve the robustness (especially toward commercial ASR models) and broader applicability of Devil in real-world scenarios, future work can apply the AdvCAPTCHA scheme to digits beyond numeric digits (e.g., alphabetical digits, words), and languages beyond English.

In the context of *targeted-defense* audio CAPTCHA, our findings show the potential of the unique advantage of Volcano: attacker models with a high attack success rate (e.g., Microsoft Azure) that are capable of breaking untargeted-defense audio CAPTCHAs (e.g., Baseline), were prone to be identified as machines by Volcano. That is, ASR models' capability to recognize speech also leads them to be "captured" by Volcano. On the other hand, models with a lower attack success rate against untargeted-defense audio CAPTCHA (e.g., Self-trained models) are unable to break Volcano anyway due to their lower accuracy in recognizing speech. In other words, regardless of which models attackers are using, Volcano is capable of providing adequate security. On the other hand, our findings also show that there is room for improving the usability of Volcano. Future work can explore ways to increase the easiness of Digits-For-Humans-Only. For example, lowering the audio component removal threshold when reconstructing the audio digits might trade security with better audio clarity and hence increase usability. Additionally, when creating Volcano, one could ensure the created audio digit should not be distorted above a certain threshold (e.g., measured with the minimum square error), and thus be easier for users to complete the tasks. Such an approach can also be applied to the improvement of the two other AdvCAPTCHAs, Kenan and Devil.

In this paper, we present the threshold selection of AdvCAPTCHA as a preliminary attempt to quantify the trade-off between the security and usability of a CAPTCHA scheme. When deploying AdvCAPTCHA in the real world, however, one should adaptively prioritize, based on actual use logs, either security or usability according to different use cases, similar to the way we explored the T for Devil in this paper. For example, for CAPTCHA prone to higher user errors (e.g., Volcano), while we suggest first exploring approaches to decrease task difficulty as discussed above, one can also fine-tune the threshold by starting from the suggested optimal global threshold and then gradually lowering (or increasing) the threshold through iterative testing until finding the acceptable security-usability balance. In addition, after the deployment of an AdvCAPTCHA scheme, adaptive threshold adjustments can be made while the system measures users' accuracy in solving CAPTCHA tasks — i.e., selecting the threshold that yields the optimal CAPTCHA performance, concerning both security and usability based on actual use logs. We also see such threshold examination and selection processes applicable to other CAPTCHA schemes more broadly.

A. Limitations

This study has several limitations. Firstly, our utilization of an off-the-shelf Google Cloud service for generating base digit audio with varied pitches and tones may present a constraint in terms of diversity, potentially limiting a comprehensive evaluation of AdvCAPTCHA. Future research should aim to broaden the spectrum of audio inputs, incorporating a more diverse set of voices, languages, accents, and digit types beyond numeric. Secondly, the demographic composition of our user studies is predominantly skewed toward a younger demographic, potentially failing to accurately represent the full spectrum of audio CAPTCHA users. It is essential to include a wider range of demographics in future studies, particularly focusing on older adults and individuals with different hearing abilities. Such inclusivity will provide a more holistic understanding of the usability and accessibility of audio CAPTCHAs across diverse user groups. Lastly, given the dynamic and rapidly evolving nature of adversarial machine learning and speech recognition, the efficacy of our proposed AdvCAPTCHA may fluctuate over time. As these technologies continue to advance, our approach might become less effective against newer ASR models and self-training algorithms. Therefore, ongoing testing and adaptation are imperative to maintain the robustness of AdvCAPTCHA. Future research can replicate and validate our results against the latest state-of-the-art models.

IX. Conclusion

In this paper, we present AdvCAPTCHA, a set of audio CAPTCHA designs driven by different adversarial machine learning techniques that can resist common speech-to-text attacks while preserving usability. We also create a novel targeted-defense audio CAPTCHA that enables detection against attackers by implementing a honeypot-like defense in AdvCAPTCHA. We launch a user study on both non-PVIs and PVIs to measure usability in a real-world environment. We search for a custom criterion for each type of audio CAPTCHA, which models the security-usability trade-off in the design of CAPTCHA. In a nutshell, our findings suggest that AdvCAPTCHAs handle the security-usability trade-off better than the extant audio CAPTCHA design. As commercial ASR services become more easily accessible for attackers, our work provides an effective defense mechanism for audio CAPTCHA to distinguish machines from humans.

Acknowledgment

This work was supported in part by the National Science and Technology Council under Grants MOST 110-2634-F002-051, MOST 110-2222-E-002-014-MY3, NSTC 113-2923-E-002-010-MY2, and NSTC-112-2634-F-002-002-MBK. We thank the CMU SPUD Lab for sharing the source code of the audio CAPTCHA test bed.

References

[1] H. Abdullah, W. Garcia, C. Peeters, P. Traynor, K. R. Butler, and J. Wilson, "Practical hidden voice attacks against speech and speaker recognition systems," *arXiv preprint arXiv:1904.05734*, 2019.

[2] H. Abdullah, M. S. Rahman, W. Garcia, L. Blue, K. Warren, A. S. Yadav, T. Shrimpton, and P. Traynor, "Hear" no evil", see" kenansville": Efficient and transferable black-box attacks on speech recognition and voice identification systems," *arXiv preprint arXiv:1910.05262*, 2019.

[3] H. Abdullah, K. Warren, V. Bindschaedler, N. Papernot, and P. Traynor, "The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems," 07 2020.

[4] T. Ahmed, R. Hoyle, K. Connelly, D. Crandall, and A. Kapadia, "Privacy concerns and behaviors of people with visual impairments," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 3523–3532.

[5] M. Alnfiai, "A novel design of audio captcha for visually impaired users," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 168–179, 2020.

[6] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[7] S. Baheti, H. Shah, R. Sherathia, and P. Waghamode, "Captcha for visually impaired people: A review," 2021.

[8] D. Berrar, "Cross-validation." 2019.

[9] J. P. Bigham and A. C. Cavender, "Evaluating existing audio captchas and an interface optimized for non-visual use," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1829–1838. [Online]. Available: https://doi.org/10.1145/1518701.1518983

[10] K. Bock, D. Patel, G. Hughey, and D. Levin, "uncaptcha: A low-resource defeat of recaptcha's audio challenge," in *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. Vancouver, BC: USENIX Association, Aug. 2017. [Online]. Available: https://www.usenix.org/conference/woot17/workshop-program/presentation/bock

[11] E. Bursztein and S. Bethard, "Decaptcha: Breaking 75% of ebay audio captchas," in *Proceedings of the 3rd USENIX Conference on Offensive Technologies*, ser. WOOT'09. USA: USENIX Association, 2009, p. 8.

[12] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving captchas? a large scale evaluation," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 399–413.

[13] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 513–530. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/carlini

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

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

[16] S. Choudhary, R. Saroha, Y. Dahiya, and S. Choudhary, "Understanding captcha: text and audio based captcha with its applications," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, 2013.

[17] Y.-W. Chow, W. Susilo, and P. Thorncharoensri, "Captcha design and security issues," in *Advances in Cyber Security: Principles, Techniques, and Applications*. Springer, 2019, pp. 69–92.

[18] V. Fanelle, S. Karimi, A. Shah, B. Subramanian, and S. Das, "Blind and human: Exploring more usable audio CAPTCHA designs," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, Aug. 2020, pp. 111–125. [Online]. Available: https://www.usenix.org/conference/soups2020/presentation/fanelle

[19] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[20] Y. Feng, Q. Cao, H. Qi, and S. Ruoti, "Sencaptcha: A mobile-first captcha using orientation sensors," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–26, 2020.

[21] H. Gao, H. Liu, D. Yao, X. Liu, and U. Aickelin, "An audio captcha to distinguish humans from computers," in *2010 Third International Symposium on Electronic Commerce and Security*. IEEE, 2010, pp. 265–269.

[22] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[23] M. Jain, R. Tripathi, I. Bhansali, and P. Kumar, "Automatic generation and evaluation of usable and secure audio recaptcha," in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 2019, pp. 355–366.

[24] K. A. Kluever and R. Zanibbi, "Balancing usability and security in a video captcha," in *Proceedings of the 5th Symposium on Usable Privacy and Security*, 2009, pp. 1–11.

[25] J. B. Li, S. Qu, X. Li, J. Szurley, J. Z. Kolter, and F. Metze, *Adversarial Music: Real World Audio Adversary against Wake-Word Detection System*, 2019.

[26] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[27] Y. Qin, N. Carlini, I. J. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," in *ICML*, 2019.

[28] S. Sano, T. Otsuka, and H. G. Okuno, "Solving google's continuous audio captcha with hmm-based automatic speech recognition," in *Advances in Information and Computer Security*, K. Sakiyama and M. Terada, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 36–52.

[29] H. Shekhar, M. Moh, and T.-S. Moh, "Exploring adversaries to defend audio captcha," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1155–1161.

[30] S. Solanki, G. Krishnan, V. Sampath, and J. Polakis, "In (cyber)space bots can hear you speak: Breaking audio captchas using ots speech recognition," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 69–80. [Online]. Available: https://doi.org/10.1145/3128572.3140443

[31] C. Szegedy, G. Inc, W. Zaremba, I. Sutskever, G. Inc, J. Bruna, D. Erhan, G. Inc, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[32] J. Tam, S. Hyde, J. Simsa, and L. V. Ahn, "Breaking audio captchas," in *Proceedings of the 21st International Conference on Neural Information Processing Systems*, ser. NIPS'08. Red Hook, NY, USA: Curran Associates Inc., 2008, p. 1625–1632.

[33] E. Tulving and N. Kroll, "Novelty assessment in the brain and long-term memory encoding," *Psychonomic Bulletin & Review*, vol. 2, no. 3, pp. 387–390, 1995.

[34] J. Yan and A. S. El Ahmad, "Usability of captchas or usability issues in captcha design," in *Proceedings of the 4th Symposium on Usable Privacy and Security*, ser. SOUPS '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 44–52. [Online]. Available: https://doi.org/10.1145/1408664.1408671

[35] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," ser. CCS '17, 2017, p. 103–117.

[36] Z. Zhang, Z. Zhang, H. Yuan, N. M. Barbosa, S. Das, and Y. Wang, "{WebAlly}: Making visual task-based {CAPTCHAs} transferable for people with visual impairments," in *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, 2021, pp. 281–298.

APPENDIX

A. Victim Model Selection

For untargeted-defense AdvCAPTCHA such as Kenan and Devil, we first generate AdvCAPTCHAs against different victim models as described in Section II-A. Then, we calculate the digit error rate averaged over all the commercial ASR models and select the model with the highest digit error rate as the victim model. This is because a higher digit error rate implies that the generated CAPTCHA is more likely to defend unseen models from attackers. For Devil and Kenan, we chose Google Cloud Speech (Phone Model) as the victim model. For targeted defense AdvCAPTCHA (i.e., Volcano), since Volcano is composed of two parts, Digit-For-Human-Only and Digit-For-Machine-Only, we first select different victim models and generate the two parts separately, then we estimate the probability for commercial ASR models to identify an attack-detection digit (Digit-For-Machine-Only) on every combination.

For simplicity, in Table VIII, we only show the rows with victim models of Google Cloud Speech (Phone Call Model)

and Microsoft Azure because of their higher attack success rate.

B. Mixed-effects Regression Model

To assess if the designs of AdvCAPTCHA impact users' ability to provide accurate responses, we ran a mixed-effects regression model predicting if a task was accurately solved by users, with the independent variable of the CAPTCHA design (with Baseline as the reference) and with random intercepts to account for variances among participants and iterations (i.e., the three-time task iteration for each CAPTCHA) within each testing session. We found that Devil has a higher accuracy than Baseline with statistical significance. Overall, our participants achieved better or at least similar accuracy when solving tasks of AdvCAPTCHA compared to Baseline (see Table IX).

As for completion time, we ran another regression model to predict participants' completion time for each challenge. We set the CAPTCHA design as an independent variable with random intercepts to account for variances among participants and iterations within each testing session. We found higher completion time for Kenan and Volcano than Baseline with statistical significance. Overall, participants spent more time solving AdvCAPTCHAs than they did for Baseline (see Table IX).

Similarly, we built two regression models to assess if the differences in our participants' perceptions of AdvCAPTCHA to Baseline design were statistically significant. Specifically, we labeled *satisfaction* and *easiness* into binary variables with the response of "4" or "5" in 5-Likert scales to be "satisfying" and "easy to answer", or otherwise the opposite (i.e., not "satisfying" and not "easy to answer"). We found that Baseline and Devil were perceived as more satisfying and easy to answer than the other CAPTCHA designs (see Table IX).

C. ROC Curves Against Commercial ASRs

The ROC curves against commercial ASR models besides the Ensemble model are summarized in Figure 4. From the view of ROC curves, when encountering Google Cloud Speech (Phone Call Model), Google Cloud Speech (Default Model), and Facebook Wit, the ROC curve of Kenan is closer to the upper left perfect point than all the other CAPTCHAs. As for Microsoft Azure, Volcano's ROC curve is the closest to the upper left perfect point. Our proposed AdvCAPTCHAs only fell short of Baseline against IBM with slight differences in the human-machine gap as the curves almost overlap. For more precise results, Table VII shows the exact local thresholds and corresponding human-machine gaps for each CAPTCHA type against all of the attacker models. Volcano achieves human-machine gap $g^3_{Volcano}$ higher than 90% with local threshold $T_{Volcano} = 3$ against Microsoft Azure and Ensemble Commercial ASRs. Kenan achieves $g^4_{Kenan} = 96\%$ against Google Cloud Speech (Phone Call Model), $g^4_{Kenan} = 93\%$ against Facebook Wit with $T_{Kenan} = 4$, and $g^3_{Kenan} = 97\%$ against Google Cloud Speech (Default Model) with $T_{Kenan} = 3$. Even though the proposed AdvCAPTCHAs fall short of Baseline against IBM, they still have human-machine gaps of about 90%. Notably, Baseline only has a human-machine gap of 58% against Microsoft Azure and 26% against Google Cloud Speech (Phone Call Model). The results show that Baseline is not secure enough to resist a single strong commercial ASR, not to mention more powerful ones such as the Ensemble Commercial ASRs.

D. Cross-validation on Humans & Attacker Models

As mentioned in Section VII, threshold T_c is selected for each CAPTCHA type c to maximize the human-machine gap according to the accuracy of a group of humans and the attack success rate of a set of attacker models. We take a closer look at whether the threshold selected is still effective even when encountering unseen attackers and humans, as mentioned in Section VII-B.

To compare the effectiveness of AdvCAPTCHAs and Baseline when encountering unseen clients (machines or humans), we provide cross-validation analysis on both human and attacker models to prepare a hidden validation set for each CAPTCHA. First, to simulate encountering unseen humans, we use K-Fold cross-validation [8] with $K = 5$ to separate humans into five folds. Each fold would contain 24 to 25 PVIs plus two to three non-PVIs. A hidden validation human set would be one of the five folds, while the other four form a training human set. To simulate unseen attacker models, we use Leave-one-out cross-validation [8] to keep a single attacker model as a validation attacker model while the others serve as training attacker models. When selecting thresholds, only training human set and training attacker models would be used with the same procedure described in Section VII-B. After the threshold is determined, we check the human-machine gap of each CAPTCHA on the validation human set and attacker model, which is hidden during threshold selection. The result of cross-validation is summarized in Table X.

TABLE VIII. THE DIGIT ERROR RATE FOR UNTARGETED CAPTCHA DESIGNS (I.E., DEVIL AND KENAN), AND THE ATTACK-DETECTION DIGIT TRIGGER RATE FOR TARGETED CAPTCHA DESIGN (I.E., VOLCANO), WHEN GENERATED BY DIFFERENT VICTIM MODELS. FOR DEVIL AND KENAN, WE CHOSE GOOGLE CLOUD SPEECH (PHONE MODEL) AS THE VICTIM MODEL BECAUSE THOSE CAPTCHAS HAVE THE LOWEST ATTACK SUCCESS RATE. FOR VOLCANO, WE CHOSE THE COMBINATION OF GOOGLE CLOUD SPEECH (PHONE MODEL) FOR DIGIT-FOR-HUMAN-ONLY AND MICROSOFT AZURE FOR DIGIT-FOR-MACHINE-ONLY.

	Victim Model		**Digit Error Rate of Commercial ASR**					
			Google (Default)	Google (Phone)	Microsoft	IBM	Facebook	Average
Kenan	*Google Cloud Speech (Phone Call)*		88.75%	85.63%	35.00%	77.50%	74.38%	**72.25%**
	Microsoft Azure		80.63%	32.50%	50.62%	68.75%	71.25%	60.75%
Devil	*Google Cloud Speech (Phone Call)*		23.57%	15.71%	25.71%	65.00%	52.14%	**36.43%**
	Microsoft Azure		23.13%	14.37%	28.12%	68.13%	46.25%	36.00%
	Digit-For-Human-Only	**Digit-For-Machine-Only**	**Attack-detection Digit Trigger Rate**					
Volcano	*Google (Phone)*	*Google (Phone)*	49.00%	31.00%	29.50%	34.50%	16.00%	32.00%
	Google (Phone)	*Microsoft*	44.00%	30.00%	25.50%	40.50%	15.00%	31.00
	Microsoft	*Google (Phone)*	62.00%	49.00%	64.00%	34.50%	18.00%	**45.50%**
	Microsoft	*Microsoft*	52.00%	11.50%	54.50%	13.50%	23.50%	31.00%

TABLE IX. MIXED-EFFECTS REGRESSIONS PREDICTING ACCURACY, COMPLETION TIME, SATISFACTION, EASINESS AGAINST CAPTCHA DESIGN IN STUDY I.

	Acuracy		**Completion Time**		**Satisfaction**		**Easiness**	
	Est.	*Sig.*	*Est.*	*Sig.*	*Est.*	*Sig.*	*Est.*	*Sig.*
Kenan v. Baseline	–0.0312		3.799	***	–1.3143	***	–1.7112	***
Devil v. Baseline	0.8684	**	0.892		–0.2633		–0.3811	
Volcano v. Baseline	0.0320		6.575	***	–3.0162	***	–3.7286	***
Devil v. Kenan	0.8997	**	–2.907	**	1.5010	***	1.3301	***
Volcano v. Kenan	0.0632		2.776	**	–1.7020	***	–2.0174	***
Volcano v. Devil	-0.8385	**	5.683	***	–2.7529	***	–3.3475	***

Significance: * p<.05; ** p<.01; *** p<.001

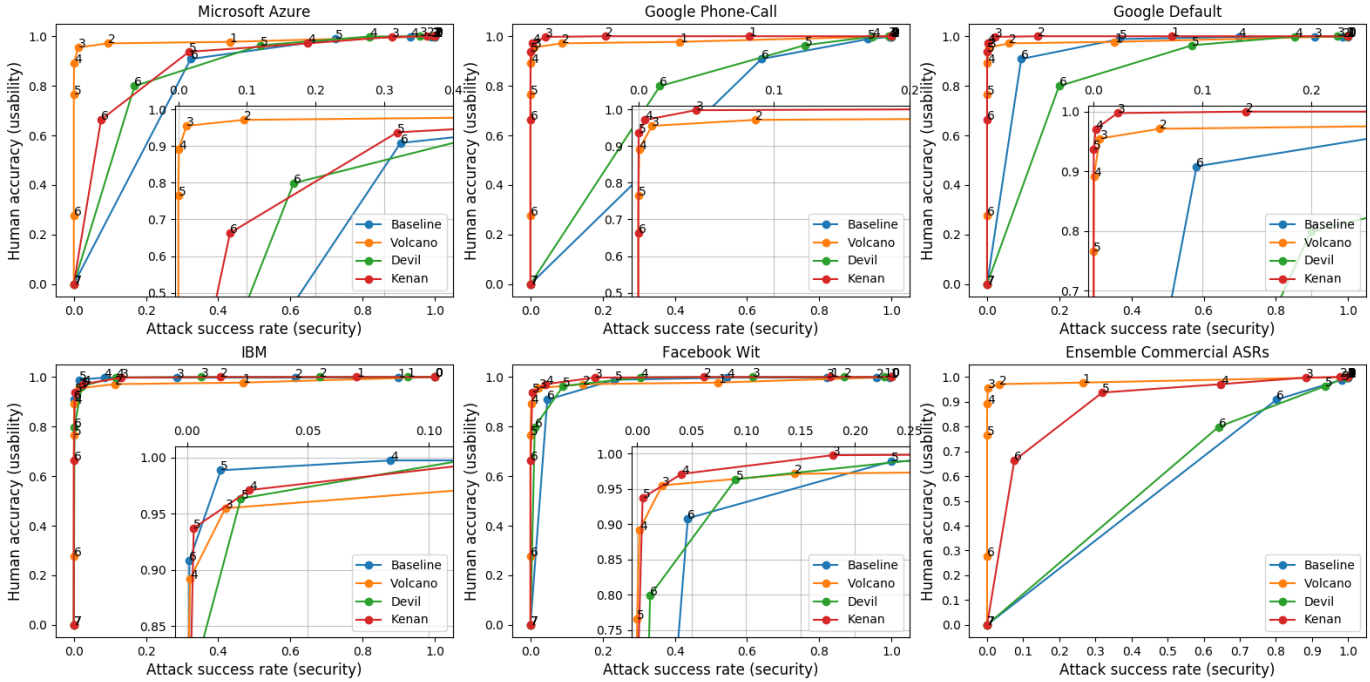

Fig. 4. The ROC curves of AdvCAPTCHAs against each commercial ASR model and self-trained model. Overall, AdvCAPTCHAs are closer to the upper left point than Baseline against almost all the attacker models.

TABLE X. THE RESULTS OF THE K-FOLD ON HUMANS AND LEAVE-ONE-OUT ON ATTACKER MODELS CROSS-VALIDATION. WE CALCULATE THE AVERAGE HUMAN-MACHINE GAP ACROSS ALL OF OUR THREAT MODELS.

Attacker Model Leave-one-out	Human Fold Number	Baseline	Volcano	Devil	Kenan
Microsoft Azure	1	0.2477	0.9482	0.4006	0.6209
	2	0.2744	0.9188	0.4689	0.5628
	3	0.2675	0.9535	0.4759	0.6601
	4	0.2466	0.9257	0.4411	0.6115
	5	0.2744	0.9465	0.4550	0.6323
	average	0.2621	0.9385	0.4483	0.6175
Google Cloud Speech (Default)	1	0.8322	0.9542	0.3496	0.9399
	2	0.8083	0.9248	0.4179	0.8818
	3	0.8499	0.9595	0.4249	0.9791
	4	0.843	0.9317	0.3901	0.9305
	5	0.718	0.9525	0.4040	0.9513
	average	0.8103	0.9445	0.3973	0.9365
Google Cloud Speech (Phone Call)	1	0.2867	0.9499	0.1581	0.9397
	2	0.2628	0.9205	0.2264	0.8816
	3	0.3044	0.9552	0.2334	0.9789
	4	0.2975	0.9274	0.1986	0.9303
	5	0.1725	0.9482	0.2125	0.9511
	average	0.2648	0.9402	0.2058	0.9363
IBM	1	0.9257	0.9441	0.8955	0.9372
	2	0.9018	0.9147	0.9638	0.8791
	3	0.9434	0.9494	0.9708	0.9764
	4	0.9365	0.9216	0.9360	0.9278
	5	0.8115	0.9424	0.9499	0.9486
	average	0.9038	0.9344	0.9432	0.9338
Facebook Wit	1	0.8800	0.9368	0.8272	0.9348
	2	0.8561	0.9074	0.8955	0.8767
	3	0.8978	0.9421	0.9025	0.9740
	4	0.8908	0.9143	0.8677	0.9254
	5	0.7658	0.9351	0.8816	0.9462
	average	0.8581	0.9271	0.8749	0.9314
Ensemble Commercial ASRs	1	0.1251	0.9578	-0.0192	0.6209
	2	0.1013	0.9284	0.0491	0.5628
	3	0.1429	0.9631	0.0561	0.6601
	4	0.1360	0.9353	0.0213	0.6115
	5	0.011	0.9561	0.0352	0.6323
	average	0.1033	0.9481	0.0285	0.6175
Black-box Self-trained Model	1	0.9251	0.917	0.9177	0.913
	2	0.9012	0.8876	0.9860	0.8549
	3	0.9429	0.9223	0.9930	0.9522
	4	0.9360	0.8945	0.9582	0.9036
	5	0.8110	0.9153	0.9721	0.9244
	average	0.9032	0.9073	0.8008	0.9096
White-box Self-trained Model	1	0.9253	0.9049	0.9165	0.8890
	2	0.9014	0.8755	0.9848	0.8309
	3	0.9431	0.9102	0.9918	0.9282
	4	0.9362	0.8824	0.9570	0.8796
	5	0.8112	0.9032	0.9709	0.9004
	average	0.9034	0.8952	0.9642	0.8856
Average across hidden commercial ASRs		0.7127	0.9353	0.6001	0.8812
Average across threat models		0.7801	0.9138	0.8042	0.8747