THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2008-31

Improving Auditory CAPTCHA Security

Sonja Bohr, Andrea Shome and Jonathan Z. Simon

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the A. James Clark School of Engineering. It is a graduated National Science Foundation Engineering Research Center.

www.isr.umd.edu

Improving Auditory CAPTCHA Security

Sonja Bohr[1*], Andrea Shome[1*] and Jonathan Z. Simon[1,2,3†]

[1] Institute for Systems Research, University of Maryland, College Park, MD 20815, USA.

[2] Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20815, USA.

[3] Department of Biology, University of Maryland, College Park, MD 20815, USA.

Abstract

CAPTCHAs are tests used by resource-rich websites to ensure that humans, but not malicious automated programs, have access to their resources. Most CAPTCHAs are visual tests (e.g. identifying distorted text), but auditory versions are necessary to provide access to the visually impaired, and are currently deployed at commonly used websites such as Google and Facebook. To be effective at deterring automated programs, they must be at least as secure as their visual counterparts. Assuming that the attacks against auditory CAPTCHAs will depend on automatic speech recognition systems (ASRs), we undertook the project of designing auditory CAPTCHAs that would take advantage of the weaknesses in ASRs as compared to the human auditory system. Examples of such weaknesses of ASRs, relative to humans, include impeded recognition in the presence of broadband and time-varying noise such as multiple simultaneous speakers. Results show that a combination of such disruptive noise types can outperform currently employed techniques while still maintaining human intelligibility.

[*] These authors contributed equally to this work.

[†] To whom correspondence should be addressed. E-mail: jzsimon@umd.edu

Introduction

The notion of a machine imitating human intelligence was first addressed as early as 1950 by English mathematician and logician Alan Turing. Acknowledged as the father of modern computing, Turing recognized that computers might eventually be able to imitate human thought in very convincing ways. Therefore, he suggested what is now known as the Turing test, where a human converses with a computer without seeing it. If the human is convinced by the computer's answers that it is human, then the machine passes the test and is deemed to have some level of human-like intelligence. The idea of a reverse Turing test, wherein a computer attempts to differentiate between a human and a computer, arose during the late 1990s when computer programs began to imitate humans in order to misuse the resources of internet-based systems. Tests developed to differentiate these programs from real humans took the form of what would come to be known as CAPTCHAs. [1-3]

In a general sense, CAPTCHAs are used to prevent automated software (i.e. "bots") from abusing the resources of various systems. They arose from a need to reduce or eliminate mass spam on websites and in web-based mail, to prevent bots posing as humans from entering chat rooms, to eliminate voting fraud on online polls, and most importantly to protect sensitive data that could be accessed by a simulated user. For example, a bot might use an email provider to create multiple fake email accounts in order to send spam. Similarly, it could use a social networking site to spam or otherwise annoy the other users. Or, it might simply create fake accounts in order to use up system resources. CAPTCHAs can be used to prevent this misuse of resources by requiring that users prove that they are human before using a system.

The first CAPTCHAs were visual, requiring users to enter a series of letters or numbers based on a given distorted image. The image was meant to create a barrier for simulated users because a program would presumably not be able to interpret the image in order to enter the correct string of characters. Visual CAPTCHAs, however, are an accessibility barrier to users who are visually impaired or have cognitive disabilities such as dyslexia. Auditory CAPTCHAs attempt to provide accessibility where visual ones cannot.[4]

Since auditory CAPTCHAs are necessary for every website which must provide access to all users, it is important that they be at least as secure as their visual counterparts—a website is only as secure as its weakest link.

It is important to understand the advantages as well as the vulnerabilities of auditory CAPTCHAs. For example, a bot might attempt to crack an auditory CAPTCHA by filtering out excess noise from the sound clip in order increase its signal quality. It may be stumped, then, if a human voice is mixed with background noise that shares similar frequencies; it then cannot distinguish which frequencies belong to the voice and which belong to the background noise. A human user may be able to understand the voice in that same situation, because the human brain has specialized ways to segment sound.

For instance, the “cocktail party effect” [5] is a well-known example of the human brain’s surprisingly facile ability to separate sound from one source among many. A common example of this is that someone at a noisy party can understand what another person is saying, despite the fact that the noise in this situation (i.e. all the other speakers) covers the same frequency range as the speech. A computer would likely be unable to accomplish this since it would be confused by so many voices of similar frequency and volume. This is the idea behind typical auditory CAPTCHAs—to utilize the human brain’s ability to segregate sound streams from disruptive noise.

In our research, we looked at some of the techniques currently used in some of the most advanced CAPTCHAs. We attempted to duplicate some of these techniques and to test how well they stand up to an automatic speech-recognition program (ASR). We also formulated some of our own distortion methods to observe how well they fared against the ASR. We focused exclusively on CAPTCHAs that contained a series of spoken digits, partly for simplicity, but also because, despite the fact that they require knowledge of spoken English, their pronunciation is more widely known among non-English speaking users than almost any other English words.

Hypotheses

The auditory CAPTCHAs used by the companies of Google [6] and Facebook [7] to protect their websites should represent some of the most advanced distortion techniques used today, since their high-profile internet presence makes their resources so desirable. In listening to these auditory CAPTCHAS, and ones from other popular websites, we observed a variety of techniques used to distort the spoken text. For example, many of the spoken digits sounded as if reverberation had been added. Other spoken text sounded like a tempo or pitch change had been applied. Both Google and Facebook intermixed both male and female voices in one CAPTCHA file and added background noises derived from human speech. For instance, Facebook’s CAPTCHA background noise was simply reversed human speech. Many CAPTCHAs also featured “false leads”— sound elements which were as loud as the spoken digits, but to the human ear were clearly reversed numbers or gibberish. Our work focused on the distortion to the numbers themselves but not the false leads intended to fool an automatic speech recognizer.

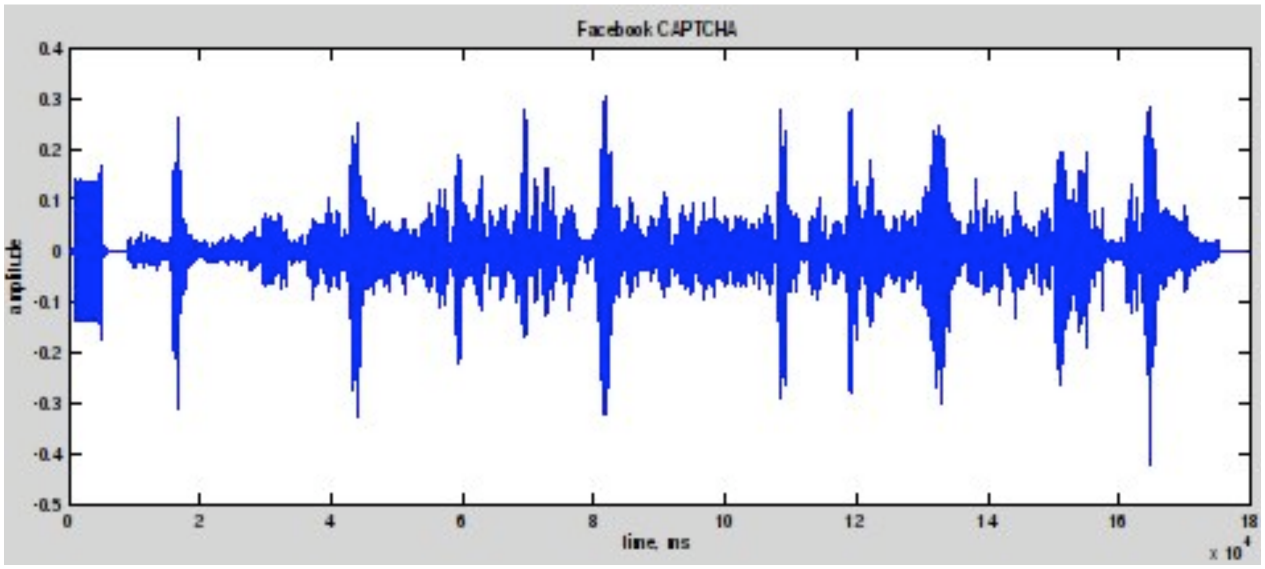

Figure 1: The sound waveform of an example Facebook CAPTCHA in its entirety.

We set out to test our own versions of some of these observed distortion techniques, to determine which were most effective at making spoken digits unrecognizable by an ASR while still recognizable to a human. We tested a variety of background sounds derived from human speech, as well as other methods such as cutting out part of the speech signal. The specific techniques are discussed below. We expected each of our distortions to decrease the recognition rate of a speech recognition program when that program was trained on normal speech, although we did not know by how much. In addition, we hypothesized that combinations of these techniques would decrease the ASR's recognition rate to lower than those found with Facebook or Google's CAPTCHAs.

Methods

We used the freely available speech recognizer HTK (Hidden Markov Model Toolkit)[8] to test the power of our various distortion techniques. This speech recognizer can be trained on a number of sound files to recognize the difference between different words, or, in our case, digits. It can then be tested on new sound files, outputting which of these test words were identified correctly in a given test. In this experiment, it used a six-step Markov modeling technique in order to create guesses and provide recognition statistics for the input files. In order to use this speech recognizer, we needed to obtain a large database of spoken digits to train and test on.

To reflect the wide range of voices used to create advanced CAPTCHAs, we chose as our training corpus a collection of spoken digits from a subset of the TIDIGITS database.[9] There are approximately 2000 separate files in this database—almost 200 recordings of each digit zero through nine, spoken by different individuals. The people recorded were men and women with varied regional American accents, speaking at a variety of speeds and rhythms. For example, the final /r/ of "four" is not always present. Our hypothesis was that by training on a wide variety of speaking styles, our speech recognizer would likewise be able to identify the different speaking styles featured in the online CAPTCHAs.

The files were downsampled from 20 kHz sampling frequency to 8 kHz, in order to match the sampling frequency of typical CAPTCHAs. The sounds were also normalized in amplitude and their mean value set to zero. Furthermore, initial and final periods of silence were trimmed, in order to increase the performance of the ASR. Trimming was accomplished by a combination of automatic and manual methods.

To determine human intelligibility level of the distorted spoken digits, a computer script was created to play randomly chosen files from the distorted testing database to human listeners. These subjects then entered via the computer keyboard their best guess for the digit, with feedback provided for correct vs. incorrect answers. A batch of digits distorted by a particular method was judged to be “intelligible” if no more than 3% were incorrect. Some digits were more prone to errors than others, e.g. some of the shorter ones such as “six”.

Only two human subjects were used to rate intelligibility (the first two authors), an issue which is discussed below. In future studies, more test subjects (and test subjects from a greater variety of backgrounds) should be used to get an accurate measure of intelligibility.

As mentioned above, our distortion techniques consisted either of adding background sound derived from human voices and other sources, or manipulating the sound itself. Preliminary attempts to manipulate the sounds included changing the tempo and/or pitch. It was found, however, that these changes did not change HTK’s recognition significantly from the original, undistorted files, so the technique was not pursued further.

Another preliminary technique was to modify a file sound by cutting out various parts of its signal. Kochanski et. al. [10] suggested that cutting out 60 ms of every 100 ms should decrease an ASR’s recognition rate to nearly zero without affecting human recognition. We tested this idea by breaking up the digits into 8 ms intervals and randomly removing half of the intervals. Besides removing parts of the signal in this way, we also attempted a similar technique where we replaced the deleted portions with white noise. We hoped that the human ear would be able to filter out this background noise and ignore it, but that the random signal would cause errors in HTK.

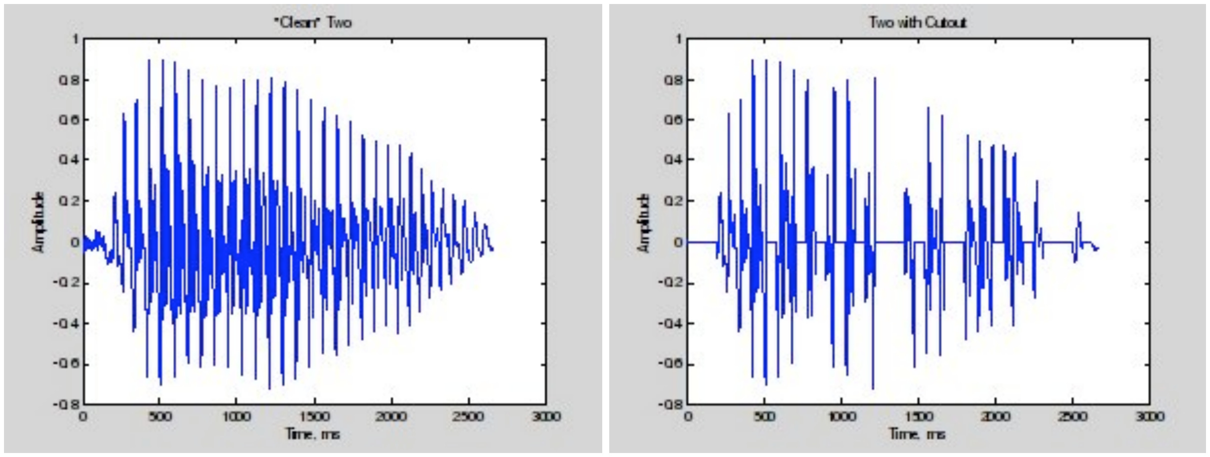

Figure 2: The spoken digit “two”, before and after the cutout technique is applied.

At first, these techniques did degrade HTK’s recognition significantly. HTK uses windows of a specified duration (typically 25 ms) to analyze small intervals of the sound, so it was found that as long as the window size was somewhat greater than the size of the removed portions, recognition did not decrease dramatically. Even with as large an interval as 60 ms cut out, we were able to increase the window size such that the recognition rate did not degrade significantly. Intervals larger than 60 ms, however, would have degraded human recognition too much.

Another form of distortion that we attempted was to add echo. An echo is created by taking the sound samples and adding them back into a set number of samples later (the delay). A double echo is created by performing this same process but for two different delays, and a triple echo with three, etc. We hypothesized that this distortion technique would achieve low recognition from HTK, since Facebook and Google CAPTCHAs include reverberation, which is similar to echo. The results were poor, however – HTK had high recognition rates, within about 90%, for most all of the echo files we tested.

We also tried adding reverberation directly, adding either mathematically simple or “natural” sounding reverberation. However, as with echo, after several trials it was found that these reverberation techniques did not have a significant impact on HTK’s recognition rate. For this reason they were not pursued further.

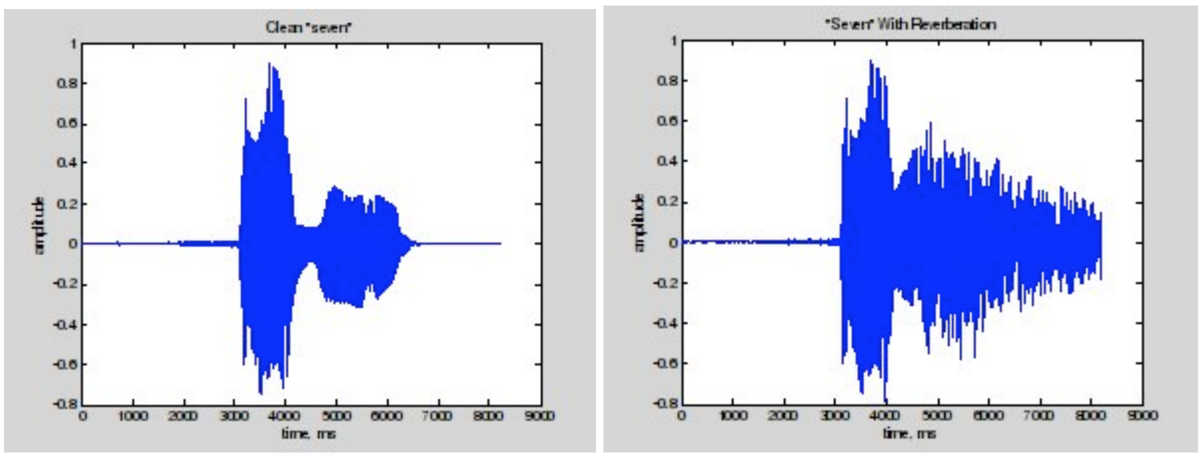

Figure 3: The spoken digit “seven”, before and after reverberation is applied.

As discussed above, the human auditory system can easily distinguish one human speaker from a cacophony of other voices, a task typically difficult for a computer. For this reason, we also decided to use background noises which themselves featured human speech in an attempt to better mask the target digit. This technique was particularly effective because the background noises share a similar frequency range with the spoken digit. Whereas the Facebook and Google CAPTCHAs use the reversed speech of one or two people speaking English, we decided to test backgrounds featuring more voices and speech of different languages. Each background noise was superimposed on the clean speech sounds. Each noise waveform was individually normalized and then the entire digit with distortion was again normalized.

Most of our background noise examples files were found in the audio file database Soundsnap, [11] a freely available database of users-contributed sound effects, consisting of a few types of background noise. One type was *cheer*, which consisted of recordings of people cheering at sporting events. This background noise covered a broad range of frequencies, especially in the higher range. Another type was *chatter*, e.g. the sounds of many people speaking simultaneously in a crowded place such a restaurant, with enough speakers that no one individual can be understood. The frequency range is lower, and has more temporal structure, than the *cheer* type. We hypothesized that the broad spectrum of frequencies found in the cheer and chatter backgrounds would help prevent cracking the CAPTCHA because a simple frequency selective filters(e.g. low pass or high pass) used to extract the spoken digit would not be effective.

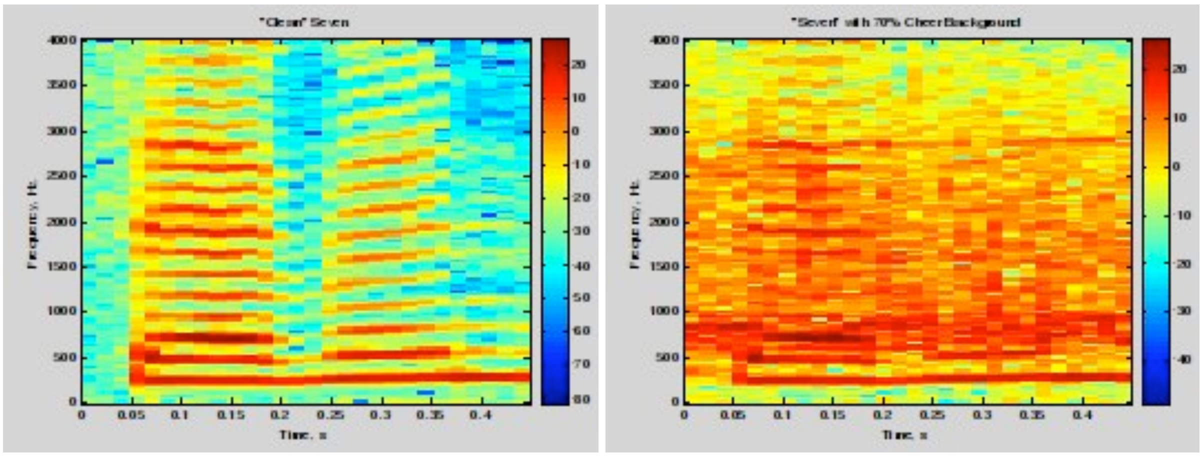

Figure 4: The spoken digit “seven” before and after *cheer* is added.

Another type of background noise was created by combining ten random digits from our speech database. The resulting sound was not recognizable as any single number, but was rather a confusing mix that we called *babble*. Since this background noise was created using speech samples from the same pool used to create the testing files, this background was like *cheer* and *chatter* in that it featured many voices, but the frequency range was more similar to that of the digits themselves. We also tried temporally reversing the babble, to prevent any recognition of the digits used to make the background noise from distracting listeners from the target digit.

We also used the babble background in another way. We realized that, for human recognition, hearing the end of a number is not as important as hearing the first part of the number. Because the digits zero through nine all begin with different sounds, humans can recognize a digit before it is even finished. Knowing that we could take advantage of this ability, the *babble* noise was added to the last 75% of the target digit waveform, giving a new type called *end-babble*, in an attempt to mask the end of the digit but leave the beginning with cleaner speech.

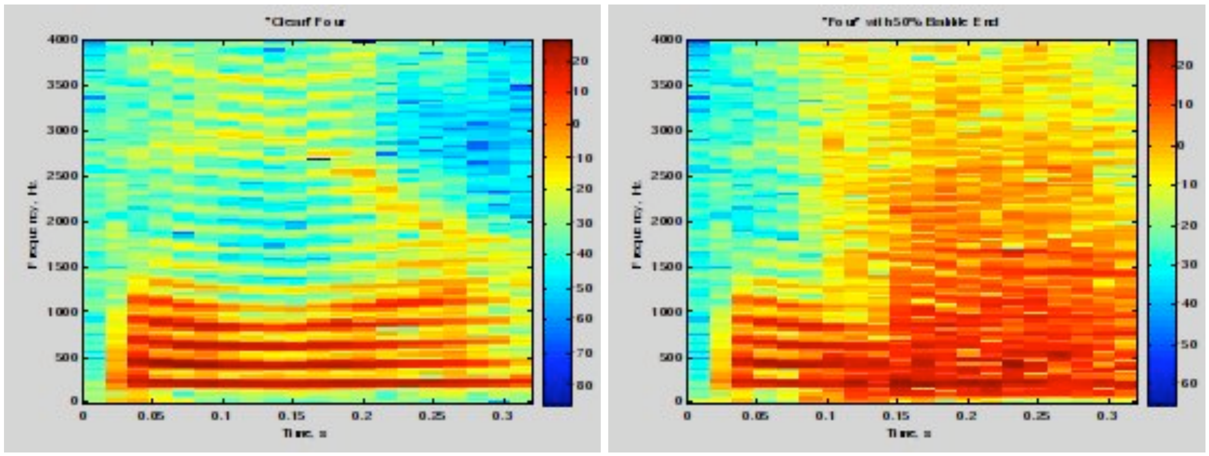

Figure 5: The spoken digit “four” before and after *end-babble* is applied.

We hypothesized that background of another language would be less distracting to a human user than an English background, and therefore added speech in a foreign language as background noise (French). We used the AT&T Labs Text-to-Speech converter [12] to convert several paragraphs of French text into artificial speech waveforms, allowing us access to multiple voices, male and female. This additional variation is important when working with HTK because it reduces the ability of the computer to recognize speech patterns. The French distortion was similar to babble in intelligibility levels for human subjects.

Because Facebook’s CAPTCHAs achieved significantly lower recognition rates than Google’s (in fact an earlier version of the Google CAPTCHA was cracked in March 2008 [13]), our goal was to create CAPTCHAs which would achieve even lower rates than Facebook’s, while still maintaining human intelligibility. In using combinations of the different types of distortion, we found at least one combination of background noises that achieved this goal – mixing *cheer*, *chatter*, and *end-babble* types. The figure below displays the spectrogram of a spoken digit before and after this combination is applied.

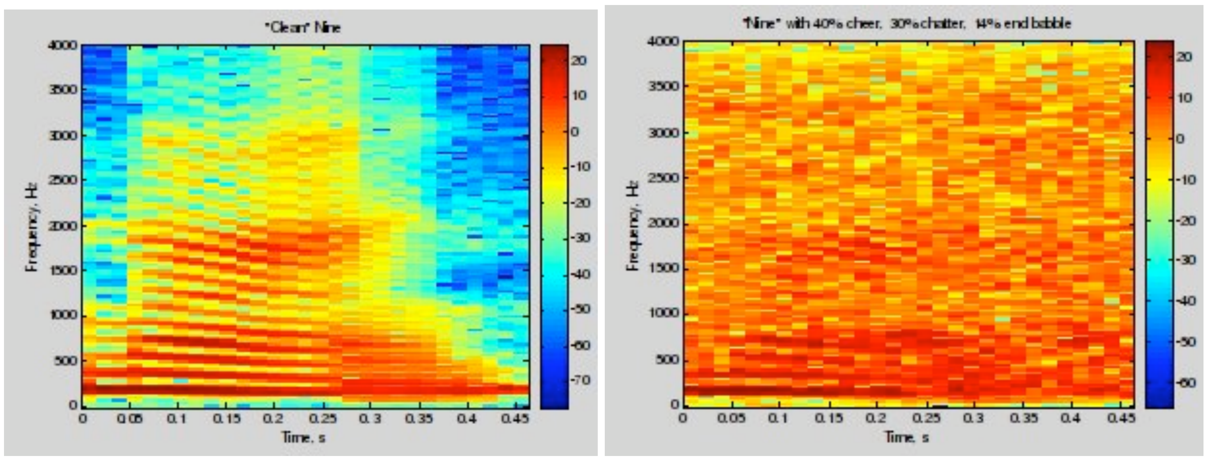

Figure 6: An example of the most effective combination type applied to the spoken digit “nine”.

Results

Figure 7 shows the trends of individual distortions as higher percentages of each distortion are applied. As the corruption percentage increases, the graph shows which techniques better degrade HTK’s ability to recognize the testing database. The second-to-last data point for the trend lines corresponds to 100% corruption (the point deemed to be just-barely-intelligible). In other words, the techniques with a lower recognition rate at this second-to-last point were overall more effective at masking the identity of the digits to the ASR while still maintaining human intelligibility.

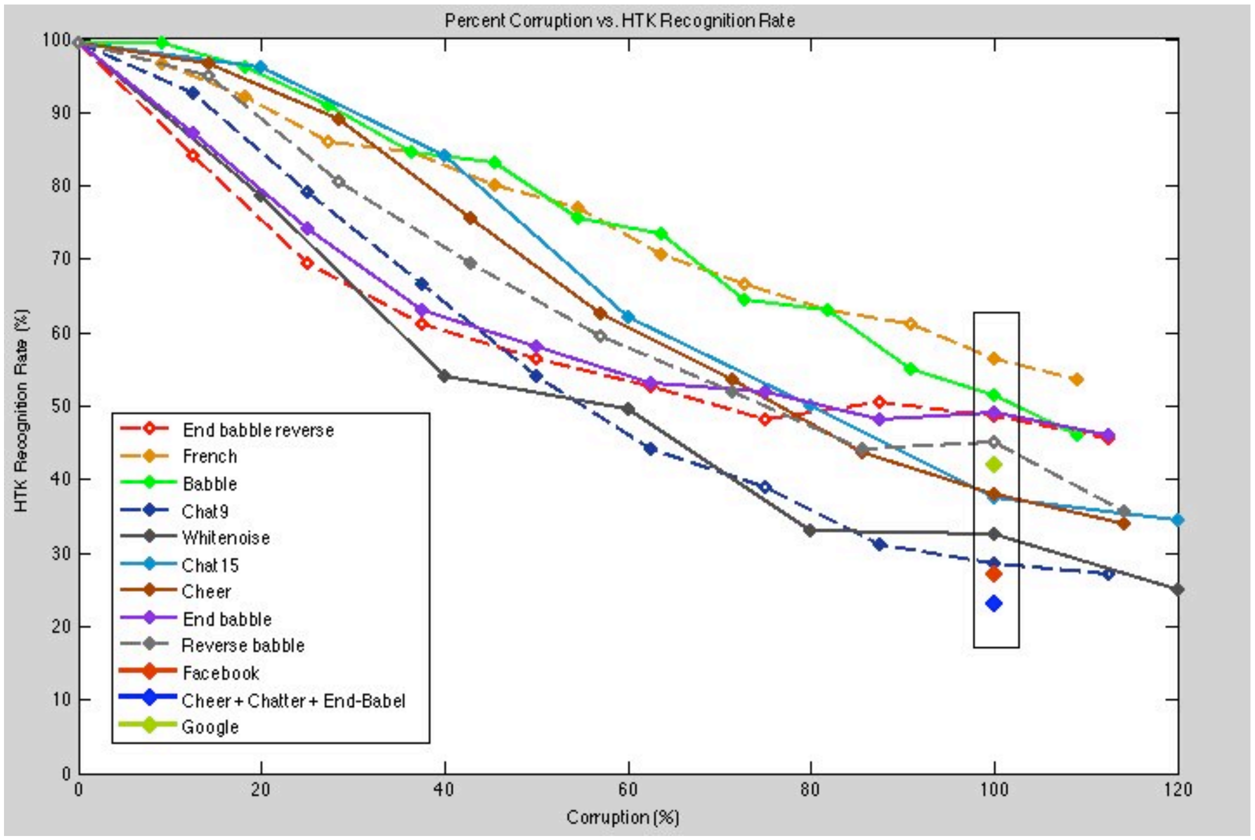

Figure 7: HTK recognition rates of our various distortion methods. The horizontal axis corresponds to an increasing corruption of the spoken digits, where 100% corruption (the second-to-last data point) means that the digits are just barely intelligible. Lower HTK recognition rates at this point indicate a more effective distortion.

Table 1 shows the automatic recognition rates for the more successful distortion techniques, and extracted digits from Google and Facebook CAPTCHAs. The second column lists recognition rates of the various distortions when HTK was trained on the 2,000 clean digit files. The third column shows the recognition rates after HTK was instead trained on already digits distorted using the same technique as those tested on. These percentages are important because a technique which is varied enough to withstand being trained on examples of itself is more resilient to attack than one which can be easily cracked by discovering or mimicking the distortion. Distortions with a higher percentage in the third column are more susceptible to this kind of attack than those with lower percentages.

Distortion Type	HTK Recognition trained on entire database (higher is worse)	HTK Recognition trained on self (higher is worse)
end-babble	74%	93%
french	56.5%	88.0%
babble	55.0%	90.0%
Google CAPTCHA	42.0%	98.0%
cheer (70%)	38.0%	73.0%
chatter (60%) + *end-babble* (14%)	33.5%	84.5%
chatter (80%)	28.5%	80.0%
chatter (70%) + *end-babble* (14%)	28.0%	83.0%
Facebook CAPTCHA	27.0%	62.5%
cheer (40%) + *chatter* (30%) + *end-babble* (14%)	23.0%	80.0%

Table 1: HTK Recognition of our Distortion Techniques and CAPTCHAs. Percentages in the Distortion Type (first) column refer to fraction of full amplitude. The second column shows the recognition percentages when HTK was trained on clean speech. The third column shows the recognition percentages when HTK was trained on digits distorted in an identical way as the testing files. Distortion types are sorted by the second column: lower is better.

This table shows that our most successful technique was a combination of cheer, chatter, and end-babble background noise. When trained on undistorted digits, this combination achieved a lower recognition rate (23%) than did the digits from Google (42%) or Facebook (27%) CAPTCHAs.

Conclusions

This project was limited in several ways which could be improved on in the future. Even though our training database contained a variety of speech styles, pitches, and intonation, an even more extensive database would likely improve a speech recognizer's accuracy. The ASR could also be trained to recognize the silence before and after the spoken digits. This was something which proved to be too time-consuming, but training the recognizer to automatically distinguish between silence and the number would allow it to recognize digits which had not been hand-trimmed. Similarly, the recognizer could be trained to distinguish between a given background noise and the digit.

The work would be most improved by using more human subjects to estimate the intelligibility of different distortion types. Using more subjects, unfamiliar with the distortion techniques and digit database, should help get a more objective assessment of intelligibility.

We discovered that several of our results were very promising: several combinations of distortions were able to achieve lower recognition rates than did Facebook's CAPTCHAs, as low as 23% recognition rate by HTK (Figure 8). Variations of this combination such as increased babble and decreased cheer also achieved similar recognition rates. The variations most challenging to the human ear, though still intelligible by the criteria above, went as low as 20.5%.

Our *end-babble* technique was also surprisingly successful. Although on its own this technique did not degrade recognition very much, *end-babble* was essential to create the low recognition rates obtained with our combination techniques.

Our testing not only showed how well these techniques fared against HTK when trained on clean speech, but how well they performed when HTK was trained on a database distorted in the same way. These figures are important because they indicate how difficult it would be to crack a CAPTCHA by simply training using other examples the same website successfully identified by a human aide. Facebook's recognition rate when trained on itself was only 62.5%, whereas for Google the rate jumped to 98.0%. This shows that Google CAPTCHAs may be especially vulnerable to attack which use samples derived from Google itself to train their ASRs. Our *cheer* + *chatter* + *babble* combos consistently got around 80.0% recognition rates when trained on themselves, and other distortions lay between 73% and 93%. None of our distortions achieved as low a recognition rate against themselves as did Facebook, showing that these techniques are weaker than Facebook's CAPTCHAs in this respect. However, using different techniques for different digits in the same CAPTCHA might alleviate this vulnerability independent of the distortion type.

The 4% discrepancy between the recognition rates of our combination distortions and those of Facebook CAPTCHAs is not great enough to show that our technique is necessarily superior to those currently used in advanced CAPTCHAs. However, the similarly low recognition rates suggest that our technique can also be used successfully. Thus, in combination with existing methods, our techniques of background noise to mask the digits can be used to further increase the security of auditory CAPTCHAs.

Acknowledgements

We would like to especially thank Nai Ding for his help throughout this entire project. This work was funded by a National Science Foundation grant #0647321.

.

References

1. von Ahn L, Blum M, Hopper NJ, Langford J: CAPTCHA: Using hard AI problems for security. Advances in Cryptology-Eurocrypt 2003, 2656:294-311.
2. von Ahn L, Blum M, Langford J: Telling humans and computers apart automatically. Communications of the ACM 2004, 47:57-60.
3. http://www.captcha.net/, URL: <http://www.captcha.net/>
4. Yan J, Ahmad Salah El A: Usability of CAPTCHAs or usability issues in CAPTCHA design. In Proceedings of the 4th symposium on Usable privacy and security. Edited by. Pittsburgh, Pennsylvania: ACM; 2008.
5. Cherry EC: Some Experiments on the Recognition of Speech, with One and with 2 Ears. Journal of the Acoustical Society of America 1953, 25:975-979.
6. Google (Create New Account), URL: < https://www.google.com/accounts/NewAccount>
7. Facebook (Create New Account), URL: <http://www.facebook.com/r.php>
8. Hidden Markov Model Toolkit, URL: <http://htk.eng.cam.ac.uk/>
9. Leonard RG, Doddington G: TIDIGITS. Published: Linguistic Data Consortium; 1991. URL: <http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S10>
10. Kochanski G, Lopresti D, Shih C: A Reverse Turing Test using Speech. In 7th International Conference on Spoken Language Processing (ICSLP2002 - INTERSPEECH 2002); Denver, Colorado, USA: 2002:1357-1360.
11. Soundsnap, URL: <http://www.soundsnap.com>
12. AT&T Labs Text-to Speech Demo, URL: <http://www.research.att.com/~ttsweb/tts/demo.php>
13. Wintercore Labs: Breaking Gmail's Audio Captcha, URL: <http://blog.wintercore.com/?p=11>