Balancing Usability and Security in a Video CAPTCHA

Kurt Alfred Kluever
Google, Inc.
76 Ninth Ave.
New York, NY 10011
kak@google.com

Richard Zanibbi
Document and Pattern Recognition Lab
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
rlaz@cs.rit.edu

ABSTRACT

We present a technique for using content-based video labeling as a CAPTCHA task. Our CAPTCHAs are generated from YouTube videos, which contain labels (*tags*) supplied by the person that uploaded the video. They are graded using a video's tags, as well as tags from *related* videos. In a user study involving 184 participants, we were able to increase the human success rate on our video CAPTCHA from roughly 70% to 90%, while keeping the success rate of a tag frequency-based attack fixed at around 13%. Through a different parameterization of the challenge generation and grading algorithms, we were able to reduce the success rate of the same attack to 2%, while still increasing the human success rate from 70% to 75%. The usability and security of our video CAPTCHA appears to be comparable to existing CAPTCHAs, and a majority of participants (60%) indicated that they found the video CAPTCHAs more enjoyable than traditional CAPTCHAs in which distorted text must be transcribed.

Categories and Subject Descriptors

H.5.2 [**HCI**]: Web-based interaction; D.4.6 [**Security and Protection**]: Access Control and Authentication

Keywords

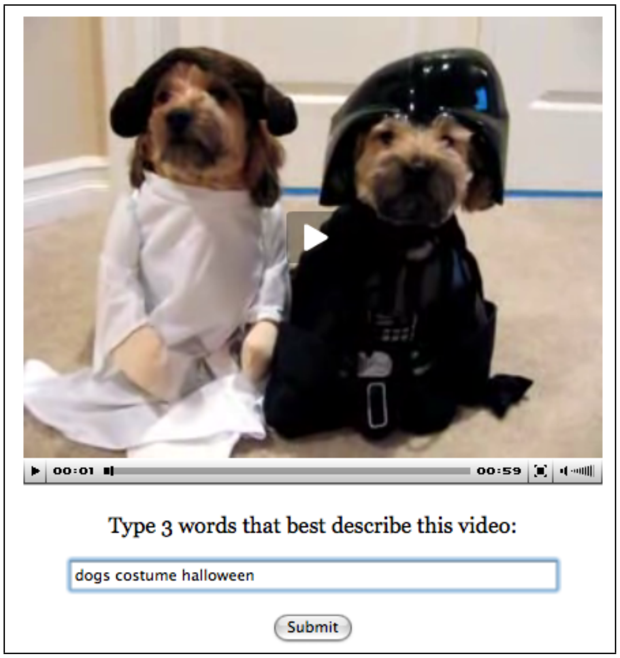
Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA); Human Interactive Proof (HIP); video understanding; tagging

1. INTRODUCTION

A Completely Automated Public Turing test to tell Computers and Humans Apart (*CAPTCHA*) is a variation of the Turing test [24], in which an online challenge is used to distinguish humans from computers. They are commonly used to prevent the abuse of online services, such as a program creating thousands of free email accounts and then using them to send SPAM. A number of hard artificial intelligence problems including natural language processing [8], character recognition [3, 4, 23, 26], speech recognition [16], and image understanding [5, 6, 11, 22] have been used as the basis for CAPTCHAs [19].

Various criteria have been proposed in the literature for evaluating CAPTCHAs [1, 22, 26]. We propose the following four desirable properties for CAPTCHAs:

1. **Automated:** Challenges should be easy to automatically generate and grade by a computer.
2. **Open:** The underlying database(s) and algorithm(s) used to generate and grade the challenges should be public. This property is in accordance with Kerckhoffs' Principle, which states that a system should remain secure even if everything about the system is public knowledge [14].
3. **Usable:** Challenges should be easily solved in a reasonable amount of time by humans. Furthermore, challenges should strive to minimize the effect of a user's language, physical location, education, and/or perceptual abilities.
4. **Secure:** Challenges should be difficult for machines to solve algorithmically.



Type 3 words that best describe this video:

dogs costume halloween

Submit

Figure 1: An example of our video CAPTCHA.

Symposium on Usable Privacy and Security (SOUPS) 2009, July 15–17, 2009, Mountain View, CA USA.

The most common type of CAPTCHA requires a user to transcribe distorted characters displayed within a noisy image (such as in [4]). The algorithms and data used to automatically generate these challenges are publicly available, but not only do many users find them frustrating, automated programs have been successful at defeating them. For example, researchers have developed an attack against Microsoft's Hotmail CAPTCHA that yields a 60% success rate [28]. The need for a new CAPTCHA that is automated, open, usable, and secure arises.

We present a new type of CAPTCHA, in which a user must provide three words (*tags*) describing a video taken from a public database (see Figure 1; an online demonstration is also available[1]). Words may be submitted as the video plays, i.e. the user does not have to wait for the video to finish before submitting their three words. In its simplest form, a challenge is passed if any of the three submitted tags match an author-supplied tag associated with the video. This challenge is similar to the image labeling game known as ESP created by von Ahn *et. al* [27], in which people are randomly paired up and then try to guess a common tag for an image. Our video CAPTCHA is similar to playing a game of ESP using videos, but where one player's responses (the ground truth set) are automatically generated from tags associated with videos in the database.

Due to the inherent ambiguity of natural language, misspellings by the authors of the videos, and inconsistencies in tagging behaviors, we hypothesized that exact matching of author-supplied tags would be a difficult task. However, in our first user study, the human success rate for exact matching of author-supplied tags was 75%. The goal of this research was to further improve the human success rate on our video CAPTCHA, while maintaining security against a tag frequency-based attack, where the three tags estimated to have the highest frequency (i.e. are associated with the largest number of videos) are submitted. We improve usability by expanding the user-supplied tags and the ground truth tags, and by allowing approximate string matching. To maintain security, we reject tags estimated to have a frequency greater than or equal to a given rejection threshold.

To test the usability and security of our CAPTCHA, we have conducted two user studies and simulated a frequency-based attack against a sample of challenges. Our first user study was used to explore possible grading functions and to determine the appropriate parameter values for the second user study. In the first user study, participants were only instructed to tag the videos and their responses were not graded. However, participants in the second user study were told whether they had passed or failed the video CAPTCHAs. For both user studies and the frequency-based attack, success rates were observed over the space of usability and security parameters. In the second user study, it was possible to increase human success rates from 70% (exact matching author tags) to 90% while maintaining an attack success rate of approximately 13%. These success rates are comparable to existing CAPTCHAs. In addition, we observed that different balances between security and usability could be achieved by modifying the generation and grading function parameters (see Table 8).

From our initial investigation, it appears that our video CAPTCHA is usable and secure. In addition, it is semi-automated (a human may be needed to ensure that the content is appropriate and the tags are in a given language), and open (all algorithms and the database used to generate challenges are publicly available). Our video CAPTCHAs may not be accessible to those with hearing or visual disabilities, and toward that end we would like to compare the usability of our system to strictly image-based or audio-based versions in the future. Currently we have explored only a single attack type, and acknowledge that other attacks may be more successful, such as submitting words detected in video frames or audio.

In the remaining sections of this paper, we outline our data sampling technique, the definition of our CAPTCHA generation and grading functions, report results from an attack simulation and two user studies, and finally conclude and recommend future avenues of research.

2. COLLECTING VIDEO SAMPLES

For our video dataset, we chose to utilize YouTube.com, which is currently the largest user-generated content video system available [2]. YouTube currently stores and indexes close to 150 million videos. Ideally, we would like to randomly sample from this large database, but this is not possible, as no comprehensive list of videos is available [20]. There are also restrictions on the number of API requests allowed per day and the number of results returned per query.

Randomly generating YouTube video identifiers (IDs) would yield a true random sample, but collecting a large sample in this fashion is impractical. YouTube video IDs are 11 characters long with a character set consisting of lower case letters (`a-z`), uppercase letters (`A-Z`), numbers (`0-9`), dashes (`-`), and underscores (_) for a total of 64 different characters. Therefore, there are $64^{11} \approx 7.4 \times 10^{19}$ possible IDs. Given that there are approximately 1.5×10^8 videos on YouTube, the probability of randomly generating a valid video ID is approximately 2×10^{-12}. Clearly, this is not a tractable method for collecting large samples.

A common method used for sampling hidden populations where direct interaction with individuals would be difficult is known as *snowball sampling* [10]. An s stage k name snowball sample is similar to a breadth-first search where a fixed number of children are selected at random at each node in the search tree. The sampling procedure is as follows:

1. From the population, pick a random sample of individuals (Stage 0).
2. Each individual in the current stage names k individuals (children) at random.
3. Repeat for s stages.

Recently, this sampling technique has been used to sample large social networks, including YouTube.com [20]. A common criticism of snowball sampling is that it biases results towards individuals who are connected to the entry points. Therefore, we chose to use random walks, which are a form of randomized local search. This technique has been previously used for sampling video data [12].

One can model YouTube as an undirected, bipartite graph G. The vertices in the graph consist of two disjoint sets: tags U and videos V. The edges in the graph are of the form (u, v) and (v, u) such that $u \in U$ and $v \in V$; edges represent associations between videos and tags. Given the YouTube video-tag graph G, a maximum walk depth m, and a dictionary D, the algorithm below returns a random walk of the graph in the form of an ordered list P of video-tags pairs (v, A).

```
RANDOMWALK($G, m, D$)
1. Create an empty list, $P \leftarrow \emptyset$, and counter $i \leftarrow 0$.
2. Randomly select a walk depth $d$, where $1 \leq d < m$.
3. Randomly select a starting tag $t$ from dictionary $D$.
4. Located the tag vertex $u$ corresponding to $t$ in $G$.
```

[1] Online at `http://www.cs.rit.edu/~dprl`

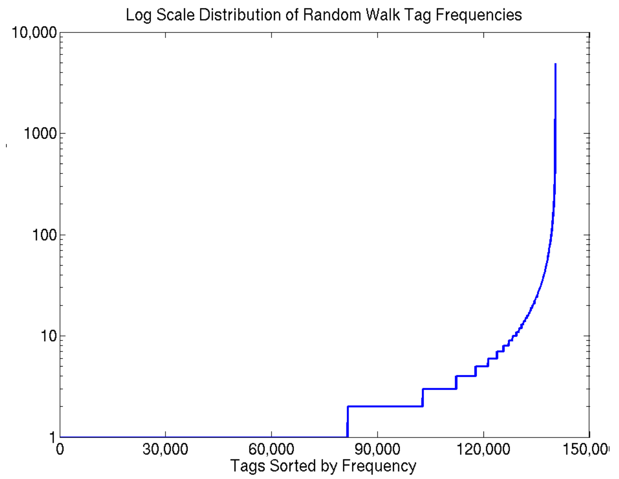


Figure 2: Log scale plot of estimated tag frequencies for 86,368 YouTube videos. Tags are listed in increasing frequency along the x axis, and the y axis shows tag counts.

5. While $i < d$:
   (a) Select a random edge (u, v) in G, where v is a video vertex.
   (b) Given the tags A on the video v, append (v, A) to P.
   (c) Select a random edge (v, w) in G where w is a tag associated with video v.
   (d) Assign $u \leftarrow w$ and increment i.
6. Return the list of video-tag pairs P.

In our experiments, we used a maximum depth of 100 ($m = 100$), to allow walks of reasonable depth, while preventing walks from becoming stuck in local neighborhoods or connected components in the graph. For our dictionary D, we used the English word list available on most Unix-based computers. We used YouTube API calls to obtain the video and tag vertices; note that there is a limit on the number of videos returned for a given tag query (a maximum of 1000).

Our tag frequency distribution was estimated from a set of 86,368 videos collected from many random walks. We plotted the tags in increasing order of frequency and observed that the shape of the curve was exponential (see Figure 2). A small number of tags are used very frequently, while most others are used infrequently.

3. CHALLENGE GENERATION

Given the YouTube graph G, a maximum walk depth m, a dictionary D, a tag frequency distribution estimate F, a maximum number of related tags to add n, and a rejection threshold t, the CAPTCHA generation algorithm below returns a pair (v, GT) containing a video and a set of acceptable ground truth tags.

$\textsc{VideoCaptcha}(G, m, D, F, n, t)$

1. A random walk of the video-tag graph G is performed to maximum depth m using dictionary D to choose a video. The last (most recent) video v and its tags A are stored: $(v, A) = \textsc{RandomWalk}(G, m, D)$
2. A list of videos R which are related to video v is obtained from G. In our case, this was performed using the YouTube API, which returns at most 100 video-tags pairs: (v_i, A_i)
3. Generate up to n additional tags from related videos: $E = \textsc{RelatedTags}(A, R, n)$
4. Using the tag frequency distribution estimate F, remove tags with a frequency greater than or equal to t: $GT = \textsc{RejectFrequentTags}(A \cup E, F, t)$
5. Return the selected video and a preprocessed version of the ground truth tag set: $(v, \textsc{PreProcess}(GT))$

To improve the usability of our CAPTCHA, we add tags from *related* videos to the ground truth tag set. RelatedTags, RejectFrequentTags, and PreProcess are defined in Sections 3.3, 3.4 and 4.1, respectively. To maintain security we filter tags estimated to occur frequently in the database. Details regarding ground truth tag set generation are described in the following subsections.

3.1 Related Videos

YouTube provides a list of up to 100 related videos for each video. Unfortunately, the details of how the related videos are selected are not public. *Relatedness* seems to involve some combination of the similarity of tags, the number of viewings a video has received, video co-views and possibly other factors. The use of related videos exploits social structure within the video database. The hope is that accepting tags from related videos will be helpful for users and difficult for attackers to construct or learn models for these social tagging patterns. For example, consider a video tagged with $\{obama, president\}$ which has a related video that is tagged with $\{barack, obama, president\}$. In our approach we assume that "*barack*" is likely a valid tag for the original video, even though the person that that posted the video did not provide it.

For this first investigation, we chose to use the set of related videos that YouTube provided, and left other techniques as future work. An alternate strategy would be to query using combinations of the tags on a video, the maximum number of which would be:

$$\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$$

Note that each tag-based query returns at most 1000 videos, so this technique only provides a partial view of videos in the database (i.e. our access to the video graph G is limited).

Tags from related videos also provide a form of social spell checking. For example, we observed that a video of the magician Criss Angel had many related videos which had been tagged as "*Chris Angel*" or "*Kris Angel*". By adding related tags, we are able to allow for common misspellings.

While there are often additional words to be obtained from a video's title [7], in our preliminary user study we found that adding titles did not substantially increase the usability of the system (e.g. we observed a decrease in security of 5% and only an increase in usability of 0.3% relative to matching against only author-supplied tags). In addition, we could not estimate the security impact of adding title words using our tag frequencies (which are calculated over tag space, not title space), and so we decided not to allow title words.

3.2 Cosine Similarity of Tag Sets

To select tags from those videos that have the most similar tag set to the challenge video, we performed a sort using the cosine similarity of the tags on related videos and the tags on the challenge video. The cosine similarity metric is commonly used in information retrieval to compare text documents [25]. The cosine similarity between two vectors A and B can simply be expressed as follows:

$$\textsc{Sim}(A, B) = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

The dot product and product of magnitudes are:

$$\begin{aligned} A \cdot B &= \sum_{i=1}^{n} a_i b_i \\ \|A\|\|B\| &= \sqrt{\sum_{i=1}^{n}(a_i)^2}\sqrt{\sum_{i=1}^{n}(b_i)^2} \end{aligned}$$

In our case, A and B are binary *tag occurrences vectors* (i.e., they only contain 1's and 0's) over the union of the tags in both videos. Therefore, the dot product simply reduces to the intersection size of the two tag sets (i.e., $|A_t \cap R_t|$) and the product of the magnitudes reduces to the square root of the number of tags in the first tag set times the square root of the number of tags in the second tag set (i.e., $\sqrt{|A_t|}\sqrt{|R_t|}$). Therefore, the cosine similarity between a set of author tags and a set of related tags can easily be computed as:

$$\cos\theta = \frac{|A_t \cap R_t|}{\sqrt{|A_t|}\sqrt{|R_t|}}$$

Tag Set	Occ. Vector	dog	puppy	funny	cat
A_t	A	1	1	1	0
R_t	B	1	1	0	1

Table 1: Example of a tag occurrence table.

Consider an example where $A_t = \{dog, puppy, funny\}$ and $R_t = \{dog, puppy, cat\}$. We can build a simple table which corresponds to the tag occurrence over the union of both tag sets (see Table 1). Reading row-wise from this table, the tag occurrence vectors for A_t and R_t are $A = \{1, 1, 1, 0\}$ and $B = \{1, 1, 0, 1\}$, respectively. Next, we compute the dot product:

$$A \cdot B = (1 * 1) + (1 * 1) + (1 * 0) + (0 * 1) = 2$$

The product of the magnitudes can also easily be computed:

$$\|A\|\|B\| = \sqrt{3}\sqrt{3} = 3$$

Thus, the cosine similarity of the two videos is $\frac{2}{3} = 0.\bar{6}$.

3.3 Adding Related Tags

Once the related videos are sorted in decreasing cosine similarity order, we introduce tags from the related videos into the ground truth. The maximum number of characters allowed in a YouTube tag set is 120. Therefore, the tag set could theoretically contain up to 60 unique words (each word would have to be a single character). The maximum number of related videos which YouTube provides is 100. Therefore, adding all of the related tags could potentially add up to 6000 new tags. We chose to limit the upper bound by adding up to n additional unique tags from the related videos (sorted in decreasing cosine similarity order). The following function produces up to n related tags, given a challenge video's tags A, and a set of related videos R.

```
RelatedTags($A, R, n$)

1. Create an empty set, $Z \leftarrow \emptyset$.
2. Sort related videos $R$ in decreasing cosine similarity order of
   their tag sets relative to the tag set $A$.
3. For each related video $r \in R$:
   (a) If the number of new tags on the related video $r$ is $\leq$
       $n - |Z|$, add them all to $Z$.
   (b) Otherwise, while the related video $r$ has tags and while
       $|Z| < n$:
       i. Randomly remove a new tag from the remaining
          tags on the related video $r$, and add this tag to $Z$.
4. Return $Z$.
```

This technique will introduce up to n additional tags to the ground truth set. In the case where we have already generated $n - b$ related tags and the next related video contains more than b new, unique tags, we cannot add all of them without exceeding our upper bound of n tags. For example, consider the case in which we wish to generate 100 additional tags ($n = 100$) and we have already generated 99 tags. If the next related video has 4 new tags, we cannot include all of these in the new tag set, and so we randomly pick one to avoid bias.

3.4 Rejecting Frequent Tags

Security against *frequency-based attacks* (an attack where the three most frequent tags are always submitted) is maintained through the parameters F and t in the challenge generation function VideoCaptcha (see earlier in this section). F is a tag frequency distribution (see Figure 2) and t is a frequency rejection threshold. During challenge generation, after author-supplied tags and tags from related videos have been added to the ground-truth set, tags with a frequency greater than or equal to t in F are removed from the ground-truth tag set.

```
RejectFrequentTags($S$, $F$, $t$)

1. Initially, $GT \leftarrow S$.
2. For each tag $g \in GT$:
   (a) If $F(g) \geq t$, remove $g$ from $GT$.
3. Return $GT$.
```

4. GRADING FUNCTION

The generation of a video CAPTCHA (see Section 3) returns a challenge video v and a set of ground truth tags GT. Given the challenge video v, the set of ground truth tags GT, the set of user response tags U, and binary variables s and l determine whether to perform stemming (s) and/or to use inexact matching (l), we grade responses as follows:

```
Grade($v, GT, U, s, l$)

1. Preprocess the user supplied tags:
   $P \leftarrow \textsc{PreProcess}(U)$.
2. If $s = \textsc{True}$, $P \leftarrow P \cup \textsc{Stem}(P)$
3. If $l = \textsc{True}$
   (a) If $\exists t \in GT$ and $\exists p \in P$ such that
       $\textsc{NormLevenshtein}(t, p) \geq 0.8$, return Pass.
   (b) Otherwise, return Fail.
4. Otherwise,
   (a) If $GT \cap P \neq \emptyset$, return Pass.
   (b) Otherwise, return Fail.
```

Details about PreProcess, Stem and NormLevenshtein are provided in the following subsections.

4.1 Preprocessing

A *stop word list* is a list of common words which are filtered prior to processing because they are unlikely to add additional information or context. For instance, it has been shown that over 50% of all words in a typical English passage can be constructed using a list of only 135 words [13]. We chose to utilize a list of 177 stop words provided in the popular Snowball string processing language developed by Martin F. Porter. Users are prevented from submitting stop words as tags.

Prior to grading, all tags are preprocessed using the function PREPROCESS, described here. The tags are converted to lower case and punctuation is stripped to remove the effects of inconsistent capitalization or punctuation. Additionally, only the first three tags are used in grading. For example, given the input string "*Barack Obama U.S.A. man*", the preprocessor will output the set: {*barack, obama, usa*}.

4.2 Expanding Tags through Word Stemming

To increase the likelihood of passing challenges, the user-supplied tags U may be expanded through word stemming using the STEM function. A *stemmer* is an algorithm for reducing inflected or derived words to their root [18]. The root of a word is the word minus any inflectional endings, such as 's', 'ing', etc. The Porter Stemmer[2] is frequently used in information retrieval systems; it uses a deterministic set of rules to recover word roots [21].

For example, if we allow stemming and if "*dogs*" $\in U$ and "*dog*" $\in GT$, the challenge is passed (where as it otherwise might not be, depending on the set of related tags). A significant benefit of this type of expansion is that it is a repeatable, algorithmic technique which, at most, doubles the cardinality of U. If a response tag is already in the stemmed form, for example "*dog*", the stemmer will simply return the same tag.

Chew suggested the use of a thesaurus to accept synonyms in the image-based naming CAPTCHA [5] where the task was to guess the common subject of six images. For example, a video about carbonated soft drinks might be tagged as "*soda*" by one user and "*pop*" by another; using synonyms we might identify a match. To obtain synonyms, we used the freely available thesaurus from the Moby Project[3]. However, in our first user study we found that that the addition of synonyms drastically compromised security and only marginally improved usability, so we decided not to use this technique.

4.3 Allowing Inexact Matching

Many users may make spelling or typing mistakes when completing a challenge. Therefore, we can also boost usability by performing inexact matching between user tags and ground truth. We utilized the well known *string edit distance*, or Levenshtein distance [17]. The Levenshtein distance is the minimum number of operations (insertions, deletions, or substitutions) required to convert one string into the other. After computing the Levenshtein distance, we normalize it into the interval [0.0, 1.0], using the length of the longer string. Given the two strings, s_1 and s_2, we compute the normalized Levenshtein distance as follows:

$$\text{NORMLEVENSHTEIN}(s_1, s_2) = 1.0 - \frac{\text{LEVENSHTEIN}(s_1, s_2)}{\text{MAX}(|s_1|, |s_2|)}$$

As per Chew's recommendation in [5], we have chosen to define a match as a minimum normalized similarity of 0.8. This means that the larger of two strings of length $1 \leq l < 5$ are allowed no edits, strings of length $5 \leq l < 10$ are allowed one edit, strings of length $10 \leq l < 15$ are allowed two edits, etc. More generous or conservative approximate matches could be used with corresponding usability/security tradeoffs.

5. ATTACK SIMULATION

The best way to attack a video CAPTCHA using tag frequency data alone is to submit the three tags which label the largest set of videos (i.e. where the union of the video sets is the largest). Increasing usability by extending the ground truth tag set (as explained in the previous sections) will typically result in decreasing security because it allows an attacker a larger set of tags to match against.

The attack success rate may be reduced by pruning frequently occurring tags from the ground truth tag set, so that tags with an estimated frequency $\geq t$ are not accepted. However, an intelligent attacker would then select the three most frequent tags such that their estimated probabilities are slightly less than the pruning threshold (i.e. $t - \epsilon$). This is the attack which we replicated.

We performed multiple random walks to obtain a testing sample for this attack. The sample contained 5146 challenge videos, with 295,274 related videos used for challenge generation, (299,796 unique videos in total). For our experiment, we varied t in the interval $0.001 \leq t \leq 0.01$ by steps of 0.001, and the number of related tags n in the interval $0 \leq n \leq 200$ in steps of 5 tags. Note that $t = 1.0$ represents the case of no tag pruning. For each of 11 rejection threshold values, we calculated the best set of attack tags and used these to attack the 5146 videos, using the tag frequency estimate described in Section 2 (see Table 2). The results of the experiment are shown in Figure 3.

Given an attack response (A, a set of three tags) and an estimate of the frequency of tags labeling a video in the database (F), we can estimate the success rate of the attack for the control condition, where tags on a video must be matched exactly ($\hat{S}_c$):

$$\hat{S}_c(A) = \sum_{a \in A} F(a)$$

$\hat{S}_c$ is a pessimistic estimate, as it assumes that each tag labels different videos (i.e. the sets of videos labeled by each tag are disjoint).

Table 2 shows the tags used in our frequency-based attack, along with $\hat{S}_c$, and the number of tags that were pruned from our tag frequency estimate for each threshold value t. For the control con-

t	Best Attack Tags	# Pruned	$\hat{S}_c(A)$
1.0	[music, video, live]	0	0.1377
0.01	[dj, remix, vs]	37	0.0291
0.009	[girl, school, el]	44	0.0256
0.008	[animation, michael, star]	49	0.0237
0.007	[concert, news, day]	67	0.0207
0.006	[fantasy, dragon, rb]	92	0.0179
0.005	[islam, humor, blues]	129	0.0148
0.004	[real, bass, 12]	184	0.0120
0.003	[uk, spoof, pro]	302	0.0090
0.002	[seven, jr, patrick]	570	0.0060
0.001	[ff, kings, ds]	1402	0.0030

Table 2: Tags used in our frequency-based attack. For each pruning threshold, we show the attack tags used, their estimated success rate for the control condition ($\hat{S}_c$), and the number of tags pruned from our tag frequency estimate.

[2] Online at `http://tartarus.org/~martin/PorterStemmer/`

[3] Online at `http://www.gutenberg.org/etext/3202`

dition, $\hat{S}_c$ provided estimates that were always slightly larger than the observed success rates shown in Figure 3. As tags are added however, the estimate becomes increasingly optimistic and unreliable.

We observed that, in general, a smaller pruning threshold reduces the success of the attack and a larger number of related tags increases the success of the attack. Figure 3 plots the attack success rate as the tag rejection threshold t and the number of related tags n is varied. There is a nearly linear trade-off between t and n for the attack success rate (see the roughly linear cut across the colored tiles in Figure 3). Note that the attack success rate of the control (no pruning and no additional related tags) is approximately 13%.

6. USER STUDIES

To analyze the usability of our video CAPTCHA, we conducted two anonymous, online user studies. IP addresses of participants were recorded to protect against multiple responses from a single user but were discarded during analysis. Friends, family, and colleagues were invited to participate, and a college-wide invitation was also emailed to students. The majority of our participants were males in the 18-24 age group with at least some college experience and were familiar with online videos. We acknowledge the fact that participants of this demographic may perform better than other demographics (for example, elderly people with little familiarity with online videos). Complete demographics are presented in Table 3.

6.1 User Study 1: Video Tagging

To study tagging behavior and to choose appropriate parameters for our grading function, we first conducted a user study in which we had participants tag a set of 20 randomly ordered videos with 3 unique tags each. The videos were manually selected to ensure appropriate content (this is a modification of the first step in the VIDEOCAPTCHA function for generating challenges).

In order to familiarize the participants with the task, two practice videos were shown to the participants, one of which was particularly challenging due to the use of a foreign language in the video. The tags from the practice videos were recorded, but were not used during analysis. Participants were instructed to *tag* each video with three unique, non-stop words. The participants were not required to watch the entire video before submitting their tags. We recorded the time it took the participants to complete each challenge using both client-side Javascript and server-side logs analysis. The recorded times included the time needed to: 1) watch some (or all) of the video, 2) think of three reasonable tags, 3) type their responses, and 4) press the submit button. The participants were then instructed to *rate* how difficult it was to tag the video using the following scale (both numbers and descriptions were shown): 5 (Great Effort), 4 (Moderate Effort), 3 (Some Effort), 2 (Little Effort), and 1 (No Effort).

After completing the *tag and rate task* for each of the 20 videos, the participants were asked the following questions in an exit survey:

1. Which task do you enjoy completing more?
   (a) Guessing an appropriate tag for a video
   (b) Transcribing a string of distorted text
   (c) No preference
2. Which task do you find faster to complete?
   (a) Guessing an appropriate tag for a video
   (b) Transcribing a string of distorted text
   (c) Neither

	User Study 1	User Study 2
Age group		
18-24	74.82% (107)	77.71% (143)
25-34	13.28% (19)	11.95% (22)
35-44	3.496% (5)	4.891% (9)
45-54	4.195% (6)	2.173% (4)
55-65	2.797% (4)	2.717% (5)
65-74	0.699% (1)	0.543% (1)
75+	0.699% (1)	0.0% (0)
Gender		
Male	79.02% (113)	83.69% (154)
Female	20.97% (30)	16.30% (30)
Highest level of education completed		
Some High School	0.0% (0)	0.543% (1)
High School	2.797% (4)	4.891% (9)
Some College	46.85% (67)	47.82% (88)
Associate's	4.895% (7)	6.521% (12)
Bachelor's	33.56% (48)	30.43% (56)
Master's	11.18% (16)	4.347% (8)
Professional Degree	0.699% (1)	0.0% (0)
PhD	0.0% (0)	5.434% (10)
Number of online videos watched per month		
0-4	17.48% (25)	17.93% (33)
5-14	30.76% (44)	30.43% (56)
15-30	23.07% (33)	20.65% (38)
31+	28.67% (41)	30.97% (57)
Have you ever uploaded a video before?		
Yes	60.83% (87)	64.67% (119)
No	39.16% (56)	35.32% (65)
Which do you find more enjoyable?		
Transcribing Distorted Text	15.38% (22)	20.10% (37)
Tagging a Video	61.53% (88)	58.15% (107)
No Preference	23.07% (33)	21.73% (40)
Which do you think is faster?		
Transcribing Distorted Text	64.33% (92)	59.78% (110)
Tagging a Video	19.58% (28)	27.17% (50)
Neither	16.08% (23)	13.04% (24)

Table 3: Participant demographics and exit survey responses.

See Table 3 for the results of the exit survey. The participants were also given a chance to provide additional comments and a field to enter their email address if they wished to be contacted again in the future.

6.2 User Study 2: Video CAPTCHAs

This study was nearly identical to the first with the following modifications:

- Users were told whether they had *passed* or *failed* each challenge.
- Challenge videos were selected using a random walk with manual filtering.
- An open source flash video player was used to stream the videos instead of the YouTube.com player to mask the ID of the challenge video.

An effort was made to keep the user interface similar across both the user studies. In the first user study, participants were instructed to submit three tags for each video (the challenges were not graded). However, in the second user study, the instructions emphasized that the participants were completing a challenge, or

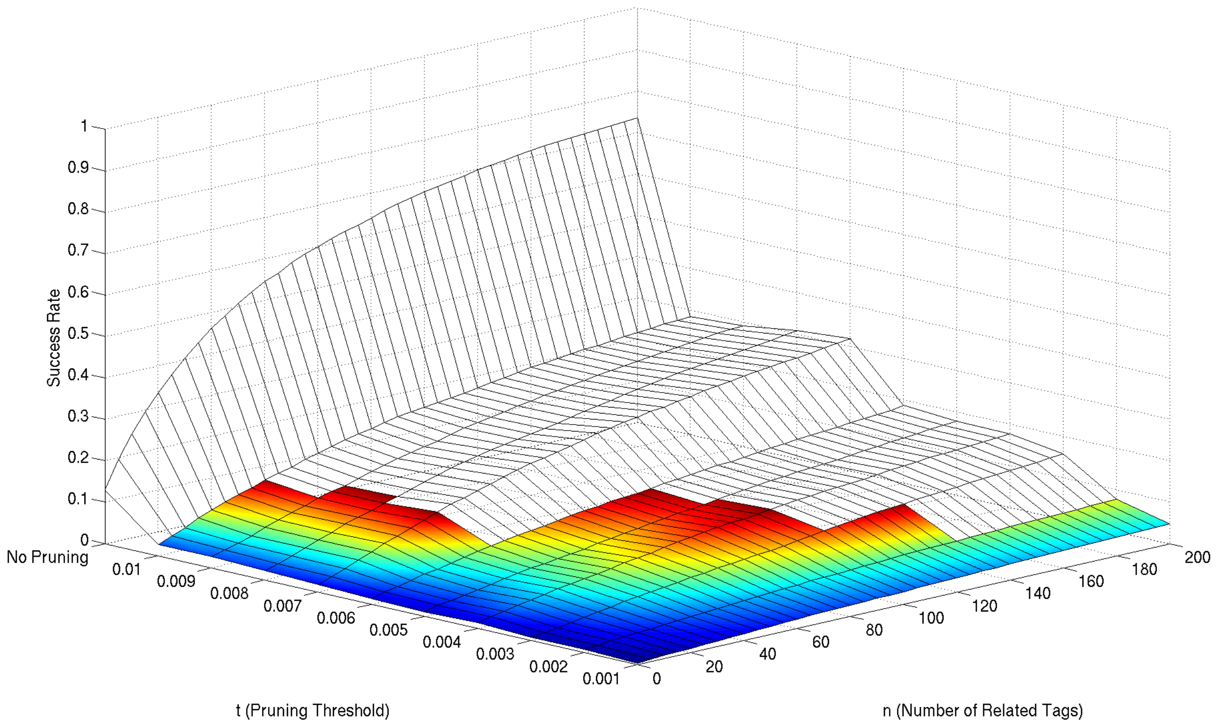


Figure 3: Success rates for frequency-based attack on 5146 videos (no stemming and exact matching of tags). The control is located at the leftmost corner (0 related tags added, no pruning, and an attack success rate of 12.86%). If all four corners of a tile have equal or better security than the control, the tile is shaded. Tags used for each pruning threshold differ (see Table 2).

test, which would be graded.

Unlike the first user study, the 20 challenge videos were selected using a random walk (see Section 3). However, the videos were manually inspected for inappropriate content; we rejected two videos which had questionable adult content and five videos which contained strictly non-English tags. Other than that, all other videos, regardless of length, content, or rating were allowed.

In this user study, we were also concerned with people trying to defeat our video CAPTCHA. We pre-fetched the video files from YouTube and streamed them from our own servers using a free open source flash video player. If we had chosen to use the YouTube flash video player, the participants could either view the page's source to expose the YouTube video ID or click on the player itself to be redirected to the video on YouTube.com (which would reveal the author's tags).

In order to inform the user whether they passed or failed the challenges, we had to grade responses. The selection of parameters for the grading function was based on an analysis of the human success rates (S_h) in the first user study, and the attack success rates (S_a) in our simulated attack. We provided feedback using the most usable generation parameters for VIDEOCAPTCHA that did not rely on stemming or inexact matching but whose parameters still provided better or equal security than the control. We chose to use this parameter setting so as to avoid discouraging participants, while using the strictest grading protocol (exact matching of tags).

We will define our parameter space τ as a 4-tuple $\langle n, t, s, l \rangle$. Our control (no related tags, no pruning, no stemming, and exact matching) is $\tau_c = \langle 0, 0, \text{FALSE}, \text{FALSE} \rangle$. From our first user study, we observed a human success rate for the control condition of $S_h(\tau_c) = 0.75$. In the attack simulation, we observed a success rate of $S_a(\tau_c) = 0.1286$. We fixed s and t as false, and then searched over n and t to find a condition that maximized the human success rate, while insuring that the attack success rate was no better than the control condition. We found that $\hat{\tau} = \langle 110, 0.005, \text{FALSE}, \text{FALSE} \rangle$ satisfied these criteria. The second user study was conducted using this parameter setting ($\hat{\tau}$).

We computed the effect of varying the generation and grading parameters on human success rates ($S_h(\tau)$) in a post-processing fashion. These results are summarized in Table 8; complete results may be found in [15].

6.3 User Study Results

A set of three metrics for evaluating the usability of CAPTCHAs are presented in [29]. To assess *errors*, we observed the human success rates (measuring how accurately users can complete the task). To evaluate *efficiency*, we measured user response time, and to evaluate *satisfaction* we measured the *perceived difficulty* of the users using a 1-5 scale.

The median completion time for our task was 17 seconds (see Table 6). The mode of the perceived difficulty for our CAPTCHA was 2 (see Table 7). As expected, the difficulty ratings and the median completion times are strongly correlated (the Pearson's coefficients were $\rho = 0.9492$ and $\rho = 0.9898$ for the first and second user studies, respectively). Detailed completion times and difficulty ratings can be found in Tables 4 and 5.

We also allowed participants to provide comments on the experiment. Here are a few of the comments we received:

- "You overestimate the public's ability to spell"
- "Deciphering the scrambled text of some sites is almost impossible, and it has stopped me from entering several online

contests that were using it."

- "The only reason I prefer distorted text to video tagging is the time it takes."
- "This is a great idea, and it's ... more fun than [a popular text-based CAPTCHA]"
- "CAPTCHAs have become too distorted to read. It usually takes me three or four tries to get one right!"
- "Some videos were very easy to figure out. Others were cryptic."

Difficulty	User Study 1	User Study 2
1	17.344	13.449
2	20.696	15.668
3	24.640	20.579
4	29.334	24.967
5	42.798	30.967

Table 4: Median completion time (in seconds) grouped by perceived difficulty ratings.

Difficulty	User Study 1	User Study 2
1	27.657% (791)	23.315% (858)
2	41.118% (1176)	37.853% (1393)
3	22.972% (657)	26.413% (972)
4	6.643% (190)	9.701% (357)
5	1.608% (46)	2.717% (100)

Table 5: Distribution of perceived difficulty ratings.

	User Study 1	User Study 2
Mean (μ)	29.688	22.038
StdDev (σ)	34.746	23.578
Median	20.642	17.062

Table 6: Completion time statistics (in seconds).

In both the attack simulation and user studies, we varied the challenge generation parameters n and t in the ranges: $t \in \{0.001, 0.002, \ldots, 0.01, 1.0\}$ and $n \in \{0, 5, \ldots, 195, 200\}$. As the pruning threshold t decreases, more tags are pruned from the ground truth set and the human success rate decreases (see Figures 4 and 5). The human success rate increases as the number of additional related tags n increases.

The human success rate for the control in the first user study is located at the leftmost corner of Figure 4. The addition of only 5 related tags improves the usability of the CAPTCHA approximately 6% regardless of the pruning level. While many of the parameter settings yield a higher human success rate than the control, a parameter setting is generally only useful if it does not have a higher attack success rate than the control.

The human success rates from the first user study with no stemming and exact matching are plotted in Figure 4 while the corresponding human success rates from the second user study are plotted in Figure 5. A comparison of the human success rates in the second user study, and attack success rates in the attack simulation over the parameter space are presented in Table 8.

	User Study 1	User Study 2
Mean (μ)	2.1343	2.3066
StdDev (σ)	0.9482	1.0181
Mode	2	2

Table 7: Perceived difficulty rating statistics.

	Parameter set (τ)				Success Rates		
Condition	n	t	s	l	$S_h(\tau)$	$S_a(\tau)$	$Gap(\tau)$
Control	0	1.0			0.6973	0.1286	0.5687
Most Usable	100	0.006			0.8828	0.1220	0.7608
Most Secure	30	0.002			0.7502	0.0239	0.7263
Largest Gap	45	0.006			0.8682	0.0750	0.7931
Most Usable	100	0.006	✓		0.8896	0.1226	0.7670
Most Secure	25	0.002	✓		0.7548	0.0209	0.7339
Largest Gap	45	0.006	✓		0.8755	0.0750	0.8005
Most Usable	100	0.006		✓	0.9000	0.1280	0.7719
Most Secure	15	0.003		✓	0.7671	0.0233	0.7438
Largest Gap	25	0.006		✓	0.8611	0.0526	0.8084
Most Usable	90	0.006	✓	✓	0.9019	0.1263	0.7755
Most Secure	15	0.003	✓	✓	0.7690	0.0237	0.7453
Largest Gap	25	0.006	✓	✓	0.8649	0.0526	0.8122

Table 8: Human (S_h) vs. attack (S_a) success rates, for the second user study (Section 6.2) and attack simulation (Section 5). The parameter space for τ includes the number of related tags added (n), the tag frequency rejection threshold (t), and whether word stemming (s) and approximate tag matching (l) were allowed. $Gap(\tau)$ is the difference between the human and attack success rates for parameter set τ.

As Table 8 indicates, the human success rate on the control is only 69.73%, and the attack success rate is 12.86%. For all combinations of whether stemming (s) and inexact matching (l) are used, the table provides the most usable and secure related tag (n) and frequency threshold (t) values where the human rate does not drop below the control, and the attack rate does not exceeed the control. For parameter set $\tau = \langle 90, 0.006, \text{TRUE}, \text{TRUE} \rangle$ we were able to boost the usability (S_h) to over 90% and even increase security slightly (decreasing S_a by 0.23% from the control).

Table 8 illustrates that our video CAPTCHA can be parameterized to allow for different tradeoffs between usability and security. As one would expect, fewer tags need to be added to increase human success rates over the control if inexact matching is permitted ($l = \text{TRUE}$). The largest gap between human and attack performance is observe in the bottom entry of Table 8, where both stemming and inexact matching are used, with an 86% human success rate and only a 5% attack success rate.

In the first user study, we were able to outperform the control by including as few as 5 additional related tags. However, in the second user study, we must include 10 or more related tags for all $t < 1.0$. In the first user study, we were able to reduce the attack success rate to nearly 1.2% (adding 5 related tags, pruning at 0.003, using stemming and exact matching). However, in the second user study, the best security level which we were able to achieve while maintaining the control success rate for humans was 2.1% (adding 25 related tags, pruning at 0.002, using stemming and exact matching).

The human success rates are slightly lower in the second user study than in the first user study. This can be explained by the sampling method used: the videos used for the first user study were manually selected while the videos used in the second user study

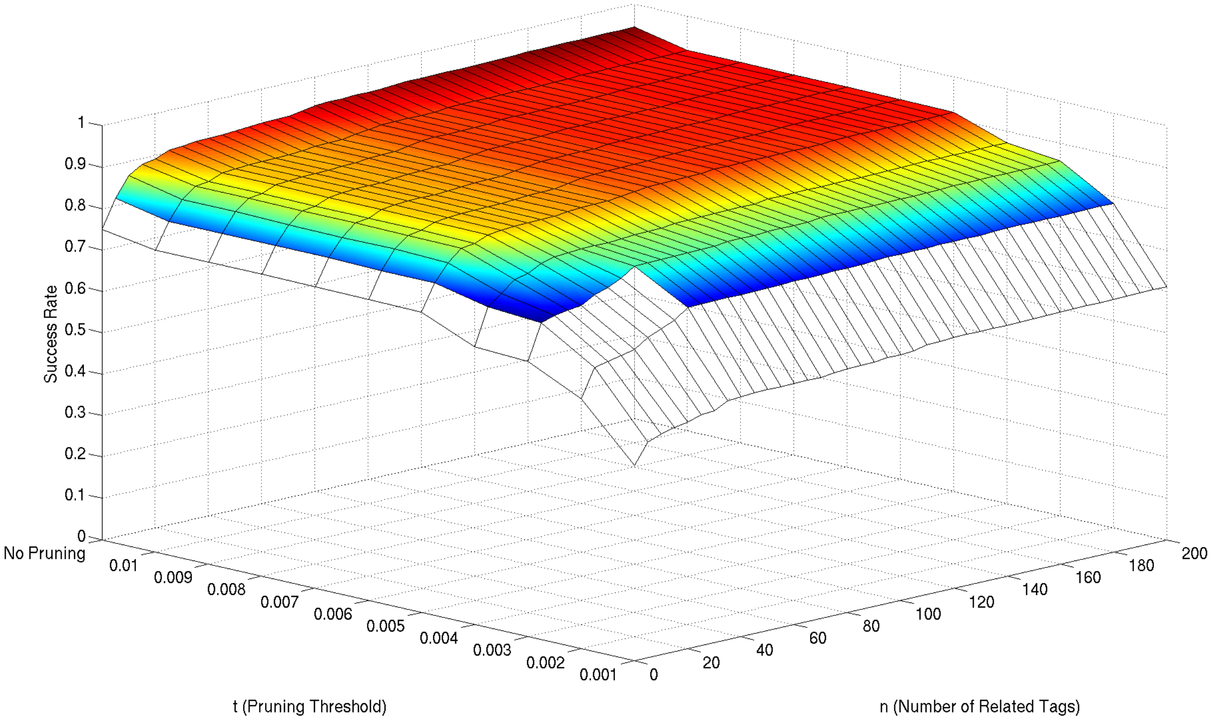


Figure 4: The human success rates from the first user study with no stemming and exact matching. The control is located at the leftmost corner (0 related tags added, no pruning, and a human success rate of 75%). If all four corners of a tile have better usability than the control, the tile is shaded.

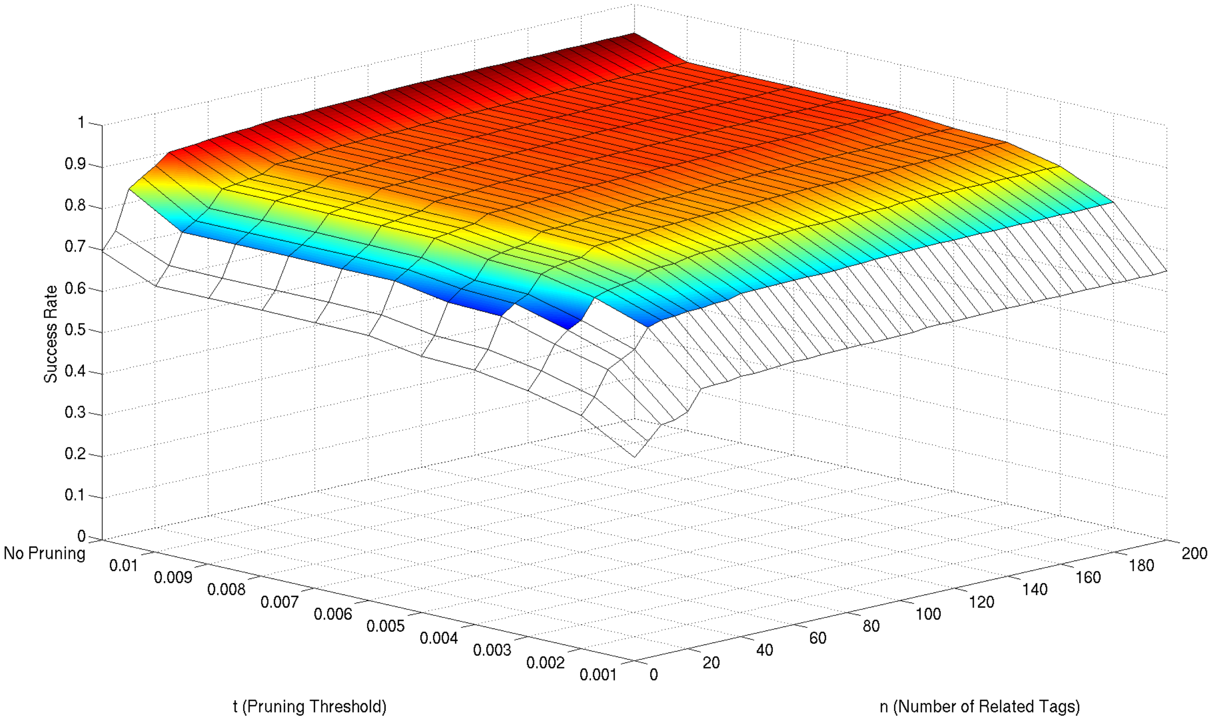


Figure 5: The human success rates from the second user study with no stemming and exact matching. The control is located at the leftmost corner (0 related tags added, no pruning, and a human success rate of 69.73%). If all four corners of a tile have better usability than the control, the tile is shaded.

were randomly selected. The trends and patterns of the human success rates are uniform across both samples as shown in Figure 4 and Figure 5. The other conditions (using/not using stemming and/or exact matching) also exhibit similar trends to that of the samples presented (see Appendix C of [15]).

7. CONCLUSION AND FUTURE WORK

We have proposed the first CAPTCHA that uses video understanding to distinguish between humans and machines. It has nearly all of the desirable properties outlined in the introduction: challenges can be semi-automatically generated, graded automatically, the challenge design and data are publicly available, and challenge generation and grading may be parameterized in order to achieve a desired balance between usability and security. Using a video database known to be free of inappropriate content, our video CAPTCHA has all four desirable properties (no human inspection is needed, and generation becomes fully automatic). The results of our attack estimate and second user study suggest that our video CAPTCHAs have comparable usability and security to existing CAPTCHAs (see Table 9). In fact, more than half (60%) of the participants in our second user study indicated that they found the video CAPTCHA more enjoyable than traditional CAPTCHAs in which distorted text must be transcribed. These results are encouraging and suggest that video CAPTCHAs may provide a viable alternative to text-based CAPTCHAs.

		Success Rates	
CAPTCHA	Type	Human	Machine
Microsoft	Text-based	0.90 [3]	0.60 [28]
Baffletext	Text-based	0.89 [4]	0.25 [4]
Handwritten	Text-based	0.76 [23]	0.13 [23]
ASIRRA	Image-based	0.99 [6]	0.10 [9]
Video	$\tau = \langle 15, 0.003, \mathrm{T}, \mathrm{T} \rangle$	0.77	0.02
	$\tau = \langle 25, 0.006, \mathrm{T}, \mathrm{T} \rangle$	0.86	0.05
	$\tau = \langle 90, 0.006, \mathrm{T}, \mathrm{T} \rangle$	0.90	0.13

Table 9: A comparison of human and attack success rates for our video CAPTCHA (for different parameter settings) with other CAPTCHAs.

In this first investigation, the security of the video CAPTCHA was only tested with a tag frequency-based attack. We acknowledge that other attacks may perform better. For example, computer vision could be used to locate frames with text-segments in them, and then detect and submit words using optical character recognition (*OCR*). If videos were pre-scanned for text content, text could be detected in a pre-processing phase. These words could then be marked as *taboo tags* (similar to how taboo tags are used in the ESP game [27]), or be weighted down (requiring at least one additional matching tag). Another attack could use Content-based Video Retrieval systems to locate videos with similar content (and then submit their tags). Audio analysis might also give an indication as to the content of the video.

It would be interesting to compare the usability of the video CAPTCHA under all combinations of audio and video being present or absent. Such a study would help us evaluate the usability of our video CAPTCHA for individuals with limited vision or hearing abilities. The current CAPTCHA was tested only for English-speaking users located in the United States, trying to match English tags. Another interesting experiment would be to see if using dictionaries from other languages to seed random walks during generation would yield usable challenges for other geographic regions and cultures.

Finally, the tag-based challenge generation technique presented is not video-specific. We can imagine CAPTCHAs being developed which utilize social structure in other types of tagged data, for example using images from Flickr.com. An additional study could compare the usability of our video CAPTCHA to one where only a single frame of the video is shown to the user. This would test the hypothesis that tagging full motion video is easier for users than tagging individual video frames (still images).

8. ACKNOWLEDGMENTS

We wish to thank Ben Hughes for help with implementing the user interface, and the numerous volunteers that participated in our user studies. This work was partially supported by Xerox Corporation through a University Affairs Committee grant. Much of this work was completed as part of the first author's Master's thesis [15].

9. REFERENCES

[1] H. S. Baird and K. Popat. Human Interactive Proofs and Document Image Analysis. In *Proc. IAPR DAS 2002*, ACM Press (2002), 507–518.

[2] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. In *Proc. IMC 2007*, ACM Press (2007), 1–14.

[3] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski. Building Segmentation Based Human-friendly Human Interaction Proofs (HIPs). In *Proc. HIP 2005*, LNCS (2005), 1–26.

[4] M. Chew and H. S. Baird. Baffletext: A Human Interactive Proof. In *Proc. DRR 2003*, IST/SPIE (2003), 305–316.

[5] M. Chew and J. D. Tygar. Image Recognition CAPTCHAs. In *Proc. ISC 2004*, LNCS (2004), 268–279.

[6] J. Douceur, J. Elson, J. Howell, and J. Saul. Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization. In *Proc. CCS 2007*, ACM Press (2007), 366–374.

[7] G. Geisler and S. Burns. Tagging Video: Conventions and Strategies of the YouTube Community. In *Proc. JCDL 2007*, ACM/IEEE (2007), 480–480.

[8] P. B. Godfrey Text-based CAPTCHA algorithms. In *Proc. HIP 2002*.

[9] P. Golle. Machine Learning Attacks Against the ASIRRA CAPTCHA. In *Proc. CCS 2008*, ACM Press (2008), 535–542.

[10] L. A. Goodman. Snowball sampling. *The Annals of Mathematical Statistics 32*, 1 (1961), 148–170.

[11] R. Gossweiler, M. Kamvar and S. Baluja. What's Up CAPTCHA? A CAPTCHA Based on Image Orientation. In *Proc. WWW 2009*, ACM Press (2009), 841–850.

[12] M. J. Halvey and M. T. Keane. Analysis of Online Video Search and Sharing. In *Proc. Hypertext 2007*, ACM Press (2007), 217–226.

[13] G. W. Hart. To Decode Short Cryptograms. *Communications of the ACM 37*, 9 (1994), 102–108.

[14] A. Kerckhoffs. La Cryptographie Militaire. *Journal des Sciences Militaires 9*, (1883), 161–191.

[15] K. A. Kluever. Evaluating the Usability and Security of a Video CAPTCHA. Master's thesis, Rochester Institute of Technology, 2008.

[16] G. Kochanski, D. P. Lopresti and C. Shih. Using a Text-to-Speech Synthesizer to Generate a Reverse Turing Test. In *Proc. ICTAI 2003*, IEEE Press (2003), 226-232.

[17] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady 10*, (1966), 707–710.

[18] J. B. Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics 11*, (1968), 22–31.

[19] M. Naor. Verification of a human in the loop or Identification via the Turing Test. *Unpublished manuscript*, (1996).

[20] J. C. Paolillo. Structure and Network in the YouTube Core. In *Proc. HICSS 2008*, IEEE Press (2008), 156–166.

[21] M. F. Porter. An Algorithm for Suffix Stripping. *Program 14*, 3 (1980), 130–137.

[22] Y. Rui and Z. Liu. ARTiFACIAL: Automated Reverse Turing test using FACIAL features. *Multimedia Systems Journal 9*, 6 (2004), 493–502.

[23] A. Rusu. *Exploiting the Gap in Human and Machine Abilities in Handwriting Recognition for Web Security Applications*. PhD thesis, University of New York at Buffalo, 2007.

[24] A.M. Turing. Computing Machinery and Intelligence. *Mind 59*, 236 (1950), 433–460.

[25] C. van Rijsbergen. *Information Retrieval, Second edition*. Butterworth-Heinemann Ltd, London, UK, 1979.

[26] L. von Ahn, M. Blum, and J. Langford. Telling Humans and Computers Apart Automatically. *Communications of the ACM 47*, 2 (2004), 56–60.

[27] L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. In *Proc. CHI 2004*, ACM Press (2004), 319–326.

[28] J. Yan and A. S. E. Ahmad. A Low-cost Attack on a Microsoft CAPTCHA. In *Proc. CCS 2008*, ACM Press (2008), 543–554.

[29] J. Yan and A. S. E. Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proc. SOUPS 2008*, ACM Press (2008), 44–52.