Journal of Information Processing Vol.23 No.6 814–826 (Nov. 2015)

[DOI: 10.2197/ipsjjip.23.814]

**Regular Paper**

# HMM-based Attacks on Google's ReCAPTCHA with Continuous Visual and Audio Symbols

Shotaro Sano[1,a)] Takuma Otsuka[1,b)] Katsutoshi Itoyama[1,c)] Hiroshi G. Okuno[1,d)]

Received: January 27, 2014, Accepted: August 12, 2015

***Abstract:*** CAPTCHAs distinguish humans from automated programs by presenting questions that are easy for humans but difficult for computers, e.g., recognition of visual characters or audio utterances. The state of the art research suggests that the security of visual and audio CAPTCHAs mainly lies in anti-segmentation techniques, because individual symbol recognition after segmentation can be solved with a high success rate with certain machine learning algorithms. Thus, most recent commercial CAPTCHAs present continuous symbols to prevent automated segmentation. We propose a novel framework that can automatically decode continuous CAPTCHAs and assess its effectiveness with actual CAPTCHA questions from Google's reCAPTCHA. Our framework is constructed on the basis of a sequence recognition method based on hidden Markov models (HMMs), which can be concisely implemented by using an off-the-shelf library HMM toolkit. This method concatenates several HMMs, each of which recognizes a symbol, to build a larger HMM that recognizes a question. Our experimental results reveal vulnerabilities in continuous CAPTCHAs because the solver cracks the visual and audio reCAPTCHA systems with 31.75% and 58.75% accuracy, respectively. We further propose guidelines to prevent possible attacking from HMM-based CAPTCHA solvers on the basis of synthetic experiments with simulated continuous CAPTCHAs.

***Keywords:*** CAPTCHA, human interaction proof, hidden Markov model, continuous character/speech recognition

## 1. Introduction

CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) are programs that distinguish humans from automated programs by presenting tasks that humans can easily solve but computers cannot [1]. Many websites use CAPTCHAs to prevent their services from various abuses such as flooding from spam accounts. While they have been widely used in recent web services, even CAPTCHAs in popular web services (such as Google, Microsoft, Yahoo!) are sometimes easily solved by simple machine learning algorithms [2], [3]. This fact immediately threatens the quality of many web services with unauthorized accesses by malicious programs. Thus, there has been a huge demand to organize guidelines for the design of secure CAPTCHAs by examining breaking techniques.

This paper discusses the attacking techniques on both visual and audio CAPTCHAs. While most CAPTCHAs display images of characters, famous CAPTCHA services also provide audio versions for accessibility reasons. Because a user that solves either a visual or audio question is authorized by the CAPTCHA, sometimes audio CAPTCHA systems provide another loophole for malicious programs. The security of both types of CAPTCHAs should be equally examined.

In both visual and audio CAPTCHAs, recent systems protect against automated programs by presenting continuous symbols. For example, visual CAPTCHAs of Google, Yahoo!, and Amazon present continuous characters as shown in **Fig. 1**. This is because continuous symbols make the segmentation task difficult for machines but easy to manage for humans. In accordance with Bursztein *et al.*, using continuous symbols so far is the best option to avoid automatic segmentation [4].

**Fig. 1** Visual CAPTCHAs of Google (upper left), Yahoo! (upper right), Microsoft (lower left), and Amazon (lower right). Characters are connected except for Microsoft CAPTCHA.

Here, we discuss a framework that automatically solves continuous CAPTCHAs and is applicable to both visual and audio ones. The framework formulates the decoding process with a well-known sequence recognition method based on hidden Markov models (HMMs) [5] that has been successfully used in automatic speech recognition (ASR) and cursive handwriting recognition systems [6], [7].

We tested the efficiency of our framework with actual CAPTCHA data collected from Google's reCAPTCHA [*1]. The solvers cracked the current version of the visual reCAPTCHA with 31.75% accuracy and that of the audio reCAPTCHA with 58.75% accuracy (as of July 2013), which means continuous

[1] Graduate School of Informatics, Kyoto University, Kyoto 606–8501, Japan
[a)] sano@kuis.kyoto-u.ac.jp
[b)] ohtsuka@kuis.kyoto-u.ac.jp
[c)] itoyama@kuis.kyoto-u.ac.jp
[d)] okuno@kuis.kyoto-u.ac.jp

[*1] reCAPTCHA [8] is an application programming interface (API) provided by Google to embed the CAPTCHA system, which has been used by various web services including Google, Twitter, and Facebook.

CAPTCHAs are no longer safe.

The HMM-based sequence recognition method discussed here has been black-boxed and is easily available in an off-the-shelf library called the HMM toolkit (HTK) [9]. Therefore, our method may further threaten the security of continuous CAPTCHAs.

The rest of this paper is organized as follows. Section 2 summarizes previous works on CAPTCHA breaking. Section 3 outlines the HMM-based framework to crack continuous CAPTCHAs that is applicable for both visual and audio CAPTCHAs. Sections 4 and 5 describe the implementation of the visual and audio reCAPTCHA solvers based on our framework. Section 6 presents experiments to evaluate the performance of the reCAPTCHA solvers. Sections 7 and 8 demonstrate synthetic experiments using simulated continuous CAPTCHAs to find out the weakness of our framework for various kinds of popular defensive techniques. Section 9 discusses guidelines to design better CAPTCHAs that are based on our experimental results. Finally, Section 10 concludes the paper.

## 2. Related Works on Breaking CAPTCHAs

This section describes how continuous CAPTCHAs have become the mainstream CAPTCHA system. We then explain the contributions of our paper.

Hindle et al. [10] summarized the attacking process of automated programs as three steps: preprocessing, segmentation, and classification (**Fig. 2**). The preprocessing stage reduces redundant information on the question such as background clutter and noise. The segmentation stage divides the preprocessed information into regions of single symbols. Finally, the classification stage labels each symbol with a certain supervised method.

Based on this attacking process, numerous security assessments have been undertaken on both visual and audio CAPTCHAs. Recent comprehensive assessments have been conducted by Bursztein et al. [4], [11] for visual CAPTCHAs from 15 web services and audio CAPTCHAs from six web services, including Google, Yahoo, eBay, etc., with a breaking tool called Decaptcha.

Many previous works concluded that the strength of a CAPTCHA depends on the difficulty of its segmentation, since given a perfect segmentation, a machine often attains superior accuracy of classification to humans [12], [13]. Thus, recent CAPTCHAs attach much importance to anti-segmentation techniques where a question is presented with continuous symbols (Fig. 1), which makes use of Sayre's paradox: a word cannot be segmented before being recognized and cannot be recognized before being segmented [14]. The above-mentioned works [4], [11] also concluded that continuous CAPTCHAs are unsolvable with their methods or were not even applied to continuous CAPTCHAs.

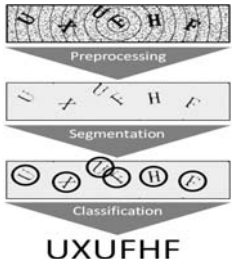



**Fig. 2** Attacking process of automated programs. CAPTCHAs are in most cases broken into three stages: preprocessing, segmentation, and classification.

Specifically in the field of visual CAPTCHAs, heuristic segmentation methods for continuous CAPTCHAs have been proposed by Yan et al. [15] and Cruz et al. [16]. Although these methods perform well on specific CAPTCHAs, they fail in their robustness for configurations of CAPTCHAs such as font types.

The contribution of this paper is to construct a general framework to break continuous CAPTCHAs. Our work is advantageous in the following ways.

- Based not on heuristic segmentation but on HMMs, our framework is robust against various configurations of CAPTCHAs.
- Our framework can be made applicable to both visual and audio continuous CAPTCHAs by modifying an HMM-based sequence recognition method where the solver simultaneously conducts segmentation and classification.
- We demonstrate the effectiveness of our framework with attacking experiments using actual data of the reCAPTCHA, one of the most secure CAPTCHA systems.
- Based on synthetic experiments in which our solvers break simulated continuous CAPTCHAs, we propose several workarounds for possible attacks from HMM-based CAPTCHA solvers.

## 3. HMM-based CAPTCHA Solver Framework

### 3.1 Overview

**Figure 3** depicts the overview of a CAPTCHA solver based on HMMs. Given the target CAPTCHA, the solver finds the optimal answer sequence of labels $\hat{W}$, out of all possible answers $L$. (Basically, all possible answers of a CAPTCHA can be described with a generation language as shown in **Fig. 4**.) The question's signal is represented as a sequence of feature vectors $O = o_1, \ldots, o_T$,

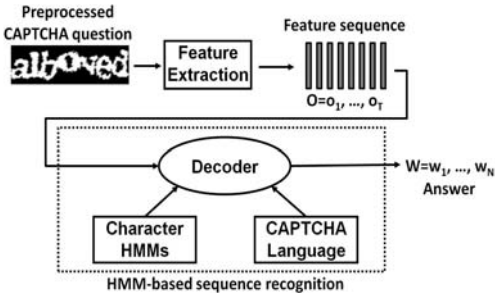



**Fig. 3** Solver overview. Input question is converted into a sequence of feature vectors and labeled by decoder. Decoder consists of a language to describe possible answers and set of HMMs, each of which corresponds to each label.

<captcha> ::= <captcha$_3$>|<captcha$_4$>;
<captcha$_3$> ::= <digit><digit><digit>;
<captcha$_4$> ::= <captcha$_3$><digit>;
<digit> ::= '0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9';

**Fig. 4** Example of Backus-Naur form to generate language $L$. Start symbol is < *captcha* > and terminal symbols are digit labels. This grammar meets schema of CAPTCHA that consists of three or four digits.

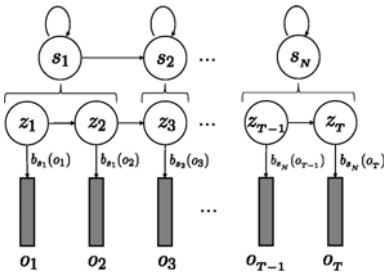


**Fig. 5** N-state linear HMM. Observed sequence $O$ is generated by sequence of hidden states $Z$. $Z$ has the form of Markov chain in states $S$. Each HMM corresponds to each label in this framework.

e.g., the sequence of pixel values for each column.

This problem is formulated as Eqs. (1) to (3), which means the problem can be broken down into the computations of $P(W)$ and $P(O|W)$:

$$\hat{W} = \arg\max_{W \in L} P(W|O) \tag{1}$$

$$= \arg\max_{W \in L} \frac{P(W)P(O|W)}{P(O)} \tag{2}$$

$$= \arg\max_{W \in L} P(W)P(O|W) \tag{3}$$

where Eq. (2) is obtained with Bayes' theorem. The denominator, $P(O)$, may be left out because it is always the same for given feature vectors $O$. Thus, we can obtain Eq. (3).

We assume that $P(W)$ is equal for all possible answers:

$$P(W) = \frac{1}{|L|}, \tag{4}$$

where $|L|$ is the number of sentence patterns generated from the generation language of $L$. This is because an answer is randomly given in most ordinary CAPTCHAs.

From Eqs. (3) and (4),

$$\hat{W} = \arg\max_{W \in L} P(O|W). \tag{5}$$

Thus, the rest of the problem is the likelihood $P(O|W)$, which is computed by the decoder. The rest of this section gives an overview of how the decoder labels a sequence of feature vectors $O$ after it describes the mechanism of an HMM and a concatenated HMM.

### 3.2 HMM

As depicted in **Fig. 5**, an HMM is a probabilistic model for a sequential observation. Given an observed sequence, $O = o_1, \ldots, o_T$, an HMM, $\lambda = \{\pi, A, B\}$ outputs the likelihood, $P(O|\lambda)$. $O$ is assumed to be generated by a sequence of hidden states $Z = z_1, \ldots, z_T$, which has the form of a Markov chain in states $S = \{s_1, \ldots, s_N\}$, i.e., $z_t \in S$. An observation value, $o_t$, is generated by a state, $s_n$, with probability $b_{s_n}(o_t)$. Thus, an HMM $\lambda$ is defined with three parameters:

- An initial probability vector of hidden states: $\pi = [\pi_n | 1 \leq n \leq N]$.
- A transition matrix of hidden states: $A = \{a_{i,j} | 1 \leq i, j \leq N\}$ where each element $a_{i,j}$ corresponds to $P(s_j|s_i)$, which means the transition probability from state $s_i$ to $s_j$.
- Observation likelihood functions: $B = \{b_s(o) | s \in S\}$ where $o$ may be a continuous value by defining $b_s(o)$ as a continuous density function.

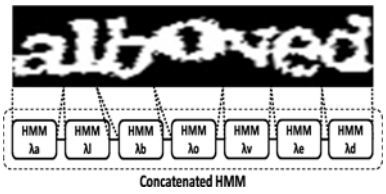



**Fig. 6** Concatenated HMM. By connecting several HMMs to recognize characters, a larger HMM to recognize a word is obtained.

The probability that hidden state $Z$ generates observation $O$ can be calculated with the parameter of HMM $\lambda$:

$$P(O, Z|\lambda) = P(O|Z, \lambda)P(Z|\lambda) \tag{6}$$

$$= \left(\prod_{t=1}^{T} b_{z_t}(o_t)\right)\left(\pi_{z_1} \prod_{t=1}^{T-1} a_{z_t, z_{t+1}}\right). \tag{7}$$

Thus, $P(O|\lambda)$ is obtained as:

$$P(O|\lambda) = \sum_{Z} P(O, Z|\lambda) \tag{8}$$

$$= \sum_{Z}\left(\prod_{t=1}^{T} b_{z_t}(o_t)\right)\left(\pi_{z_1} \prod_{t=1}^{T-1} a_{z_t, z_{t+1}}\right). \tag{9}$$

We can efficiently calculate the summation over $Z$ in Eq. (9) by using the forward algorithm [5].

The decoder has a set of HMMs $\{\lambda_{l_1}, \ldots, \lambda_{l_M}\}$ where $M$ is the number of labels, and HMM $\lambda_l$ corresponds to a label $l$. An HMM $\lambda_l$ is supposed to return higher likelihood given a part of the question's feature sequence representing $l$.

### 3.3 Concatenated HMM and Decoder

As depicted in **Fig. 6**, several HMMs are concatenated to build a larger HMM that recognizes a sequence of labels. For example, the concatenation of HMMs $\lambda_a$, $\lambda_l$, $\lambda_b$, $\lambda_o$, $\lambda_v$, $\lambda_e$, and $\lambda_d$ is expected to return a higher likelihood for a feature sequence representing "alboved." In the following, the concatenated HMM corresponding to the answer $W$ is represented as $\Lambda_W$.

To enable the concatenation, the non-emission states, the initial state $s_{\text{initial}}$ and the final state $s_{\text{final}}$, are appended to the head and tail of each HMM. No state transits to $s_{\text{initial}}$ and $s_{\text{final}}$ has no transition to a next state:

$$P(s_{\text{initial}}|s_n) = 0 \quad (1 \leq n \leq N), \tag{10}$$

$$P(s_n|s_{\text{final}}) = 0 \quad (1 \leq n \leq N), \tag{11}$$

A concatenated HMM is created, the final state of an HMM becomes the initial state of the next one, and the process continues.

For an arbitrary possible answer $W$, the concatenated HMM $\Lambda_W$ to calculate $P(O|W)$ can be made up in this way. Therefore, the optimal answer $\hat{W}$ is obtained by searching it from the set of all possible answers $L$ to maximize $P(O|W)$. In general for HMM-based sequence recognition methods, this searching task is enabled by constructing a network of concatenated HMMs [17], [18] where a certain sentence $W$ corresponds to a path in the network.

The parameters of the HMMs are obtained with the concatenated training [19] based on the Baum-Welch algorithm [20]. In

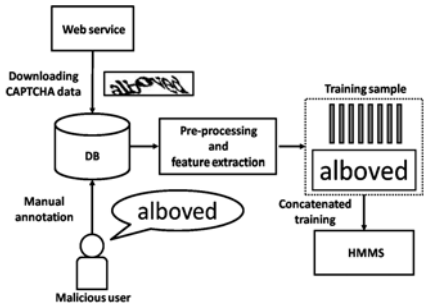



**Fig. 7** Training framework of CAPTCHA solvers. User manually annotates dataset downloaded from target CAPTCHA. HMMs are trained with annotated data using concatenated training.

this setup, the training data are provided as pairs of a sequence of feature vectors and the corresponding label transcription. Note that the pairwise data do not necessarily include manual alignment.

### 3.4 Training Framework

The HMMs are trained with actual data of the target CAPTCHA. As outlined in **Fig. 7**, the training set of questions is downloaded and stored in the database (DB). The data are manually annotated for each question by the solver's user. Then, the HMMs are trained with the feature sequences extracted from the data in the DB and corresponding transcriptions using the concatenated training [19].

## 4. Visual reCAPTCHA Solver

This section shows how our framework is applied to visual CAPTCHAs.

### 4.1 Visual reCAPTCHA Schema

An example of the visual reCAPTCHA questions is shown in upper-left of Fig. 1 [*2]. A question consists of six to eight alphabetic characters including both upper and lower cases. When all characters of the question are correctly estimated, the CAPTCHA is solved (or equivalently broken using a certain algorithm), i.e., the visual reCAPTCHA confirms that the user is a human.

To prevent segmentation from automated programs, the visual reCAPTCHA removes the space between characters of a question. The experimental results in Section 6 demonstrate that our framework efficiently cracks this anti-segmentation technique.

In addition, the visual reCAPTCHA distorts the image of the question. The distortion is applied in two steps: the linear transformation, which transforms the image with an affine filter, and the wavy transformation, which waves the image. Effective preprocessing techniques for these distortions are discussed later in this section.

To increase usability, the visual reCAPTCHA regards a response as correct even when one of the characters in a question is misestimated in terms of Levenshtein distance. For example, a question whose correct answer is "abcdefg" may be labeled as "bbcdefg," "bcdefg," or "aabcdefg."

[*2] Although an actual question of the visual reCAPTCHA presents two images of words, a control word and an unknown word [8], we ignore the unknown word. This is because the user have only to answer the control word to pass a reCAPTCHA question.

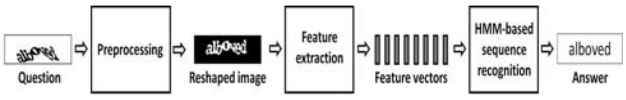



**Fig. 8** Decoding process of visual reCAPTCHA solver. Question is decoded in three steps: preprocessing, feature extraction, and HMM-based recognition.

**Fig. 9** Preprocessing of visual reCAPTCHA solver. Input image (upper left) is binarized (upper right), nonlinearly reshaped (lower left), and linearly reshaped (lower right).

**Fig. 10** Center line detection. Image is nonlinearly reshaped to straighten detected line.

In summary, the visual reCAPTCHA adopts the following defensive techniques: using continuous characters, randomized text length, linear transformation, and wavy transformation. The visual reCAPTCHA also adopts an additional idea to ensure usability by allowing an off-by-one error to when labeling a question.

### 4.2 Solver Overview

**Figure 8** depicts the pipeline of the visual reCAPTCHA solver. The input to the solver is an image of the question, and the solver outputs the answer label sequence of the word. The solver decodes an input in three steps: (1) the input image is nonlinearly reshaped, and then linearly reshaped (preprocessing); (2) the preprocessed image is converted into a sequence of feature vectors (feature extraction); and (3) the feature sequence is labeled with the HMM-based sequence recognition method (HMM-based sequence recognition).

### 4.3 Preprocessing

**Figure 9** shows the images through preprocessing. First, the input image is binarized by thresholding. The solver uses a fixed threshold that is defined as the intermediate value between maximum and minimum pixel intensities. Then the linear transformation and the wavy transformation should be reshaped to reduce the horizontal overlap between each character before the question is recognized by the HMMs. This is because the HMM-based recognition method implicitly performs vertical segmentation of the image.

#### 4.3.1 Nonlinear Reshaping

Suppose that each pixel value of the binarized image in the question is represented as $I(x, y)$ where $x$ and $y$ is the horizontal and vertical positions of the pixel respectively:

$$I(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ belongs to the text area,} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

First, as shown in **Fig. 10**, the center line of the distorted word $c(x)$ is detected:

$$c(x) = \frac{\sum_{w \geq |x-x'|} \{y \cdot I(x', y)\}}{\sum_{w \geq |x-x'|} I(x', y)}, \tag{13}$$

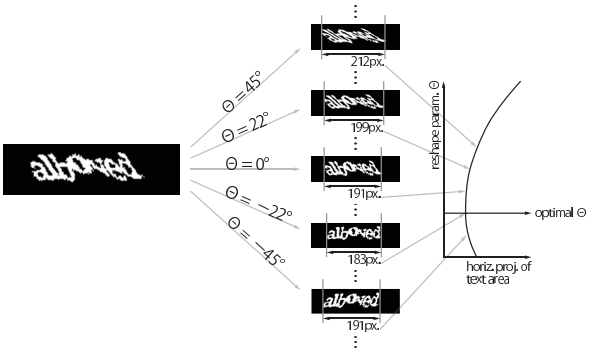



**Fig. 11** Finding the optimum linear reshaping parameter $\theta$.

where $w$ is the window size. In this study, we set $w = 20$. The reshaped image $J(x, y)$ is obtained by vertically aligning each column of $I(x, y)$ to straighten the detected center line:

$$J(x, y) = I\left(x, y - c(x) + \frac{h}{2}\right), \tag{14}$$

where $h$ is the height of the image.

#### 4.3.2 Linear Reshaping

As shown in **Fig. 11**, the linear reshaping can be represented as a transformation matrix $A$ that has one parameter $\theta$ as the following equation:

$$A = \begin{pmatrix} 1.0 & \tan\theta \\ 0.0 & 1.0 \end{pmatrix}. \tag{15}$$

The solver finds the optimum reshaping parameter $\theta$ that minimizes the horizontal projection of the text area.

### 4.4 Feature Extraction

A sliding window along the horizontal axis is used to extract feature vectors. The window size is set to five columns in this study, and the height of each reCAPTCHA image is 57 pixels. Thus, a 285-dimensional vector is extracted for each window. The feature vectors are decomposed into 25 dimensions with a sparse principal component analysis (sparse PCA) [21].

### 4.5 HMM Topology

A character is modeled as a left-to-right HMM. Each HMM has 20 states including the initial and final states. The observation likelihood function is a 25-dimensional Gaussian distribution for each state.

## 5. Audio reCAPTCHA Solver

This section shows how our framework is applied to audio CAPTCHAs.

### 5.1 Audio reCAPTCHA Schema

#### 5.1.1 Current Version of Audio reCAPTCHA

**Figure 12** shows the waveform of an audio clip from the current version of audio reCAPTCHA as of April 2013. A question consists of three utterances, which we refer to as clusters, with distinct intervals, and a cluster contains three or four digit utterances spoken in English. Digit utterances in a cluster overlap at random intervals. When all digits in a question are correctly estimated, the CAPTCHA is solved.

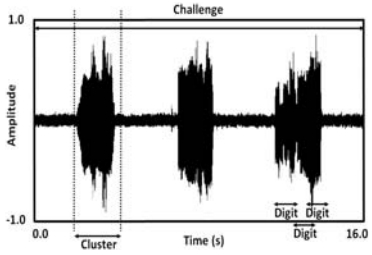



**Fig. 12** Waveform of audio reCAPTCHA question. Question consists of three clusters, and each cluster contains three or four overlapping digits. If all digits in question are correctly identified, audio reCAPTCHA is solved.

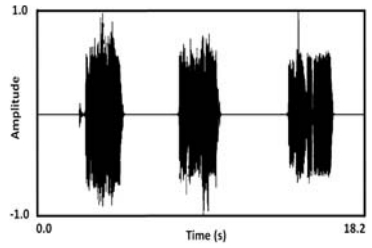



**Fig. 13** Waveform of previous version of audio reCAPTCHA question. Intervals between clusters are completely silent.

The audio reCAPTCHA also protects against automated programs with two types of noise: additive stationary noise and non-additive convolutive noise. The former covers the entire audio signal of the question to prevent both clusters from being detected and recognized. On the other hand, the latter is applied for each digit. **Figure 14** shows an example spectrogram of a digit voice distorted with the convolutive noise of the reCAPTCHA that is pronounced "zero" (left), comparing it to that of a clear digit voice (right). Some of the distorted digit's features collapse, especially in the high frequency range. Although the convolutive noise seems to prevent clusters or digits from being recognized, it degrades usability since it is often too strong even for humans to hear, in the authors' opinion.

Similarly to the visual reCAPTCHA, the audio reCAPTCHA regards a response as correct even when one of the digits in a question is deleted or replaced to increase usability. For example, a question whose correct answer is "012 345 6789" may be labeled as "012 345 678" or labeled as "112 345 6789." On the other hand, the audio reCAPTCHA does not allow insertion errors, and the question should not be mislabeled as "0012 345 6789."

In summary, the audio reCAPTCHA adopts four defensive techniques: overlap of target voices, random number of target voices in a cluster, a stationary noise signal that entirely covers a question, and filtering that collapses high frequency features of digits. The audio reCAPTCHA also adopts an additional idea to ensure usability by allowing an off-by-one error when labeling a question while disallowing insertion errors.

#### 5.1.2 Previous Version of Audio reCAPTCHA

**Figure 13** shows an audio clip from the previous version of reCAPTCHA, which had been used until February 2013. As this version did not adopt the additive stationary noise, the intervals between clusters were completely silent. In Section 8, we evaluate the solver's accuracy both for the previous and current ver-

sions to assess the efficiency of the additive stationary noise.

### 5.2 Solver Overview

**Figure 15** depicts the audio reCAPTCHA solver. The input to the solver is a question's audio signal of the reCAPTCHA, and the solver outputs the question's answer. The system solves a question in three steps: (1) the input question is segmented into three clusters with a voice activity detection algorithm (preprocessing); (2) each cluster's audio signal is converted into feature vectors (feature extraction); and (3) the feature vectors of each cluster are labeled with the HMM-based sequence recognition method (HMM-based sequence recognition). Note that, because digits are connected for each cluster, the recognition method is applied not for questions but for clusters.

### 5.3 Preprocessing

This component segments a question audio signal into three clusters. Clusters are extracted with a power-based algorithm for voice activity detection. The question audio signal, $f_1, \ldots, f_N$, is split into segments of length $l$ and is subsampled as Power($t$):

$$\text{Power}(t) = \frac{1}{l}\sum_{n=t}^{t+l-1}(\bar{f} - f_n)^2, \tag{16}$$

where $\bar{f}$ is the mean of $f_t, \ldots, f_{t+l-1}$.

**Figure 16** plots the power analysis of a question. There are three cluster segments between four noise segments in which every power value is less than a threshold, $\theta$. First, this component removes the four longest segments in which every window has a lower power than the threshold, $\theta$, and it then returns the remaining three segments as clusters. We set $l = 512$ and $\theta = 0.01$ where

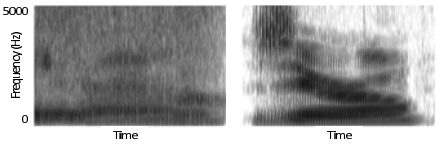



**Fig. 14** Comparison of spectrograms for distorted digital voice of reCAPTCHA (left) and clear digital voice (right). Both are pronounced "zero."

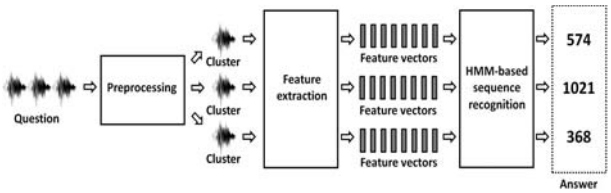



**Fig. 15** Audio reCAPTCHA solver. Preprocessing module splits question into three clusters. Feature extraction module converts each cluster into a feature sequence, and HMM-based sequence recognition method labels the feature vectors of each cluster.

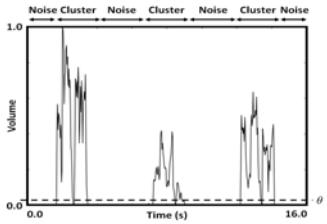



**Fig. 16** Power analysis of question. Power does not reach threshold $\theta$ in non-utterance sections.

the sampling rate of the question audio signal is 16 kHz and the amplitude of the input waveform is normalized to 1.0.

### 5.4 Feature Extraction

We adopt the Mel-frequency cepstral coefficient (MFCC) [22]. MFCC is one of the best transformation techniques successfully used in recent ASR systems that is based on the mechanism for human auditory perception. An MFCC vector is extracted for each short time window of the source audio signal. In addition, the first and second derivatives of MFCC are called a delta MFCC and a delta-delta MFCC, both of which are also effective temporal representations [23].

A feature vector consists of a 13-dimensional MFCC, a 13-dimensional delta MFCC, and a 13-dimensional delta-delta MFCC, and is in total a 39-dimensional vector. We set the window size to 25 ms and the frame shift to 10 ms.

### 5.5 HMM Topology

A digit is modeled as a linear HMM whose transition matrix meets the following condition:

$$P(s_j|s_i) = 0 \quad \text{if } i \neq j \text{ and } i+1 \neq j, \tag{17}$$

where $P(s_j|s_i)$ is the transition probability from state $s_i$ to $s_j$.

Each HMM has 20 states including the initial and final states. The observation likelihood function is a 39-dimensional Gaussian distribution for each state. In addition, we adopt the triphone model [24], [25], [26]; thus, each HMM denotes each triplex pattern of digits.

## 6. Solver Evaluation

We evaluated our solvers' performances for both the visual and the audio versions of reCAPTCHA.

### 6.1 Data

As listed in **Table 1**, the experiments were performed with two datasets. In dataset A-1, 2000 questions downloaded from the actual visual reCAPTCHA as of July 2013 what used to evaluate the performance of the visual reCAPTCHA solver. In dataset A-2, 400 questions downloaded from the actual audio reCAPTCHA as of April 2013 what used to evaluate the performance of the audio reCAPTCHA solver.

### 6.2 Metrics

#### 6.2.1 Strict Accuracy and Off-by-one Accuracy

We evaluated the solvers' performance with two metrics (strict accuracy and off-by-one accuracy) defined as follows.

- Strict accuracy is the ratio of strictly correct questions:

$$\{\text{strict accuracy}\} = \frac{T_{\text{strict}}}{R}, \tag{18}$$

where $R$ is the number of questions, and $T_{\text{strict}}$ is the number of strictly correct questions.

**Table 1** Datasets for evaluating reCAPTCHA solvers.

| Dataset | Version | Amount | Collected date |
|---|---|---|---|
| A-1 | Visual (current) | 2,000 questions | July 2013 |
| A-2 | Audio (current) | 400 questions | April 2013 |

**Table 2** Average performance of visual reCAPTCHA solver.

| | Closed test | Open test |
|---|---|---|
| Strict accuracy | 18.38% | 10.50% |
| Off-by-one accuracy | 44.94% | **31.75%** |

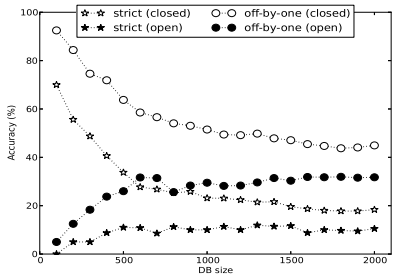



**Fig. 17** Relationship between visual reCAPTCHA solver's accuracy and data size. Performance saturated when data size reached around 1,500.

**Table 3** Average performance of audio reCAPTCHA solver.

| | Closed test | Open test |
|---|---|---|
| Strict accuracy | 22.06% | 22.50% |
| Off-by-one accuracy | 59.50% | **58.75%** |

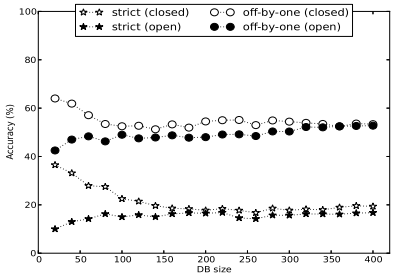



**Fig. 18** Relationship between audio reCAPTCHA solver's accuracy and data size. Performance saturated when data size reached around 200.

- Off-by-one accuracy evaluates the actual vulnerability of the reCAPTCHA:

$$\{\text{off-by-one accuracy}\} = \frac{T_{\text{off-by-one}}}{R}, \tag{19}$$

where $R$ is the number of all questions, and $T_{\text{off-by-one}}$ is the number of correct questions. Here, a question is regarded as correct when the Levenshtein distance between the solver's output and the correct answer is less than two. This excludes the case of insertion error for the evaluation of the audio reCAPTCHA solver, because the audio reCAPTCHA disallows insertion error as described in Section 5.1.

Note that the actual vulnerability of the reCAPTCHA is represented by off-by-one accuracy, because the reCAPTCHA regards a response as correct even when there is an off-by-one error.

#### 6.2.2 Open Test and Closed Test

The solver was trained with four-fifth of the entire dataset. Open test used the remaining one-fifth data for the evaluation. This test measured the practical performance of the system because it used unknown data excluded from the training. We performed this five times to obtain the average performance, which is called five-fold cross validation. Closed test used the same data as used in the training phase to check whether over-training has not occured.

#### 6.2.3 Transition of Performance

The transition of the solver's performance was also evaluated. We changed the number of training data at intervals of 100 samples for the visual reCAPTCHA solver and 20 samples for the audio reCAPTCHA solver.

### 6.3 Performance of reCAPTCHA Solvers

**Table 2** lists the average results when all data in dataset A-1 were used. The solver cracked the current version of the visual reCAPTCHA with 31.75% accuracy. **Figure 17** plots the trend in terms of strict accuracy and off-by-one accuracy. The solver's performance saturated when the data size reached around 1,500.

**Table 3** lists the average results when all data in dataset A-2 were used. The solver cracked the current version of the audio reCAPTCHA with 58.75% accuracy. **Figure 18** plots the trend in the performance. The performance saturated when the data size reached around 200. Note that the closed accuracy is almost the same as the open accuracy. This means that the solvers did not over-trained and have appropriate complexity for the given datasets.

To evaluate the computational costs of our reCAPTCHA solvers, we used an Amazon AWS M1.medium instance that has one virtual CPU and 3.75 GB of memory (the physical processor was Intel Xeon E5645, 2.4 GHz). Our audio reCAPTCHA solver takes 0.86 seconds to solve one audio reCAPTCHA. This computational time is sufficiently reasonable since it is as fast as a human solves an audio reCAPTCHA. Our visual reCAPTCHA solver takes 11.3 seconds to solve one visual reCATPCHA. The breakdown of the time is: 11.2 seconds for preprocessing and 0.1 seconds for feature extraction and HMM decoding.

## 7. Bottleneck Evaluation of Visual CAPTCHA Solvers

To clarify effective countermeasures against the HMM-based method, we carried out two synthetic experiments for visual CAPTCHAs. In the first experiment, the solver attacked various defensive techniques one by one. The second experiment was conducted to prove that one of the defensive techniques of using multiple fonts, which showed fine defensive performance in the first experiment, can be easily broken by a simple counter technique.

We evaluated the following defensive techniques adopted by recent CAPTCHA systems in popular web services as listed in **Table 5**. They have been also known as being practical against previous breaker frameworks [4].

- Using multiple fonts and large set of characters is effective for anti-classification methods. They slightly affect performances of most classification methods.
- Conjunction of overlapping characters and obscuring their segmentation point with randomized factors (e.g., random font size, random text length, and character rotation) is one of the best anti-segmentation techniques.
- The linear transformation of a word makes vertical segmentation of characters more difficult.
- The wavy transformation of a word increases the difficulty of finding characters' positions and prevents automated recognition by nonlinearly distorting their shapes.

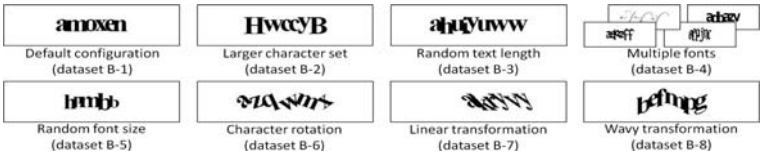



**Fig. 19** Simulated CAPTCHA examples. Dataset B-1 was the default configuration; dataset B-2 included a larger character set; dataset B-3 consisted of randomized text length; dataset B-4 had multiple fonts; dataset B-5 randomized font sizes; dataset B-6 rotated characters; dataset B-7 used the linear transformation; and dataset B-8 adopted the wavy transformation.

**Table 4** Datasets for evaluating robustness of visual CAPTCHA solver.

| Dataset | Character set | Text length | Font | Font size | Rotation | Linear transformation | Wavy transformation |
|---|---|---|---|---|---|---|---|
| B-1 | a-z | 6 characters | 1 kind | 50 point | N/A | N/A | N/A |
| B-2 | a-zA-Z | 6 characters | 1 kind | 50 point | N/A | N/A | N/A |
| B-3 | a-z | 6 to 8 characters | 1 kind | 50 point | N/A | N/A | N/A |
| B-4 | a-z | 6 characters | 10 kinds | 50 point | N/A | N/A | N/A |
| B-5 | a-z | 6 characters | 1 kind | 30 to 50 point | N/A | N/A | N/A |
| B-6 | a-z | 6 characters | 1 kind | 50 point | −45° to 45° | N/A | N/A |
| B-7 | a-z | 6 characters | 1 kind | 50 point | N/A | adopted | N/A |
| B-8 | a-z | 6 characters | 1 kind | 50 point | N/A | N/A | adopted |

**Table 5** Defensive techniques in popular services.

| Name | Description |
|---|---|
| reCAPTCHA | Six to eight continuous characters, lower/upper case alphabets, linear transformation, and wavy transformation. |
| Yahoo! | Five to seven continuous characters, lower case alphabets, multiple fonts, and character rotation. |
| Microsoft | Eight to ten non-continuous characters, lower/upper case alphabets and digits, multiple fonts, random font size, and character rotation. |
| Amazon | Six continuous characters, upper case alphabets and digits, and character rotation. |

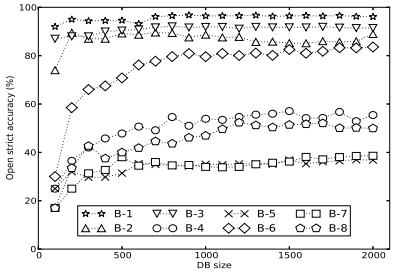



**Fig. 20** Effectiveness of visual defensive techniques for our framework. Solver cracks almost all questions of default configuration (dataset B-1). Multiple fonts (dataset B-4), randomized font sizes (dataset B-5), linear transformation (dataset B-7), and wavy transformation (dataset B-8) greatly degrade solver's performance. Character rotation (dataset B-6) requires many training samples to obtain saturated performance. Large character set (dataset B-2) and random text length (dataset B-3) scarcely contribute to security.

### 7.1 Experiment 1: Bottleneck Defensive Techniques

The experiment was conducted with eight types of simulated visual CAPTCHA whose examples are shown in **Fig. 19**. We evaluated the solver's performance for each configurations of the simulated CAPTCHA listed in **Table 4**.

The default configuration of the simulated CAPTCHA (dataset B-1) was a fixed text length with six lower case letters, single font, fixed font size, no character rotation, no linear transformation, and no wavy transformation. The other datasets were designed to evaluate each defensive technique described above in isolation. Dataset B-2 included both lower and upper case letters. Dataset B-3 consisted of randomized numbers of six to eight characters. Dataset B-4 had multiple font faces. Dataset B-5 varied font sizes from 30 to 50 points. Dataset B-6 rotated each character with a random angle from −45° to 45°. Dataset B-7 adopted the linear transformation, and dataset B-8 did the wavy transformation.

The solver's strict accuracy for each dataset listed in Table 4 was evaluated with five-fold cross validation. The number of data varied from 100 to 2,000 in increments of 100. The solver's schema was the same as the visual reCAPTCHA solver described in Section 4.

**Figure 20** shows the trend of open strict accuracy for each dataset. The solver cracked almost all questions of the default configuration, which simply overlapped characters (dataset B-1). Among the other configurations, multiple fonts (dataset B-4), randomized font size (dataset B-5), linear transformation (dataset B-7), and wavy transformation (dataset B-8) greatly degraded the solver's performance. While character rotation (dataset B-6) was less effective with the maximum DB size, it required many training samples to obtain the saturated performance. Large character set (dataset B-2) and random text length (dataset B-3) scarcely contributed as a defensive technique compared with the other techniques.

### 7.2 Experiment 2: Counter Attack on Multiple Fonts

This experiment proved that even multiple fonts can be compromised by a simple counter technique: composing a solver with multiple sub-solvers each of which recognizes a single font. **Figure 21** depicts the overview of the counter technique against multiple fonts. The feature sequence $O$ is passed into sub-solvers $\phi_1, \ldots, \phi_i, \ldots, \phi_I$, where $i$ means the index of the $i$-th font, and $I$ means the number of font types. Given the feature sequence $O$, a sub-solver $\phi_i$ outputs the label sequence $W_i$ and the accompanying likelihood $P(O|W_i; \phi_i)$. Finally, the solver outputs the best one among all answers of sub-solvers that has the highest likelihood.

In the following, we refer a solver as a parallel solver if

it adopts the counter technique described in this section and as a non-parallel solver otherwise. We also refer a simulated CAPTCHA system with multiple fonts as a multi-font CAPTCHA and as a single-font CAPTCHA otherwise.

To prove the effectiveness of a parallel solver, the performance evaluation was carried out in the following three conditions:

**Condition 1** A multi-font CAPTCHA generated 2,000 questions. A non-parallel solver was trained with 1,600 of the questions. The solver's open strict accuracy was evaluated with the other 400 questions.

**Condition 2** For each font listed in **Table 6**: a single-font CAPTCHA generated 2,000 questions; a non-parallel solver was trained with 1,600 of the questions; and its open strict accuracy was evaluated with the other 400 questions.

**Condition 3** In the training process, each sub-solver of a parallel solver was trained with 1,600 questions generated by each single-font CAPTCHA. In the evaluation process, the solver's open strict accuracy was evaluated with 400 questions generated by a multi-font CAPTCHA.

Table 6 shows the solvers' performance for each single-font CAPTCHA under condition 2. Each single-font CAPTCHA was solved with adequate accuracy: from 82.75% to 100.00% (95.30% on average).

**Table 7** shows the solver's performance against multi-font CAPTCHA under conditions 1 and 3, and the average performance against single-font CAPTCHAs under condition 2. The result of condition 3 proves that just by adopting a parallel solver, the multi-font CAPTCHA is broken with almost the same accuracy as that for single-font CAPTCHAs. Hence, the defensive technique of using multiple fonts is easily surpassed by the HMM-based method.

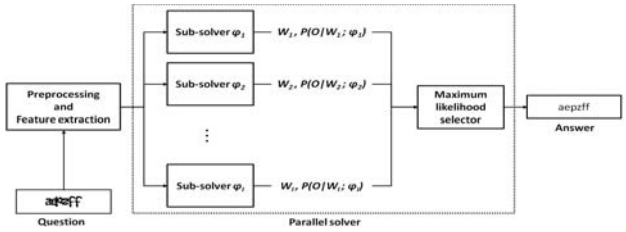



**Fig. 21** Parallel CAPTCHA solver to deal with multiple fonts. Feature sequence is passed into sub-solvers. Parallel solver outputs best answer among sub-solvers in terms of likelihood.

**Table 6** Example of simulated single font CAPTCHAs and performance of non-parallel solvers.

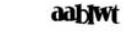
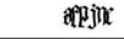
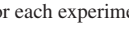
| Font example | Accuracy | Font example | Accuracy |
|---|---|---|---|
|  | 99.50% |  | 95.75% |
|  | 100.00% |  | 99.25% |
|  | 98.00% |  | 94.25% |
|  | 82.75% |  | 96.00% |
|  | 90.50% |  | 100.00% |

**Table 7** Solvers' performance for each experimental conditions in Section 7.2.

| | CAPTCHA type | Solver type | Accuracy |
|---|---|---|---|
| Condition 1 | multi-font | non-parallel | 55.50% |
| Condition 2 | single-font | non-parallel | 95.30% |
| Condition 3 | multi-font | parallel | 90.80% |

## 8. Noise Evaluation of Audio CAPTCHA Solvers

As described in Section 5, there are two defensive techniques specific to audio CAPTCHAs: additive noise, which covers the entire part of a question to prevent target voices being detected and recognized, and convolutive noise, which distorts each target voice with an unknown filtering process to collapse a part of the spectral feature. Specifically, additive noise is adopted in most famous commercial audio CAPTCHAs including those of Microsoft and Yahoo!; thus, this section evaluates our solver's robustness for various kinds of additive noise with simulated audio CAPTCHAs. Convolutive noise, on the other hand, is not discussed here because this noise is adopted only in the audio reCAPTCHA and the unknown filtering process is difficult to reproduce on a simulated CAPTCHA.

We conducted two experiments. The first experiment compared the solver's performance for the current audio reCAPTCHA, which adopted the additive stationary noise, and the previous audio reCAPTCHA, which did not adopt the additive noise, to show that the additive stationary noise in the audio reCAPTCHA scarcely contributes to the security. The second experiment examined the robustness for several types of additive noise with simulated CAPTCHAs that were generated by adding those kinds of noise to the previous version of audio reCAPTCHA.

### 8.1 Data

As listed in **Table 8**, the experiments were performed with three datasets. Dataset C-1 was a set of questions downloaded from the current version of audio reCAPTCHA that was the same as dataset A-2 in Table 1. Dataset C-2 was that from the previous version. We downloaded 400 questions both from the previous and the current versions. (As described in Section 5, the difference between the previous and current versions of reCAPTCHA was that the current version adopted the additive stationary noise that entirely covered a question.) Dataset C-3 was obtained by segmenting dataset C-2 into clusters. The first experiment was carried out with datasets C-1 and C-2. The second experiment was conducted with dataset C-3.

### 8.2 Experiment 1: Effectiveness of reCAPTCHA's Additive Noise

We compared the performance of our solver for the current version of audio reCAPTCHA (dataset C-1) with that for the previous version (dataset C-2). We evaluated the open accuracies for both datasets with five-fold cross validation. The solver's schema was the same as that of the audio reCAPTCHA solver described in Section 5.

**Table 8** Datasets for evaluating robustness of audio CAPTCHA solver.

| Dataset | Version | Amount | Collected date |
|---|---|---|---|
| C-1 | Audio (current) | 400 questions | April 2013 |
| C-2 | Audio (former) | 400 questions | December 2012 |
| C-3 | Audio (former) | 1200 clusters | December 2012 |

**Table 9** Performance for current version of audio reCAPTCHA (dataset C-1) and that of the previous version (dataset C-2).

| | Dataset C-1 | Dataset C-2 |
|---|---|---|
| Open strict accuracy | 20.00 | 22.50 |
| Open off-by-one accuracy | 56.75 | 58.75 |

**Table 10** Description of noise signals used in synthetic evaluation for audio CAPTCHA solver.

| Class | Name | Description |
|---|---|---|
| Stationary | White | White noise. |
| | Brown | Brown noise. |
| | Pink | Pink noise. |
| Semantic | Speech | Spoken audio signal. Noise audio is selected for each cluster from a corpus of spontaneous Japanese dialogue [28]. |
| | Music | Music audio signal. Noise audio is randomly clipped from "I Saw Her Standing There" by the Beatles for each cluster. |

**Table 9** compares the performances when using all questions in datasets C-1 and C-2. Both performances were almost the same; hence the additive stationary noise adopted in the audio reCAPTCHA is ineffective for preventing the solver's recognition.

### 8.3 Experiment 2: Robustness of Recognition for Various Kinds of Additive Noise

The result obtained from experiment 1 proved that the HMM-based recognition method is scarcely disturbed by the stationary noise adopted in the current version of audio reCAPTHCA. In this experiment, we evaluated the robustness of the method against various kinds of additive noise where the method decoded several simulated CAPTCHAs.

We generated the simulated CAPTCHAs by adding one of the noise signals listed in **Table 10** to each cluster signal detected from the previous version of audio reCAPTCHA, which did not have additive noise. Note that to evaluate the recognition method in isolation from the preprocessing, the simulated CAPTCHAs were generated not from questions but from clusters. This was intended to make the comparison straightforward; the parameter for the preprocessing should be suitably configured depending on the strength of the noise signals.

We tested five kinds of noise that can be divided into two classes: stationary noise, such as white noise, and semantic noise. Semantic noise has characteristics more similar to CAPTCHA's target voices like those in spoken audio. (A typical example of semantic noise is a male voice when the target is a female voice.) This has been known to be an effective technique that defends against the classification of non-continuous audio CAPTCHAs [11]. We tested white noise, brown noise, and pink noise [27] as stationary noise and spoken sentences and music as semantic noise.

We also examined how the solver's performance was affected by various noise levels. The noise level was controlled by changing the signal-to-noise ratio (SNR) calculated as:

$$\mathrm{SNR} = 10 \log_{10} \sqrt{\frac{\sum_{t=1}^{T} s_t}{\sum_{t=1}^{T} n_t}}, \tag{20}$$

where $s_1, \ldots, s_T$ is the source audio signal and $n_1, \ldots, n_T$ is the noise signal. Note that the lower the SNR, the stronger the noise. The SNR ranges from $-25$ to 25 at five intervals.

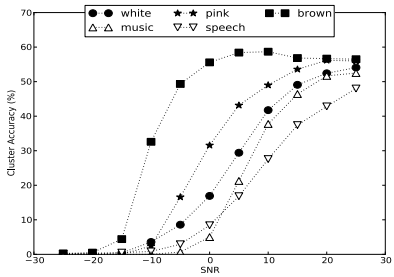



**Fig. 22** Relationship between SNR and performance of solver for several kinds of additive noise. Semantic noise results in lower accuracy for each value of SNR.

We conducted five-fold cross validation for each type of noise and for each value of SNR. **Figure 22** plots the relationship between SNR and the solver's per-cluster accuracy for each noise. Semantic noise resulted in lower accuracy for each value of SNR, which means semantic noise enables more secure distortion without increasing the strength of noise.

## 9. Discussion and Guidelines

Our solvers cracked the current version of visual reCAPTCHA with 31.75% accuracy and that of audio reCAPTCHA with 58.75% accuracy. We conclude that our solvers disclosed the vulnerability of the current reCAPTCHA systems in accordance with the criteria for breaking CAPTCHAs claimed in previous works, e.g., "automatic scripts should not be more successful than 1 in 10,000" by Chellapilla et al. [29] or "a CAPTCHA schema is broken when the attacker is able to reach a precision of at least 1%" by Bursztein et al. [4].

The security of the reCAPTCHA is further threatened by the fact that the solvers were easily implemented by using an off-the-shelf library, HTK, and their performance saturated with just a small amount of training data — 1,500 and 200 questions for the visual and audio reCAPTCHA, respectively.

On the other hand, we found out several bottlenecks of the HMM-based method in the synthetic experiments. The rest of this section discusses the guidelines to design more secure continuous CAPTCHAs in accordance with those experiments.

### 9.1 Visual CAPTCHAs

The previous security guidelines of continuous visual CAPTCHAs have been organized by Burszein et al. [4]: (1) using continuous characters is considered to be the most secure anti-segmentation technique; and (2) using continuous characters is only effective with randomized factors such as the number and size of the characters. In accordance with the experimental results in Section 7, we propose the following complementary guidelines for continuous visual CAPTCHAs against HMM-based attacks.

Guideline 1

Continuous CAPTCHAs scarcely prevent HMM-based attacks, even in conjunction with random text length, character rotation, large character set, and multiple fonts.

Random text length, character rotation, and large character set showed poor defensive performance for the HMM-based method with enough training data with the result of synthetic evaluation shown in Fig. 20. Even using multiple fonts, which prevented the solver's attack in the first synthetic experiment, turned out to be ineffective by using a simple counter technique of composing a solver with multiple sub-solvers.

Guideline 2

To prevent HMM-based attacks, continuous CAPTCHAs should be used in conjunction with random font size, the linear transformation, and the wavy transformation.

Randomizing font size greatly affected the performance of the HMM-based method. This is because feature vectors were extracted as raw pixel information for each column in our solver while an HMM poorly works with shift-variant observations.

The linear transformation and the wavy transformation also exceedingly deteriorated the performance of the solver. Essentially, the HMM-based method is poor at dealing with these distortions, because the HMM-based method implicitly performs vertical segmentation of characters, while the linear transformation further overlaps horizontal positions of them. Additionally, an HMM especially with a Gaussian observation likelihood function is not suitable for recognizing nonlinearly distorted sequences caused by the wavy transformation.

Note that this guideline may not be applicable in the future. The preprocessing of our proposed visual reCAPTCHA solvers uses a naïve technique for linear reshaping and nonlinear reshaping. Sophisticated image processing techniques for text dewarping and improvement of the features will improve the preprocessing in terms of performance and computational cost.

### 9.2 Audio CAPTCHAs

Guideline 3

For additive background noise, audio CAPTCHAs should adopt semantic noise rather than stationary noise in favor of security.

The experimental results shown in Table 9 revealed that the stationary noise adopted in reCAPTCHA scarcely prevented HMM-based recognition. In addition, the results in Fig. 22 indicated that semantic noise was better than stationary noise at deteriorating the performance of an HMM-based solver.

Semantic noise also can be advantageous because humans can easily handle such noise even at low SNRs when the target audio differs semantically from the noise. This is known as the cocktail party effect [30]. Thus, we can expect that adopting proper semantic noise will enhance the security of CAPTCHAs as well as retain excellent usability.

Guideline 4

Additive background noise scarcely prevents automated segmentation, without similar power level to target voices.

Additive background noise is adopted to prevent not only recognizing but segmenting target voices. However, they can be easily segmented by the power-based voice activity detection algorithm described in Section 5.3, as long as there is a clear difference in sound power. Thus, background noise and target voices desirably have the same power level.

This paper did not consider the solver's workaround for the convolutive noise in the audio reCAPTCHA, since the specific filtering process of the convolutive noise is unknown other than it collapses the high frequency range of the original signal's power spectrum. Identifying the filtering process and assessing its effectiveness for defense is one future direction.

Auditory characteristics, or auditory illusions, such as phonemic restoration [31], [32] might be helpful to develop an audio CAPTCHA which supports both security and usability at the same time. The relationship between the level of noise and the word intelligibility [33] is helpful in designing an optimal level of noise.

## 10. Conclusion

We developed and evaluated an HMM-based framework to automatically solve recent CAPTCHAs. It could overcome one of the most secure defensive techniques of using continuous symbols.

We demonstrated how our framework was applied for actual CAPTCHA breakers targeting the current version of Google's reCAPTCHA for both the visual and audio versions. Our solvers cracked the visual and audio reCAPTCHA systems with 31.75% and 58.75% accuracy, which threatens the security of recent CAPTCHAs.

We also conducted synthetic experiments to find out the weakness of our HMM-based method. The experimental results led us to the principles in Section 9 to protect against our HMM-based CAPTCHA solver. Future works will involve user testing on each noise and/or distortion technique, to enrich the guidelines with quantitative discussion on the trade-off between security and usability.

### References


[1] Ahn, L.V., Blum, M. and Langford, J.: Telling Humans and Computers Apart Automatically, *Comm. ACM*, Vol.47, No.2, pp.56–60 (2004).

[2] Mori, G. and Malik, J.: Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA, *Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol.1, pp.134–141 (2003).

[3] Yan, J. and Ahmad, A.S.E.: A Low-cost Attack on a Microsoft CAPTCHA, *Proc. 15th ACM Conference on Computer and Communications Security*, pp.543–554 (2008).

[4] Bursztein, E., Martin, M. and Mitchell, J.: Text-based CAPTCHA Strengths and Weaknesses, *Proc. 18th ACM Conference on Computer and Communications Security*, pp.125–138 (2011).

[5] Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proc. IEEE*, Vol.77, No.2, pp.257–286 (1989).

[6] Lee, A., Kawahara, T. and Shikano, K.: Julius — An Open Source Real-time Large Vocabulary Recognition Engine, *Proc. 7th European Conference on Speech Communication and Technology*, pp.1691–1694 (2001).

[7] Plamondon, R. and Srihari, S.N.: Online and Off-line Handwriting Recognition: A Comprehensive Survey, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.22, No.1, pp.63–84 (2000).

[8] von Ahn, L., Maurer, B., McMillen, C., Abraham, D. and Blum, M.: reCAPTCHA: Human-based Character Recognition via Web Security Measures, *Science*, Vol.321, No.5895, pp.1465–1468 (2008).

[9] Young, S.J.: The HTK Hidden Markov Model Toolkit: Design and Philosophy, Technical Report, Engineering Department, Cambridge

University (1994).
[10] Hindle, A., Godfrey, M.W. and Holt, R.C.: Reverse Engineering CAPTCHAs, *Proc. 15th Working Conference on Reverse Engineering*, pp.59–68 (2008).
[11] Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C. and Mitchell, J.: The Failure of Noise-based Non-continuous Audio CAPTCHAs, *Proc. 2011 IEEE Symposium on Security and Privacy*, pp.19–31 (2011).
[12] Bursztein, E., Bethard, S., Fabry, C., Mitchell, J.C. and Jurafsky, D.: How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation, *Proc. 2010 IEEE Symposium on Security and Privacy*, pp.399–413 (2010).
[13] Chellapilla, K., Larson, K., Simard, P. and Czerwinski, M.: Computers Beat Humans at Single Character Recognition in Reading Based Human Interaction Proofs (HIPs), *Proc. 2nd Conference on Email and Anti-Spam*, pp.21–22 (2005).
[14] Sayre, K.M.: Machine Recognition of Handwritten Words: A Project Report, *Pattern Recogn*., Vol.5, No.3, pp.213–228 (1973).
[15] Yan, J., Salah, A. and El Ahmad, A.S.: The Robustness of a New CAPTCHA, *3rd Workshop on System Security*, pp.36–41 (2010).
[16] Cruz-Perez, C., Starostenko, O., Uceda-Ponga, F., Alarcon-Aquino, V. and Reyes-Cabrera, L.: Breaking reCAPTCHAs with Unpredictable Collapse: Heuristic Character Segmentation and Recognition, *Pattern Recogn*., pp.155–165, Springer (2012).
[17] Brown, M.K. and Wilpon, J.G.: A Grammar Compiler for Connected Speech Recognition, *IEEE Trans. Signal Processing*, Vol.39, No.1, pp.17–28 (1991).
[18] Rabiner, L. and Juang, B.-H.: *Fundamentals of Speech Recognition*, Prentice Hall (1993).
[19] Lee, K.-F. and Hon, H.-W.: Large-vocabulary Speaker-independent Continuous Speech Recognition Using HMM, *Proc. 1988 International Conference on Acoustics, Speech, and Signal Processing*, Vol.1, pp.123–126 (1988).
[20] Baum, L.E., Petrie, T., Soules, G. and Weiss, N.: A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, *The Annals of Mathematical Statistics*, Vol.41, No.1, pp.164–171 (1970).
[21] Mairal, J., Bach, F., Ponce, J. and Sapiro, G.: Online Dictionary Learning for Sparse Coding, *Proc. 26th Annual International Conference on Machine Learning*, pp.689–696 (2009).
[22] Tiwari, V.: MFCC and Its Applications in Speaker Recognition, *International Journal on Emerging Technologies*, Vol.1, No.1, pp.19–22 (2010).
[23] Furui, S.: Speaker-independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum, *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol.34, No.1, pp.52–59 (1986).
[24] Schwartz, R., Chow, Y., Roucos, S., Krasner, M. and Makhoul, J.: Improved Hidden Markov Modeling of Phonemes for Continuous Speech Recognition, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol.9, pp.21–24 (1984).
[25] Lee, K.-F.: Context-dependent Phonetic Hidden Markov Models for Speaker-independent Continuous Speech Recognition, *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol.38, No.4, pp.599–609 (1990).
[26] Nakagawa, S., Hanai, K., Yamamoto, K. and Minematsu, N.: Comparison of Syllable-based HMMs and Triphone-based HMMs in Japanese Speech Recognition, *Proc. 1999 IEEE Automatic Speech Recognition and Understanding Workshop*, pp.393–396 (1999).
[27] Halley, J.M. and Kunin, W.E.: Extinction Risk and the 1/f Family of Noise Models, *Theoretical Population Biology*, Vol.56, No.3, pp.215–230 (1999).
[28] Maekawa, K.: Corpus of Spontaneous Japanese: Its Design and Evaluation, *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition* (*SSPR 2003*), pp.7–12 (2003).
[29] Chellapilla, K., Larson, K., Simard, P. and Czerwinski, M.: Building Segmentation Based Human-friendly Human Interaction Proofs (HIPs), *Human Interactive Proofs*, Springer, pp.1–26 (2005).
[30] Bronkhorst, A.W.: The Cocktail Party Phenomenon: A Review of Research on Speech Intelligibility in Multiple-talker Conditions, *Acta Acustica united with Acustica*, Vol.86, No.1, pp.117–128 (2000).
[31] Kashino, M.: Phonetic Restoration: The Brain Creates Missing Speech Sounds, *Acoustic Science and Technology*, Vol.27, No.6, pp.318–321 (2006).
[32] Nishimoto, T. and Watanabe, T.: The Comparison between the Deletion-based Methods and the Mixing-based Methods for Audio CAPTCHA Systems, *Proc. 11th Annual Conference on the International Speech Communication Association*, pp.266–269 (2010).
[33] Sato, H., Morimoto, M., Hoshino, Y. and Odagawa, Y.: Relationship between Sound Insulation Performance of Walls and Word Intelligibility Scores, *Applied Acoustics*, Vol.73, No.1, pp.43–49 (2012).

**Shotaro Sano** received B.E. and M.Inf. from Kyoto University, Kyoto, Japan, in 2011 and 2014 respectively. His research interests include human robot interaction, image/audio processing, and machine learning. He received the Best Paper Award of IWSEC-2013.

**Takuma Otsuka** received B.E., M.Inf., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 2009, 2011, and 2014 respectively. He has been a Researcher at NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan, since 2014. His research interests include statistical signal processing, robot audition, statistical pattern recognition, and machine learning. He received the Best Paper Award of IEA/AIE-2010, NEC CDr. Otsuka is a member of IPSJ and IEEE.

**Katsutoshi Itoyama** received B.E. degree in 2006, M.S. degree in Informatics in 2008, and Ph.D. degree in Informatics in 2011 all from Kyoto University. He is currently an Assistant Professor of the Graduate School of Informatics, Kyoto University, Japan. His research interests include musical sound source separation, music listening interfaces, and music information retrieval. He recieved the 24th TAF Telecom Student Technology Award and the IPSJ Digital Courier Funai Young Researcher Encouragement Award. He is a member of the IPSJ, ASJ, and IEEE.

**Hiroshi G. Okuno** received B.A. and Ph.D. from the University of Tokyo in 1972 and 1996, respectively. He worked for NTT, JST, Tokyo University of Science, and Kyoto University. He is currently a professor of Graduate Program for Embodiment Informatics, Graduate School of Creative Science and Engineering, Waseda University, and a professor emeritus, Kyoto University. He was visiting scholar at Stanford University from 1986 to 1988. He is currently engaged in computational auditory scene analysis, robot audition and animal acoustics. He co-edited "Computational Auditory Scene Analysis" (Lawrence Erlbaum Associates, 1998), "Advanced Lisp Technology" (Taylor and Francis, 2002), and "New Trends in Applied Artificial Intelligence (IEA/AIE)" (Springer, 2007). He received various awards including the 2nd Best Paper Award of Advanced Robotics, 2014 Best Kakenhi Reviewer Award of Japan Society for the Promotion of Science, 2013 Science and Technology Award of Minister of Education, Calcure, Sports, Science Technology, 2010 NTF Award for Entertainment Robots and Systems, the Best Paper Award of IEA/AIE-2001, 2005, 2010, and 2013, and the 1990 Best Paper Award of the JSAI. He is a fellow of IEEE, JSAI, IPSJ and RSJ. He is also a member of AAAI, ACM, ASA, JSSST and JCSS.