2008

Evaluating the usability and security of a video CAPTCHA

Kurt Alfred Kluever

The Thesis Committee for Kurt Alfred Kluever
certifies that this is the approved version of the following thesis:

Evaluating the Usability and Security of a Video CAPTCHA

Committee:

Dr. Richard Zanibbi, Supervisor

Dr. Roxanne L. Canosa, Reader

Dr. Zack J. Butler, Observer

Evaluating the Usability and Security of a Video CAPTCHA

by

Kurt Alfred Kluever, B.S.

Thesis

Presented to the Faculty of the Graduate School of

Rochester Institute of Technology

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

Rochester Institute of Technology

August 2008

For Amy

Acknowledgments

I gratefully acknowledge financial support from the RIT Department of Computer Science through a Teaching Assistantship, and Xerox Corporation through a University Affairs Committee (UAC) grant.

Dr. Richard Zanibbi has provided me with excellent supervision. I am extremely grateful for his wisdom, creativity, and enthusiasm. It has been a pleasure working with him over the past year.

Thanks to YouTube.com for providing public API access to their enormous amounts of valuable data. I appreciate the time and effort of the 533 anonymous volunteers who participated in my user studies and Ben Hughes for helping me develop the data collection interface.

Thanks to all of my friends for helping me focus on more important stuff in life than my thesis, like six dollar pitcher nights.

None of this would have been possible without the loving support and encouragement of my amazing parents. I am eternally thankful to them for all that they have helped me with over the years.

Lastly, I thank my best friend and soulmate Amy for her endless love, encouragement, and patience. It is to her that I dedicate this thesis.

Kurt Alfred Kluever

Rochester Institute of Technology
August 2008

Abstract

Evaluating the Usability and Security of a Video CAPTCHA

Kurt Alfred Kluever, M.S.

Rochester Institute of Technology, 2008

Supervisor: Dr. Richard Zanibbi

A CAPTCHA is a variation of the Turing test, in which a challenge is used to distinguish humans from computers ('bots') on the internet. They are commonly used to prevent the abuse of online services. CAPTCHAs discriminate using hard artificial intelligence problems: the most common type requires a user to transcribe distorted characters displayed within a noisy image. Unfortunately, many users find them frustrating and break rates as high as 60% have been reported (for Microsoft's Hotmail).

We present a new CAPTCHA in which users provide three words ('tags') that describe a video. A challenge is passed if a user's tag belongs to a set of automatically generated ground-truth tags. In an experiment, we were able to increase human pass rates for our video CAPTCHAs from 69.7% to 90.2% (184

participants over 20 videos). Under the same conditions, the pass rate for an attack submitting the three most frequent tags (estimated over 86,368 videos) remained nearly constant (5% over the 20 videos, roughly 12.9% over a separate sample of 5146 videos). Challenge videos were taken from YouTube.com. For each video, 90 tags were added from related videos to the ground-truth set; security was maintained by pruning all tags with a frequency $\geq 0.6\%$. Tag stemming and approximate matching were also used to increase human pass rates. Only 20.1% of participants preferred text-based CAPTCHAs, while 58.2% preferred our video-based alternative.

Finally, we demonstrate how our technique for extending the ground truth tags allows for different usability/security trade-offs, and discuss how it can be applied to other types of CAPTCHAs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHAs) are a class of automated challenges used to differentiate between legitimate human users and computer programs ('bots') on the internet. CAPTCHAs have many practical security applications, including preventing the abuse of online services such as free email providers.

In order to evaluate the success of a CAPTCHA, a list of desirable properties must first be established. Building off of recommendations from researchers at Carnegie Mellon University [102], the Palo Alto Research Center [14], and Microsoft Research [82], the following set of amended properties are proposed:

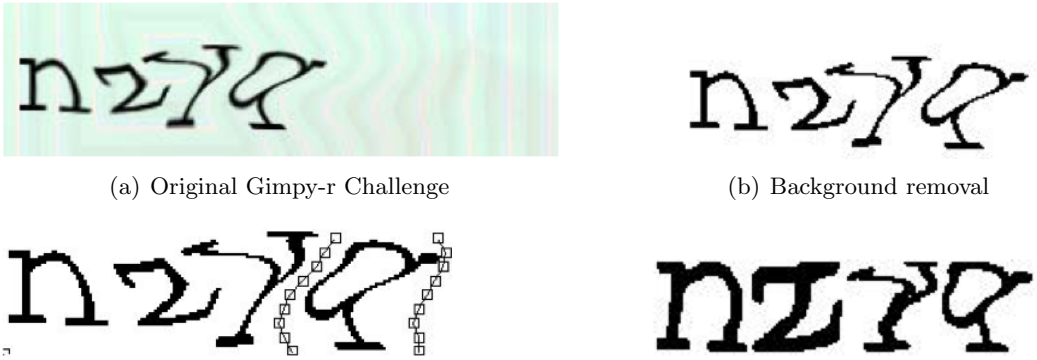
Automated It must be possible for a machine to automatically generate and grade the challenges.

Open The database(s) and algorithm(s) used to generate the challenges must be publicly available to ensure that the difficulty of the CAPTCHA stems from the underlying hard artificial intelligence problem and not a secret algorithm. This is in accordance with Kerckhoffs' Principle, which states that a system should remain secure even if everything about the system (except the key, or in the case of CAPTCHAs, the solution) is public knowledge [55].

Usable Challenges should be easily and quickly solved by humans. The test should be as independent as possible of the user's language, physical location, educational background, and perceptual disabilities.

Secure The underlying AI problem must be a well-known and well-studied problem where the best existing techniques are weaker than humans.

Current implementations are based on hard artificial intelligence problems such as natural language processing [105] (linguistic CAPTCHAs), character recognition [31] (text-based CAPTCHAs), image understanding [32] (image-based CAPTCHAs), and speech recognition [57] (audio-based CAPTCHAs). Most commercial implementations require the user to transcribe a string of distorted characters within a noisy image. CAPTCHAs which rely on a human's ability to recognize distorted characters have been broken through techniques such as shape matching [68] [97], distortion estimation [69], and even simple pixel counting [109]. For example, Moy *et al.* wrote a program to remove the distortion and solve EZ-Gimpy challenges with a 99% success rate and Gimpy-r challenges (see Fig. (1.1)) with a 78% success rate [69]. The need for a more secure yet user friendly CAPTCHA arises.



(a) Original Gimpy-r Challenge (b) Background removal

(c) Distortion estimation (d) After distortion removal

Figure 1.1: Steps in breaking the Gimpy-r challenges from [69].

While the current trend in online media is towards streaming video, the use of video in CAPTCHA challenges has not yet been explored. Chew has presented image-based CAPTCHAs which display a set of 6 images to a user and require them to correctly label the common subject [32]. In this work, we present a new CAPTCHA in which users provide three words ('tags') that describe a video. A challenge is passed if a user's tag belongs to a set of automatically generated ground truth tags. This is similar to an offline version of von Ahn's popular ESP Game, in which users are randomly paired and shown a common image [103]. Their task is to agree on an appropriate label for the image. In our video CAPTCHA, one user (the author) has provided labels for the video when they uploaded it. It is up to

the user taking the video CAPTCHA to guess one of these labels.

For our control, we will consider the case where the set of ground truth tags consist only of the author-supplied tags. In other conditions, we extend the ground truth tag set by adding tags from related videos. Tag stemming [64] [76] and approximate matching [59] are also used to increase usability. A stemmer reduces a word to it's base form; for example 'dogs' becomes 'dog'. Inexact matching is performed to allow for minor spelling errors and works by ensuring that two words have a minimum normalized Levenshtein distance of 80%. To maintain security, frequently occurring tags are pruned at various thresholds to reduce the success rates of frequency-based attacks. In our frequency-based attack, the three most frequent tags below the pruning threshold are submitted as the answer.

Thesis Statement One can increase usability while maintaining security in a video CAPTCHA by intelligently extending the set of user-supplied and ground truth tags.

To evaluate the usability of our video CAPTCHA, we have conducted an experiment in which 184 participants completed 20 video CAPTCHAs. We were able to increase the human pass rate from 69.7% to 90.2% without compromising the security of the system (we maintained an estimated 12.9% attack success rate). To evaluate the security of our video CAPTCHA, we performed a tag frequency-based attack as described above.

Let us now evaluate video CAPTCHAs with respect to the proposed list of desirable properties. It is easy to show that video CAPTCHAs satisfy two of the properties (automated and open). The remaining two properties (usable and secure) must be evaluated through experiments.

Automated The challenges can be easily generated by harvesting videos and tags from online video databases, or a large, local, dynamic database of videos may be used, if available. A human is needed in the loop to filter out inappropriate content. The set of ground truth tags can also be easily generated using the methods detailed in this thesis. While video selection is partially manual, grading is entirely automatic.

Open Challenges are generated using videos from publicly available video databases (YouTube). Algorithms for generating and grading challenges are documented

in this thesis.

Usable An experiment was conducted in which 184 participants (mostly students, faculty, and staff from the Rochester Institute of Technology) completed 20 video CAPTCHAs (see Section 3.8). International versions could be developed by serving appropriate videos from localized versions of the online video database.

Secure In this thesis, we present pass rates for a tag frequency-based attack (see Section 3.7). Other attacks may be possible (see next section).

1.1 Limitations and Assumptions

We acknowledge the following limitations of our proposed video CAPTCHA:

1. The current implementation is language dependent. We have ensured that all videos contain only English tags.
2. Streaming videos from the online repository exposes the ID of the video in the source of the page.
3. The usability and security analyses presented in this thesis are a preliminary study, not an exhaustive analysis.

We offer the following suggestions to mitigate the effects of the above limitations:

The first limitation could be addressed by obtaining videos that have been tagged in foreign languages and using a different dictionary to start the random walks. We have observed that these videos do exist (in fact, we had to filter several of these videos out from our final experiment). An algorithm could determine the language of the tags and serve appropriate content to users from other countries.

To overcome the second limitation, we could proxy the videos. Streaming the content directly from the video content providers to the user exposes the source of the video stream. An attacker would easily be able to grab the author supplied tags from the original page. Proxying the video would require the CAPTCHA service to download a large set of challenge videos ahead of time. This could be accomplished by keeping a queue of videos ready to be served. When a CAPTCHA challenge is requested, the video would be removed from the queue and streamed from the CAPTCHA service instead of from the video content provider. The service would then replenish the queue when it dropped below a certain size.

The third limitation is due to the infancy of this project. We encourage future usability and security analyses. Specifically, we have not evaluated the effectiveness of any computer vision-related attacks. Such attacks are certainly worthy of investigation but are outside the scope of this thesis. We welcome others to investigate this avenue of research.

1.2 Contributions

This work makes the following contributions:

1. The first CAPTCHA to use video understanding as the underlying hard artificial intelligence problem.
2. The ability to semi-automatically generate challenges (using a human to filter inappropriate and non english-tagged videos).
3. Empirical evidence from a preliminary study supporting our hypothesis that the usability of a video CAPTCHA can be boosted without compromising the security of the system (see Table (4.8)).
4. The ability to parameterize the grading of video CAPTCHAs to balance usability and security. Increasing the number of related tags increases usability (and decreases security). Decreasing the pruning threshold increases security (and decreases usability).

1.3 Chapter Summary

Chapter 2 summarizes previous work published in the field of CAPTCHAs. The work has been broken down by type of CAPTCHA: linguistic, text-based, image-based, and audio-based.

Chapter 3 details the three experiments performed and four video samples collected (see Table (3.1)). In the first experiment (Experiment 1: Video Tagging), we analyzed how parameter selection influenced the human success rates on our video CAPTCHA in a user study. In the second experiment (Experiment 2: Attack Estimation), we estimated the attack success rates for a frequency-based attack against a large sample of videos. In the third experiment (Experiment 3: Video CAPTCHAs), we deployed a video CAPTCHA using parameters chosen based on

the results from Experiments 1 and 2 to validate the human success rates and attack success rates.

Chapter 4 presents and discusses the results of the three experiments. It explains the effect of the parameters on the usability and security of our video CAPTCHA. And finally, it compares the human success rates and attack success rates of our video CAPTCHA to existing CAPTCHAs.

Chapter 5 summarizes the contributions and outcomes of this work and provides suggestions for future work in this area.

Chapter 2

CAPTCHAs: A Literature Review

In this chapter, we motivate the development of CAPTCHAs, list some common applications of CAPTCHAs, and provide a detailed history of research in the area.

2.1 Early Development

Early internet hackers of the 1980s are often credited with the idea of obscuring text to foil content filtering devices. When trying to obscure sensitive data online, they would use the simple technique of substituting numbers for text (e.g., $I \rightarrow 1$, $A \rightarrow 4$, etc.). The algorithms they used were extremely rudimentary but would fool content filters and web crawlers. The technique was originally used to get around profanity filters which were installed on online commenting systems and forums. These filters were extremely simple and banned a predefined set of inappropriate words. The end result was something that a fellow human could read with a minor degree of difficulty, but the existing (bot) filters would miss. This concept eventually evolved into what is currently known as "l33tspeak" (elite speak).

With the massive amounts of *spam* (unsolicited, junk email) being delivered to inboxes around the world, people have been concerned with publishing their email address online. *Spam bots* can process thousands of web pages per hour, scanning for email addresses in clear text. Many people have resorted to attempting to fool spam bots by posting their email addresses in a human readable but unconventional

form. For example, the text string "kurt AT kloover DOT com" is not easily parsed or understood by most spam bots which crawl for email addresses in conventional forms. However, humans can easily identify the intended email address. This solution is not perfect, as any text-based data that humans can easily read can also be easily read by a machine. A simple set of regular expressions could easily match many predefined text masquerading patterns and successfully harvest the obscured email address. Thus, a need to protect data against automated machine processing arose.

More recently, differentiating between a human and a machine has become important for other applications besides preventing email spam. A list of applications is presented in the next section.

2.2 Applications

Differentiating between a user and machine over the internet has significant importance in the fields of internet security, artificial intelligence, and human computer interaction. Examples of where CAPTCHAs are useful include the following applications:

- Preventing automated account registration [75]
- Preventing artificial inflation of product ratings or voting in online polls
- Preventing outgoing spam [48]
- Preventing blog comment or forum spam [29]
- Preventing automated mining or duplication of a website's content
- Preventing denial of service attacks against web servers [67, 54]
- Preventing brute force password cracking [74, 106, 38]
- Preventing phishing [40]: Humans can issue a CAPTCHA to a computer to validate that it is the computer that it claims to be to prevent masquerading attacks.
- Automating document authentication [43]: If it is hard for a computer to recognize a document, then the computer cannot manipulate it. If a human can still recognize and understand the document, they can be sure that it has not been tampered with. This type of authentication is desirable in the context of digital signatures.

- Tagging images [37]: Tags for images can be improved by learning tags from legitimate users who solve image-based CAPTCHAs.
- Digitizing books [104]: Instead of synthesizing text-based images, words that a computer struggles to digitize from old books could be used.

CAPTCHAs exploit the fact that many artificial intelligence problems remain unsolved. As observed by Luis von Ahn, CAPTCHAs that are based off of interesting AI problems present a win-win situation [101]. Either the CAPTCHA is successful and provides a method for differentiating between humans and machines on the internet, or the CAPTCHA is broken and the underlying hard artificial intelligence problem is also solved. The struggle continues to evolve this set of dynamic problems and advance the field of artificial intelligence.

CAPTCHAs have received criticism from the HCI community [65, 19]. Unfortunately most web users are unaware of the reasons why the websites are forcing them to pass these "annoying" challenges. Some CAPTCHAs are extraordinarily difficult and continue to frustrate legitimate users. For example, Yahoo! discontinued the use of the Gimpy CAPTCHA shown in Fig. (2.1(b)) due to a slew of user complaints. The key to developing a successful human verification technique is to correctly balance security and usability.

2.3 Foundation: The Turing Test

In 1950, Alan Turing provided an operational definition of intelligence using the imitation game [98]. The *Turing test* is played by a human interrogator and taken by one human participant and one machine participant, both of which attempt to appear "human-like" over a text-only channel. If the interrogator cannot reliably determine which participant is the machine and which is the human, the machine is said to have passed the Turing test.

Literature describes a *reverse Turing test* as a test in which any of the above roles have been switched. CAPTCHAs are a slight modification of a reverse Turing test, where the challenge is administered by a machine and taken by a human on a machine. The burden is on the human participant to convince the machine that he is human. Furthermore, ideally the challenge should not be solvable by any machine. Notice the paradox that this creates: the machine can automatically create, administer, and grade a test that it cannot reliably pass [102]. A team of researchers

at Carnegie Mellon University are credited with the first formal mathematical definition of the problem [101]. They define a CAPTCHA test V as $(\alpha, \beta)-$*human executable* if at least an α portion of the human population has success greater than β over V. The more common metric used to measure usability is the average pass rate over a sample of humans [31].

2.4 Types of CAPTCHAs

While often uncredited, Moni Naor was the first to informally define the concept of online human verification [71]. In a 1996 unpublished draft, he proposed nine possible sources for CAPTCHAs which can be separated into four logical categories:

Linguistic CAPTCHAs (Hard AI Problem: natural language processing)

1. **Filling in words** Given a sentence without a subject, select which noun best fits the sentence.
2. **Disambiguation** Given a set of sentences with a pronoun, determine what noun the pronoun refers to.

Text-Based CAPTCHAs (Hard AI Problem: character recognition)

3. **Handwriting understanding** Given an image of a handwritten word with noise added, transcribe the word.

Image-Based CAPTCHAs (Hard AI Problem: semantic image labeling)

4. **Gender recognition** Given an image of a face, determine which gender it is.
5. **Facial expression understanding** Given an image of a face, determine the mood of the person.
6. **Find body parts** Given an image of a body, locate a certain body part.
7. **Deciding nudity** Given several images, determine which image contains the nude person.
8. **Drawing understanding** Given an image, determine what the subject of the image is.

Audio-Based CAPTCHAs (Hard AI Problem: understanding spoken languages)

9. **Speech recognition** Given a clip of spoken audio, transcribe the text.

It is important to note that all popular CAPTCHA implementations that have been suggested in the past 12 years of literature have been either based on, or are a slight variation from the above list.

2.4.1 Linguistic CAPTCHAs

An important distinction must be made between true linguistic CAPTCHAs and image-based CAPTCHAs containing text. Linguistic CAPTCHAs include challenges that are rendered in standard character encoding schemes, such as ASCII. The text appears as normal, inline text on a web page. Humans who are both blind and deaf still frequently use the internet through the aid of Braille readers [73], but are often locked out of many services which rely on CAPTCHAs which do not fall into this category. The CAPTCHAs in this category are the only ones that are truly accessible regardless of perceptual abilities.

The problem with developing such CAPTCHAs is that they are extremely difficult to automatically generate. Common techniques utilize a finite set of phrase templates in which variables are substituted. This finite set of templates increases the likelihood of a successful attack. CAPTCHAs of this type are also vulnerable against machine attack because they do not require any audio or image understanding abilities; they only require a trivial amount of natural language processing related to templates. Due to this, very little research has been devoted to this type of CAPTCHA.

At the CAPTCHA 2002 workshop, the concept of text-based CAPTCHAs was presented in two extended abstracts [46] [78]. Several months later in a preliminary manuscript, the benefits and difficulties of constructing a *Natural Language CAPTCHA* (NLC) [45] were explored. A NLC could be presented in a variety of formats: rendered in an image, spoken as audio, or in plain text. When presented in plain text, it could be solved by users with visual and/or hearing disabilities. While the appeal of a NLC is strong, the author was not able to find a suitable generation algorithm. It seems that a NLC based on recognizing coherent natural language is most likely impossible using current models of natural language.

In 2004, researchers attempted to use word-sense ambiguity to construct a CAPTCHA [18], similar to Naor's 2nd recommendation. They attempted to exploit the fact that different words can have similar meaning, depending on the surrounding

context. They also showed that it is possible to use input from previous tests to train the underlying linguistic model. Two challenges were presented to the user; the answer was only known for one of them. If the user correctly completed the known challenge, it was assumed that their additional challenge was correct as well. Through this method, they build up additional templates to use for future tests.

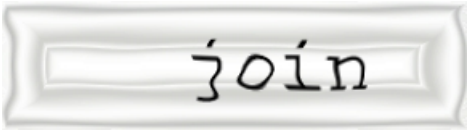
In 2006, a natural language CAPTCHA was presented that required the user to determine the funny knock-knock joke out of a set of 3 jokes (1 real and 2 constructed jokes) [105]. While it was demonstrated that a gap between humans and machines did exist, a machine can pass such a challenge 33.3% of the time by random guessing. Requiring a user to successfully pass such a challenge twice in a row (serial repetition) drops the machine success rate to a more reasonable 11.1%.

2.4.2 Text-Based CAPTCHAs

By far the most popular type of CAPTCHAs are those containing strings of distorted text (a variation of Naor's 3rd recommendation).

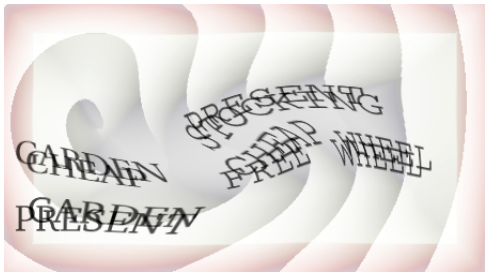
In 1997, one year after Naor's recommendations, AltaVista developed the first concrete implementation of a CAPTCHA. AltaVista had been receiving automated URL submissions to their search engine database by spam bots for the purpose of artificially inflating their search ranking. A group of researchers from the Digital Equipment Systems Research Center were contracted to develop a solution to prevent such an attack [61]. To combat this, the team of developers created a verification system similar to Naor's suggestion of recognizing handwritten images. However, they soon realized that although an image containing text was a step in the right direction, it could easily be foiled by use of OCR software. *Optical Character Recognition* (OCR) software is designed to translate images of text into a machine editable form. The team researched the limitations of scanners with OCR capabilities, and exploited the weaknesses of the OCR systems when rendering their CAPTCHAs. In order to improve OCR results, the manual suggested using similar typefaces, plain backgrounds, and no skew or rotation. To create an image that was resilient to OCR, they did the exact opposite of the suggestions.

In the summer of 2000, Yahoo! began to experience a similar problem where their chat rooms were being spammed by chatbots. Dr. Udi Manber, Yahoo!'s chief scientist, contacted researchers at Carnegie Mellon University to develop a solution



(a) EZ-Gimpy CAPTCHA

(b) Gimpy-r CAPTCHA



(c) Gimpy CAPTCHA

Figure 2.1: Examples of EZ-Gimpy, Gimpy-r, and Gimpy CAPTCHAs

to this problem. This gave birth to the CAPTCHA project, which was led by Manuel Blum and his graduate student, Luis von Ahn.

The researchers at CMU provided Yahoo! with three options (see Fig. (2.1)): EZ-Gimpy renders a single, distorted English word on a noisy background, Gimpy-r renders a random string of distorted characters on a noisy background, and Gimpy renders 5 pairs of overlapping distorted words (of which you must type 3).

In June 2003, Mori and Malik used shape context matching to solve EZ-Gimpy with 92.1% accuracy and Gimpy with 33% accuracy [68]. Also in June 2003, shape context matching was used again to achieve 93.2% accuracy on EZ-Gimpy [97]. This technique computed the cost between templates and the images as the average symmetric chamfer distance of the letters and the variance of the letter distances. The use of two templates raised their accuracy from 89.5% to 93.2%. In June 2004, distortion estimation techniques were used to solve EZ-Gimpy with 99% accuracy and Gimpy-r with 78% accuracy [69]. Due to the limited and fixed size of EZ-Gimpy's dictionary, every challenge image was easily compared against a template database. The distorted template image with the best correlation was returned as the result. Gimpy-r does not rely on a dictionary, and therefore required local distortions to be removed via distortion estimation techniques.

Meanwhile, researchers at Georgia Institute of Technology developed a simi-

lar technique as CMU. A string of characters is rendered into an image form which humans can easily transcribe, but an automated robot cannot [107]. They compared the problem to a one-way trapdoor (non-reversible) hash function, which they termed a Turing-resistant hash. The Turing-resistant hash turns a string of characters (the preimage) into a graphical form (the image) such that a human can easily recover the preimage but a machine cannot.

(a) PessimalPrint CAPTCHAs from [9].

(b) Baffletext CAPTCHAs from [31].

(c) Scattertype CAPTCHAs from [13].

Figure 2.2: Examples of PessimalPrint, Baffletext, and ScatterType CAPTCHAs.

In 2001, researchers at the Xerox Palo Alto Research Center and the University of California at Berkeley synthesized low quality images of machine printed text using a range of words, fonts, and image degradations [35, 9]. Following Baird's quantitative stochastic model of document image quality [7] and a list of problematic OCR examples [70], noise was introduced into the rendered strings by using two image-degradation parameters: blurring and thresholding (see Fig. (2.2(a))). A couple of years later, a reading based CAPTCHA known as BaffleText was developed [31, 11]. BaffleText exercised the Gestalt perception abilities of humans; humans are extremely good at recognizing and understanding pictures despite incomplete, sparse, or fragmented information, whereas machines are not. Baird's model was again used to apply three types of mask operations to binary images: addition, subtraction, and difference (see Fig. (2.2(b))). The first step was to create a random

mask using the following parameters: masking shape (circles, squares, or ellipses), minimum radius of the masking shape, maximum radius of the masking shape, and density (percentage of black pixels in the mask). The mask was then applied to the rendered string using addition (the equivalent of boolean OR), subtraction (the equivalent of NOT-AND), or difference (the equivalent of XOR). A significant contribution was the use of pronounceable, non-dictionary character strings which prohibits a simple dictionary attack while still being user friendly. The character strings were generated by a character trigram Markov model which was trained on the Brown corpus [58].

Typically, OCR systems separate recognition into two sub tasks: segmentation and classification. In 2003, researchers at Microsoft Research exploited the fact that segmentation is much more difficult than classification for OCR systems. They developed a CAPTCHA based on hard segmentation problems, as opposed to hard classification problems [94]. Although character classification was still required, the main challenge was correctly segmenting the string. Another contribution was the observation that website owners with CAPTCHAs have the advantage in the battle against CAPTCHA attackers. This is because CAPTCHA generation is a synthesis task while attacking a CAPTCHA is an analysis task. Analysis is orders of magnitude more difficult than synthesis, especially during lossy synthesis. In the synthesis task, the creator has the ability to use randomness and creativity, whilst in the analysis task, the attackers are tightly constrained by the decisions made by the creator. Unfortunately, legitimate human users also are at the mercy of the CAPTCHA creator and are required to perform the analysis task as well.

A formal study of user friendliness for transcription tasks was conducted at Microsoft Research [27, 25, 26]. They studied the effects of varying the distortion parameters and attempted to determine the optimal parameters where the CAPTCHAs prove hard for machines but easy for humans. As researchers found in the past, the most effective CAPTCHAs are segmentation based challenges, which continues to be a computationally difficult task (see Fig. (2.3)). In 2004, researchers at Microsoft Research attacked several commercial CAPTCHA implementations with surprisingly high accuracy (80%-95%) [28]. Convolutional neural networks were used to perform character recognition. Their attacks had the most difficulty with the segmentation task, not the recognition task. Therefore, they suggested that researchers focus their efforts on building CAPTCHAs which rely on the segmenta-

Figure 2.3: Example of Microsoft's segmentation-based CAPTCHAs from [27].

tion task instead of the recognition task. It was later confirmed in July 2005 that computers are as good as, or better than humans at classifying single characters under common distortion and clutter techniques [26]. However, other researchers have developed an attack that recognizes the "hard-to-segment" Microsoft CAPTCHA more than 60% of the time [110].

Building off of the idea that segmentation is more difficult than recognition, ScatterType was developed at Lehigh University [13, 15, 12]. The challenges are pseudorandomly generated images of text which have been fragmented using horizontal and vertical cuts and scattered using horizontal and vertical displacements (see Fig. (2.2(c))). To defend against dictionary attacks, the text strings are English-like, but non-dictionary words (similar to BaffleText). A human study was performed and showed that human pass rates averaged only 53%, but exceeded 73% on the easiest challenges.

Table 2.1: The average familiarity and average human pass rate as a function of different string generation techniques [16].

Technique	Familiarity	Pass Rate
English	4.79	75.1%
Markov	3.41	61.1%
CVC	2.63	58.7%
Random	1.47	46.3%

In 2006, a formal evaluation of string generation methods was conducted at Avaya Labs [17]. Three challenge string generation techniques were considered: dictionary words, Markov text, and random strings. The authors argued that all of the existing models exhibit substantial weaknesses. It was suggested to use *consonant-vowel-consonant* (CVC) trigrams of psychology as an improvement to the Markov text model. The three original techniques and the improved technique were evalu-

ated using ScatterType to determine the correct balance of image degradations and usability [16]. This user study asked 86 participants to rank the familiarity of 6 character strings on a scale from 1-5 and solve Scattertype CAPTCHAs using each of the string generation techniques. The results are summarized in Table (2.1). Since CVC strings provide more security than Markov strings (they are random and not generated probabilistically based on trigrams), the use of CVC was recommended.

A researcher from the National Chengchi University in Taiwan developed an interesting CAPTCHA using textured patterns [60]. Instead of rendering distorted characters in a different color than the background, the foreground is rendered as a different texture than the background (see Fig. (2.4)). As stated before, segmentation is a difficult task for most OCR systems. Segmenting such an image is extremely hard because the texture of the characters must be analyzed.

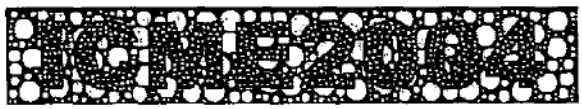


Figure 2.4: Example of a textured pattern-based CAPTCHA from [60].

Researchers at the University of Buffalo's Center of Excellence for Document Analysis and Recognition have contributed a great deal of research building off of Naor's 8th recommendation ("Handwriting recognition") [85, 86, 87, 88, 89]. Features are removed and non-textual strokes/other noise is added to images of handwritten words, while preserving Gestalt segmentation and grouping principles (see Fig. (2.5)). Attempts at attacking the CAPTCHA using state of the art handwritten word recognizers yields a success rate of less than 10%, while the accuracy of human readers is over 75%.

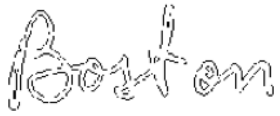


Figure 2.5: Example of a CAPTCHA based on handwritten text from [88].

2.4.3 Image-Based CAPTCHAs

While requiring a user to recognize distorted characters is the most common type of CAPTCHA, semantic image understanding tasks have also been proposed. Chew and Tygar from the University of California at Berkeley investigated a set of three image recognition tasks using a fixed English dictionary of 627 words and Google Images [32, 33]:

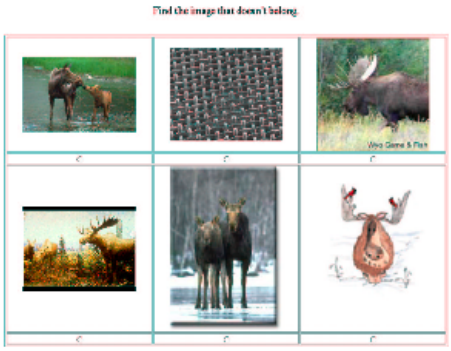
1. **Naming images** Determine the common term associated with a set of 6 images (see Fig. (2.6(a))). They used approximate matching to grade the responses.
2. **Distinguishing images** Determine if two sets of images contain the same subject.
3. **Identifying anomalies** Identify the "odd one out" from a set of 6 images (see Fig. (2.6(b))).

The problems which affected human performance were evaluated and tested during an in-depth user study. Two formal metrics for evaluating CAPTCHAs were also proposed as well as attacks on the three image-based CAPTCHAs. The first metric evaluated CAPTCHA efficacy with respect to the number of rounds of a CAPTCHA and the second metric measured the expected time required for a human to pass the CAPTCHA. Further details can be found in the appendix of [33].

In late 2003, researchers at Microsoft Research argued that the most familiar objects to humans are human faces. They developed a CAPTCHA designed to confuse face recognition algorithms while still being easy to use [80] [81] [82] [83]. Images are automatically synthesized from facial models and the task is to locate and click on the 4 corners of the eyes and 2 corners of the mouth (6 points in total). However, the images looked eerie to many users (see Fig. (2.7)). For this reason, the system was never adopted.

A similar approach to a face recognition based CAPTCHA was developed in 2006 by researchers at George Mason University [66]. Photographs of human faces were mined from a public database and distorted. The user is then prompted to match distorted photographs of several different humans. This CAPTCHA has the benefit of being language independent (ignoring textual instructions for completing the task).

(a) The imaged-based naming CAPTCHA.



(b) The imaged-based anomaly CAPTCHA.

Figure 2.6: Examples of imaged-based naming and anomaly CAPTCHAs from [32].

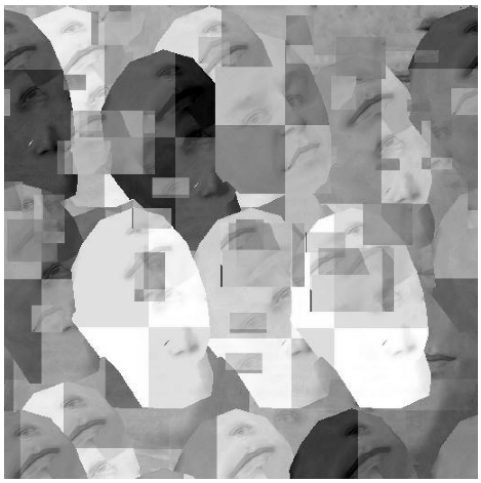


Figure 2.7: Example of an ARTiFACIAL CAPTCHA from [82].

A novel approach to image-based CAPTCHAs was presented by researchers at the University of Memphis where they used a 3D model to generate a 2D image [53]. Random rotations, distortions, translations, lighting effects, and warping were applied to the 3D model to result in a 2D image (see Fig. (2.8)). After these manipulations, the image is displayed to the user who is prompted to determine the subject contained in the image from a list of 30 different objects (similar to Fig. (2.6(a))). By rendering and morphing the images from models, the image database is infinite in size. However, the down side is that the user must choose a label from a fixed, finite set of approximately 30 words. While the test data is infinite, the subject selection is severely limited. The security of the system is directly proportional to the number of words in the system. A random guess attack succeeds with probability $1/n$.

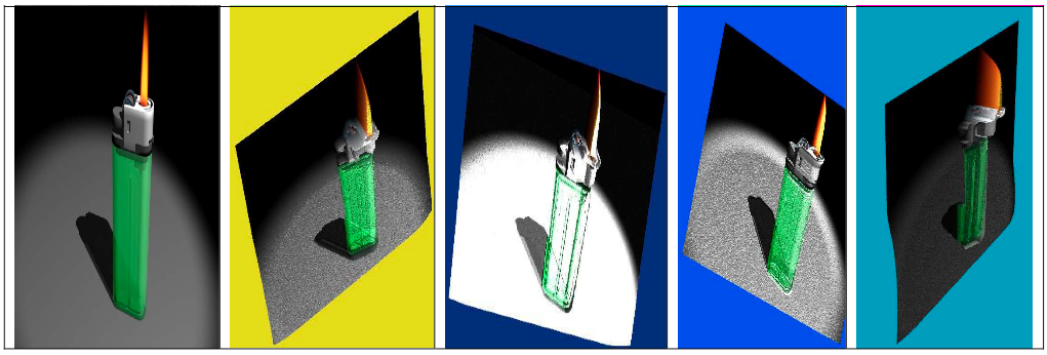


Figure 2.8: Example of several 3D-to-2D CAPTCHAs from [53].

An interesting attempt was made to harness optical illusions [21]. Conventional machine vision systems do not detect visual illusions, though some have been specifically designed to detect them [91]. Existing CAPTCHAs use problems where humans have surpassed a machine's abilities. However, visual illusions present a situation in where the visual system fails in comparison to machines. This gap, albeit a negative gap, can still be exploited in the form of a CAPTCHA. For example, consider a CAPTCHA that asks if the lines in Fig. (2.9) appear to be parallel or not. A human would tell you no, but if a machine measured the lines, it would tell you that they are in fact parallel.

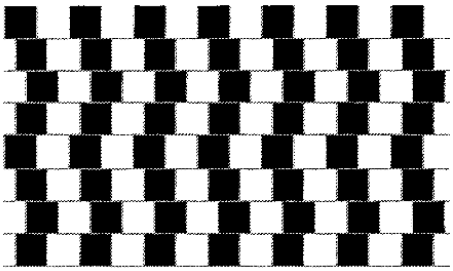


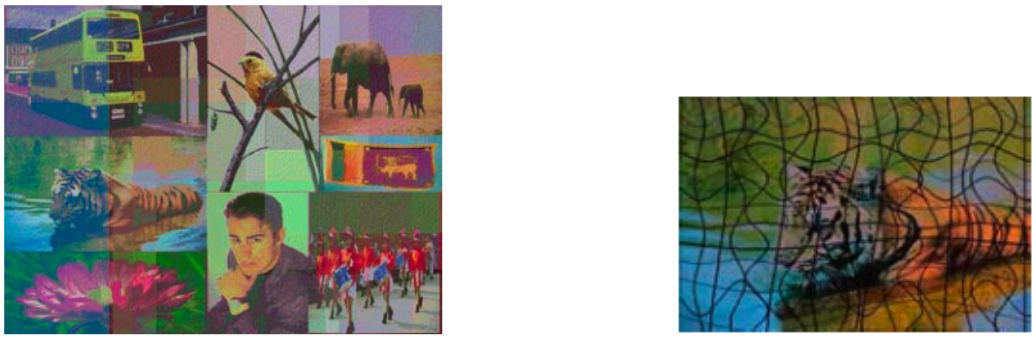
Figure 2.9: Example of an optical illusion from [30].

In January 2005, some researchers thought that current CAPTCHAs were too demanding of legitimate human users. Instead, they proposed Implicit CAPTCHAs which require as little as a single click [8]. The challenges are so elementary that a failed challenge indicates an attempted bot attack. The authors suggest disguising necessary browsing links in images and claim that bots would not be able to find these hidden links (see Fig. (2.10)). While the usability of the system is attractive, the system could easily be attacked on a case-by-case basis. For example, if the user is told to click on a specific, static place on an image, an attacker would only have to solve this once (challenges are static and therefore are reused). This type of CAPTCHA may work for low traffic or low value services, but it would never survive in a large scale application, as it is impossible to automate the generation of challenges.

Figure 2.10: Example of an implicit CAPTCHA where the user is instructed to click on the top of the mountain to continue from [8].

Building off of Chew's approach, a robust image-based CAPTCHA was developed in 2005 [39]. The system, called IMAGINATION, performs controlled dis-

tortions on randomly chosen images and presents them to the user. The distortions are chosen to have a negligible effect on human performance, while simultaneously confusing content based image retrieval systems. The word list was carefully chosen to avoid ambiguous labels while still providing security against attacks. In the first step, the user must click the center of one of the images (see Fig. (2.11(a))). In the second step, the user has to select the correct label for the image after distortions have been applied to it (see Fig. (2.11(b))).



(a) Users must click the center of one of the images.

(b) Users must select the correct label for the image.

Figure 2.11: Example of the IMAGINATION CAPTCHA from [39].

A recent image-based CAPTCHA called ASIRRA, prompts the user to identify images of cats from a set of 8 images of cats and dogs [42] (see Fig. (2.12)). Through a partnership with PetFinder.com, Microsoft is able to harness the power of their large database of manually labeled images. However, a downside of the approach is that the underlying image database has not been publicized (in fact, this violates one of the desirable properties of CAPTCHAs). A 10.3% attack success rate on the ASIRRA CAPTCHA has been achieved using a binary classifier which is 82.7% accurate [47].

2.4.4 Audio-based CAPTCHAs

Automatic speech recognition (ASR) is severely affected by background noise, music, or other speech, while human perception of speech in a noisy environment is very robust. In fact, humans need only a *signal-to-noise* (SNR) ratio of approximately

Figure 2.12: Example of the ASIRRA CAPTCHA from [42].

1.5 dB to recognize and understand speech [96], while even the most state of the art ASR systems require a SNR between 5 and 15 dB [100].

At HIP2002 [20], two audio-based CAPTCHAs were presented: one by a team from Bell Laboratories [63] and one by Nancy Chan of the City University of Hong Kong [23]. Chan's system overlaid white noise and other distractions onto a clip of spoken text to baffle ASR systems. The user's task would be to transcribe the spoken clip. The team from Bell Labs also chose to exploit the gap between humans and machines in natural language processing. Answering a simple spoken question requires processing the audio stream as well as understanding the semantic meaning of the question. For example, the system might ask "What color is the sky?". A further analysis of 18 different sets of distortions on speech data was later performed at Bell Laboratories [57].

A year later, Chan attempted to break the audio-based CAPTCHA using a *Hidden Markov Model* (HMM) [24]. While the results demonstrate that a small gap does exists in the understanding of synthesized speech with background noise, the gap is so small that Chan actually discourages against building audio-based CAPTCHAs until the "naturalness" of synthesized speech improves.

A research group from the Sharif University of Technology has developed a similar approach to that of Bell Labs'. They suggest creating a large database of question templates which are then rendered into an audio clip using a *Text-To-Speech* (TTS) synthesizer [93]. An example question from this CAPTCHA: "There are 5 cats, 3 apples, and 4 dogs on a table. How many pets are there on the table?". This

challenge is trivial for humans, but computers would be required to possess three difficult abilities: speech recognition, natural language processing, and reasoning. The problem with such a CAPTCHA is that the templates for the questions must be manually generated (even if random values can be substituted into the templates). The list of possible challenge questions is therefore small and prone to attack. They have also suggested such a technique for mobile phones [92].

2.4.5 Other CAPTCHAs

At HIP2005 [10], Lopresti presented his plans to leverage the CAPTCHA problem by constructing CAPTCHA challenges which offer simultaneous benefits to both security and pattern recognition [62]. The goal is to harness the cognitive "horsepower" of the millions of legitimate users who are required to solve the CAPTCHAs. Instead of constructing a contrived challenge, real world problems which current computers struggle with are used. In order to determine the ground truth answer for the challenge, prior user input is used.

Also at HIP2005 [10], Chew and Tygar presented a similar idea known as *Collaborative Filtering CAPTCHAs* which are challenges where the ground truth answer is not known [34]. Instead, challenges are graded by comparing their response to the responses of other users. They suggest designing a challenge which evokes some aspect of humanity which is difficult to quantify, such as quality in art, emotion, philosophical views, or humor. While their results are inconclusive, the direction for future work is an exciting area.

An interesting project known as reCAPTCHA [104], which was developed by the original CMU team, follows from Lopresti's [62] and Chew's [34] ideas. The project is implemented as a web service (first suggested by Converse in [36]) that provides images of words from physical books as CAPTCHA challenges. The images are of words which modern OCR systems have failed to recognize with high confidence. Since no ground truth is known for the word, two words are presented: 1 which the ground truth is known for and 1 which the ground truth is not known for (see Fig. (2.13)). If the user correctly transcribes the word for which the ground truth is known, it is assumed that the other transcription is correct as well. After several people have transcribed an unknown word as the same string, it can then be labeled as ground truth with the transcription. It then can be recycled as the known

word for future challenges and reinserted into the appropriate digitized book.

Figure 2.13: Example of a reCAPTCHA from [104].

In October 2007, researchers attempted to avoid *laundry attacks* by animating the answer of the CAPTCHA inside the test itself [6]. A laundry attack is when the attacker posts the challenge online and convinces visitors to solve the challenge for them (also known as the *pornographer-in-the-middle attack* in [62]). The CAPTCHA proposed is a Java applet where various objects are randomly animated inside the test window. The user is given a question and told to click on the correct answer. The animation prevents the unsuspecting user from inadvertently telling the attack where the answer is (as the object is constantly moving).

In hopes of providing an alternate for blind users, researchers at Towson University and the University of Notre Dame developed an audio and image-based CAPTCHA [52]. The proposed CAPTCHA combines a photo of an object and the sound which the object makes (e.g., a photo of a cow and a clip of "moo-ing"). The challenge then becomes to identify the object that they see and hear. While this type of CAPTCHA may be easy for humans and hard for machines, these challenges cannot be automatically generated. Currently only 15 different image/audio combinations have been proposed. Since these challenges must be manually constructed and the list is finite in length, an attack against this CAPTCHA would be quite simple.

2.4.6 Attacks on CAPTCHAs

While there has been a lot of research devoted to creating usable CAPTCHAs, there has also been a considerable amount of interesting research devoted to the breaking of CAPTCHAs [79]. Note that the definition of "breaking a CAPTCHA" is to frequently pass the CAPTCHA with an automated computer program (see [108] for

further discussion).

In June 2004, the Clock Face CAPTCHA, which required users to tell time from a distorted clock face, was attacked [111]. Researchers were able to successfully solve this CAPTCHA with 87.4% accuracy. Their method used 4 types of Harr wavelet-like features and AdaBoost to detect the clock face. In March 2005, the Holiday Inn Priority Club CAPTCHA was broken using linear regression and rotation of axes to undo the rotation of the string and bivariate Haar wavelet filters to recognize the characters with near 100% accuracy [4].

In December 2007, simple pattern recognition algorithms were used to exploit design errors in several commercially available CAPTCHAs [109] and achieved near 100% accuracy. All previous attacks have focused on computer vision or machine learning algorithms, but researchers at Newcastle University chose to use a much simpler approach: letter-pixel counting. They found that even after applying distortions, the pixel counts of the letters remained constant. For example, if a character with no distortions has a pixel count of n, after warping, the same character still has n pixels (just randomly warped). In May 2008, several simple classification methods were investigated to break the PayPal.com CAPTCHA and a 100% success rate was achieved using a simple template correlation technique [56].

Many non-academic related attacks have also been informally documented on blogs and websites. Security analysts recently confirmed that automated attacks have been approximately 20% successful at solving Google's CAPTCHA [77], 30-35% successful at solving Microsoft's CAPTCHA [2], and 35% successful at solving Yahoo!'s CAPTCHA [3]. While there is no formal published literature on the attacks, security logs show that spammers are becoming more and more successful at attacking existing CAPTCHAs. In March 2008, analysts from Wintercore Labs achieved a 90% success rate at solving Google's audio-based CAPTCHA [90].

Security experts are also concerned with the *pornographer-in-the-middle* attack where a spammer captures a CAPTCHA challenge and serves it to an unsuspecting user who is trying to access a website which the spammer has control over [62]. Furthermore, it has been suggested that spammers are outsourcing the solving of CAPTCHAs to third world countries. While this does require a spammer to pay a human for the answers to these CAPTCHAs, the value of the email account is often much more than the cost of paying a human to solve the CAPTCHA.

2.5 Summary

Significant amounts of research have gone into the development of CAPTCHAs over the past 12 years. The first CAPTCHAs required users to transcribe strings of distorted text. Later, more advanced CAPTCHAs which relied on image understanding emerged. In an effort to allow access to blind users, audio-based CAPTCHAs have been developed. Our design of a new video CAPTCHA is motivated by the fact that automated attacks have been successful at breaking nearly all existing CAPTCHAs and usability concerns that many have for existing CAPTCHAs. We also wished to explore the use of video in constructing CAPTCHA challenges, and discover if we could increase usability without sacrificing security.

Chapter 3

Methodology

In this chapter, the methodology for our experiments is presented. In experiment 1 (Video Tagging), we aimed to discover video tagging behaviors of human participants. In experiment 2 (Estimating Attack Rates), we estimated attack success rates on 3 samples (A, C, and D). In experiment 3 (Video CAPTCHAs), we validated our parameter choice through a second user study. The three experiments performed and the four samples collected are summarized in Table (3.1). The results of the experiments are presented in Chapter 4.

Also in this chapter, we define three metrics for measuring success: usability, security, and the gap between usability and security. We visually present our video CAPTCHA design, which requires a user to submit three tags for a video. We also how discuss challenges are graded and how ground truth tags are generated. To test security, we developed a frequency-based attack which submits the most frequent tags below a certain threshold. And finally, we define the parameters of our video CAPTCHA.

3.1 Data Sets and Sampling

For our data, we chose to utilize YouTube.com, which is currently the largest user generated content video system [22]. YouTube currently stores close to 150 million videos[1]. Although all of the videos are publicly accessible, the size of this repository makes sampling a difficult task. YouTube places restrictions on the number of API

[1]Current count is available at `http://gdata.youtube.com/feeds/api/videos?vq=*`

Table 3.1: Experiments and samples

(a) Experiments

Experiment	Description	Samples Used
1	Video Tagging and Human Success Rates	A, B
2	Attack Success Rates and Gap Estimation	B, C
3	Video CAPTCHAs and Human Success Rates	B, D

(b) Samples

Video Sample	Sampling Method	Size
A	Manual Selection	20
B	Random walk recording tag frequencies	86,368
C	Random walk recording all data (incl. related videos)	5146
D	Random walk recording the last video only	20

requests allowed per day and the number of results returned for a query. Therefore, we compared two sampling techniques: using a fixed dictionary of query terms, and performing random walks of the YouTube graph, selecting videos or tags alternately at random.

It is worth noting that our samples represent a snapshot of YouTube at the time they were collected. It has been reported that over 65,000 new videos are uploaded to YouTube every day [72]. Therefore, our samples are already out dated. Additionally, tag frequencies can be heavily influenced by current events. For example, if an earthquake was to strike tomorrow, a slew of new earthquake videos may be uploaded the following day and our sample would not capture this. However, an attacker is limited, just as we are, by the API query limit and the problem of stale tag frequencies. We assume that the attacker has access to our most recent tag frequency list. In practice, this most likely would not be the case. Additionally, frequency samples would need to be refreshed from time to time by performing additional random walks of the YouTube graph for the CAPTCHA to work properly.

3.1.1 Dictionary Sampling

In our first approach, we queried YouTube.com to determine the frequency of all words in a fixed dictionary of common American English words. The dictionary was a newline delimited list of words called the `words` file, which can be found on most Unix-based operating systems (typically at `/usr/share/dict/words`). However, the contents of the file do vary from distribution to distribution. The version which we selected was provided by the `wordlist` package on a Ubuntu 7.10 Gutsy Gibbon machine. It contains 98,569 common American English words, including proper nouns (names, cities, etc.).

Tags are not the only type of metadata that YouTube uses to index videos; they also store the video's category, title, author, etc. Therefore, to obtain tag frequencies, a tag-based YouTube API query was issued for each word in the dictionary. A tag-based API query returns a partial set of videos containing this tag, as well as the total number of videos tagged with it (no other metadata is matched). Using the number of videos that have been tagged with the word, we computed the frequencies by dividing by the total number of videos on YouTube (at the time of our snapshot, the number was 149,661,893). The frequencies for the entire dictionary were collected in approximately 4 hours. Of the 98,569 words in the dictionary file, only 68,377 ($\approx 69.4\%$) were used to tag at least 1 video.

We later learned that the API returns *hints* as to the total number of videos and not the exact number. Unfortunately, we had no way of estimating the accuracy of the hints. Therefore, we investigated an alternate method of collecting frequency data: random walks.

3.1.2 Random Walk

While the dictionary sampling method was a good baseline, we would not be able to discover any tags which were not in the original dictionary. Due to the social nature of YouTube, we hypothesized that non-dictionary words, such as slang, would be extremely common. In hopes of discovering these non-dictionary tags, a random walk of the YouTube graph was necessary. However, purely random sampling is not possible because no comprehensive list of videos is publicly available [72].

Attempting to randomly generate a valid YouTube video ID is a poor random sampling method. YouTube video IDs are 11 characters long, include lowercase

letters (`a-z`), uppercase letters (`A-Z`), numbers (`0-9`), dashes (`-`), and underscores (_). Therefore, the size of the character set allowed is $(26 + 26 + 10 + 1 + 1) = 64$, giving a total of $64^{11} = 7.37 \times 10^{19}$ unique combinations. 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 sampling method.

A common method used for sampling hidden populations where direct interaction with participants would be difficult is known as snowball sampling [49]. An s stage k name snowball sample is similar to a breadth-first search over a fixed number of children, but is performed as follows:

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

Recently, this sampling technique has been used to sample large social networks, including YouTube.com [72]. We chose to use a similar sampling technique, but employed a search technique more closely resembling a randomized depth-first search called a random walk [5]. In [50], 10 random videos from each of the 12 YouTube categories were used as start points for a depth-first crawl. Unfortunately, further details of their crawl are unavailable.

While many types of connections (links) are available on YouTube, the only type we are considering for the random walk are video-tag connections. A connection between a video and a tag exists when a video is associated with a given tag (and vice versa). 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 in the form (u, v) and (v, u) such that $u \in U$ and $v \in V$.

As mentioned before, the entire graph structure is not publicly available, so a random sampling, in the strictest sense, was impossible. Therefore, we used the RandomWalk(MaxDepth) algorithm to randomly walk the YouTube graph:

1. Randomly select a depth d for the walk, such that $1 \leq d <$ MaxDepth.
2. Randomly select a word t from the dictionary.
3. Query the YouTube API to locate the tag vertex u corresponding to t.
4. While $i < d$.

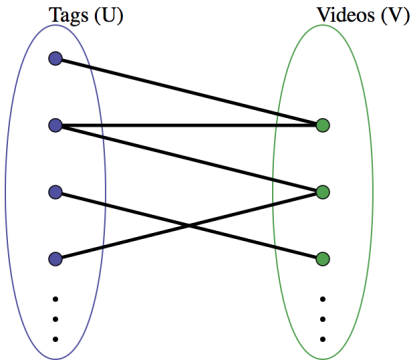


Figure 3.1: Visualization of the undirected, bipartite video-tag graph.

(a) Select a random edge (u, v) where v is a video vertex. Note that only the first 1000 video vertices are visible through the YouTube API.
(b) Select a random edge (v, w) where w is a tag associated with video v.
(c) Assign $u \leftarrow w$ and increment i.

Many random walks were performed using the above procedure (see Table (3.1)). Tag frequencies were collected in Sample B using the random walk (it visited a total of 86,368 videos and discovered a total of 140,439 unique tags). Both the tags and related videos (and their tags) were recorded for the 5146 videos in Sample C. For Sample D, only the video (and its tags) located at the end of the walk were recorded. The value of MaxDepth = 100 was chosen heuristically to ensure good walk depth while avoiding becoming stuck in a local neighborhood or a single connected component. The decision to perform many bounded, random walks instead of a single, unbounded random walk was made in case the YouTube graph structure contained more than 1 connected component (whether it does or not remains unknown). It also mitigates the bias towards a single entry point. If a single, unbounded random walk was performed, the walk would only explore the component of the YouTube graph from which it entered.

We can represent the random walk using a tree. We begin at $u_{1,1}$, the starting tag vertex corresponding to tag t. Next, we randomly select a video vertex to visit, v_{1,r_1} such that $1 \leq r_1 < \text{MIN}(1000, \text{NumOfVideos}(u_{1,1}))$. Next, we randomly select a tag vertex to visit, u_{2,r_2} such that $1 \leq r_2 \leq \text{NumOfTags}(v_{1,r_1})$. Next, we randomly select a video vertex to visit, v_{2,r_3} such that $1 \leq r_3 < \text{MIN}(1000, \text{NumOfVideos}(u_{2,r_2}))$.

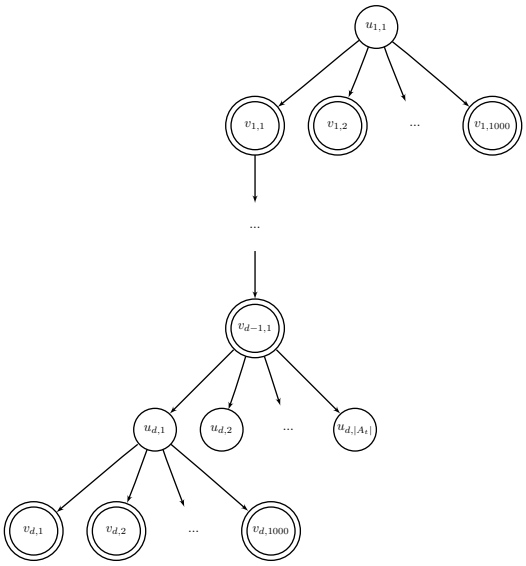


Figure 3.2: A random walk. The smaller, single circled vertices represent tags and the large, double circled vertices represent videos. The nodes may not be unique.

This process is repeated for MaxDepth iterations.

3.1.3 Comparison of Dictionary and Random Walk Samples

As shown in Table (3.2), both samples tend to agree on what the most frequent tags are. As hoped, the random walk results introduced new terms which were not in the dictionary such as years (such as '2008', '2007', etc.) and acronyms (such as 'tv', etc.). The log scale plots of the tag frequencies in Fig. 3.3 both show the exponential drop offs of the tag frequencies. That is to say, many videos are tagged with a small range of words.

However, the random walk yielded higher tag frequencies than the dictionary sample. Although it was originally hypothesized that the dictionary sample would provide better estimates of tag frequencies, the random walk data was more representative of the actual tag frequencies observed on our challenge videos. Our attack rate estimate on Sample C more closely matches the random walk data. This is because the challenge videos were sampled from YouTube using the same random walk technique. For this reason, the dictionary frequencies were discarded and the

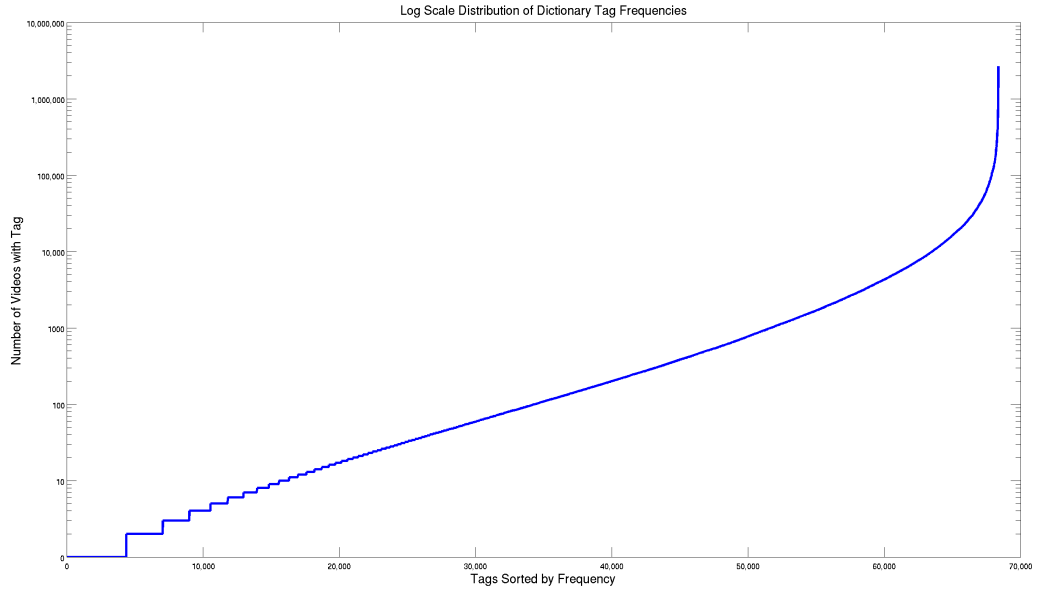
Table 3.2: The 25 most frequent tags from each sample. Entries marked with a * were not in the dictionary but were discovered during the random walk.

(a) Dictionary Sample. The total number of videos is 149,661,893.

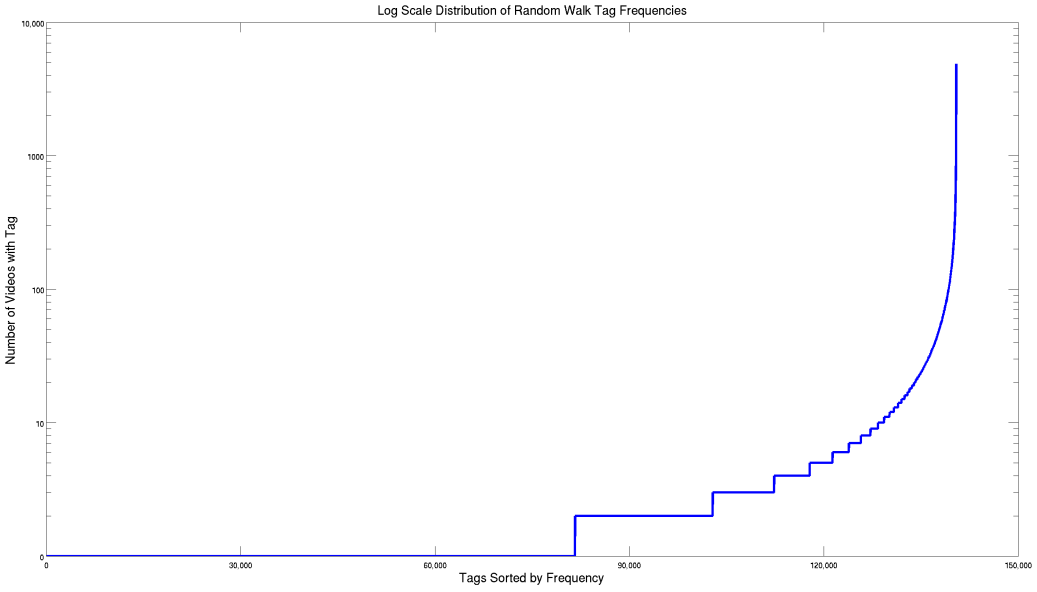
n	Tag	Count	Frequency
1	music	2667006	1.78%
2	video	2627336	1.75%
3	funny	2447215	1.63%
4	rock	1922957	1.28%
5	dance	1424866	0.95%
6	film	1325076	0.88%
7	live	1171594	0.78%
8	short	1045413	0.69%
9	love	937340	0.62%
10	animation	904670	0.60%
11	new	847074	0.56%
12	game	837900	0.55%
13	la	836814	0.55%
14	world	834335	0.55%
15	guitar	823881	0.55%
16	fun	776754	0.51%
17	pop	762626	0.50%
18	rap	757514	0.50%
19	hop	727545	0.48%
20	hip	725533	0.48%
21	comedy	719484	0.48%
22	baby	712005	0.47%
23	show	706895	0.47%
24	dancing	618886	0.41%
25	song	614846	0.41%

(b) Random Walk Sample. The total number of videos is 86,368.

n	Tag	Count	Frequency
1	music	4880	5.65%
2	video	4110	4.75%
3	live	2904	3.36%
4	rock	2680	3.10%
5	funny	2273	2.63%
6	de*	2021	2.33%
7	love	1810	2.09%
8	dance	1734	2.00%
9	new	1707	1.97%
10	world	1563	1.80%
11	guitar	1548	1.79%
12	2007*	1518	1.75%
13	2008*	1499	1.73%
14	rap	1434	1.66%
15	tv*	1409	1.63%
16	comedy	1378	1.59%
17	game	1374	1.59%
18	show	1350	1.56%
19	movie	1312	1.51%
20	episode	1310	1.51%
21	anime*	1296	1.50%
22	pop	1290	1.49%
23	part	1281	1.48%
24	song	1241	1.43%
25	la	1165	1.34%



(a) Log scale plot of tag frequencies using dictionary sampling.



(b) Log scale plot of tag frequencies using random walk sampling.

Figure 3.3: Log scale plots of the tag frequencies for both sampling techniques. The x axis represents the tags in increasing frequency. The y axis represents the tag counts in log scale. Note that the y axis is $10^0 = 1$. In (a), words with a count of 0 are not shown.

random walk frequencies were used for our experiments.

3.2 Usability and Security Metrics

We quantify the usability of a CAPTCHA using the success rates of humans in an experimental setting. The estimated probability that a human passes the CAPTCHA is represented by $P_r(H)$. We quantify the security of a CAPTCHA as the probability that a specific, automated attack passes the challenge. Unlike the usability metric, security for our attack can be both estimated empirically and upper-bounded. The probability that an attack passes the CAPTCHA is $P_r(A)$.

Striking a balance between security and usability is a difficult task. It is most often the case with CAPTCHAs that the two metrics vary inversely: if you wish to have higher security, the usability will suffer and if you wish to have higher usability, the security will suffer. One measure of balance, the *semantic gap*, is the difference between a human's and a machine's ability to recognize visual content [95]. In fact, any positive gap between the success of humans and machines can be amplified to a gap arbitrarily close to 1 through *serial repetition* [101]. In our work, we define the gap as $G = P_r(H) - P_r(A)$. The maximal gap represents the point at which human and machine performance is most dissimilar. However, a maximal gap may not occur for the most secure or most usable conditions.

3.3 Challenge Design

Since the use of video in CAPTCHAs is a novel idea, we needed to determine what we wanted to show to the user and what we wanted the user to tell us. Both proposed designs have their own limitations and drawbacks: Asking users to pick a set of tags requires significant time to read the choices. Asking users to type three tags assumes that their words are correctly spelled (we later mitigate the effects of incorrect spelling through inexact matching).

3.3.1 Pick the Correct Tag Set

In our first Video CAPTCHA design, the user was prompted to select the correct set of tags for the given video from a list of tag sets (see Fig. (3.4)). Each tag set was pulled from a random video. In the example, one video from each of the 11

YouTube categories was used. We speculate that the usability of such a system may have been high, but if n choices are given, a random guess attacker will succeed with probability $1/n$. If we wish to reduce the attack success rate to 2%, 50 tag sets would have to be presented to the user, rendering this design unusable (or serial repetition could be used where users must solve X number of challenges correctly in a row).

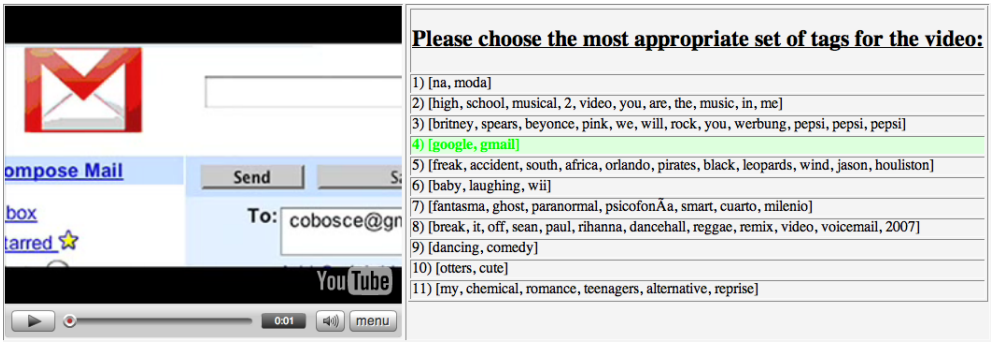


Figure 3.4: A video CAPTCHA in which the user must pick the correct set of tags.

3.3.2 Submit a Correct Tag

Our next Video CAPTCHA design was to present a video to the user and require the user to input three tags (see Fig. (3.5)). A user passes the challenge if any of their preprocessed (see Section 3.6.1) response tags U match any of the author-supplied tags A_t associated with the video. This challenge is similar to the image labeling game known as ESP created by von Ahn [103]. This was used as our control for the experiments.

Given a video with a set of author-supplied tags $A_t = \{a_1, a_2, \ldots a_n\}$ and the user-supplied response tags $U = \{u_1, u_2, u_3\}$, we can grade a challenge as follows:

$$\text{GRADE}(A_t, U) = \begin{cases} \text{PASS} & \text{if } U \bigcup A_t \neq \emptyset \\ \text{FAIL} & \text{otherwise} \end{cases}$$

The human success rates of the control were computed for human subjects over 20 videos in two experiments. To analyze the security of the control, we performed a tag frequency-based attack in which the three most common tags were

(a) A practice video being solved.

(b) A video being solved.

(c) A practice challenge which has been passed.

(d) A challenge which has been failed.

Figure 3.5: Screenshots of the Video CAPTCHA experiment.

always submitted. We simulated such an attack on a random sample of 5146 videos (Sample C). The results may be found in Section 4.4.

3.4 Increasing Usability

In the control, the user must exactly match one of the author-supplied tags for a given video. Due to the ambiguity of natural language and inconsistencies in tagging behaviors, this proves to be a difficult task. The techniques to improve usability that we explored may be broken into two categories: those which expand the user-supplied response tags U, and those which expand the set of ground truth tags T.

3.4.1 Expanding User-Supplied Response Tags

When completing a challenge, the user supplies a set the of three tags, $U = \{u_1, u_2, u_3\}$. We expanded the tag sets through stemming and addition of synonyms. This increases the likelihood of matching a ground truth tag.

A *stemmer* is an algorithm for reducing inflected or derived words to their root [64]. The *root* of a word is the word minus any inflectional endings such as 's', 'ing', etc. The Porter Stemmer is frequently used in *information retrieval* (IR) systems and was developed in 1980 by Martin F. Porter [76]. It's implementation is freely available and uses a deterministic set of rules to stem all words in the English language.

By stemming the response tags, we increase usability by allowing for the following: if "dogs"$\in U$ and "dog"$\in T$, we still accept. We define $\textsc{Stem}(x)$ as the stemmed form of x and U_{stem} as the stemmed forms of the tags in U, such that $U_{stem} = \{\textsc{Stem}(u_1), \textsc{Stem}(u_2), \textsc{Stem}(u_3)\}$. 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. In this way, the stemming approach adds between 0 and 3 tags to U.

Monica Chew suggested the use of a thesaurus to accept synonyms in the image-based naming CAPTCHA [32]. For example, a video about carbonated soft drinks might be tagged as "soda" by one user and "pop" by another. Following this recommendation, we chose to experiment with expanding the user response tags by adding synonyms for the tags in U.

Our first approach was to utilize the `thesaurus.reference.com` website. While this method was resulting in the generation of a small set of seemingly usable synonyms, it suffered from two problems: non-repeatability and lag. Since the reference.com website provides no guarantee as to the stability and permanence of the synonyms provided, we could not be assured that a query issued today would return the same results as a query issued tomorrow. Additionally, submitting a web query for each of the response tags would add a significant delay to the grading of a challenge.

To overcome both of these limitations, we chose to use the freely available,

fixed thesaurus from the Moby Project[2]. In June 1996, the creator of the Moby Project, Grady Ward, released the project into the public domain. The Moby Thesaurus II is a standard text file, where each line contains a word followed by its synonyms. It contains 30,260 words and 2,520,264 synonyms. We define $\textsc{Syn}(x)$ as the set of synonyms of x and U_{syn} as the set of sets of synonyms of the tags in U, such that $U_{syn} = \{\textsc{Syn}(u_1), \textsc{Syn}(u_2), \textsc{Syn}(u_3)\}$.

While the Moby Thesaurus solved the two problems above, inclusion of all of the synonyms for each response tag significantly increases the cardinality of the set. For example, the average number of synonyms per word is 83.3 (the largest number of synonyms is 1447). Note that these counts were computed prior to preprocessing. In the worst case, 1447 synonyms are generated in addition to the original three tags in U. Note that if the response tag is incorrectly spelled or not in the thesaurus, 0 synonyms are returned.

$$\begin{aligned} |U| &\leq |U_{syn} \bigcup U| \\ &\leq 1448 * |U| \end{aligned}$$

However, addition of synonyms in our first experiment proved to compromise the security of the system while adding little usability, so we decided not to use it in experiment 3. For example, when adding synonyms to Sample A, we observed a decrease in security of 50% and only an increase in usability of 2% in our control condition (matching author-supplied tags only).

3.4.2 Expanding Ground Truth Tags

In addition to expanding the user response tags U, we can also intelligently expand the set of ground truth tags T. To do so, we can use additional metadata available from the video as well as metadata from related videos. YouTube provides a list of up to 100 related videos for each video. Unfortunately, the details of how the related videos are computed is not public; all that is known is that they "algorithmically select the set of related videos"[3]. This is less than desirable, as repeatability is questionable. If YouTube tunes the algorithm, the set of related videos may change overnight. Had we wished to exhaustively compute a set of related videos from n

[2]Available online at `http://www.gutenberg.org/etext/3202`

[3]Online at `http://code.google.com/apis/youtube/reference.html`

author tags, we would have had to perform a large number of tag-based queries (one query for each combination of the tags):

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

Note that each tag-based query only returns at most 1000 videos. We could have alternately estimated the set of related videos by only performing some of the $2^n - 1$ queries. For example, given a video with 10 tags, we could have queried on all unique sets of size 10, 9, 8, etc. until we discovered a desired number of videos. While this approach is certainly slower than using the related videos from the YouTube API, it is worthy of future investigation.

On a sample of over one million YouTube.com videos, it was reported that a significant majority (66%) of author-supplied tags (A_t) were tags that did not appear in the video's title [44] (no pre-processing or stop word removal was performed). This indicates that there is additional information in the video's title which is not captured in the tag set. Therefore, a logical expansion is to tokenize the video's title into a set of words A_l and include these words in T as well. The maximum number of characters allowed in a YouTube video title is 60. In the worst case, the title could contain 30 unique words (each word would be a single character), separated by spaces. While this is extremely unlikely (the title would have to be of the form "a b c d e ..." or similar), it is possible.

However, adding titles did not substantially increase the usability of the system. For example, when adding titles to Sample A in experiment 1, we observed a decrease in security of 5% and only an increase in usability of 0.3% as compared to our control (which only used 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). Therefore, we decided not to use titles in the experiments.

We can also learn information about a given video from the tags on *related* videos. For example, consider a video tagged with $A_t = \{bush, president\}$. A related video may be tagged with $R_t = \{george, bush, president\}$. We therefore can assume that "george" might be a good tag for the original video, but the author forgot to provide it. We can also use this as a form of a social spell checker. For example, we noticed that a video of the magician Criss Angel had many related

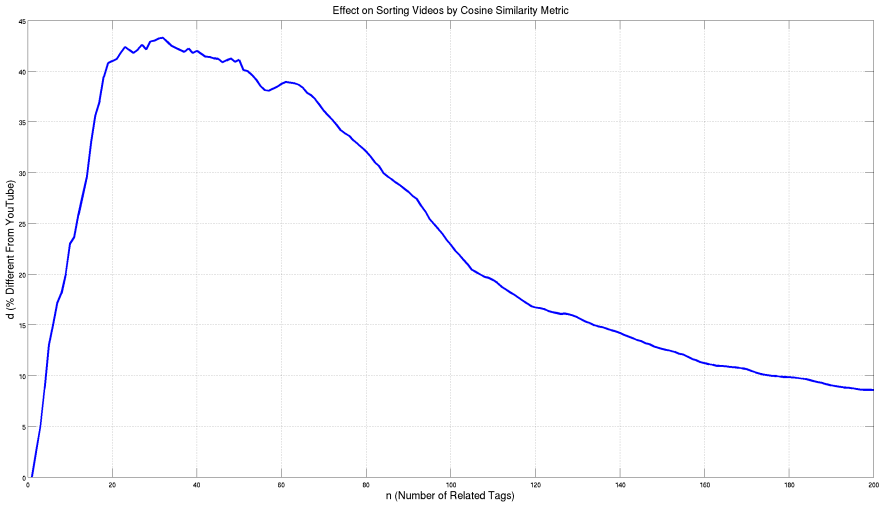


Figure 3.6: The average (across 20 videos) effect of sorting using the cosine similarity metric on Sample A. The y axis is calculated as $d = \frac{k-n}{n}$ where k is the size of the union of the top n related tags from YouTube and the top n tags after the cosine similarity sort. The x axis is the number of related tags.

videos which had been tagged as "Chris Angel" or "Kris Angel". By adding related tags, we are able to allow for common misspellings of tags.

In an effort to select tags in a principled way from the related video set, we performed a sort using the cosine similarity of the tags on related videos and tags on the challenge video. The effects of the cosine similarity sort are shown in Fig. (3.6). The cosine similarity metric is a standard similarity metric often used in IR to compare text documents [99]. The cosine similarity $\textsc{Sim}(A, B)$ can be easily computed:

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

A and B are vectors of occurrence over all tags in the 2 videos. The dot product between A and B is simply:

$$A \cdot B = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

Since the vectors represent tag occurrences, they only contain 0s and 1s. Therefore,

the dot product simply reduces to the size intersection of the two tag sets (i.e., $|A_t \cap R_t|$). The magnitude of a tag occurrence vector reduces to the square root of the number of tags (i.e., $\sqrt{|A_t|}$). Therefore, the cosine similarity $\textsc{Sim}(A_t, R_t)$ between a set of author tags and a set of related tags is:

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

Table 3.3: Example of a tag occurrence table.

	dog	puppy	funny	cat
A_t	1	1	1	0
R_t	1	1	0	1

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. From Table 3.3 (reading row-wise), the tag occurrence vectors for A_t and R_t are $A = \{1, 1, 1, 0\}$ and $B = \{1, 1, 0, 1\}$, respectively. The tag occurrence vectors can be up to length 120 (assuming 60 unique tags on each video). Next, we compute the dot product:

$$\begin{aligned} A \cdot B &= a_1b_1 + a_2b_2 + \ldots + a_nb_n \\ &= (1*1) + (1*1) + (1*0) + (0*1) = 2 \end{aligned}$$

The denominator can also easily be computed:

$$\begin{aligned} \|A\|\|B\| &= \sqrt{(a_1)^2 + (a_2)^2 + \ldots (a_n)^2} * \sqrt{(b_1)^2 + (b_2)^2 + \ldots (b_n)^2} \\ &= \sqrt{1^2 + 1^2 + 1^2 + 0^2} * \sqrt{1^2 + 1^2 + 0^2 + 1^2} = \sqrt{3}\sqrt{3} = 3 \end{aligned}$$

Thus, for our example, the cosine similarity of A_t and R_t is $\frac{2}{3} = 0.\bar{6}$.

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. In the worst case, the tag set could contain 60 unique words (each word would be a single character), separated by spaces. While this is extremely unlikely (the tags would have to be of the form "a

b c d e ..." or similar), it is possible. The maximum number of related videos which YouTube provides is 100. Therefore, we have both a lower and an upper bound on the number or related tags.

$$|T| \leq |T \bigcup R_t| \leq |T| + (60 * 100)$$

One way to reduce the upper bound is to only add n additional unique tags from the related videos sorted in decreasing cosine similarity order. Note that while there is a limit to the number of related tags added, there is no limit to the number of related videos (other than the total number we have available) from which these related tags can come. To generate n related tags, we use the following algorithm, TopRelatedTags(n):

1. Create an empty set, $Z = \emptyset$.
2. Sort related videos in decreasing cosine similarity order relative to the challenge video.
3. For each of the related videos, in sorted order:
   (a) While the related video has tags and while $|Z| < n$:
      i. If the number of new tags on the related video is $\leq n - |Z|$, add them all to Z.
      ii. Otherwise, randomly remove a tag from the remaining tags on the current related video, and add this tag to Z.
4. Return Z.

This technique will introduce up to n additional tags to the ground truth set. Tags are randomly selected from the sorted related videos because 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 must randomly pick one.

3.4.3 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 the user tags and the ground truth tags. We utilize the well known Levenshtein distance [59]. The Levenshtein distance is the minimum number of operations (insertion, deletion, or substitution) required to convert one string into the other. It does not allow for transposition of characters. After computing the Levenshtein distance, we normalize it into the interval [0.0, 1.0], using the maximum length of the two strings s_1 and s_2 in question:

$$\textsc{NormalizedLevenshtein}(s_1, s_2) = 1.0 - \frac{\textsc{Levenshtein}(s_1, s_2)}{\textsc{Max}(|s_1|, |s_2|)}$$

We then define a match as follows:

$$\begin{cases} true & \text{if } \textsc{NormalizedLevenshtein}(s_1, s_2) \geq 0.8 \\ false & \text{otherwise} \end{cases}$$

As per Chew's recommendation in [32], 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. A more conservative similarity bound would have only allowed edits on longer strings (most tags are relatively short).

3.5 Increasing Security

Increasing usability by extending the ground truth tag set will typically result in decreasing security because it allows an attacker a larger set to match against. However, our hypothesis is that we can boost usability without compromising security. If any of the tag set expansions mentioned above were used in isolation, the security would either remain the same (assuming they added 0 tags), or suffer due to the inclusion of additional ground truth tags.

To maintain, or ideally increase, the security even when using tag set expansion techniques, we have to *prune* frequent tags from the ground truth. To do

so, we use the tag frequency estimates from a random walk (Sample B). Given a pruning threshold t, we simply remove all tags from the ground truth which have a estimated frequency $\geq t$ in the random walk data. Note the difference between the ground truth tags T and the pruning threshold t.

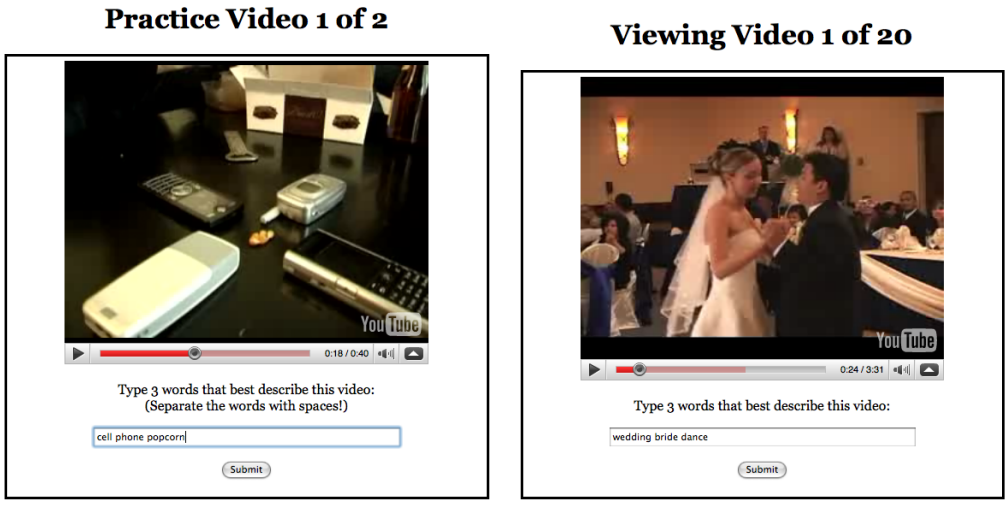
Using this pruning process, we can also provide an upper bound on the success of an attack. If our ground truth tag set T contains the author-supplied tags A_t and n additional tags generated by the TopRelatedTags function, and we prune our ground truth tag set at threshold t, then:

$$P_r(A) < \textsc{Minimum}(1.0, t * |T|)$$

Note that the expression above is extremely pessimistic and assumes that each tag has the highest frequency possible and that all tags label different videos (the sets of videos labeled by each tag are disjoint). That is to say that there is no video on the YouTube website which is tagged with more than 1 of the tags. Table (3.4) contains the estimated attack success rate of tag sets at different pruning thresholds (for the control condition). Note that the estimates in Table (3.4) are lower than the upper bound provided above. Also note that the inner expression above $t * |T|$ is not upper bounded by 1.0 (and a probability must be), so we bound it ourselves using the Minimum function.

3.6 Experiment 1: Video Tagging

In this experiment, we wanted to analyze how people tag online videos by observing participants tagging a set of 20 videos with 3 unique tags each. The videos used in this experiment were individually selected by browsing YouTube ensuring that the videos did not contain any inappropriate content. The experiment was conducted online using a Ruby on Rails based web service to easily collect data from a large number of users. We recorded the IP address of participants to watch for multiple responses from a single user. Friends, family, and colleagues of ours were invited to participate in the experiment. A college-wide email invitation was also sent out to students in the Golisano College of Computing and Information Sciences at the Rochester Institute of Technology. The experiment ran for 2 weeks and we received 233 participants (of which 143 yielded usable responses: see Section (4.1)).



(a) A practice video being tagged.

(b) A video being tagged.

Figure 3.7: Screenshots of Experiment 1.

After participants provided informed consent, they were asked about their age group, gender, education level, the number of online videos they watch per month, and if they have ever uploaded a video to a video sharing website before (see Table (4.2)).

In order to familiarize the participants with the task before beginning the experiment, two practice videos were shown to the users, one of which was a particularly challenging video due to the use of a foreign language. The tags from these practice trials were recorded, but they were not used during analysis. The participant was instructed to *tag* the video with three unique, non-stop words and then *rate* how difficult it was to come up with the tags. A *stop word list* is a list of frequent words which are filtered prior to processing because they are unlikely to add additional information or context. In fact, over 50% of all words in a small typical English passage can be constructed using a list of only 135 words [51]. We chose to utilize a list of 177 stop words provided in the popular Snowball string processing language developed by Martin F. Porter (see Appendix A).

Once the participant submitted the tags for a video, they were asked to subjectively rank how difficult they felt it was to come up with three unique tags. They were given the following choices:

- 5 (Great Effort)
- 4 (Moderate Effort)
- 3 (Some Effort)
- 2 (Little Effort)
- 1 (No Effort)

After completing two practice trials, the participant was required to repeat the *tag-and-rate* process an additional 20 times. The videos were presented in random order to prevent bias due to familiarity with the task. See Section (4.3) for the results of this experiment. The following was recorded for each submission:

Video ID: The ID of the video which was watched.
User ID: The ID of the user who watched the video.
Server Begin Time (C_b): The time at which the server received the request to view the video (computed server-side in Ruby).
Server End Time (C_e): The time at which the server received the tags from the user (computed server-side in Ruby).
Client Time (C_c): The number of milliseconds from when the client's browser rendered the page and when the submit button was pressed (computed client-side in JavaScript).
Tags (Z): The string of tags which the user entered.
Difficulty Rating: The difficulty rating which the user entered (1-5).

After completing the 20 trials, the participant was asked to complete a brief exit survey. The exit survey asked the following questions:

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

See Table (4.2) 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.

3.6.1 Preprocessing

During data collection, no effort has been made to ensure correct spelling, capitalization, or punctuation of user supplied tags. Prior to grading, all tags are preprocessed. First, tags are converted to lowercase to remove the effects of inconsistent capitalization. Any punctuation is also removed. Finally, the string is tokenized into a set of tags U, keeping only the first 3 tags. Consider the following example where the user supplies the string of tags Z:

$$
\begin{aligned}
Z &= \text{“George Bush U.S.A. man”} \\
U &= \textsc{StripPunctuation}(\textsc{LowerCase}(\textsc{KeepFirstThreeTags}(Z))) \\
&= \{\text{“george”}, \text{“bush”}, \text{“usa”}\}
\end{aligned}
$$

3.6.2 Tagging Time

The time take it takes a user to come up with three unique tags is an important usability metric. Therefore, we wanted a two-stage verification for the timings. The client time (C_c) measures the number of milliseconds between the time the page is rendered on the client and the time the submit button is pressed. However, since the time is computed on the client, it could potentially be tampered with by a malicious participant. In order to mitigate the effects of a malicious user providing bad data to us, server-side times are also recorded in seconds. The benefit of recording the time on the server is that it cannot be tampered with. The elapsed server time (C_s) is the difference between the server end time (C_e) and the server begin time (C_b). If we suspect that the client-side times have been tampered with, we can compare the elapsed server time with the client time. If it is greater than some threshold, we can use the more conservative elapsed time of the server. In our results, we did not find any inconsistencies and have therefore used the client-side times.

3.7 Experiment 2: Estimating Attack Rates

A separate random sampling was performed using the previously discussed random walk technique (see Section 3.1.2) to estimate attack rates. This sample, Sample C, contained 5146 videos (and 295,274 related videos) for a total of 299,796 unique videos. Using this sample, a frequency-based attack was performed to estimate $P_r(A)$ over a parameter space τ defined as a 4-tuple:

$$\tau = \langle n, t, s, l \rangle$$

Additional Related Tags (n): The maximum number of new related tags from the cosine-sorted list of related videos to be added to the ground truth tags T.

Pruning (t): Any tags with a frequency at or above this threshold are to be removed from the ground truth tags T.

Stemming (s): A boolean variable indicating whether or not stemming of the user response tags U is to be used.

Inexact Matching (l): A boolean variable indicating whether or not inexact matching (Levenshtein similarity metric) is to be used.

The pruning threshold was varied in the interval $0.001 \leq t \leq 0.01$ by steps of 0.001. Note that $t = 1.0$ was also computed as this represents the case of no pruning. The number of related tags was varied in the interval $0 \leq n \leq 200$ in steps of 5. Stemming and inexact matching were tested under both the TRUE and FALSE conditions. The control performs no pruning ($t = 1.0$), adds no related tags ($n = 0$), does not perform stemming ($s =$ FALSE) and performs exact matching ($l =$ FALSE).

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 (the union of the video sets is the largest). The computation of the trigram frequencies on Sample B (see Section 3.1.2) was not performed. We instead used the three most frequent tags. As explained above, the attack success rate is mitigated by pruning frequently occurring tags from the ground truth tag set. Any tags which have 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 $< t$. This is the attack which we replicated.

For each of the 11 pruning threshold values in the parameter space, we calculated the best set of attack tags from Sample B and used these to attack the 5146 main videos in Sample C. In Table (3.4), we present the attack tags as well as the number of tags which have been pruned at a given t.

Table 3.4: List of attack tags, the number of tags pruned, and the estimated upper bound on $P_r(A)$ for a given pruning threshold. The estimated upper bound on $P_r(A)$ is calculated as the sum of each attack tags' estimated frequency from Table (3.2(b)).

t	Best Attack Tags	# Pruned	Upper Bound on $P_r(A)$
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

3.8 Experiment 3: Video CAPTCHAs

This experiment was nearly identical to the video tagging experiment (Experiment 1) with the following modifications:

- Users are told whether they have passed or failed the challenge.
- The 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 hide the ID of the challenge video.

An effort was made to keep the user interface similar across both the Video Tagging and the Video CAPTCHA experiments. However, in the first experiment, users were only instructed to tag videos and the feedback of "pass" or "fail" was not

presented to them. In this experiment, the wording emphasized that the participants were completing a challenge, or test, which would be graded.

In the Video Tagging experiment, the videos were manually selected. Since a requirement of a CAPTCHA is that they are automatically generated, this time the videos were sampled using a variation of a random walk. Instead of recording information about the videos during the walk, we walk a fixed number of steps and then record information about the final video. Note that there was a human in the loop during this process. Although the random walk was selecting the videos, the videos were manually inspected for inappropriate content; approximately 2 of the random videos contained questionable adult content and approximately 5 of the videos contained foreign tags. These were the only reasons for removing a video; all other videos, regardless of length, content, or rating were allowed. This process was repeated until 20 videos had been selected.

In this experiment, we were also concerned with people trying to "break" the Video CAPTCHA. Therefore, we pre-fetched the video files from YouTube and streamed them from our own servers using the JW FLV Media Player[4], a free player which is licensed under Creative Commons by-nc-sa (Attribution, Noncommercial, and ShareAlike) [1]. 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.

3.8.1 Grading of User Response Tags

In order to inform the user whether they passed or failed the challenge, we had to grade their response. The selection of parameters for the grading function was based on an analysis of the usability $P_r(H)$ of Experiment 1 and the security $P_r(A)$ of Experiment 2. Our hypothesis is that by expanding the ground truth tags, we can develop a more usable Video CAPTCHA that is as secure as the control. We chose to not use stemming or inexact matching during the experiment because we wanted to run using the simplest (non-control) condition that was most usable and confirmed our hypothesis in Experiment 1. Instead, we computed the results of varying these parameters in a post-processing fashion.

[4]Online at `http://www.jeroenwijering.com/`

Using our control parameter set, $\tau_c = \langle 0, 1.0, \textsc{False}, \textsc{False} \rangle$ (no related tags, no pruning, no stemming, and exact matching), we found that $P_r(H|\tau_c) = 0.75$ and $P_r(A|\tau_c) = 0.1286$. To test our hypothesis we wished to select a $\hat{\tau} = \langle n, t, s, l \rangle$ such that our estimates indicate:

1. $P_r(A|\tau_c) \geq P_r(A|\hat{\tau})$
2. $P_r(H|\tau_c) < P_r(H|\hat{\tau})$
3. $P_r(H|\hat{\tau})$ is maximized.

Using data from our first experiment, we found that $\hat{\tau} = \langle 110, 0.005, \textsc{False}, \textsc{False} \rangle$ satisfied these criteria. We wanted to give feedback on exact matching and compute the results of the more 'generous' matching post-hoc. Using this parameter set, $P_r(A|\hat{\tau}) = 0.1223$ and $P_r(H|\hat{\tau}) = 0.9101$. The ground truth tags T for the videos were constructed as:

$$T = A_t \bigcup \textsc{TopRelatedTags}(n) - \textsc{TagsWithFreqGreaterOrEqualTo}(t)$$

where n is the number of related tags to add, and t is the pruning threshold.

The Ruby on Rails web service performs a system call to a Java client, which contacts a Java daemon, which then evaluates the response and returns the result. The client-server approach was used to improve evaluation speed (approximately 1 ms per request), as the database of ground truth tags would always be stored in memory for the Java daemon.

3.9 Summary

In this chapter, we have presented the methodology for our three experiments. In addition, we have described the collection of our four video samples. The final CAPTCHA is parameterized with four variables: n (the number of new, additional related tags to add to ground truth), t (the pruning threshold), s (whether we perform stemming on the user submitted tags), and l (whether we perform inexact matching when grading). In the next chapter, we present and discuss the results of our three experiments. We compute human success rates on Samples A and D and also estimate attack success rates on Sample C over the parameter space. We then

calculate the optimal values for our parameters to find the conditions which yield the most usable, the most secure, and the largest gap CAPTCHA configurations.

Chapter 4

Results and Discussion

The results from each experiment described in Chapter 3 are presented in this chapter. Section (4.1) explains how the participants were selected and compares the demographics collected in the first and third experiments. Section (4.2) presents the difficulty ratings and completion times given by the participants of experiments 1 and 3. Section (4.3) contains the results and discussion for the Video Tagging experiment. Section (4.4) contains the results and discussion for the attack estimation experiment. Section (4.5) contains the results and discussion for the video CAPTCHA experiment.

In our Video CAPTCHA experiment, humans passed roughly 90% of the time when adding 90 related tags, pruning at 0.006, stemming the user-supplied tags, and using inexact matching. We perform a tag frequency-based attack using estimated frequencies and achieve roughly 13% success under the same conditions. We also demonstrate how our CAPTCHA can be parameterized to trade-off between usability and security.

4.1 Participants

Participants for both experiments were selected by word of mouth, flyers placed around campus, direct invitation, and an email sent to all students in the Golisano College of Computing and Information Science at the Rochester Institute of Technology. Faculty and staff in the college were also invited via email to participate in the third experiment (they were not emailed directly for the first experiment

but may have responded to the flyers). Demographics and exit survey responses for the usable participants are summarized in Table (4.2). We have grouped the participants into three categories:

Incomplete, Unusable This group of participants did not complete the entire experiment. The responses in this group were discarded.

Complete, Unusable This group of participants completed the entire experiment, but did not provide at least 3 tags for each video. The responses in this group were discarded.

Complete, Usable This group of participants completed the entire experiment correctly. The responses in this group were used during analysis.

Table 4.1: Participants by category for Experiment 1: Tagging and Experiment 3: CAPTCHAs

	Exp 1: Tagging	Exp 3: CAPTCHAs
Incomplete, Unusable	82 (35.19%)	108 (36%)
Complete, Unusable	8 (3.43%)	8 (2.66%)
Complete, Usable	143 (61.37%)	184 (61.33%)
Total	233	300

4.1.1 Discussion

In both experiments, we observed that participants typically completed 4 or 5 submissions and gave up, or they completed the entire experiment. One of the drawbacks of conducting online experiments is that participants obligated to complete the entire experiment. It seems as if many people just wanted to take a look at the task but were not willing to devote 15 minutes to complete the entire experiment. If we had included the submissions from incomplete participants in the analysis, we would have had unequal samples for the videos. We suspect that tagging a video and solving video CAPTCHAs are learnt tasks. Therefore, we expect the quality of responses to improve with repeated practice. It is interesting that we had an almost identical percentage of complete, usable participants for both experiments (61.37% vs. 61.33%).

Table 4.2: Participant demographics and exit survey responses.

	Exp 1: Tagging	Exp 3: CAPTCHAs
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)

The demographics of the participants is not representative of the global population (see Table (4.2)). While we cannot be sure, we suspect that the bulk of the participants were recruited as a result of the college-wide email. Most of the students in the college are males, aged 18-24, with some college experience. Both of our samples reflect this demographic. We were also surprised at the number of participants who have uploaded a video to an online sharing website before (roughly 35% in both samples). We suspect that users who have uploaded a video before are already familiar with the tagging task.

In the exit surveys, we asked participants to compare the task of transcribing distorted text with tagging a video (see Section (3.6) for complete questions). The results are summarized in Table (4.2). In both experiments, we observed that approximately 60% of participants find the video tagging task more enjoyable. In the first experiment, approximately 20% of participants found tagging a video faster than transcribing distorted text. However, in the third experiment, 27% of the participants reported that tagging a video would be faster. This is somewhat surprising due to the fact that the videos in the third experiment were psuedo-randomly selected (and therefore providing tags might be more difficult). These statistics indicate that inclusion of a video CAPTCHA on a website as an alternate form of human verification might lead to a more enjoyable user experience. However, if a user is rushed, they might choose to solve a text-based CAPTCHA rather than a video CAPTCHA.

The exit surveys also allowed participants to provide comments on their experience. In the first experiment, many people were concerned with the bandwidth and time required to watch and tag a video. In the third experiment, the participants appeared to be more concerned with how well they performed the task (and seemed less certain of themselves). However, people generally completed the task extremely well. Other issues that were raised were the cultural 'American-specific' references in the videos. As discussed before, an international version could be generated by pulling videos from an alternate database and seeding the random walk with a different dictionary. Below are some excerpts from the comments section:

- "Deciphering the scrambled text of some sites is almost impossible, and it has stopped me from entering several online contests that were using it" (Exp 1)
- "The only reason I prefer distorted text to video tagging is the time it takes."

(Exp 1)

- "This is a great idea, and it's ... more fun than [a text-based CAPTCHA]" (Exp 3)
- "You overestimate the public's ability to spell" (Exp 3)
- "It would be interesting to see how people perform using video with audio compared with video without audio." (Exp 3)
- "CAPTCHAs have become too distorted to read. It usually takes me three or four tries to get one right!" (Exp 3)
- "Some videos were very easy to figure out. Others were cryptic." (Exp 3)

4.2 Participant Difficulty Ratings and Completion Times

After supplying three unique tags for a video, participants were asked to rate the difficulty (range 1-5) of coming up with the tags. The median completion time for Experiment 1 was 20.642 seconds and 17.062 seconds for Experiment 3. The median completion time for each difficulty rating is presented in Table (4.3). The difficulty rating and the median completion time both increase with one another. The distribution of difficulty ratings is presented in Table 4.4. The mean, standard deviation, and mode of the difficulty ratings for both experiments can be found in Table (4.5).

Table 4.3: Median completion time in seconds by difficulty ratings

Difficulty	Experiment 1	Experiment 3
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 4.4: Distribution of difficulty ratings

Difficulty	Experiment 1	Experiment 3
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 4.5: Difficulty rating statistics

	Experiment 1	Experiment 3
Mean (μ)	2.1343	2.3066
StdDev (σ)	0.9482	1.0181
Mode	2	2

Table 4.6: Completion time statistics in seconds

	Experiment 1	Experiment 3
Mean (μ)	29.688	22.038
StdDev (σ)	34.746	23.578
Median ($\mu_{1/2}$)	20.642	17.062

4.2.1 Discussion

When compared with the difficulty distribution in Experiment 1, the difficulty distribution of Experiment 3 is shifted upward. The videos in Experiment 3 may have been more challenging since they were not manually selected. Several users commented that the content in the videos was not always very clear. Additionally, the participants may have felt more "pressured" because they were being provided with feedback in Experiment 3. Even though the participants provided the difficulty ratings before they were told if they had passed or failed, the outcome might have influenced their future responses.

The median completion times for the experiments were 20.642 seconds and 17.062 seconds, respectively. We expect in a real-world CAPTCHA that these completion times would be even lower as people are typically in a rush. In an experimental setting, they may have taken more care and time before providing their set of tags.

4.3 Experiment 1 Results: Video Tagging

In the Video Tagging experiment, 20 videos were manually selected (Sample A). The subjects of the videos are presented in Appendix (B). Participants were asked to submit 3 tags for each of the videos (presented in random order). After tagging a video, they were asked to rate the difficulty of providing 3 unique tags. For further details, see Section (3.6).

The human success rates were calculated on the tagging data (Sample A) and are plotted in Fig. (4.1). If all four corners of a tile have better usability than the control, the tile is shaded. The variables ranges are $t \in \{0.001, 0.002 \ldots, 0.01, 1.0\}$ and $n \in \{0, 5, \ldots, 195, 200\}$. As the pruning threshold t decreases, more tags are pruned and the human success rate decreases. The number of tags pruned for each threshold level is determined by the tag frequencies in Sample B (see Table (3.4)). As the number of additional related tags n increases, more tags are added to the ground truth set and the human success rate increases. Pruning seems to have little effect on the human success rates until $t = 0.004$ (184 tags are pruned at this level). At this point, the human success rate begins to rapidly decline. Also of interest is that adding additional related tags substantially boosts the human success rate

up to $n = 30$ at which point it has boosted roughly 15% over the control. From $n = 35$ to $n = 200$, the human success rate only increases roughly 4%. The human success rates on Sample A for no stemming and inexact matching, stemming and exact matching, and stemming and inexact matching can be found in Fig. (C.1).

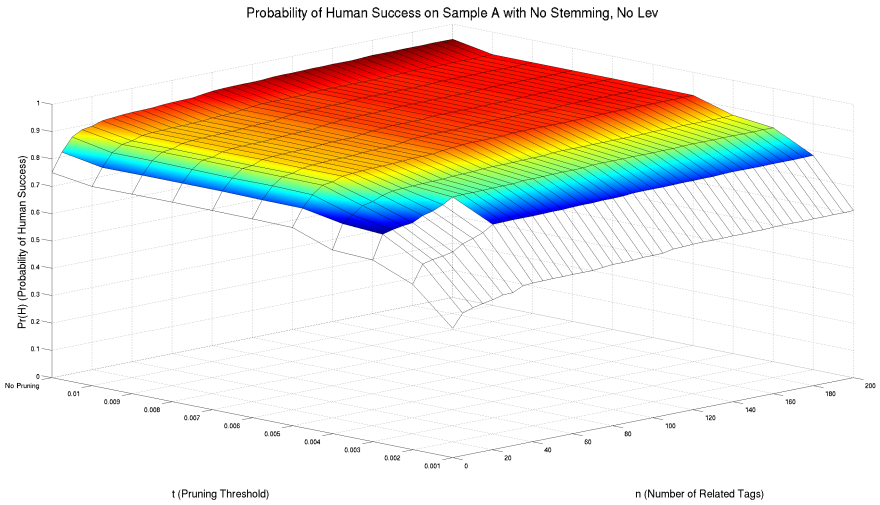


Figure 4.1: The human success rates on Sample A with no stemming and exact matching. The control is located in the top-left corner (0 related tags added, no pruning, and a human success rate of 75%). If all four corners of a tile have better usability than the control, the tile is shaded.

4.3.1 Discussion

The human success rate on the control (located at the leftmost corner of the mesh-plot: $P_r(H) = 0.75$) is surprisingly high. Based on personal experiences, we believe that it might be higher than many text-based CAPTCHAs currently in use today. The addition of only 5 related tags improves the usability of the CAPTCHA approximately 6% regardless of the pruning level. Pruning has very little (3 to 4%) effect on usability until the pruning level reaches approximately $t = 0.004$. At this point, 184 words are being pruned from the ground truth and the human success rate drops.

While many of the parameter settings yield a higher human success rate than the control, a parameter setting is only worth considering if it does not have a higher

attack success rate than the control. In the next experiment, we empirically test the success rate of a tag frequency-based attack.

4.4 Experiment 2: Frequency-Based Attack

Given a pruning threshold t, our tag frequency-based attack submits the three most frequent tags from an estimate over 86,368 videos (Sample B), such that each of their probabilities are $< t$ (see Section (3.7)). This attack was performed on the 20 videos in Sample A as the number of related tags n and pruning threshold t were varied. The attack success rates are plotted in Fig. (4.2). 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 (see Fig. (4.3)).

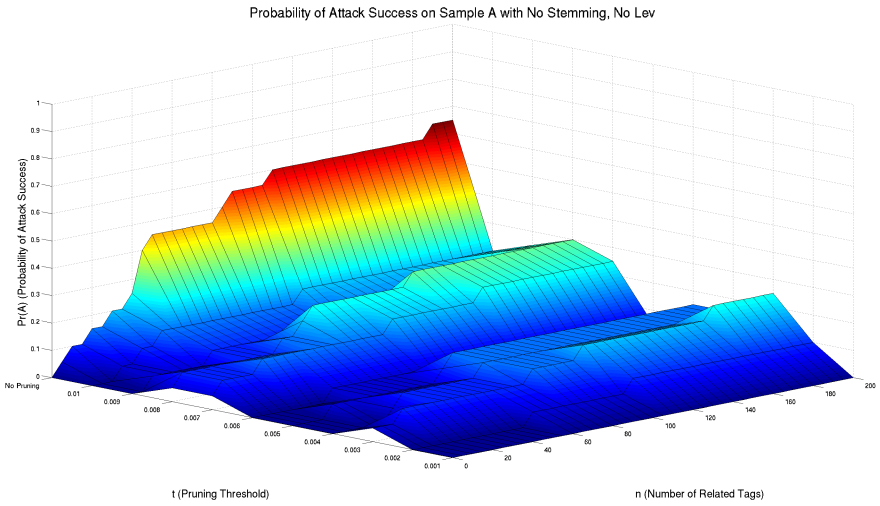


Figure 4.2: The attack success rates on Sample A with no stemming and exact matching. The control is located in the bottom-left corner (0 related tags added, no pruning, and an attack success rate of 0%)

The 5146 videos in Sample C which were discovered during a random walk (see Section 3.1.2) were also attacked with the same tag frequency-based attack. The attack success rates on Sample C are plotted in Fig. (4.3). The gap between the human success rates on Sample A and the attack success rates on Sample C are plotted in Fig. (4.4).

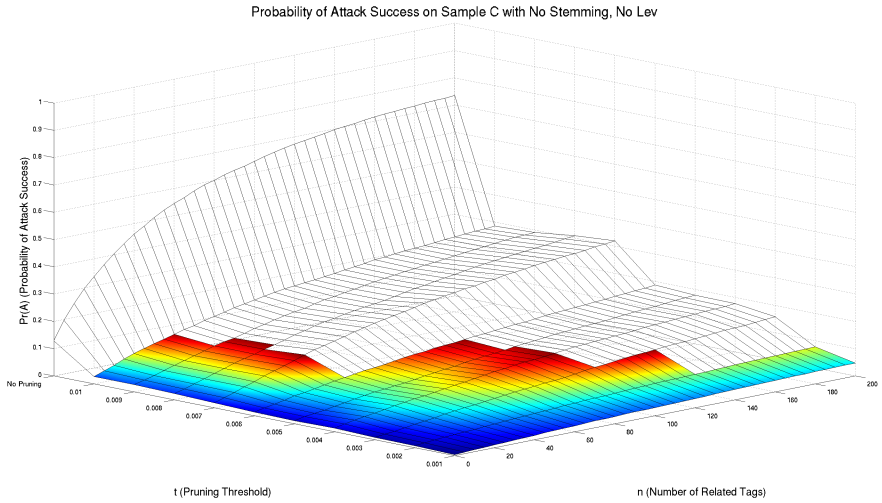


Figure 4.3: The attack success rates on Sample C with no stemming and exact matching. The control is located in the bottom-left corner (0 related tags added, no pruning, and an attack success rate of 12.86%). If all four corners of a tile have equal or better security than the control, the tile is shaded.

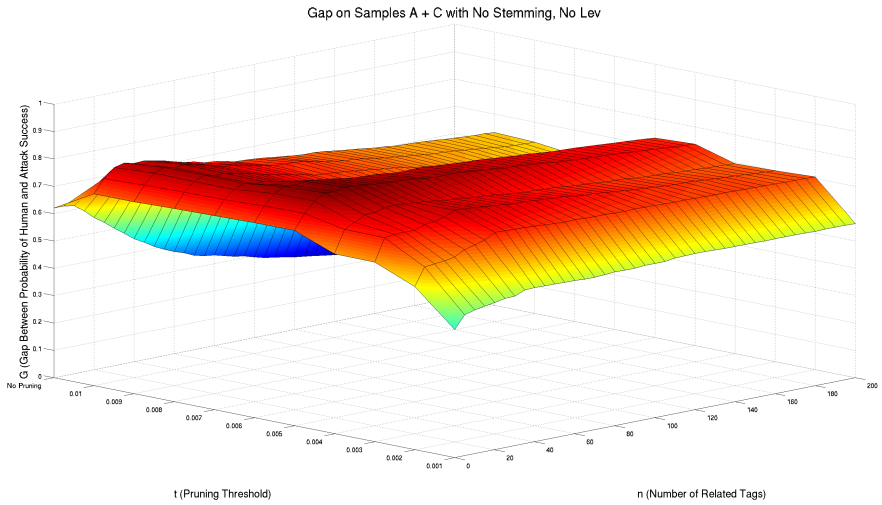


Figure 4.4: The gap between the human success rates on Sample A and the attack success rates on Sample C, computed as the human success rate minus the attack success rate. The control is located in the top-left corner (0 related tags added, no pruning, and a gap of 62.14%). The underside of the meshplot is visible for $t = 1.0$.

A summary of the human success rates, attack success rates, and gap values over the parameter space is presented in Table (4.7). Condition 0 represents the control (recall from Fig. (4.1) that the human success rate on Sample A is 75% and from Fig. (4.3) that the attack success rate on Sample C is 12.86%). Conditions 1-3 give the success rates with no stemming (the user supplied tags are not stemmed) and exact matching (the strings must match exactly; no edit distance is allowed). Conditions 4-6 give the success rates with stemming and exact matching. Conditions 7-9 give the success rates with no stemming and inexact matching. Conditions 10-12 give the success rates with stemming and inexact matching. The conditions were computed after applying constraints on usability and security (human success rate must be $> 75\%$ and the attack success rate must be $\leq 12.86\%$).

Table 4.7: A summary of the human success rates on Sample A, attack success rates on Sample C, and gap values, where n is the number of additional related tags added, t is the pruning threshold, s is whether stemming is enabled, and l is whether inexact matching is used or not. $P_r(H) : A$ is the human success rate on Sample A, $P_r(A) : C$ is the attack success rate on Sample C, and Gap is the difference between the human success rate and the attack success rate.

Condition		n	t	s	l	$P_r(H) : A$	$P_r(A) : C$	Gap
0	Control	0	1.0			0.7500	0.1286	0.6214
1	Most Usable	110	0.005			0.9101	0.1222	0.7879
2	Most Secure	5	0.003			0.7517	0.0128	0.7389
3	Largest Gap	25	0.005			0.8762	0.0402	0.8359
4	Most Usable	105	0.006	✓		0.9199	0.1273	0.7926
5	Most Secure	5	0.003	✓		0.7720	0.0124	0.7596
6	Largest Gap	15	0.006	✓		0.8769	0.0348	0.8421
7	Most Usable	100	0.006		✓	0.9273	0.1281	0.7992
8	Most Secure	5	0.003		✓	0.7682	0.0134	0.7548
9	Largest Gap	15	0.006		✓	0.8779	0.0381	0.8399
10	Most Usable	95	0.006	✓	✓	0.9343	0.1284	0.8058
11	Most Secure	5	0.003	✓	✓	0.7790	0.0134	0.7656
12	Largest Gap	15	0.006	✓	✓	0.8874	0.0379	0.8495

4.4.1 Discussion

The attack success rates presented in Fig. (4.2) indicate that the tag frequency-based attack was unsuccessful at breaking any of the 20 videos in Sample A under the control condition. One would expect that if the number of related tags added is held constant and the pruning threshold is decreased (i.e., more tags are pruned), the attack success rate would either remain constant or drop. However, the attack success rates on Sample A do not represent this. If the attack success rate is inspected for $n = 0$, we see that there is an increase around $t = 0.008/0.009$ and then again at $t = 0.003$. This seems counter-intuitive as we would expect the attack success rate to strictly decrease as t decreases. However, since the attack always chooses the 3 most probable tags directly below the pruning threshold, it's possible that an attack which is unsuccessful for $t = t_0$ may be successful for $t = t_0 - \epsilon$. This is because the attacks are performed with a different set of tags (see Table (3.4)).

Sample A consisted of only 20 videos, so a single successful attack will change the attack success rate by 5%. The videos were manually selected, so there may be some sample bias. This, combined with the previous explanation for the increase in attack success rates as t is decreased, can explain the jagged ridges in the surface. Additionally, the tags removed due to the pruning threshold and the attack tags were determined from the same sample (Sample B). We did so assuming that an attacker would have full knowledge of our frequency lists. However, an attacker may have better success using a more accurate or larger sample.

We also notice that the attack success rates on Sample C (see Fig. (4.3)) appear to be a smoothed version of the attack success rates on Sample A. This can be explained because, as shown in Fig. (3.1), Sample C is substantially larger than Sample A. Therefore, the effects of a small sample is reduced.

Fig. (4.4) illustrates the gap between the human success rates of Sample A and the attack success rates of Sample C. At first glance, it may appear odd to be comparing two different samples. However, a gap calculation using the human success rates on Sample A and attack success rates on Sample A would yield gap values which are strongly biased towards the small sample, Sample A. By using the attack success rates of a larger sample, Sample C, we have reduced the bias resulting from the sample size of Sample A for the attack, while using our available human data.

A summary of conditions of interest is presented in Table (4.7). This table presents the effects of allowing stemming and/or inexact matching via the Levenshtein similarity metric. Conditions 1-12 out perform that of the control (condition 0), in terms of security, usability, and gap value. Notice that the number of related tags necessary to add to the ground truth to create the most usable condition decreases as stemming and inexact matching is allowed. While still meeting the control level for human pass rates, the most secure condition is $n = 5, t = 0.003$. This can be explained by the significant drop in human success when $t < 0.003$. Any additional pruning yields usability that is below that of the control. The most secure condition occurs at $n = 5$ because if $n = 0$ and any pruning is performed, the human success rates will be lower than that of the control. This means that we are pruning, but are not adding any additional ground truth tags. Therefore, the usability drops below that of the control.

The largest gap with no stemming and exact matching occurs when adding 25 related tags and pruning at 0.05% (condition 3 in Table (4.7)). At $n = 25$, we have already achieved an increase in the human success rate of nearly 15%. As mentioned before, the impact of adding more related tags seems to diminish past $n = 30$. Additionally, the security does not seem to be impacted much by adding only 25 additional tags. The human success rate seems to remain fairly constant (notice the large flat region in Fig. (4.1)) for $n \geq 25$ and $t \geq 0.005$. The attack success rates remain fairly constant from $t = 0.006$ down to $t = 0.002$.

4.5 Experiment 3 Results: Video CAPTCHA

In the video CAPTCHA experiment, random walks were performed to select 20 videos (Sample D). The subjects of the videos are presented in Appendix B. Participants were asked to submit 3 tags for each of the challenges (presented in random order). After completing the challenge, they were asked to rate the difficulty of providing 3 unique tags. Once the difficulty rating was submitted, participants were told if they had "passed" or "failed" the challenge. For further details, see Section (3.8).

We chose to implement the video CAPTCHA with the parameter values in condition 1 from Table (4.7): $n = 110$ and $t = 0.005$. These parameters yielded the maximum usability while still out performing the security of the control and using

exact matching and no stemming. We also chose to use the most usable parameter setting so as to not discourage participants in the final experiment. The human pass rates were computed in a post-processing fashion for the conditions including stemming or inexact matching and can be found in Fig. (C.5).

The human success rates for Sample D with no stemming and exact matching are plotted in Fig. (4.5). The gap between the human success rates on Sample D and attack success rates on Sample C with no stemming and exact matching are plotted in Fig. (4.7).

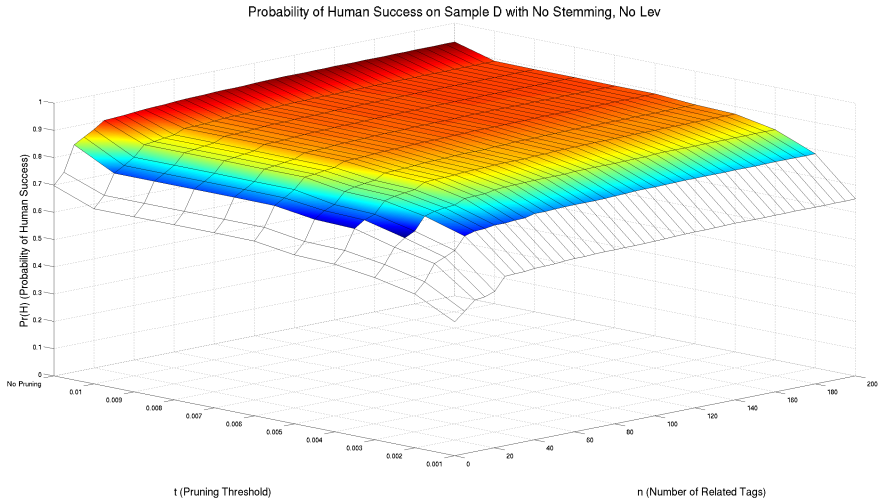


Figure 4.5: The human success rates on Sample D with no stemming and exact matching. The control is located in the top-left corner (0 related tags added, no pruning, and a human success rate of 69.73%). If all four corners of a tile have better usability than the control, the tile is shaded.

A summary of the human success rates, attack success rates, and gap values over the parameter space is presented in Table (4.7). Condition 0 represents the control. Condition 1 represents the values which the video CAPTCHA was tuned to. Conditions 2-4 give the success rates with no stemming and exact matching. Conditions 5-7 give the success rates with stemming and exact matching. Conditions 8-10 give the success rates with no stemming and inexact matching. Conditions 11-13 give the success rates with stemming and inexact matching.

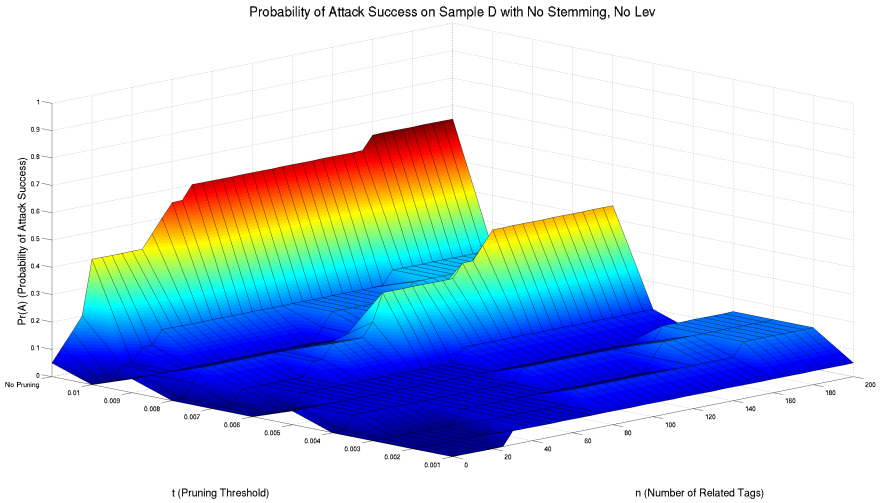


Figure 4.6: The attack success rates on Sample D with no stemming and exact matching. The control is located in the bottom-left corner (0 related tags added, no pruning, and an attack success rate of 0.05%).

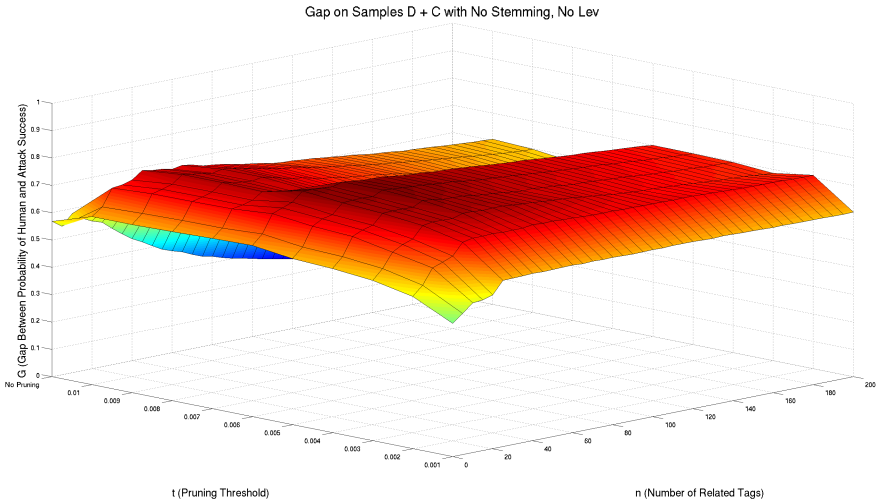


Figure 4.7: The gap between the human success rates on Sample D and the attack success rates on Sample C, computed as the human success rate minus the attack success rate. The control is located in the top-left corner (0 related tags added, no pruning, and a gap of 56.87%). The underside of the meshplot is visible for $t = 1.0$.

Table 4.8: A summary of the human success rates on Sample D, attack success rates on Sample C, and gap values, where n is the number of additional related tags added, t is the pruning threshold, s is whether stemming is enabled, and l is whether we are using inexact matching or not. $P_r(H) : D$ is the human success rate on Sample D, $P_r(A) : C$ is the attack success rate on Sample C, and Gap is the difference between the human success rate and the attack success rate.

Condition		n	t	s	l	$P_r(H) : D$	$P_r(A) : C$	Gap
0	Control	0	1.0			0.6973	0.1286	0.5687
1	Tuned Values	110	0.005			0.8696	0.1222	0.7474
2	Most Usable	100	0.006			0.8828	0.1220	0.7608
3	Most Secure	30	0.002			0.7502	0.0239	0.7263
4	Largest Gap	45	0.006			0.8682	0.0750	0.7931
5	Most Usable	100	0.006	✓		0.8896	0.1226	0.7670
6	Most Secure	25	0.002	✓		0.7548	0.0209	0.7339
7	Largest Gap	45	0.006	✓		0.8755	0.0750	0.8005
8	Most Usable	100	0.006		✓	0.9000	0.1280	0.7719
9	Most Secure	15	0.003		✓	0.7671	0.0233	0.7438
10	Largest Gap	25	0.006		✓	0.8611	0.0526	0.8084
11	Most Usable	90	0.006	✓	✓	0.9019	0.1263	0.7755
12	Most Secure	15	0.003	✓	✓	0.7690	0.0237	0.7453
13	Largest Gap	25	0.006	✓	✓	0.8649	0.0526	0.8122

4.5.1 Discussion

As Table (4.8) indicates, the human pass rate on the control is only 69.73%. By careful selection of parameters (condition 11), we are able to boost the usability to over 90% and even increase security slightly (0.23%). The difference between the usability of condition 0 (the control) and condition 1 (the live values) is 17.23%. In Experiment 1, the difference between the condition 0 (the control) and condition 1 (the tuned values used in Experiment 3) is 16.01%. Therefore, using the same parameters, we have actually seen a slight increase in usability in Experiment 3 over Experiment 1.

Additionally, the parameters appear to be relatively stable across samples. Using the parameters in condition 1, we observed a human success rate of 91.01% in Experiment 1. In Experiment 3, the human success rate dropped 4.05% to 86.96%. In general, the human success rates are slightly lower in Experiment 3 than Experiment 1. This can be explained by the sampling method used: the videos used for Experiment 1 (Sample A) were manually selected while the videos used in Experiment 3 (Sample D) were generated randomly with manual filtering (see Section 3.1.2 for details). The trends and patterns of the human success rates are uniform across both samples as seen in Fig. (4.1) and Fig. (4.5).

In Experiment 1, we were able to out perform the control by including as few as 5 additional related tags. However, in Experiment 3, this number has jumped to 10 for all $t < 1.0$. In Experiment 1, 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 Experiment 3, the best security level which we can achieve is 2.1% (adding 25 related tags, pruning at 0.002, using stemming and exact matching).

As shown, the CAPTCHA can be parameterized to allow for different trade-offs between usability and security. We also observed that it is indeed possible to out-perform the control by adding related tags and pruning frequently occurring tags.

4.6 Summary

Our experiments support our hypothesis that we can increase usability without compromising the security in our video CAPTCHA through intelligently extending the user-supplied and ground truth tag sets. In fact, in Experiment 3, we were able to boost our usability from 69.73% to 90.19% and even reduced the attack success rate slightly by adding 90 related tags, pruning at 0.006, using stemming and inexact matching.

Thus far, we have compared the video CAPTCHA in isolation (always comparing against our control). Comparing human success rates and attack success rates of different CAPTCHAs is not an entirely fair comparison because each usability study had different sample populations/tasks. Additionally, very few papers have been published which include both human success rates and attack success rates. Typically only one half of the story is told (i.e., humans passed X% of the time or our program passed Y% of the time). However, Table (4.9) provides a short list of CAPTCHAs for which human success rates and attack success rates have been published.

Table 4.9: A comparison of human success rates ($P_r(H)$) and attack success rates ($P_r(A)$) for our video CAPTCHA (for our most usable condition) against several other well-known CAPTCHAs.

	CAPTCHA Name	Type	$P_r(H)$	$P_r(A)$
0	Video CAPTCHAs	Video	0.90	0.13
1	Microsoft's CAPTCHAs [25]	Text-based	0.90 [25]	0.60 [110]
2	Baffletext [31]	Text-based	0.89 [31]	0.25 [31]
3	Handwritten CAPTCHAs [88]	Text-based	0.76 [84]	0.13 [84]
4	ASIRRA [42]	Image-based	0.99 [42]	0.10 [47]
5	Wordplays [105]	Linguistic	0.33 [105]	0.11 [105]

As shown in Table (4.9), our video CAPTCHA has competitive human/attack pass rates. With the exception of the ASIRRA CAPTCHA, it exhibits the largest gap of any of the other CAPTCHAs when using our frequency-based attack.

We've also demonstrated that our video CAPTCHA can be parameterized to trade-off between usability and security. With the correct selection of parameters, we were able to drop the attack success rate to 2.1% and still maintain a 75.5%

human success rate (adding 25 related tags, pruning at 0.002, using stemming and exact matching).

Chapter 5

Conclusion

Thesis Statement One can increase usability while maintaining security in a video CAPTCHA by intelligently extending the set of user-supplied and ground-truth tags.

We have presented empirical evidence from a preliminary study that supports our hypothesis. In our experiments, we were able to increase the human success rates from 69.7% to 90.2% by adding 90 new tags from related videos to the ground-truth tag set, stemming the user-supplied tags, performing inexact matching, and pruning tags with an estimated frequency $\geq 0.6\%$. This increase in usability did not have an adverse effect on security (the attack success rate remained nearly constant at roughly 12.9%).

Additionally, we have proposed the first CAPTCHA that uses video understanding to distinguish between humans and machines and shown it to be a viable alternative to existing CAPTCHAs. The task is similar to that of the ESP Game [103] except that only one player is online ('live'). It has comparable human/attack success rates to many well-known CAPTCHAs (see Table (4.9)). We have presented a semi-automated method for generation that yields usable challenges. The grading of challenges is completely automatic and can be parameterized to balance the desired security and usability rates.

We have demonstrated that the proposed video CAPTCHA satisfies four desirable properties; it is automated, open, usable, and secure. Additionally, more than half of the participants found the video CAPTCHA more enjoyable than text-

based CAPTCHAs, and our CAPTCHA may be parameterized for different usability and security trade-offs (as shown in Table (4.8)).

5.1 Future Work

Video CAPTCHAs could significantly benefit from collaborative filtering [34] where ground truth tags are generated by aggregating previous human responses. This direction of research would yield a more usable video CAPTCHA. In addition, such a system would be able to improve search quality on large, online video repositories by improving the tags on videos.

> Program testing can be a very effective way to show the presence of bugs,
> but is hopelessly inadequate for showing their absence.
> -Edsger W. Dijkstra (from [41])

This work has obviously not explored all possible attack vectors against video CAPTCHAs. Similar to Dijkstra's quote, it is impossible to show that a CAPTCHA is secure against all possible attacks. The authors acknowledge that other attacks might have better success than the frequency-based attack which we performed. For example, computer vision could be used to identify objects in the videos, content-based video retrieval systems could be used to retrieve similar videos (and then submit their tags), or an analysis of the audio content on the video might provides clues as to the content of the video. While it is out of the scope of this research to attack the video CAPTCHA from all possible angles, the author gladly welcomes future attacks by other researchers.

Very few CAPTCHAs are capable of being accessible by users with perceptual disabilities. Image-based CAPTCHAs are inaccessible by blind users. Audio-based CAPTCHAs are inaccessible by deaf users. Because video CAPTCHAs provided both visual and auditory cues, an interesting user study would be to have users with perceptual disabilities (simulated or real) attempt to solve a set of video CAPTCHAs.

And finally, our tag set expansion techniques may be used with other media types. For example, one could imagine a CAPTCHA where the user is presented with an audio clip of a song. The task would be to tag the song with appropriate tags

('queen', 'classic rock', etc.). The database of ground truth tags could be expanded by adding tags from related songs.

5.2 Video CAPTCHAs: Summary

Experimental evidence indicates that our video CAPTCHAs may be a more enjoyable alternative to text-based CAPTCHAs. We have provided a set of techniques that would allow for the system to be deployed immediately. Additionally, the usability and security of our video CAPTCHA is comparable to existing text-based and image-based CAPTCHAs.

Bibliography

[1] Creative commons attribution-noncommercial-sharealike 3.0 united states license. Online at http://creativecommons.org/licenses/by-nc-sa/3.0/us/legalcode.

[2] Streamlined anti-captcha operations by spammers on microsoft windows live mail. Online at http://securitylabs.websense.com/content/Blogs/2907.aspx, February 2008.

[3] Yahoo! captcha is broken. Online at http://network-security-research.blogspot.com/, January 2008.

[4] Edward Aboufadel, Julia Olsen, and Jesse Windle. Breaking the holiday inn priority club captcha. *College Mathematics Journal*, 36:101–108, March 2005.

[5] David Aldous and Jim Fill. *Reversible Markov Chains and Random Walks on Graphs*. Unpublished; Available online at http://stat-www.berkeley.edu/users/aldous/RWG/book.html, 2002.

[6] Elias Athanasopoulos and Spiros Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In *Proceedings of the 10th IFIP TC-6 TC-11 International Conference on Communications and Multimedia Security*, number 4237 in Lecture Notes in Computer Science, pages 97–108, Heraklion, Crete, Greece, October 2006.

[7] Henry S. Baird. Document image defect models and their uses. In *Proceedings of the 2nd International Conference on Document Analysis and Recognition*, pages 62–67, Tsukuba Science City, Japan, October 1993.

[8] Henry S. Baird and Jon Louis Bentley. Implicit captchas. In Elisa H. Barney Smith and Kazem Taghva, editors, *Proceedings of the IST SPIE Document Recognition and Retrieval XII Conference*, volume 5676, San Jose, CA, USA, January 2005.

[9] Henry S. Baird, Allison L. Coates, and Richard J. Fateman. Pessimalprint: A reverse turing test. *International Journal of Document Analysis and Recognition*, 5(2-3):158–163, April 2003.

[10] Henry S. Baird and Daniel P. Lopresti, editors. *Human Interactive Proofs, Second International Workshop*, volume 3517 of *Lecture Notes in Computer Science*, Bethlehem, PA, USA, May 2005.

[11] Henry S. Baird and Mark Luk. Protecting websites with reading-based captcha. In *Proceedings of the 2nd International Web Document Analysis Workshop*, pages 53–56, Edinburgh, Scotland, August 2003.

[12] Henry S. Baird, Michael A. Moll, and Sui-Yu Wang. A highly legible captcha that resists segmentation attacks. In Baird and Lopresti [10], pages 27–41.

[13] Henry S. Baird, Michael A. Moll, and Sui-Yu Wang. Scattertype: A legible but hard-to-segment captcha. In *Proceedings of the 8th International Conference on Document Analysis and Recognition*, pages 935–939, Washington, DC, USA, August 2005.

[14] Henry S. Baird and Kris Popat. Human interactive proofs and document image analysis. In *Proceedings of the 5th International Workshop on Document Analysis Systems*, volume LNCS 2423, pages 507–518, Princeton, NJ, August 2002.

[15] Henry S. Baird and Terry Riopka. Scattertype: A reading captcha resistant to segmentation attack. In Elisa H. Barney Smith and Kazem Taghva, editors, *Proceedings of the IST SPIE Document Recognition and Retrieval XII Conference*, volume 5676, pages 197–207, San Jose, CA, USA, January 2005.

[16] Henry S. Baird, Sui-Yu Wang, and Jon Louis Bentley. Captcha challenge tradeoffs: Familiarity of strings versus degradation of images. In *18th International Conference on Pattern Recognition*, volume 3, pages 164–167, 2006.

[17] Jon Louis Bentley and Colin Mallows. Captcha challenge strings: Problems and improvements. In Kazem Taghva and Xiaofan Lin, editors, *IST SPIE Document Recognition and Retrieval XIII Conference*, volume 6067, pages 1–7, January 2006.

[18] Richard Bergmair and Stefan Katzenbeisser. Towards human interactive proofs in the text-domain (using the problem of sense-ambiguity for security). In Kan Zhang and Yuliang Zheng, editors, *Proceedings of the 7th International Conference on Information Security*, volume 3225 of *Lecture Notes in Computer Science*, pages 257–267, Palo Alto, CA, USA, September 2004.

[19] Jeffrey P. Bigham, Maxwell B. Aller, Jeremy T. Brudvik, Jessica O. Leung, Lindsay A. Yazzolino, and Richard E. Ladner. Inspiring blind high school students to pursue computer science with instant messaging chatbots. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 449–453, New York, NY, USA, March 2008.

[20] Manuel Blum and Henry S. Baird, editors. *First International Workshop on Human Interactive Proofs*, Palo Alto, CA, January 2002.

[21] Gavin Brelstaff and Francesca Chessa. Practical application of visual illusions: errare humanum est. In *Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 161–161, New York, NY, USA, August 2005.

[22] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14, New York, NY, USA, October 2007.

[23] Nancy Chan. Abstract of sound oriented captcha. In Blum and Baird [20], page 35.

[24] Tsz-Yan Chan. Using a text-to-speech synthesizer to generate a reverse turing test. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 226–232, Los Alamitos, CA, USA, November 2003.

[25] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Building segmentation based human-friendly human interaction proofs (hips). In Baird and Lopresti [10], pages 1–26.

[26] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *Proceedings of the 2nd Conference on Email and Anti-Spam*, Palo Alto, CA, July 2005.

[27] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720, New York, NY, USA, April 2005.

[28] Kumar Chellapilla and Patrice Y. Simard. Using machine learning to break visual human interaction proofs (HIPs). In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 265–272, Cambridge, MA, December 2004.

[29] Casey Chesnut. Using AI to beat CAPTCHA and post comment spam. Online http://www.brains-n-brawn.com/aiCaptcha, January 2005.

[30] Monica Chew. *Automatically Distinguishing Humans From Machines in an Online Environment*. PhD thesis, University of California, Berkeley, September 2004.

[31] Monica Chew and Henry S. Baird. Baffletext: A human interactive proof. In *IST/SPIE Document Recognition and Retrieval X Conference*, pages 305–316, January 2003.

[32] Monica Chew and J. Doug Tygar. Image recognition captchas. In Kan Zhang and Yuliang Zheng, editors, *Proceedings of the 7th International Information Security Conference*, volume 3225 of *Lecture Notes in Computer Science*, pages 268–279, Palo Alto, CA, September 2004.

[33] Monica Chew and J. Doug Tygar. Image recognition captchas. Technical Report UCB/CSD-04-1333, EECS Department, University of California, Berkeley, August 2004.

[34] Monica Chew and J. Doug Tygar. Collaborative filtering captchas. In Baird and Lopresti [10], pages 66–81.

[35] Allison L. Coates, Henry S. Baird, and Richard J. Fateman. Pessimal print: A reverse turing test. In *Proceedings of the 6th International Conference on Document Analysis and Recognition*, pages 1154–1158, Seattle, WA, September 2001.

[36] Tim Converse. Captcha generation as a web service. In Baird and Lopresti [10], pages 82–96.

[37] Bruno Norberto da Silva and Ana Cristina Bicharra Garcia. A hybrid method for image taxonomy: Using captcha for collaborative knowledge acquisition. In *Proceedings of the AAAI 2006 Fall Symposium on Semantic Web for Collaborative Knowledge Acquisition*, pages 17–23, Arlington, VA, October 2006.

[38] M Dailey and C. Namprempre. A text graphics character captcha for password authentication. In *IEEE Region 10 Conference TENCON*, volume 2B, pages 45–48, November 2004.

[39] Ritendra Datta, Jia Li, and James Z. Wang. Imagination: a robust image-based captcha generation system. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 331–334, New York, NY, USA, November 2005.

[40] Rachna Dhamija and J. Doug Tygar. Phish and hips: Human interactive proofs to detect phishing attacks. In Baird and Lopresti [10], pages 127–141.

[41] Edsger W. Dijkstra. The humble programmer. *Communications of the ACM*, 15(10):859–866, October 1972.

[42] John Douceur, Jeremy Elson, Jon Howell, and Jared Saul. Asirra: a captcha that exploits interest-aligned manual image categorization. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 366–374, New York, NY, USA, October 2007.

[43] Igor Fischer and Thorsten Herfet. Visual captchas for document authentication. In *IEEE 8th Workshop on Multimedia Signal Processing*, pages 471–474, October 2006.

[44] Gary Geisler and Sam Burns. Tagging video: Conventions and strategies of the youtube community. In *Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 480–480, New York, NY, USA, June 2007.

[45] Philip Brighten Godfrey. Natural language captchas. Unpublished manuscript, Carnegie Mellon University, April 2002.

[46] Philip Brighten Godfrey. Text-based captcha algorithms. In Blum and Baird [20].

[47] Philippe Golle. Machine learning attacks against the asirra captcha. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, October 2008.

[48] Joshua T. Goodman and Robert Rounthwaite. Stopping outgoing spam. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 30–39, New York, NY, USA, May 2004.

[49] Leo A. Goodman. Snowball sampling. *The Annals of Mathematical Statistics*, 32(1):148–170, March 1961.

[50] Martin J. Halvey and Mark T. Keane. Analysis of online video search and sharing. In *Proceedings of the Eighteenth Conference on Hypertext and Hypermedia*, pages 217–226, New York, NY, USA, September 2007.

[51] George W. Hart. To decode short cryptograms. *Communications of the ACM*, 37(9):102–108, September 1994.

[52] Jonathan Holman, Jonathan Lazar, Jinjuan Heidi Feng, and John D'Arcy. Developing usable captchas for blind users. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 245–246, New York, NY, USA, October 2007.

[53] Mohammed E. Hoque, David J. Russomanno, and Mohammed Yeasin. 2d captchas from 3d models. In *Proceedings of the IEEE SoutheastCon*, pages 165–170, April 2006.

[54] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur W. Berger. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *2nd*

Symposium on Networked Systems Design and Implementation, Boston, MA, May 2005.

[55] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9:161–191, January 1883.

[56] Kurt Alfred Kluever. Breaking the PayPal.com HIP: A Comparison of Classifiers. Unpublished manuscript, May 2008.

[57] Greg Kochanski, Daniel P. Lopresti, and Chilin Shih. A reverse turing test using speech. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 1357–1360, Denver, Colorado, September 2002.

[58] Henry Kucera and W. Nelson Francis. Computational analysis of present-day american english. *International Journal of American Linguistics*, 35(1):71–75, January 1969.

[59] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[60] Wen-Hung Liao and Chi-Chih Chang. Embedding information within dynamic visual patterns. In *IEEE International Conference on Multimedia and Expo*, volume 2, pages 895–898, June 2004.

[61] Mark D. Lillibridge, Martin Abadi, Krishna Bharat, and Andrei Z. Broder. Method for selectively restricting access to computer systems, February 2001.

[62] Daniel P. Lopresti. Leveraging the captcha problem. In Baird and Lopresti [10], pages 97–110.

[63] Daniel P. Lopresti, C. Shih, and G. Kochanski. Human interactive proofs for spoken language interfaces. In Blum and Baird [20], pages 30–34.

[64] Julie Beth Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, March 1968.

[65] Matt May. Inaccessibility of CAPTCHA. http://www.w3.org/TR/turingtest/, November 2005.

[66] Deapesh Misra and Kris Gaj. Face recognition captchas. In *International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications*, page 122, Washington, DC, USA, February 2006.

[67] William G. Morein, Angelos Stavrou, Debra L. Cook, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 8–19, New York, NY, USA, October 2003.

[68] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 134–141, Madison, WI, USA, June 2003.

[69] Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter. Distortion estimation techniques in solving visual captchas. In *Conference on Computer Vision and Pattern Recognition*, volume 02, pages 23–28, Los Alamitos, CA, USA, June 2004.

[70] George L. Nagy, Stephen V. Rice, and Thomas A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, May 1999.

[71] Moni Naor. Verification of a human in the loop or identification via the turing test. Unpublished manuscript, Sept 1996.

[72] John C. Paolillo. Structure and network in the youtube core. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, pages 156–166, Washington, DC, USA, January 2008.

[73] Linda Dailey Paulson. Blind, deaf engineer develops computerized braille machine. *Computer*, 35(12):27–27, December 2002.

[74] Benny Pinkas and Tomas Sander. Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 161–170, New York, NY, USA, November 2002.

[75] Clark Pope and Khushpreet Kaur. Is it human or computer? defending e-commerce with captchas. *IT Professional*, 7(2):43–49, March 2005.

[76] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[77] Sumeet Prasad. Google's captcha busted in recent spammer tactics. Online at http://securitylabs.websense.com/content/Blogs/2919.aspx, February 2008.

[78] Bartosz Przydatek. On the (im)possibility of a text-only captcha. In Blum and Baird [20].

[79] Sara Robinson. Up to the challenge: Computer scientists crack a set of ai-based puzzles. *SIAM News*, 35(9):23–24, November 2002.

[80] Yong Rui and Zicheng Liu. Artifacial: Automated reverse turing test using facial features. In *Proceedings of the 11th ACM international conference on Multimedia*, pages 295–298, New York, NY, USA, November 2003.

[81] Yong Rui and Zicheng Liu. Excuse me, but are you human? In *Proceedings of the 11th ACM international conference on Multimedia*, pages 462–463, New York, NY, USA, November 2003.

[82] Yong Rui and Zicheng Liu. Artifacial: Automated reverse turing test using facial features. *Multimedia Systems*, 9(6):493–502, June 2004.

[83] Yong Rui, Zicheng Liu, Shannon Kallin, Gavin Janke, and Cem Paya. Characters or faces: A user study on ease of use for hips. In Baird and Lopresti [10], pages 53–65.

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

[85] Amalia Rusu and Venu Govindaraju. Handwriting word recognition: A new captcha challenge. In *Proceedings of the 5th International Conference on Knowledge Based Computer Systems*, pages 347–357, Hyderabad, India, December 2004.

[86] Amalia Rusu and Venu Govindaraju. Handwritten CAPTCHA: using the difference in the abilities of humans and machines in reading handwritten words. In *9th International Workshop on Frontiers in Handwriting Recognition*, pages 226–231, October 2004.

[87] Amalia Rusu and Venu Govindaraju. Challenges that handwritten text images pose to computers and new practical applications. In Elisa H. Barney Smith and Kazem Taghva, editors, *Proceedings of the IST SPIE Document Recognition and Retrieval XII Conference*, volume 5676, pages 84–91, San Jose, CA, USA, January 2005.

[88] Amalia Rusu and Venu Govindaraju. A human interactive proof algorithm using handwriting recognition. In *Proceedings of the 8th International Conference on Document Analysis and Recognition*, pages 967–971 Vol. 2, Aug 2005.

[89] Amalia Rusu and Venu Govindaraju. Visual captcha with handwritten image analysis. In Baird and Lopresti [10], pages 42–52.

[90] Rubén Santamarta. Breaking gmail's audio captcha. Online at http://blog.wintercore.com/?p=11, March 2008.

[91] Eric Saund. Perceptual organization of occluding contours generated by opaque surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 624–630, June 1999.

[92] Mohammad Shirali-Shahreza and M. Hassan Shirali-Shahreza. A new solution for password key transferring in steganography methods by captcha through mms technology. In *International Conference on Information and Emerging Technologies*, pages 1–6, July 2007.

[93] Mohammad Shirali-Shahreza and Sajad Shirali-Shahreza. Captcha for blind people. In *International Symposium on Signal Processing and Information Technology*, pages 995–998, December 2007.

[94] Patrice Y. Simard, Richard Szeliski, Josh Benaloh, Julien Couvreur, and Iulian Calinov. Using character recognition and segmentation to tell computers from

humans. In *International Conference on Document Analysis and Recogntion*, volume 1, pages 418–423, Los Alamitos, CA, USA, August 2003.

[95] Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[96] Andrew Stuart and Dennis P. Phillips. Word recognition in continuous and interrupted broadband noise by youn normal-hearing, older normal-hearing, and presbyacusic listeners. *Ear and Hearing*, 17(6):478–489, December 1996.

[97] Arasanathan Thayananthan, Björn Stenger, Phil H. S. Torr, and Roberto Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 127–133, June 2003.

[98] Alan M. Turing. Computing Machinery and Intelligence. *Mind*, 59(236):433–460, October 1950.

[99] C.J. van Rijsbergen. *Information retrieval, Second edition.* Butterworth-Heinemann Ltd, London, UK, 1979.

[100] Erik van Woudenberg, Frank K. Soong, and J. E. West. Acoustic echo cancellation for hands-free asr applications in noise. In *Proceedings of the Workshop on Acoustic Echo and Noise Control*, pages 160–163, September 1999.

[101] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In *Eurocrypt: Advances in Cryptology*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311, Warsaw, Poland, May 2003.

[102] Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, February 2004.

[103] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In Elizabeth Dykstra-Erickson and Manfred Tscheligi, editors, *Proceedings of the*

SIGCHI conference on Human factors in computing systems, pages 319–326, New York, NY, USA, April 2004.

[104] Luis von Ahn, Ben Maurer, Colin McMillen, Mike Crawford, Ryan Staake, and Manuel Blum. reCAPTCHA Project. Online, http://www.recaptcha.net, May 2007.

[105] Pablo Ximenes, Andre dos Santos, Marcial Fernandez, and Joaquim Celestino Jr. A captcha in the text domain. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems Workshop*, volume 4277/2006 of *Lecture Notes in Computer Science*, pages 605–615, November 2006.

[106] J. Xu, R. Lipton, I. Essa, M. Sung, and Y. Zhu. Mandatory human participation: a new authentication scheme for building secure systems. In *Proceedings of the 12th International Conference on Computer Communications and Networks*, pages 547–552, October 2003.

[107] Jun Xu, Irfan A. Essa, and Richard J. Lipton. Hello, are you human? CC Technical Report GIT-CC-00-28, Georgia Institute of Technology, November 2000.

[108] Jeff Yan. Bot, cyborg and automated turing test. Technical Report CS-TR-970, University of Newcastle, June 2006.

[109] Jeff Yan and Ahmad Salah El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Proceedings of the 23rd Annual Computer Security Applications Conference*, pages 279–291, December 2007.

[110] Jeff Yan and Ahmad Salah El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, Alexandria, VA, USA, October 2008.

[111] Zhenqiu Zhang, Yong Rui, T. Huang, and C. Paya. Breaking the clock face hip [web services human interactive proofs]. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 3, pages 2167–2170, June 2004.

Appendices

Appendix A

Stopword List

i, me, my, e, myself, we, our, ours, ourselves, you, your, yours, yourself, yourselves, he, him, his, himself, she, her, hers, herself, it, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, would, should, could, ought, ng, i'm, you're, he's, she's, it's, we're, they're, i've, you've, we've, they've, i'd, you'd, he'd, she'd, we'd, they'd, i'll, you'll, he'll, she'll, we'll, they'll, isn't, aren't, wasn't, weren't, hasn't, haven't, hadn't, doesn't, don't, didn't, won't, wouldn't, shan't, shouldn't, can't, cannot, couldn't, mustn't, let's, that's, who's, what's, here's, there's, when's, where's, why's, how's, a, an, the, so, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very

Appendix B

Videos in Sample A and D

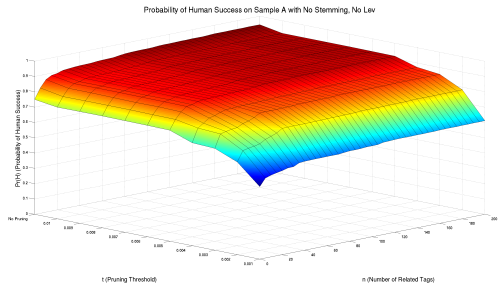
Table B.1: A brief description of the videos used in the experiments.

	Experiment 1 (Sample A)	Experiment 3 (Sample D)
1	the incredible hulk	hot air balloons
2	google maps	spongebob Square Pants
3	two babies	subaru Impeza rally
4	obama speech	seaplane landing
5	dog in swimming pool	guitar lesson
6	blending an iphone	tom and jerry cartoon
7	nba basketball	fireworks at disneyworld
8	how to perform cpr	scuba diving
9	nike beach soccer	fibonacci numbers
10	etrade laughing baby	dogs in halloween costumes
11	effiel tower	regis millionaire
12	macbook air commercial	skydiving
13	soccer zidane headbutt	nba seattle
14	playing two guitars	tetris attack game
15	robot solving rubiks cube	water pollution
16	first dance at wedding	leg workout
17	golden gate bridge	oriental medicine
18	space shuttle launch	jennifer lopez
19	kangaroos	subwoofer testing
20	tweety bird	snowboarding game

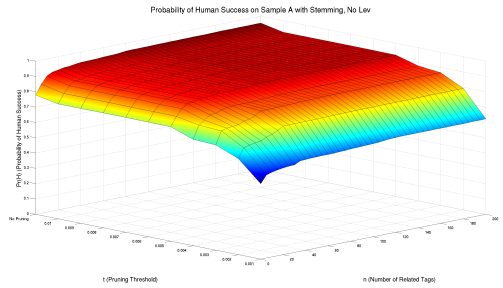
Appendix C

Usability, Security, and Gap

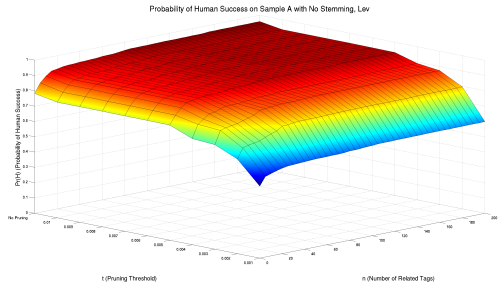
C.1 Usability of Sample A



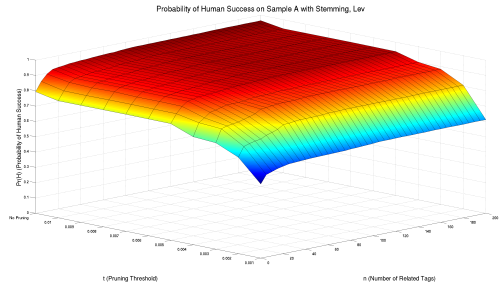
(a) Human Success Rates: No Stemming and Exact Matching



(b) Human Success Rates: Stemming and Exact Matching



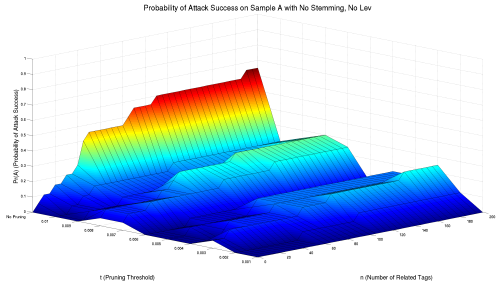
(c) Human Success Rates: No Stemming and Inexact Matching



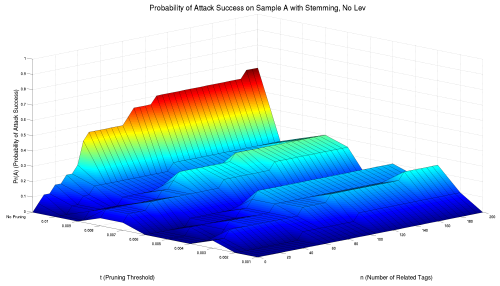
(d) Human Success Rates: Stemming and Inexact Matching

Figure C.1: The human success rates for Sample A.

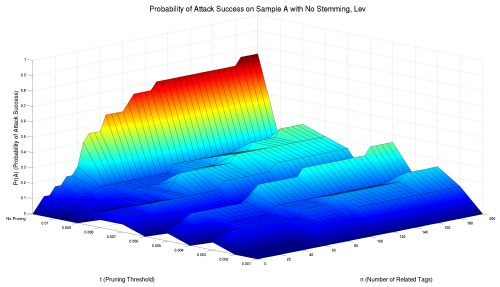
C.2 Security of Sample A



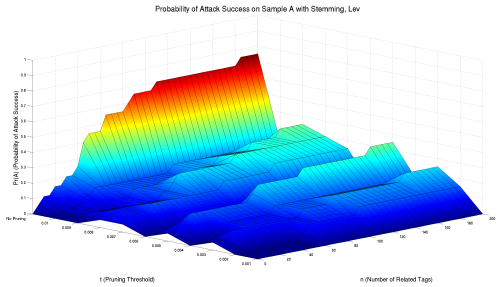
(a) Attack Success Rates: No Stemming and Exact Matching



(b) Attack Success Rates: Stemming and Exact Matching



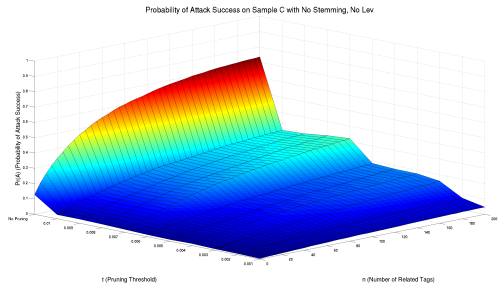
(c) Attack Success Rates: No Stemming and Inexact Matching



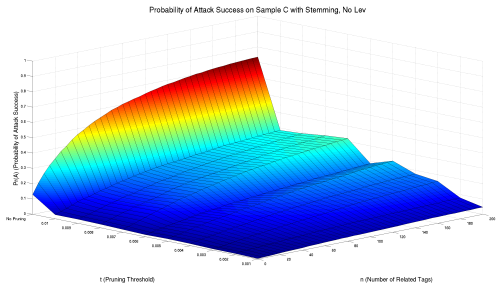
(d) Attack Success Rates: Stemming and Inexact Matching

Figure C.2: The attack success rates for Sample A.

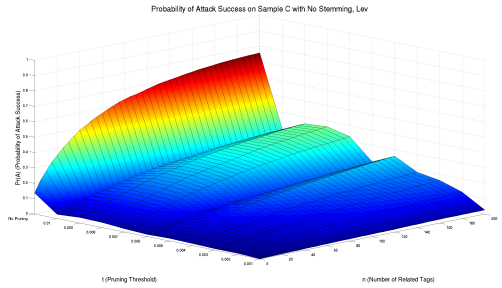
C.3 Security of Sample C



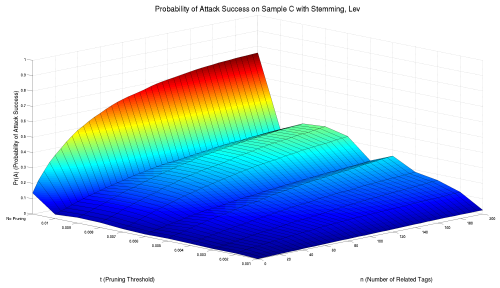
(a) Attack Success Rates: No Stemming and Exact Matching



(b) Attack Success Rates: Stemming and Exact Matching



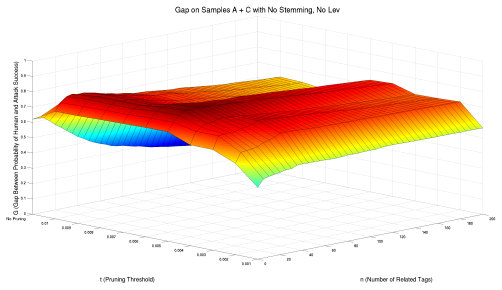
(c) Attack Success Rates: No Stemming and Inexact Matching



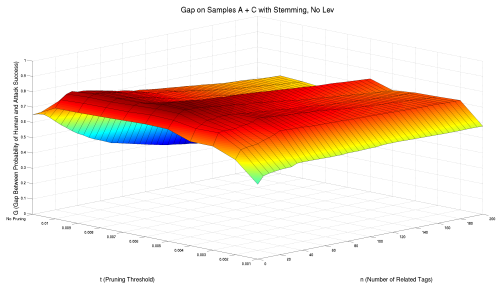
(d) Attack Success Rates: Stemming and Inexact Matching

Figure C.3: The attack success rates for Sample C.

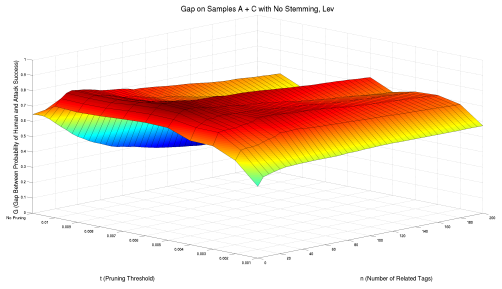
C.4 Gap of Samples A and C



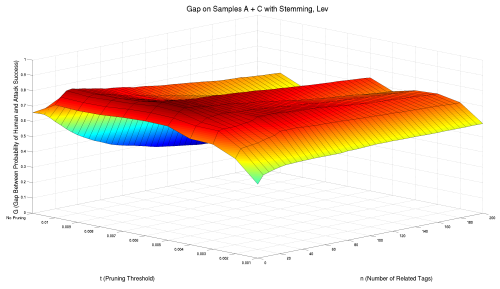
(a) Gap: No Stemming and Exact Matching



(b) Gap: Stemming and Exact Matching



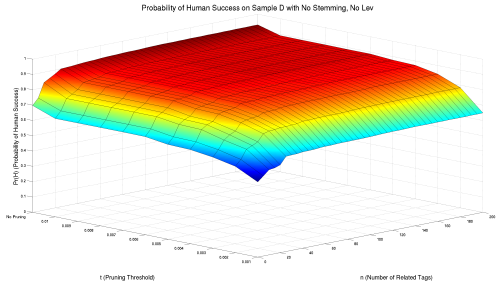
(c) Gap: No Stemming and Inexact Matching



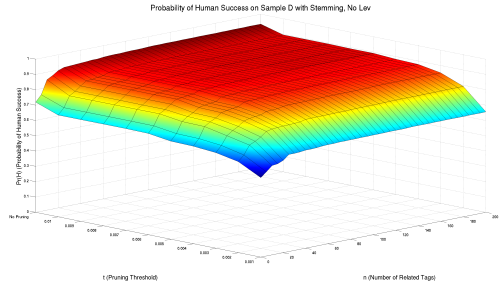
(d) Gap: Stemming and Inexact Matching

Figure C.4: The gap on between human success rate on Sample A and attack success rate on Sample C.

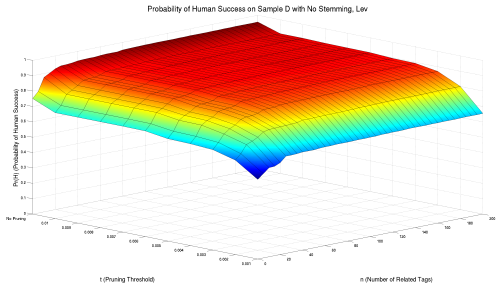
C.5 Usability of Sample D



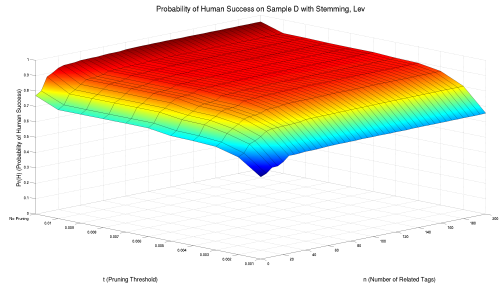
(a) Human Success Rates: No Stemming and Exact Matching



(b) Human Success Rates: Stemming and Exact Matching



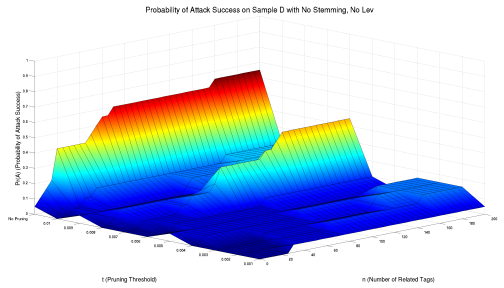
(c) Human Success Rates: No Stemming and Inexact Matching



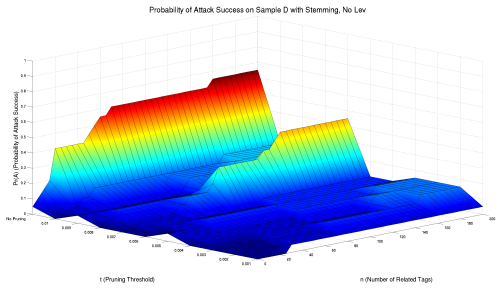
(d) Human Success Rates: Stemming and Inexact Matching

Figure C.5: The human success rates for Sample D.

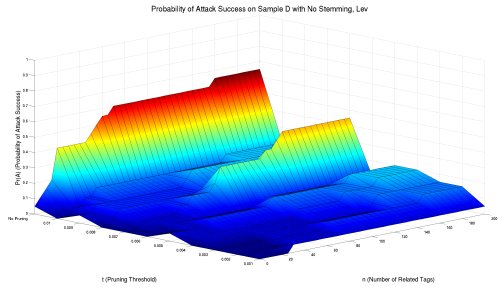
C.6 Security of Sample D



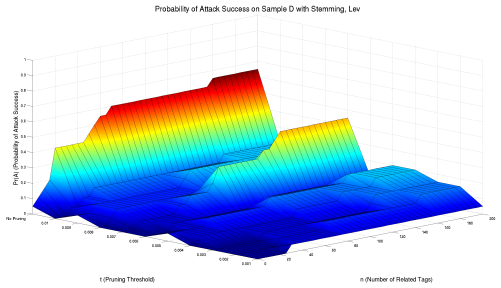
(a) Attack Success Rates: No Stemming and Exact Matching



(b) Attack Success Rates: Stemming and Exact Matching



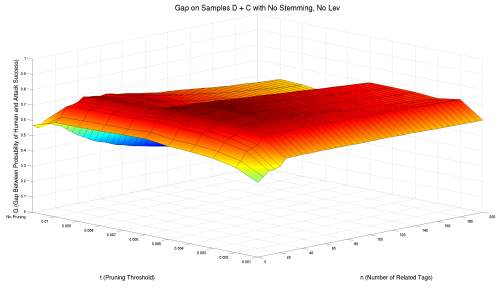
(c) Attack Success Rates: No Stemming and Inexact Matching



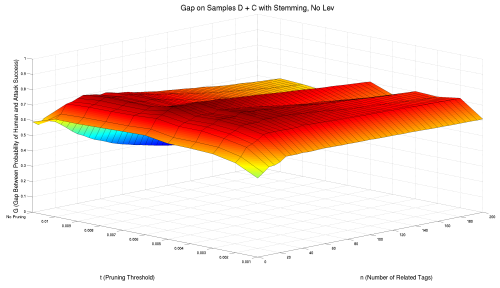
(d) Attack Success Rates: Stemming and Inexact Matching

Figure C.6: The attack success rates for Sample D.

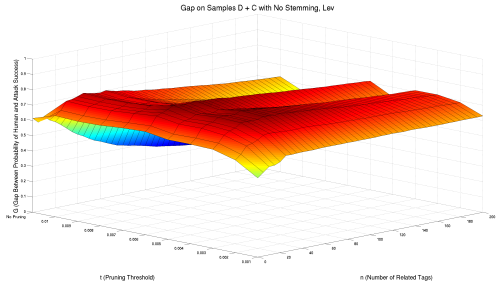
C.7 Gap of Samples D and C



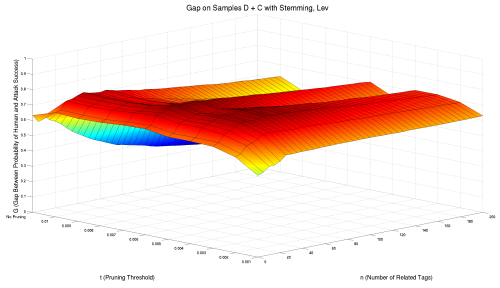
(a) Gap: No Stemming and Exact Matching



(b) Gap: Stemming and Exact Matching



(c) Gap: No Stemming and Inexact Matching



(d) Gap: Stemming and Inexact Matching

Figure C.7: The gap on between human success rate on Sample D and attack success rate on Sample C.

Vita

Kurt Alfred Kluever was born in New Milford, Connecticut on May 31, 1985, the son of Rolf Kurt Kluever and Janet Marie Kluever. He was raised in the small, rural town of Sherman, CT and attended high school in the neighboring town of New Fairfield, CT. In September 2003, he enrolled in the accelerated BS/MS Computer Science program at the Rochester Institute of Technology. During his academic career, Kurt interned with both Excellus BlueCross BlueShield in Rochester, NY and Google in Mountain View, CA. While attending graduate school, he was a Teaching Assistant for the Department of Computer Science and a Graduate Assistant in the Document and Pattern Recognition Lab. In September 2008, he will begin a full-time position with Google in New York, NY. Kurt's current contact information can be found online at `http://www.kloover.com/`.

Permanent Address: 447 Currier Road
East Falmouth, MA 02536 USA

This thesis was typeset with LaTeX 2$_\varepsilon$[1] by the author.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.