The Failure of Noise-Based Non-Continuous Audio Captchas

Elie Bursztein* Romain Beauxis† Hristo Paskov* Daniele Perito‡ Celine Fabry, John Mitchell*
*Stanford University, †Tulane University, ‡INRIA
{elie|hpaskov|jcm}@cs.stanford.edu, rbeauxis@tulane.edu, perito@inrialpes.fr, celine@celine.im

***Abstract*—CAPTCHAs, which are automated tests intended to distinguish humans from programs, are used on many web sites to prevent bot-based account creation and spam. To avoid imposing undue user friction, CAPTCHAs must be easy for humans and difficult for machines. However, the scientific basis for successful CAPTCHA design is still emerging. This paper examines the widely used class of audio CAPTCHAs based on distorting non-continuous speech with certain classes of noise and demonstrates that virtually all current schemes, including ones from Microsoft, Yahoo, and eBay, are easily broken. More generally, we describe a set of fundamental techniques, packaged together in our Decaptcha system, that effectively defeat a wide class of audio CAPTCHAs based on non-continuous speech. Decaptcha's performance on actual observed and synthetic CAPTCHAs indicates that such speech CAPTCHAs are inherently weak and, because of the importance of audio for various classes of users, alternative audio CAPTCHAs must be developed.**

I. Introduction

Many websites rely on *Completely Automated Public Turing tests to tell Computers and Humans Apart* (CAPTCHA[1]) [18] to limit abuse in online services such as account registration. These tests distinguish between humans and automated processes by presenting the user with a task that is easy for humans but hard for computers. Designing such tests, however, is becoming increasingly difficult because of advances in machine learning. In particular, the widely used category of image based captchas have received close scrutiny recently [17], [25], [30], [31].

While widely provided for accessibility reasons, audio captchas have received substantially less scientific attention. Virtually all current audio captchas on popular sites consist of a sequence of spoken letters and/or digits that are distorted with various kinds of noise. For simplicity, we will refer to such non-continuous audio captchas simply as *audio captchas* in the remainder of the paper.

Almost a decade ago, Kochanski et al [15] investigated the security of audio captchas and developed a synthetic benchmark for evaluating automatic solvers. This study, which concludes that humans outperform speech recognition systems when noise is added to spoken digits, has guided the design of modern audio captchas. Two later and independent studies [27], [23] demonstrate that a two-phase segment-and-classify approach is sufficient to break older versions of Google and Yahoo audio captchas. Two-phase solvers operate by first extracting portions of the captcha that contain a digit and then using machine learning algorithms to identify the digit. When machine learning algorithms are trained to overcome the distortions of an individual captcha scheme, they are far more effective than speech recognition systems [3], [26].

In this paper, we describe a two-phase approach that is sufficient to break modern audio captchas. One reason that audio captchas might be weaker than visual captchas stems from human physiology: the human visual system consumes a far larger portion of our brains than the human audio processing system. In addition, modern signal processing and machine learning methods are fairly advanced. As a result, the difference between human and computer audio capabilities is likely significantly less than the difference between human and computer visual processing.

While we believe our results demonstrate practical breaks, there is room for some debate on the success rate needed to consider a captcha scheme ineffective in practice. In many applications, a concerted attacker may attempt to set up fraudulent accounts using a large botnet (e.g., [16]). Since modern botnets may control millions of compromised machines [24], it is reasonable to expect that an attacker could easily afford to make one hundred attempts for every desired fraudulent account. Therefore, a computer algorithm that solves one captcha out of every one hundred attempts would allow an attacker to set up enough fraudulent accounts to manipulate user behavior or achieve other ends on a target site. A target 1% success rate is conservative relative to other studies, which hold that "*automatic scripts should not be more successful than 1 in 10,000*" attempts [11]. In fact, we greatly surpass 1% in all but one case.

Contributions. We present Decaptcha, a two-phase audio captcha solver that defeats modern audio captchas based on non-continuous speech. The system is able to solve *Microsoft*'s audio captchas with *49%* success and *Yahoo*'s with *45%* success, often achieving better accuracy than humans. This performance also comes at a low training cost because Decaptcha requires *300 labeled captchas* and approximately *20 minutes* of training time to defeat the hardest schemes. After training, tens of captchas can then be solved per minute using a single desktop computer.

We also evaluate Decaptcha on a large-scale synthetic corpus. Our results indicate that non-continuous audio captcha schemes built using current methods (without semantic noise) are inherently insecure. As a result, we suspect that it may not be possible to design secure audio captchas that are usable by humans using current methods. It is therefore important to explore alternative approaches.

[1]For readability, we will write captcha instead of CAPTCHA in the rest of this paper.

Decaptcha's success stems from the following contributions:

- **Automated segmentation.** We present a low-cost technique based on acoustic energy analysis that accurately segments noisy audio.
- **Speech analysis.** We test the performance of a variety of techniques for speech and signal analysis in the presence of various kinds of noise.
- **Fully Automated Classification.** We demonstrate the efficacy of a technique for automatic parameter tuning of the RLSC algorithm suggested by [21].
- **Real World evaluation.** We evaluate the performance of Decaptcha on a large corpus of real world Captchas from Authorize, Ebay, Microsoft, Recaptcha, Yahoo and Digg.
- **Synthetic evaluation** We perform a large scale evaluation of the effects of different kinds of noise on captcha security. The *4.2 million* synthetic captchas used for this study are generated by a method presented in [15], which is freely available.[2]

Outline. The remainder of the paper is organized as follows: In Section II we review the audio processing and machine learning techniques that are used in Decaptcha. In Section III we describe how Decaptcha is implemented and the design decisions behind each step. In Section IV we present the real captcha schemes that were evaluated, give Decaptcha's performance on them, and discuss specifics. In Section V we present the synthetic captchas that were tested using Decaptcha, give performance as a function of noise, and discuss. Finally, we present related work in Section VI and conclude in Section VII.

II. Background

Decaptcha can be split into three main components: a segmentation stage that extracts spoken digits, a representation scheme for the extracted digits, and a classification stage that recognizes each digit. While Decaptcha uses two active phases, the intermediate representation is an important part of a two-phase solver because of its impact on performance. Accordingly, the first three subsections of this section provide a high-level overview of the concepts used in each of the three components. We then discuss the metrics that are used to measure Decaptcha's performance.

A. Segmentation

An audio captcha is initially represented as a signal $S = s_0, \ldots, s_{n-1}$ where each s_i denotes the amplitude of the signal at fixed time intervals. The spacing of these time intervals is called the *sampling rate*. Segmentation finds contiguous intervals of S that contain digits and as little extraneous noise as possible. We briefly discuss useful statistics for this process, referring the reader to [13] for a more detailed discussion of concepts used in this section and the next.

[2]Those interested in the corpus should contact the authors.

Root Mean Square (RMS). RMS measures the *acoustic energy* of an audio signal and is defined as $RMS(S) = \sqrt{\frac{s_0^2+\ldots s_{n-1}^2}{n}}$. Values are reported on a logarithmic scale in units of decibels (dB).

Signal to Noise Ratio (SNR). SNR measures the relative energy between audio and noise signals as $10\log_{10}(\frac{RMS\ signal}{RMS\ noise})$.

B. Signal Representations

Once each digit has been extracted, it is represented as a temporal signal. This section discusses the transformations that can be applied to this signal to improve digit recognition.

Discrete Fourier Transform (DFT). The DFT, also referred to as the Fast Fourier Transform (FFT), of a signal decomposes it into a series of sinusoids. Formally, $DFT(S)$ reparameterizes S into n complex coefficients $f_0, \ldots, f_{n-1}$ where $f_i = \sum_{u=0}^{n-1} s_j e^{-2\pi u \frac{i}{n}}$. The f_i quantify how much of the signal occurs as a sinusoid with frequency of $\frac{i}{n}$ and therefore represent S in the frequency domain. The inverse DFT (IDFT) is computed using the following formula: $s'_i = \frac{1}{n}\sum_{u=0}^{n-1} f_j e^{2\pi u \frac{i}{n}}$.

Cepstrum. The cepstrum [6] is computed as $S \rightarrow DFT \rightarrow magnitude \rightarrow log \rightarrow IDFT \rightarrow magnitude$ where the magnitude and log operations are applied element-wise to an input sequence. The cepstrum gives a frequency analysis of the energy of the harmonics contained in the signal.

Spectro-Temporal Features (STF). STFs refer to a class of 2D representations that track how a unidimensional transform (such as a DFT or cepstrum) changes over time. Given a unidimensional transform F, the STF of S computes F on intervals of length m, called *frames*, every k time steps. We will refer to the STF using a DFT or a cepstrum as the **TFR** and **TCR** respectively.

Two-Dimensional Cepstrum (TDC). Our last representation scheme is another 2D variant of the cepstrum that has proved useful for voice recognition [1], [10]. Given the TCR of S computed over d frames of length m and stored in a $m \times d$ matrix X, the TDC computes the inverse DFT of each *row* of X. This depicts the frequencies with which cepstral coefficients change across frames.

We have experimented with the following signal analysis techniques in an attempt to provide better noise filtering.

Blackman Window. Blackman windowing is a technique used to combat deleterious effects on the DFT caused by discontinuities that appear around the edges of each frame. In particular, a frame is weighted by a function that drops to zero around its edges before computing the DFT. The weight of each sample is given by $w_i = 0.42 - 0.5\cos\left(\frac{2\pi i}{m}\right) + 0.08\cos\left(\frac{4\pi i}{m}\right)$.

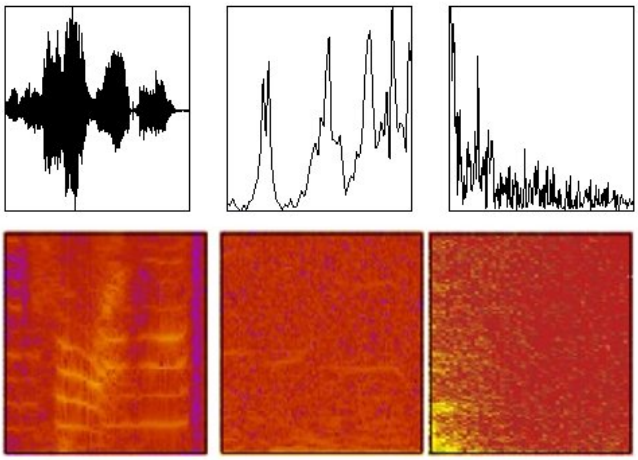

Figure 1. Representations of a "four" digit

Mel Scale. The mel scale redistributes frequencies on a logarithmic scale that matches how humans perceive frequencies. The mel value of a frequency f is given by $m = 2595 \log_{10}\left(\frac{f}{700} + 1\right)$. We also use the mel scale to shrink our data without impacting useful information by averaging intervals of mels.

Figure 1 illustrates, in top-down left to right order, the original representation of the digit four, its DFT, Cepstrum, TFR, TCR, and TDC.

C. Classification

The classification stage of Decaptcha receives a digit represented in one of the aforementioned schemes and attempts to recognize it. This section provides an overview of the classification algorithm Decaptcha employs (see [22] for a thorough description). We begin with a high level discussion of classification before moving on to the actual algorithm.

Binary Classification. Given N labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, binary classification attempts to train a classifier, i.e. find a decision rule, that accurately predicts the labels of unseen points. It is customary to assume that a classifier outputs a real number that is thresholded at some value – typically zero – to determine a binary label.

One Versus All (OVA). Multi-category classification extends the binary paradigm by allowing labels to be any integer in $1, \ldots, T$ for a fixed number, T, of classes. The OVA scheme solves multi-category classification by training T separate binary classifiers. Classifier i is trained on labels that are positive for all training points labeled as class i and negative for all others. A new point is classified by running it through all T classifiers and taking the index of the highest scoring one. For example, if we are classifying pictures of cats, dogs, and horses, we would train 3 binary classifiers. The first classifier would be trained by labeling all pictures of cats as positive and all pictures of dogs and horses as negative. The second and third classifiers would be trained in a similar fashion. Given a new picture, we would, for example, label it a cat if and only if the first classifier gives it a higher score than the other two.

Testing Error. Testing error measures how well a classifier generalizes and is defined as the expected frequency with which it will misclassify a new point. It is clear that the objective of both, binary and multi-category classification, is to come up with a decision rule that has minimal testing error. A reliable estimate of testing error can be obtained by computing a classifier's error on a sample of points that were *not* used during training.

Regularized Least Squares Classification (RLSC). RLSC is an adaptation of a regression algorithm to binary classification [22]. It was introduced as a fast yet similarly performing alternative to the Support Vector Machine (SVM), one of the most successful binary classifiers. In its simplest form, RLSC finds a hyperplane $w \in \mathbb{R}^d$ that separates its training data into positive and negative classes by solving

$$w = \min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \lambda w^T w$$

The label of a new point $x \in \mathbb{R}^d$ is found by taking the sign of $w^T x$. It takes $min(O(N^3), O(Nd^2 + d^3))$ to solve for w. We can also train an entire OVA scheme consisting of T classifiers in the time it takes to train a single binary classifier.

Leave One Out (LOO). The LOO error of a binary classifier is another estimate of the testing error. It is found as the average error the classifier makes on each point when it is held out from the training set. Computing the LOO error in general requires training N classifiers, but in the case of RLSC, it can be found at the cost of training a single binary classifier. The LOO error of an OVA scheme is similarly found by holding out each point and checking whether the OVA scheme trained on $N - 1$ points predicts its label. This multi-category LOO error can be found efficiently by computing the LOO error of each binary classifier in the OVA scheme.

Parameter Tuning. The accuracy of RLSC critically relies on a regularization parameter λ that keeps the classifier from overfitting or underfitting its training data. A good value for λ can be found by trying a range of regularization parameters and selecting the one with lowest LOO error. It is possible to compute the LOO error of $O(min(N, d))$ candidate values in the time it takes to solve a single binary classifier.

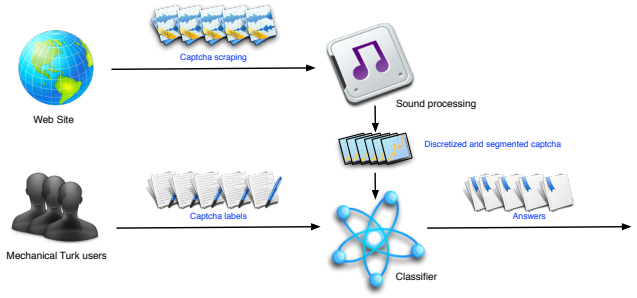

Figure 2. Overview of the System

D. Metrics

We conclude our overview with a discussion of the metrics that are used to evaluate the performance of a captcha solver.

The most basic measure of performance is *accuracy*, the fraction of captchas that are answered correctly. However, solvers may also select which captchas they respond to, in which case we need more fine grained metrics. *Coverage* is the fraction of captchas that a solver attempts to answer. For Decaptcha, this is the number of captchas that were segmented correctly. *Precision* is the accuracy of the solver, computed only on the captchas it attempts to answer. This metric is equivalent to Decaptcha's classification accuracy. Finally, as will be discussed later, Decaptcha is a supervised solver that requires a set of sample captchas to train on. A measure of a supervised algorithm's *efficiency* is given by the *corpus size*, the number of labeled captchas, it needs to train on to obtain a specific accuracy. A lower value indicates a less secure captcha scheme because it requires less training effort to break.

III. Decaptcha

This section describes Decaptcha and the design decisions behind it. We begin with an overview of the system and then discuss its segmentation, representation, and classification components.

A. Overview

Decaptcha is a supervised algorithm that must be trained on each captcha scheme. Training requires a set of captchas labeled with their answers. It outputs an automatic solver and an estimate of Decaptcha's expected performance on the scheme. Figure 2 depicts the interactions of Decaptcha's three processing stages during and after training. The segmentation stage is *unsupervised*, i.e. it is does not undergo training, and it "cuts" out the pieces of a captcha that are likely to contain digits. Each cut is then converted to a representation scheme that must be specified before training. Finally, a (supervised) classifier is trained to recognize the digit in each cut. The next three sections detail each of these stages.

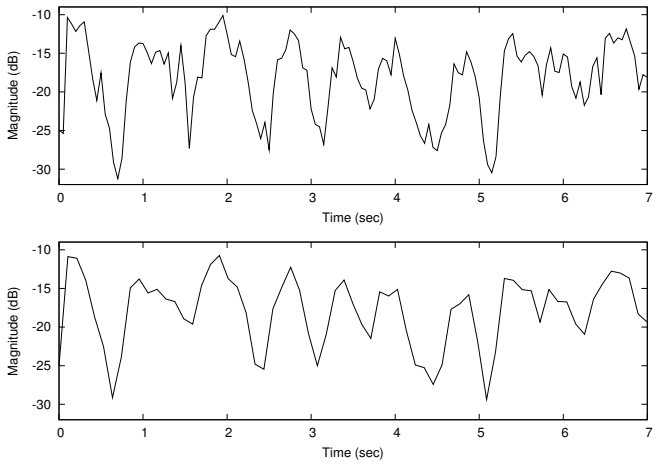

Figure 3. RMS peak analysis

B. Segmentation

Segmentation determines which pieces of the captcha are likely to contain digits by looking at a subsampled version of the signal. Given a window length l, the original signal is split into segments of length l and is subsampled by computing their RMS. A cut is extracted when a consecutive sequence of segments

- lasts a minimal duration (0.4 sec by default).
- contains at least one segment with RMS higher than a given level (-16 dB by default), referred to as the *noise level*.
- ends with a segment whose RMS is below the noise level and is a local minimum.

We determine appropriate values for the noise level and window length for each captcha. Indeed, optimal parameters differ even among captchas from the same scheme. These parameters are jointly optimized over a grid of values between -10 to -30 dB for noise level and 0.1 to 0.5 sec for window length. The largest window size and noise level that produce a correct number of segments for the given captcha scheme are selected. We prefer larger values because they give a clearer separation between noise and signal and avoid overly small cuts. Furthermore, joint optimization is necessary because the two parameters are interrelated; noise level is a measure of energy that depends on the window length.

We chose this segmentation algorithm because it is fast and does not modify the original signal. Traditional methods[2] for noise analysis filter the frequency representation of a captcha and therefor require computing the DFT and IDFT of the entire signal. This method is computationally expensive for large-scale experiments and leads to a potential loss of information when switching between frequency and temporal representations.

In order to understand the workings of our segmentation approach, it is necessary to categorize the types of noise

present in a captcha. Captchas usually feature a combination of distortions present as low energy *background* noise and a medium energy *intermediate* noise. The simplest form of noise, *constant noise*, is a constant sound such as white noise or perpetual buzzing. More complicated noise that changes over time but has similar energy level, duration, and regularity is known as *regular noise*. Unrecognizable speech that is repeated throughout the captcha falls into this category. Finally, *semantic noise* consists of a signal that has similar characteristics to a spoken digit but is not a useable digit. We consider music or vocal tracks to be semantic noise.

Our RMS subsampling acts as a *low-pass RMS filter* that eliminates constant and regular noises and only leaves peaks that correspond to digits. In particular, constant noise increases the RMS of each window uniformly and cannot destroy or add peaks. Similarly, short, regular noise retains digit peaks and impacts the RMS of each window approxmately uniformly for appropriate values of l. This last assumption is true in audio captchas because regular noise must have an average duration that is much shorter than a digit for the captcha to be understandable by humans. Figure 3 illustrates our RMS subsampling when the window length is too short (top) and when it is optimal (bottom). The bottom graph smooths out the signal but retains peaks that correspond to the 10 digits present in the captcha. In contrast, an overly short window allows noise to create a jagged energy plot.

It is important to note that our segmentation technique is not robust to semantic noise. This kind of noise creates artificial peaks in the subsampling procedure that differ from digits only on a semantic level. This weakness is tolerated in Decaptcha because semantic noise is not common in audio captchas. Moreover, semantic noise that approaches the energy of spoken digits is confusing to humans and results in frustrating captchas. This phenomenon is demonstrated in Recaptcha captchas which have a high error rate among humans.

C. Representation

Decaptcha can represent cuts in any of the schemes described in section II-B. A single representation scheme is chosen before training and must be used thereafter. Based on our experiments, the default representation is the cepstrum.

We chose each of our representation schemes because of their popularity and efficacy in analyzing speech. In particular, the cepstrum is an experimentally validated technique that is useful for speech analysis because it separates low-frequency periodic excitation from the vocal cords from the formant filtering of the vocal tract [19]. Similarly, STFs are effective in speech recognition systems and physiological studies suggest that they may be used in the brain [14]. Finally, the TDC has been applied successfully to recognizing Mandarin and slovak digits in the presence of noise, [20] , [10] respectively.

D. Classification

Decaptcha classifies digits using the RLSC algorithm in an OVA scheme. Parameter tuning is automated via the method described in [21] and is performed by computing the multi-category LOO of a range of regularization parameters. We handle the classification of unidimensional and two-dimensional representations differently. For efficiency reasons, the TDC is handled by looking at the first 5,000 dimensions when they are extracted column-wise. In constrast, only the first *75* dimensions of a unidimensional representation are used. A cut is represented by a *2850*-dimensional vector consisting of these *75* dimensions and all pair-wise products between them. This representation was chosen because it minimizes the multi-category LOO for all of our real world captcha schemes. These experiments suggest that the first 75 dimensions of the cepstrum contain all of the information necessary to identify a spoken digit and that neighboring cepstral coefficients are strongly correlated with each other.

Finally, STFs are handled differently because of misalignment. In general, cuts differ in length and the actual frame in which each digit begins. We handle these issues by classifying the contents of a window 30 frames in length. This window length is chosen for efficiency reasons because it fits an entire digit but is still small. A cut is classified by sliding the window along the time axis and selecting the digit which receives the highest vote from the OVA scheme at any time point. This approach effectively selects the maximum response obtained from correlating the cut with filters that look for each digit.

RLSC is trained with an additional noise class that forces each OVA classifier to learn a more robust decision rule. This noise class is not used during the predictive phase. Samples of digits are obtained from the first 30 frames of each cut while samples of noise are taken from the last 30. This rule is used because of its simplicity and experimental evidence that indicates that digits are more likely to occur at the beginning of the cut. It is clear that some noise samples will contain fragments of digits, and vice versa, but RLSC's robustness to noise allows it to learn when some training data is incorrect.

We chose the RLSC algorithm over its more popular counterpart, the SVM, because of its performance and efficiency. Decaptcha requires a classifier that, in addition to performing well, can be trained quickly and automatically for multi-category classification. SVM's are problematic in these respects because T separate classifiers must be computed when using an OVA scheme with T classes. Moreover, there is no efficient way to compute the LOO error of an SVM, so automatic parameter tuning is very costly. In contrast, RLSC uses dynamic programming to train an OVA scheme and tune necessary parameters in the time it takes to train a single binary classifier with a fixed regularization

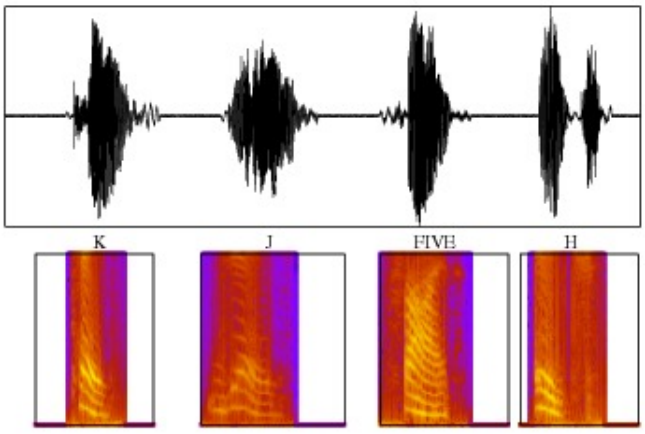

Figure 4. Authorize Captcha

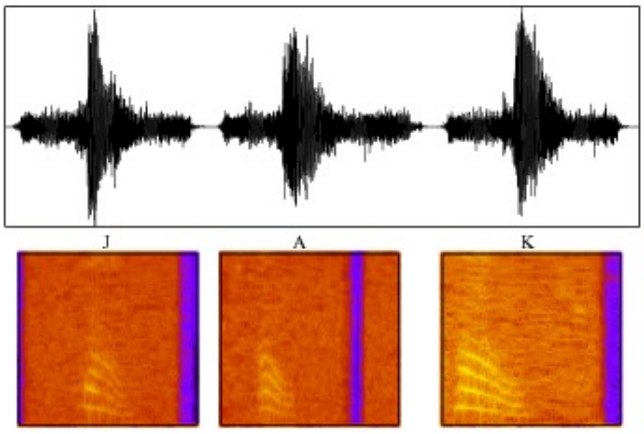

Figure 5. Digg Captcha

parameter. These properties can make RLSC orders of magnitude faster to train than an SVM [21]. This efficiency is noticeable in Decaptcha; it takes 2 minutes (5 minutes) to train on thousands of captchas with a unidimensional (two-dimensional) representation, respectively.

IV. Commercial Captchas

This section describes the commercial captchas we used to validate Decaptcha as well as our testing methodology and results. We tested audio captchas from Authorize, Digg, eBay, Microsoft, Recaptcha, and Yahoo. We were unable to test Google's captchas because of difficulties we encountered obtaining reliable annotations; they are so difficult for humans that they are ineffective as captchas.

A. Corpus description

Authorize. Audio captchas on *authorize.net* consist of five letters or digits spoken aloud by a female voice. The voice clearly articulates each character and there is minimal distortion. The waveform and spectrogram presented in Figure 4 show a portion of a captcha containing the digits/letters K, J, 5 and H. A long pause appears between spoken characters and vowels are clearly articulated. The letters K and H, which are fricative consonants, show some harmonic patterns in the spectrogram while the letter J has almost no harmonic patterns.

Digg. Audio captchas on *digg.com* consist of five letters spoken aloud by a female voice. There is random white noise in the background and sometimes an empty, but louder, segment is played between letters. The waveform and spectrogram presented in Figure 5 show a portion of a captcha containing the letters J, A and K. The overall brown color of the spectrogram shows the heavy constant noise that obscures vowels but still maintains some characteristic patterns of the letters. These patterns cannot be completely masked by the white noise since they are necessary for human recognition. Interestingly, the spectrogram of the letter J shows patterns similar to those of the letter J in the Authorize captcha (see figure 10).

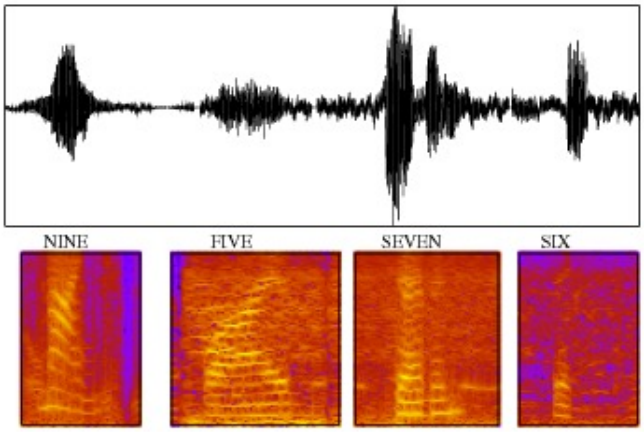

Figure 6. Ebay Captcha

eBay. Audio captchas on ebay.com consist of six digits spoken by a different speaker and in a different setting with regular background noise. The waveform and spectrogram presented in Figure 6 show part of a captcha containing the digits 9, 5, 7 and 6. The digits in these captchas are delivered much faster than those of *authorize.net* or *digg.com*. The waveform shows the variability of the various digits due to different speakers and different background noise levels, while the spectrogram shows that the vowels are short and relatively unobscured by noise.

Microsoft. Audio captchas from *live.com* consist of ten digits spoken by different speakers over a low quality recording. There is a regular background noise consisting of several simultaneous conversations. The waveform and spectrogram presented in Figure 7 show a portion of a captcha containing the digits 2, 9, 0 and 0. Like the eBay audio captchas, these digits are spoken very quickly. While all of the high amplitude sections of the waveform

Scheme	Authorize	Digg	eBay	Microsoft	Recaptcha	Yahoo
Length	5	5	6	10	8	7
Type of voice	Female	Female	Various	Various	Various	Child
Background Noise	None	Constant (random)	Constant (random)	Constant (random)	Constant (random)	None
Intermediate noise	None	None	Regular (speech)	Regular (speech)	Regular (speech)	Regular (speech)
Charset	0-9a-z	a-z	0-9	0-9	0-9	0-9
Avg. duration	5.0	6.8	4.4	7.1	25.3	18.0
Sample rate	8000	8000 8000	8000	8000	8000	22050
Beep	no	no	no	no	no	yes

Table I
COMMERCIAL AUDIO CAPTCHA FEATURE DESCRIPTION

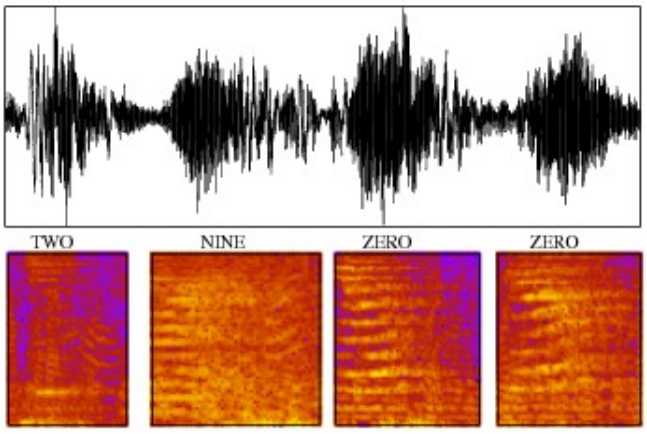

Figure 7. Microsoft Captcha

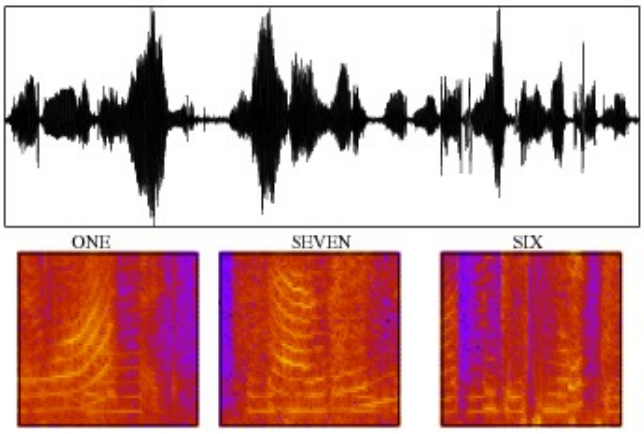

Figure 9. Yahoo Captcha

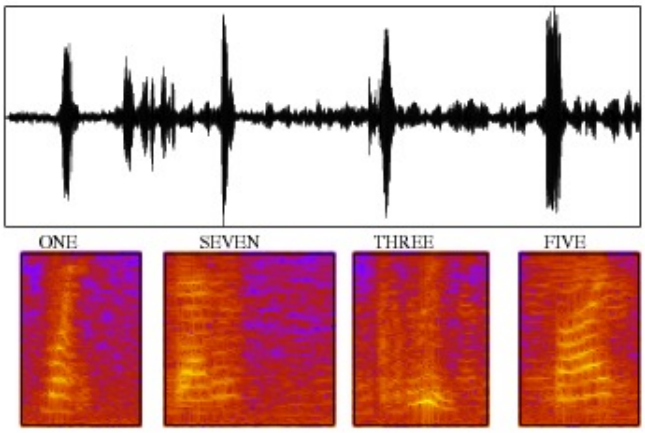

Figure 8. Recaptcha Captcha

correspond to actual digits, the spectrogram shows that the vowels are somewhat obscured by background noise. Interestingly, the two 0 digits show very similar patterns, but this pattern is not easily distinguished from the pattern observed for the 9 digit.

Recaptcha. Audio captchas from *recaptcha.net* consist of eight digits spoken by different speakers. Distortions include background conversations and approximately two semantic vocal noises that look like digits in the waveform. Apart from the presence of semantic noise, Recaptcha captchas are similar to live.com captchas, but the digits are delivered much more slowly. The waveform and spectrogram presented in Figure 8 show a portion of a captcha containing the digits 1, 7, 3 and 5. As will be discussed in 10 the five digit from this captcha shows similar harmonic patterns the five digit from Authorize and eBay captchas.

Yahoo. Audio captchas from *yahoo.com* consist of three beeps followed by seven digits spoken by a child. The captcha is obscured with other childrens' voices in the background. The waveform and spectrogram presented in Figure 9 show a portion of a captcha containing the digits 1, 7 and 6. The digits are the largest amplitude sections in the waveform and the spectrogram shows that the background voices do not confuse the patterns of the digits. This spectrogram shows different patterns than the spectrograms of the other captchas because of the use of a child's voice. It seems that the patterns induced by a child are much clearer than the patterns of an adult's voice. This makes digits easier to recognize even though the noise in Yahoo's captchas has more energy than the noise in other captchas.

Comparison. Figure 10 illustrates some differences between commercial captcha schemes. The first line presents the TFR of the digit five from Authorize, eBay, and Recaptcha

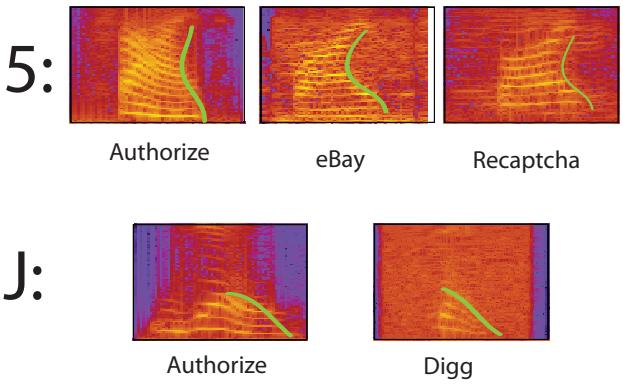

Figure 10. The TFR of the same digit/letter for different captcha schemes

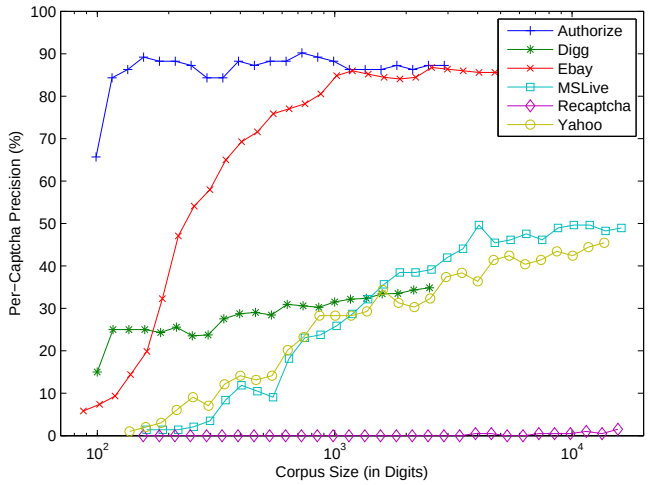

Figure 11. Per-captcha precision as a function of corpus size using the cepstrum

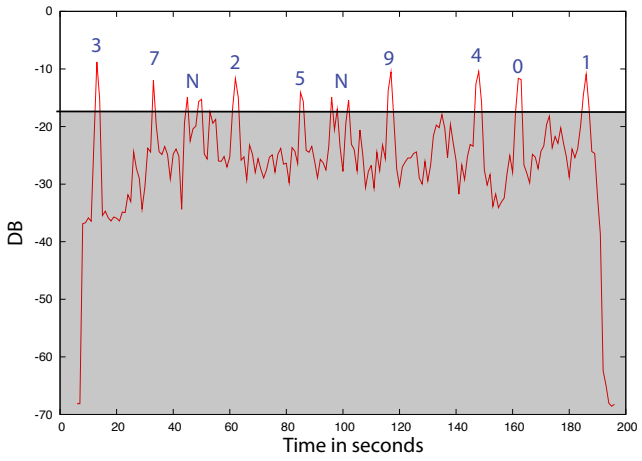

Figure 12. A standard Recaptcha captcha with the peaks annotated

captchas, respectively. All three digits show similar patterns, although the overall shape is different because of different speakers. The second line presents the TFR of the letter J from Authorize and Digg captchas, respectively. Similar patterns can also be observed: both images show few harmonic patterns because the letter J is a consonant with very few harmonic components.

B. Labeling

We used Amazon Mechanical Turk to label captchas scraped from the aforementioned commercial schemes. We considered a captcha labeled if three different Turkers independently agreed on the sequence of digits it contains. This method allowed us to reuse some labels obtained during a previous study [4]. Nonetheless, we had to label 10,000 new captchas from each of Microsoft, Recaptcha, and Yahoo because of label disagreement in our old corpus. Only approximately 10% of these new captchas satisfied our label requirement.

Obtaining reliable annotations for Microsoft, Recaptcha, and Yahoo captchas turned out to be more difficult than expected: three or more individuals agreed on the same *incorrect* labels for some captchas. These incorrect labels were tolerated when training the RLSC algorithm because of its robustness to noise. However, we were forced to manually and accurately annotate 200 captchas from these schemes for our testing sets because incorrect labels drastically affect the measured testing error. To see its impact on the per digit classification rate, suppose that a classifier has a precision of $0 \leq p \leq 1$ and w fraction of our testing set is mislabeled. The testing precision that we measure is

$$q = \frac{p(1-w) + (1-p)w}{10}$$

which is given by the likelihood of successfully labeling a sample whose label is correct and mislabeling one that was mislabeled with the same digit, assuming ten digits. If we solve for p, we get $p = \frac{10q-w}{10-11w}$ which leads to an estimate of 60% precision when our real precision is 89% in the case of Microsoft's captchas. This implies that almost one third of Microsoft's captchas were consistently labeled incorrectly.

C. Results

Decaptcha's performance on commercial captcha schemes is presented in Table II and Figure 11. Figure 11 depicts the impact of the corpus size, measured in number of labeled digits, on Decaptcha's per-captcha precision using the cepstrum. Table II is finer grained and it presents the coverage, per-digit precision, and per-captcha precision of each captcha scheme for all possible signal representations. Our best per-captcha accuracies are highlighted in bold and are always the unidimensional cepstrum. The per-captcha precision of Decaptcha is **89%** for Authorize, **41%** for Digg, **82%** for eBay, **48.9%** for Microsoft, **45.45%** for Yahoo and, **1.5%** for Recaptcha. We improve our previous work's [3] result on eBay from **75%** up to **82%**.

Recaptcha. Precision is particularly low on the Recaptcha scheme because it uses semantic vocal noise. This noise has

Scheme	Len	Coverage	FFT		Cepstrum		Cepstrum+Mel		TFR		TFR+Mel		TCR		TDC	
			Digit	Captcha	Digit	Captcha	Digit	Captcha	Digit	Captcha	Digit	Captcha	Digit	Captcha	Digit	Captcha
Authorize	5	100	93.73	80.39	96.08	87.25	97.06	**89.22**	92.55	77.45	91.76	71.57	83.14	34.31	97.25	88.24
Digg	5	100	71.08	32.07	76.77	40.84	76.61	**41.04**	62.15	35.66	74.66	36.65	70.96	27.69	72.19	31.08
eBay	6	85.60	81.58	44.36	92.48	**82.88**	92.61	80.93	81.84	47.08	81.91	44.36	45.40	0.78	90.60	75.88
Microsoft	10	80.60	76.57	14.69	89.58	**48.95**	89.30	47.55	88.95	46.85	86.99	41.26	84.48	28.67	87.20	42.66
Recaptcha	8	99.90	26.58	0.00	40.47	**1.52**	37.44	1.52	38.45	0.00	38.26	0.00	24.62	0.00	30.30	0.00
Yahoo	7	99.10	33.77	0.00	74.71	**45.45**	68.13	30.30	66.03	22.22	61.74	20.20	38.93	1.35	62.01	17.51

Table II
DECAPTCHA'S COVERAGE, PER-DIGIT PRECISION, AND PER-CAPTCHA PRECISION AS A FUNCTION OF THE REPRESENTATION SCHEME

the same RMS, harmonic content, and overall characteristics as a regular digit. These properties confuse our segmentation algorithm because the only distinction between vocal noise and proper digits is in their meaning; it is the classifier's job to distinguish between the two. As evidenced by the large proportion of mislabeled captchas, humans are also more error prone in the presence of Recaptcha's semantic vocal noise. These two factors make it difficult to acquire a good training set and lead to poor segmentation during testing.

Performance estimation. A conservative estimate of Decaptcha's precision on a specific scheme can be obtained by raising the per-digit precision to the number of digits contained in each captcha. This formula, however, consistently underestimates the per-captcha precision because it assumes that the probabilities of erring on the first, second, etc. digit of a captcha are independent. To illustrate the problem with this assumption, suppose that the segmenter skips the first digit of a captcha labeled "1234". During testing, the 2 will be labeled a 1, the 3 will be labeled a 2, and so on. This label misalignment ensures that each digit is counted as a mistake, irrespective of whether the classifier recognizes it correctly. A second way in which digit misclassification probabilities are not independent is that a mistake on the first digit of a captcha is likely to be indicative of a particularly noisy or otherwise difficult captcha. In this case, we are likely to make mistakes on the remaining digits. Conversely, misclassified digits are likely to belong to the same captcha, so per-captcha precision is higher than what it is estimated to be.

Training corpus size. Figure 11 shows the relationship between training corpus size and per-captcha success rate. For Authorize and eBay, Decaptcha achieves maximal precision early, at approximately 200 and 1000 digits, respectively. The other captcha schemes continue to benefit from additional training data, although there are diminishing returns because the x-axis is on a logarithmic scale. If we account for the number of digits contained in each captcha, a reasonable initial training corpus size requires between 100 and 300 labeled captchas. The expected precision of Decaptcha after training can then be used to decide whether to add additional data.

Note that both Digg and Authorize use letters and digits for a total of 36 possible characters whereas the other schemes only use 10 digits. The more varied a scheme is, the larger a corpus is needed to obtain the same number of examples per character. Obviously, therefore, a larger corpus is needed to provide a sufficient number of samples of each character. Similarly, we expect a lower rate of increase in precision, as a function of corpus size, in schemes with more characters. This is shown in Digg's precision curve: the rate at which precision increases is considerably lower than the rate at which the precision of the 10-digit schemes increase. Nonetheless, the high initial precision of both Authorize and Digg indicates that the distortions present in these schemes are easily handled by Decaptcha.

It is interesting to observe that corpus size serves as an effective measure of captcha difficulty that coincides with human observations. In particular Figure 11 allows us to rank the schemes in terms of difficulty as Recaptcha being the hardest, followed by a tie between Microsoft and Yahoo. Authorize is the easiest scheme, followed by eBay and then Digg. A similar ranking was observed from the percentage of captchas that were correctly labeled on Amazon Turk.

Impact of signal representation. The precision of each representation technique relative to the best performing one varies dramatically depending on the captcha scheme. For instance, the TDC is nearly optimal with Authorize, but it almost three times worse than the cepstrum on Yahoo captchas. Overall, our results show that cepstrum is the best representation scheme for commercial captchas. However, we will show in the next section that the TFR is better suited when the SNR is low.

Classifier confusion. Figure 13 shows Decaptcha's *confusion matrices* on the Microsoft, Recaptcha, and Yahoo schemes using the cepstrum. Note that we use an exponential scale because it leads to the best contrast. A confusion matrix depicts the conditional probability that a digit is classified as a y given that it is actually an x, for all pairs of digits x, y. In general, such a matrix will not be symmetric. Microsoft's matrix indicates that the digits 9, 6, and 3 are often mistaken for the digits 5, 2, and 2, respectively. These three pairs are among the most frequently confused digits for Recaptcha as well. The Recaptcha confusion matrix also reflects our overall higher error rates on the Recaptcha scheme. Finally, Yahoo's confusion matrix significantly differs from the other two and has a nearly uniform error distribution. This last property was observed because all off-diagonal elements become white when we subtract the mean from the original confusion matrix. It will be interesting to investigate how confusion patterns change as a function of the sampling rate, voice, and distortions used.

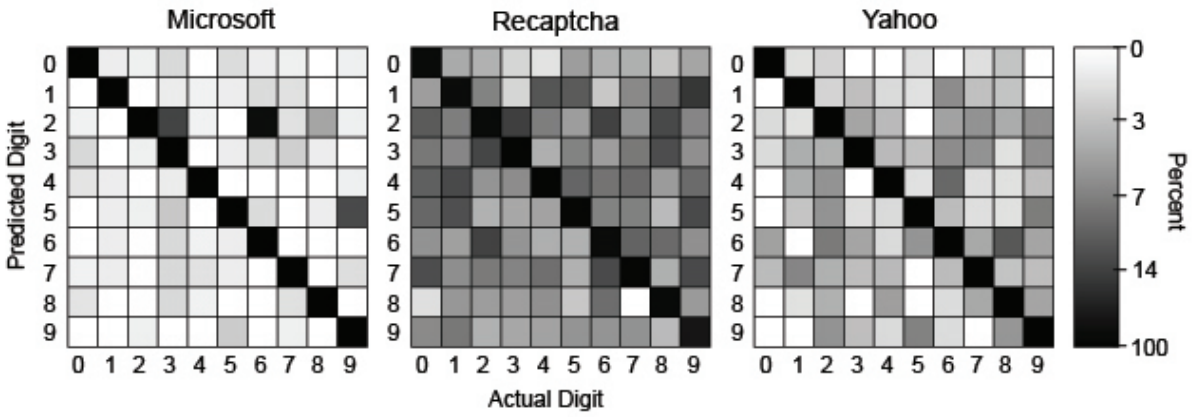

Figure 13. Decaptcha's Confusion Matrices on Microsoft, Recaptcha, and Yahoo Schemes Using Cepstrum

V. Synthetic Evaluation

This section reports Decaptcha's performance on a synthetic corpus that was generated following methodology introduced by [15]. The synthetic corpus uses nine types of noise described in Table III. We measured Decaptcha's performance on 2000 captchas for each noise type at SNRs ranging from -5 to 30 dB. Each captcha is composed of six spoken digits, spaced randomly between 0.5 and 4 seconds.

Familly	Name	Description
Constant Noise	*White*	White Gaussian noise.
	buzz	Sine waves at 700 Hz, 2100 Hz and 3500 Hz.
Regular noise	*pow*	10 ms bursts of white Gaussian noise repeated every 100 ms.
	rnoise	Every 100 ms, a section of the signal is replaced by white noise of the same RMS amplitude.
	lofi	Add distortion, cracks, bandwidth limiting and compression. Simulates old audio equipment.
	echo	The signal starts to echo at 0.6, 1.32, and 1.92 seconds.
	disintegrator	Amplifies random half-cycles of the signal by a multiplier. Simulates a bad audio channel.
Semantic noise	*chopin*	Chopin Polonaise for Piano No. 6, Op. 53.
	gregorian	Gregorian chant.
	nina	"Just in time" by Nina Simone.

Table III
Description of the Noises Used in Our Synthetic Corpus

Performance. Decaptcha's coverage of the synthetic corpus is depicted in Figure 14. Our overall coverage is between 80 and 90%, even for SNRs of -5dB. The only exceptions occur with *white* and *gregorian* noise, which achieve a coverage of 60% at low SNRs. Precision using TFR and

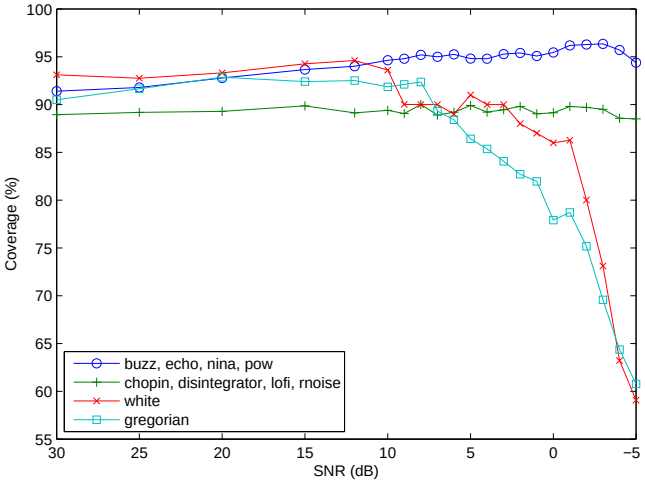

Figure 14. Coverage as a Function of Noise

cepstrum representations is presented in Figures 15 and 16, respectively. As expected, precision is directly related to SNR. For readability purposes, we have collapsed all of the figures by combining similar curves.

Constant noise. Precision drops most drastically for constant noises once the SNR is below 5. Indeed, as the SNR gets close to zero, constant noises mask any spoken digits and therefore make the captcha unintelligible. Decaptcha responds to such noise in a similar way as humans so constant noise should only be used as background noise with a low RMS.

Regular noise. The worst-measured precision of 64% is achieved on the *pow* noises. Otherwise, Decaptcha has a precision above 80% at all SNRs. We believe that the *pow* noise leads to poor precision because it confuses the segmentation algorithm when the SNR is low. Nonetheless, Decaptcha handles regular noise remarkably well, even at low SNRs, which may suggest that computers can outperform humans with this type of noise.

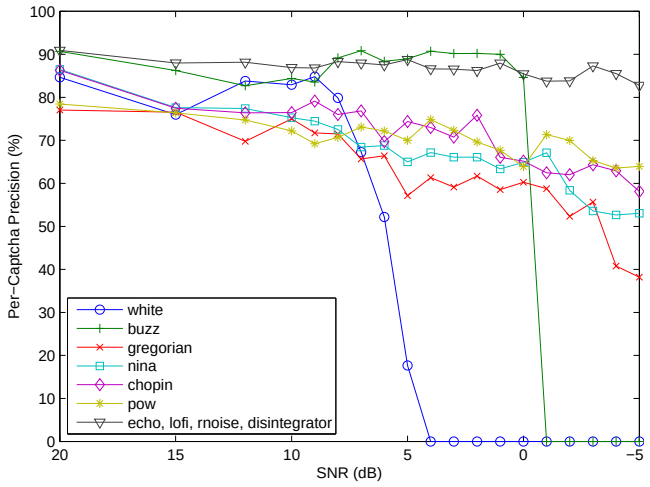

Figure 15. Precision of the TFR as a Function of Noise

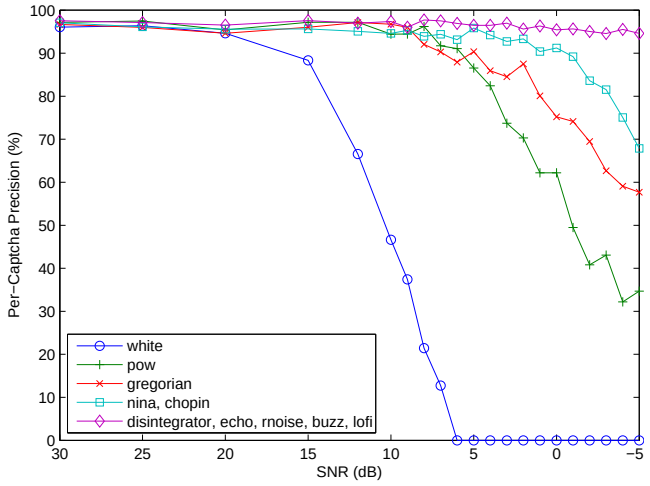

Figure 16. Precision of the Cepstrum as a Function of Noise

Semantic noise. As expected from Decaptcha's low precision on Recaptcha, the *nina*, *gregorian* and *chopin* noises produce the most robust captchas. Unlike constant noise, humans are well equipped to handle semantic noise, even at low SNRs, because we can select which voice to listen to. Furthermore, semantic noise consistently leads to lower precision than regular noise, especially at low SNRs. This noise is therefore the least harmful to human understanding at levels that hinder Decaptcha's performance.

The impact of sound representation. A final takeaway from this evaluation is that the TFR representation gives better results than the cepstrum when dealing with constant noise at low SNRs.

VI. Further Related Work

The first discussion of the captcha idea appears in [18], though the term CAPTCHA was coined later in [28]. Text/image based captchas have been studied extensively [11], [12], [5] and there is a long record of successful attempts at breaking popular sites' visual captchas [7]. For example in March 2008, a method to break 60% of MSN visual captchas was disclosed [29] and more recently an attack against the recaptcha captcha was demonstrated at the Defcon[9]. Using machine learning to break captchas applies to almost every kind of captcha and in 2008, Golle [8] successfully used machine learning attacks to break the Microsoft picture based scheme Assira.

VII. Conclusion

Decaptcha's performance on commercially available audio captchas indicates that they are vulnerable to machine learning based attacks. In almost all cases, we achieve accuracies that are significantly above the 1% threshold for a scheme to be considered broken. Compared with human studies done in [4], Decaptcha's accuracy rivals that of crowdsourcing attacks. Morever, our system does not require specialized knowledge or hardware; its simple two-phase design makes it fast and easy to train on a desktop computer. As such, automatic solvers are a credible threat and measures must be taken to strengthen existing audio captchas.

Our experiments with commercial and synthetic captchas indicate that the present methodology for building audio captchas may not be rectifiable. Besides Recaptcha, all of the commercial schemes we tested use combinations of constant and regular noise as distortions. Based on the difficulties we had with obtaining reliable annotations, human accuracy plummets when such distortions contribute significantly to the signal. On the other hand, Decaptcha's performance on our synthetic corpus indicates that automated solvers can handle such noise, even at low SNRs. All in all, computers may actually be more resilient than humans to constant and regular noise so any schemes that rely on these distortions will be inherently insecure.

Our results also pinpoint an inherent weakness of two-phase machine learning attacks that may be exploited, at least temporarily. As evidenced by Decaptcha's difficulties with Recaptcha, semantic noise hinders the segmentation stage by introducing noise that can be confused with a digit. Architectures that successfully overcome such distortions require a monolithic design that blends together classification and segmentation to endow the segmentation algorithm with semantic understanding. These designs are more difficult to realize than the simple two-phase approach and have received little attention. We therefore recommend that future designs for audio captchas investigate the use of semantic noise.

Future directions. We plan to extend our work in two directions. First, we would like to modify Decaptcha to handle audio captchas that contain spoken words. It is important to understand whether such "continuous" designs lead to more secure captchas. Secondly, we would like to investigate a series of design principles that may lead to more secure captchas. These include the use of semantic noise and leveraging differences between the ways that humans and computers make mistakes so as to maximize

an attacker's difficulty and cost.

Acknowledgment

We thank David Molnar and anonymous reviewers for their comments and suggestions. This work was partially supported by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research.

References

[1] Y. Ariki, S. Mizuta, M. Nagata, and T. Sakai. Spoken-word recognition using dynamic features analysed by two-dimensional cepstrum. In *Communications, Speech and Vision, IEE Proceedings I*, volume 136, pages 133–140. IET, 2005. 2

[2] B. Boashash. *Time frequency signal analysis and processing : a comprehensive reference / edited by Boualem Boashash*. Elsevier, Amsterdam ; Boston :, 2003. 4

[3] E. Bursztein and S. Bethard. Decaptcha: breaking 75% of eBay audio CAPTCHAs. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, page 8. USENIX Association, 2009. 1, 8

[4] E. Bursztein, S. Bethard, C. Fabry, J. Mitchell, and D. Jurafsky. How good are humans at solving CAPTCHAs? a large scale evaluation. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 399–413. IEEE, 2010. 8, 11

[5] K. Chellapilla and P. Simard. Using machine learning to break visual human interaction proofs. In M. Press, editor, *Neural Information Processing Systems (NIPS)*, 2004. 11

[6] D. Childers, D. Skinner, and R. Kemerait. The cepstrum: A guide to processing. *Proceedings of the IEEE*, 65(10):1428 – 1443, 1977. 2

[7] D. Danchev. Microsoft's captcha successfully broken. blog post http://blogs.zdnet.com/security/?p=1232, May 2008. 11

[8] P. Golle. Machine learning attacks against the asirra captcha. In *ACM CCS 2008*, 2008. 11

[9] C. Houck and J. Lee. Decoding recaptcha. http://www.defcon.org/html/links/dc-archives/dc-18-archive.html. 11

[10] R. Jarina, M. Kuba, and M. Paralic. Compact representation of speech using 2-d cepstrum - an application to slovak digits recognition. In V. Matousek, P. Mautner, and T. Pavelka, editors, *TSD*, volume 3658 of *Lecture Notes in Computer Science*, pages 342–347. Springer, 2005. 2, 5

[11] P. S. K Chellapilla, K Larson and M. Czerwinski. Building segmentation based human- friendly human interaction proofs. In Springer-Verlag, editor, *2nd Int'l Workshop on Human Interaction Proofs*, 2005. 1, 11

[12] P. S. K Chellapilla, K Larson and M. Czerwinski. Designing human friendly human interaction proofs. In ACM, editor, *CHI05*, 2005. 11

[13] S. Kay and J. Marple, S.L. Spectrum analysis: A modern perspective. *Proceedings of the IEEE*, 69(11):1380 – 1419, 1981. 2

[14] M. Kleinschmidt. Localized spectro-temporal features for automatic speech recognition. In *Proc. Eurospeech*, pages 2573–2576, 2003. 5

[15] G. Kochanski, D. Lopresti, and C. Shih. A reverse turing test using speech. In *Seventh International Conference on Spoken Language Processing*, pages 16–20. Citeseer, 2002. 1, 2, 10

[16] R. McMillan. Wiseguy scalpers bought tickets with captcha-busting botnet. *Computerworld*, Nov. 2010. http://www.computerworld.com/s/article/9197278/Wiseguy_scalpers_bought_tickets_with_CAPTCHA_busting_botnet. 1

[17] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 134–141, 2003. 1

[18] M. Naor. Verification of a human in the loop or identification via the turing test. Available electronically: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps, 1997. 1, 11

[19] A. M. Noll. Cepstrum pitch determination. *Acoustical Society of America Journal*, 41:293–+, 1967. 5

[20] H. Pai and H. Wang. A study of the two-dimensional cepstrum approach for speech recognition. *Computer Speech & Language*, 6(4):361–375, 1992. 5

[21] H. Paskov and L. Rosasco. Notes on Regularized Least Squares: Multiclass Classification. Technical report, MIT, 2011. 2, 5, 6

[22] R. M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches*. PhD thesis, MIT, 2002. 3

[23] R. Santamarta. Breaking gmail's audio captcha. http://blog.wintercore.com/?p=11. 1

[24] Shadowserver. Conficker. http://www.shadowserver.org/wiki/pmwiki.php/Stats/Conficker, 2010. 1

[25] P. Y. Simard. Using machine learning to break visual human interaction proofs (hips. In *Advances in Neural Information Processing Systems 17, Neural Information Processing Systems (NIPS'2004*, pages 265–272. MIT Press, 2004. 1

[26] Y. Soupionis and D. Gritzalis. Audio CAPTCHA: Existing solutions assessment and a new implementation for VoIP telephony. *Computers & Security*, 29(5):603–618, 2010. 1

[27] J. Tam, J. Simsa, S. Hyde, and L. Von Ahn. Breaking audio captchas. *Advances in Neural Information Processing Systems*, 1(4), 2008. 1

[28] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In Sringer, editor, *Eurocrypt*, 2003. 11

[29] J. Yan and A. S. E. Ahmad. A low-cost attack on a microsoft captcha. Ex confidential draft http://homepages.cs.ncl.ac.uk/jeff.yan/msn_draft.pdf, 2008. 11

[30] J. Yan and A. S. El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 543–554, New York, NY, USA, 2008. ACM. 1

[31] J. Yan, A. Salah, and E. Ahmad. Breaking visual captchas with naïve pattern recognition algorithms. In *Twenty-Third Annual In Computer Security Applications Conference*, 2007. 1