

Utilizing The Power of Human Cycles

(Thesis Proposal)

Luis von Ahn

Committee

Manuel Blum (Chair), Carnegie Mellon University

Josh Benaloh, Microsoft Research

Takeo Kanade, Carnegie Mellon University

Jitendra Malik, UC Berkeley

Mike Reiter, Carnegie Mellon University

Abstract

We propose several novel techniques for exploiting the computational abilities (or “cycles”) of humans.

One technique is CAPTCHA, an automated test that humans can pass but that current computer programs cannot. CAPTCHAs take advantage of the power of human cycles to differentiate people from computers. We propose several constructions of CAPTCHAs and we show that they have many applications in practical security — the results of our work, for instance, are used by Yahoo to ensure that only humans obtain free email accounts.

We also introduce techniques to harvest human cycles for the purpose of solving large-scale problems that computers cannot yet solve. There many people on the internet constantly doing things that computer programs cannot currently do: understanding images and sentences, chatting, passing CAPTCHA tests, etc. We show how to “reuse” or “steal” such human cycles to do useful work. We introduce a game, The ESP Game, that is fun — many people play over 40 hours a week! — and when people play they help determine the contents of images on the Web by providing meaningful labels for them. If the game is played as much as other popular online games, all images on the Web can be labeled in just a few weeks. Attaching proper labels to all images on the Web would allow for more accurate image search engines, would improve the accessibility of Web sites (by providing descriptions of images to visually impaired individuals), and would help Web browsers block

pornography. Our approach is simple but novel: rather than using computer vision techniques that don't work well enough, we encourage people to do the work for us by taking advantage of their desire to be entertained. In fewer than 6 months The ESP Game has collected over 3 million labels for images on the Web. The data collected by The ESP Game can also be used to improve computer vision algorithms.

We present other ideas to steal cycles from humans — a shooting game that helps computers locate specific objects in images, a memory game that helps decide whether two media objects are similar, etc.

Finally, we consider an alternative view of CAPTCHAS. Much like research in cryptography has had a positive impact on algorithms for factoring and discrete log, the use of difficult AI problems for security catalyzes the advancement of Artificial Intelligence. As an example, we introduce families of AI problems that can be used to construct CAPTCHAS and we show that solutions to some of these problems allow for *robust steganographic communication* and *watermarking*. This, coupled with the positive results of the computer vision community against distorted-text CAPTCHAS, suggests a novel approach for dealing with unsolved AI problems.

Chapter 1

CAPTCHA: Telling Humans and Computers Apart Automatically*

A CAPTCHA is a program that can generate and grade tests that: (A) most humans can pass, but (B) current computer programs can't pass. Such a program can be used to differentiate humans from computers and has many applications for practical security, including (but not limited to):

- **Online Polls.** In November 1999, *slashdot.org* released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting "bots". MIT finished with 21,156 votes, Carnegie Mellon with 21,032

*Adapted from von Ahn, L., Blum, M., Hopper, N. and Langford, J. CAPTCHA: Using Hard AI Problems for Security, in *EUROCRYPT 2003* and von Ahn, L., Blum, M. and Langford, J. CAPTCHA: Telling Humans and Computers Apart Automatically, in *Communications of the ACM, Feb. 2004*.

and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote.

- **Free Email Services.** Several companies (Yahoo!, Microsoft, etc.) offer free email services, most of which suffer from a specific type of attack: “bots” that sign up for thousands of email accounts every minute. This situation can be improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a CAPTCHA of our design to prevent bots from registering for accounts. Their CAPTCHA asks users to read a distorted word such as the one shown below (current computer programs are not as good as humans at reading distorted text).



Figure 1.1: **The Yahoo!** CAPTCHA.

- **Search Engine Bots.** Some web sites don’t want to be indexed by search engines. There is an *html* tag to prevent search engine bots from reading web pages, but the tag doesn’t guarantee that bots won’t read the pages; it only serves to say “no bots, please”. Search engine bots, since they usually belong to large companies, respect web pages that don’t want to allow them in. However, in order to truly *guarantee* that bots won’t enter a web site, CAPTCHAS are needed.
- **Worms and Spam.** CAPTCHAS also offer a plausible solution against email worms and spam: only accept an email if you know there is a human behind the other computer. A few companies, such as *www.spamarrest.com* are already marketing this idea.
- **Preventing Dictionary Attacks.** Pinkas and Sander [41] have suggested using CAPTCHAS to prevent dictionary attacks in password systems. The idea is simple:

prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

CAPTCHA stands for “Completely Automated Public Turing Test to Tell Computers and Humans Apart”. The P for Public means that the code and the data used by a CAPTCHA should be publicly available. This is not an open source requirement, but a security guarantee: it should be hard to write a computer program that can pass the tests generated by a CAPTCHA even if it’s known exactly how the CAPTCHA works (the only hidden information is a small amount of randomness utilized to generate the tests). The T for “Turing Test to Tell” is because CAPTCHAs are like Turing Tests [51]. In the original Turing Test, a human judge was allowed to ask a series of questions to two players, one of which was a computer and the other a human. Both players pretended to be the human, and the judge had to distinguish between them. CAPTCHAs are similar to the Turing Test in that they distinguish humans from computers, but they differ in that the judge is now a computer. A CAPTCHA is an *Automated Turing Test*. Notice that we do not use the term “*Reverse Turing Test*” (or even worse, “RTT”) because it can be misleading—“Reverse Turing Test” has been used to refer to a form of the Turing Test in which both players pretend to be a computer.

Examples of CAPTCHAs

CAPTCHAs also differ from the original Turing Test in that they can be based on a variety of sensory abilities. The original Turing Test was conversational—the judge was only allowed to ask questions over a text terminal. In the case of a CAPTCHA, the computer judge can ask any question that can go over a network.

- **Gimpy and OCR-based CAPTCHAs.** Humans are still better than computers at reading distorted characters. GIMPY [1] works as follows: it picks seven words out of a dictionary and renders a distorted image containing the words (as shown in Figure 1.2). GIMPY then presents a test to its user, which consists of the distorted image



Figure 1.2: Can you read 3 words in this image?

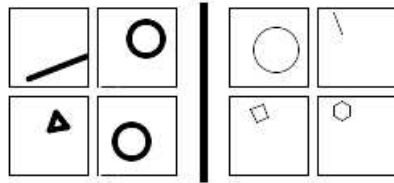


Figure 1.3: Everything on the LEFT is drawn with thick lines, while everything on the RIGHT is drawn with thin lines.

and the directions: “type three words appearing in the image”. Given the types of distortions that GIMPY uses, most humans can read three words from the distorted image, but current computer programs can’t. The majority of CAPTCHAs used on the Web today are similar to gimpy in that they rely on the difficulty of Optical Character Recognition (i.e., the difficulty of reading distorted text).

- **Bongo.** Another example of a CAPTCHA is the program we call BONGO [1]. BONGO is named after M.M. Bongard, who published a book of pattern recognition problems

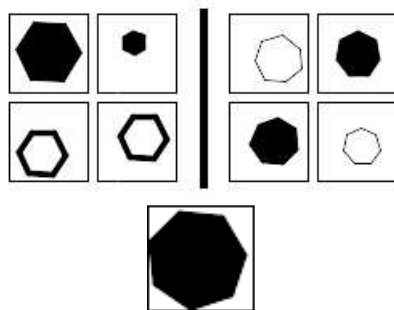


Figure 1.4: To which side does the block on the bottom belong?

([10]) in the 70s. BONGO asks the user to solve a visual pattern recognition problem. It displays two series of blocks, the LEFT and the RIGHT. The blocks in the LEFT series differ from those in the RIGHT, and the user must find the characteristic that sets them apart. A possible LEFT and RIGHT series are shown in Figure 1.3. After seeing the two series of blocks, the user is presented with a single block and is asked to determine whether this block belongs to the RIGHT series or to the LEFT. The user passes the test if he or she correctly determines the side to which the block belongs to. Try it yourself: to which side does the isolated block belong in Figure 1.4? (answer: RIGHT)

- **Pix.** PIX is a program that has a large database of labeled images. All of these images are pictures of concrete objects (a horse, a table, a house, a flower, etc). The program picks an object at random, finds 6 images of that object from its database, presents them to the user and then asks the question *“what are these pictures of?”* Current computer programs should not be able to answer this question, so PIX should be a CAPTCHA.

However, PIX, as stated, is not a CAPTCHA: it is very easy to write a program that can answer the question *“what are these pictures of?”* Remember that all the code and data of a CAPTCHA should be publicly available; in particular, the image database that PIX uses should be public. Hence, writing a program that can answer the question *“what are these pictures of?”* is easy: search the database for the images presented and find their label. Fortunately, this can be fixed. One way for PIX to become a CAPTCHA is to randomly distort the images before presenting them to the user, so that computer programs can not easily search the database for the undistorted image.

- **Sound-Based CAPTCHAs.** The final example that we mention is based on sound. The program picks a word or a sequence of numbers at random, renders the word or the numbers into a sound clip and distorts the sound clip; it then presents the

distorted sound clip to its user and asks them to enter its contents. This CAPTCHA is based on the difference in ability between humans and computers in recognizing spoken language. Nancy Chan of the City University in Hong Kong was the first to implement a sound-based system of this type.

It is extremely important to have CAPTCHAs based on a variety of sensory abilities. All CAPTCHAs presented so far, except for the sound-based CAPTCHA, rely on the user being able to see an image. However, there are many people who use the Web that are visually impaired. Thus, for accessibility purposes, CAPTCHAs based on sound are necessary. Unfortunately, images and sound alone are not sufficient: there are people who use the Web that are both visually and hearing impaired. The construction of a CAPTCHA based on a text domain such as text understanding or generation is an important open problem for the project.

Who Knows What?

Our definitions imply that an adversary attempting to write a program that has high success over a CAPTCHA knows exactly how the CAPTCHA works. The only piece of information that is hidden from the adversary is a small amount of randomness that the verifier uses in each interaction.

This choice greatly affects the nature of our definitions and makes the problem of creating CAPTCHAs more challenging. Imagine an Automated Turing Test that owns a large secret book written in English and to test an entity A it either picks a paragraph from its secret book or generates a paragraph using the best known text-generation algorithm, and then asks A whether the paragraph makes sense (the best text-generation algorithms cannot produce an entire paragraph that would make sense to a human being). Such an Automated Turing Test might be able to distinguish humans from computers (it is usually the case that the best text-generation algorithms and the best algorithms that try to determine whether something makes sense are tightly related). However, this test is not a CAPTCHA: an

adversary with knowledge of the secret book could achieve high success against this test without advancing the algorithmic state of the art.

Chapter 2

Stealing Cycles From Humans: The ESP Game*

Images on the Web present a major technological challenge. There are millions of them, there are no guidelines about providing appropriate textual descriptions for them, and computer vision hasn't yet produced a program that can determine their contents in a widely useful way. However, accurate descriptions of images are required by several applications like image search engines and accessibility programs for the visually impaired. Current techniques to categorize images for these applications are insufficient in many ways, mostly because they assume that the contents of images on the Web are related to the text appearing in the page. This is insufficient because the text adjacent to the images is often scarce, and even when it's there it can be misleading and hard to process.

The only method currently available for obtaining precise image descriptions is manual labeling, which is tedious and thus extremely costly. But, what if people labeled images without realizing they were doing so? What if the experience was enjoyable? We introduce a new interactive system in the form of a game with a unique property: the people who play the game label images for us.

*Adapted from von Ahn, L. and Dabbish, L. Labeling Images with a Computer Game, in *CHI 2004*.

The labels generated by our game can be useful for a variety of applications. For accessibility purposes, visually impaired individuals need textual descriptions of images to be read out loud. For computer vision research, large databases of labeled images are needed as training sets for machine learning algorithms. For image search over the Web and inappropriate (e.g., pornographic) content filtering, proper labels could dramatically increase the accuracy of the current systems. We believe our system makes a significant contribution, not only because of its valuable output, but also because of the way it addresses the image-labeling problem. Rather than making use of computer vision techniques, we take advantage of people’s existing perceptual abilities and desire to be entertained.

Our goal is extremely ambitious: to label the majority of images on the World Wide Web. If our game is deployed at a popular gaming site like Yahoo! Games and if people play it as much as other online games, we estimate that most images on the Web can be properly labeled in a matter of weeks. As we show below, 5,000 people playing the game for 24 hours a day would assign a label to all images indexed by Google in 31 days. We stress that our method is not necessarily meant to compete against the other techniques available for handling images on the Web. The labels produced using our game can usually be combined with these techniques to provide a powerful solution.

Our work is similar in spirit to the Open Mind Initiative (e.g., [48, 49]), which is a worldwide effort to develop “intelligent” software. Open Mind collects information from regular Internet users (referred to as “netizens”) and feeds it to machine learning algorithms. Volunteers participate by answering questions and teaching concepts to computer programs. Our method is similar to Open Mind in that we plan to use regular people on the Internet to label images for us. However, Open Mind has not considered the problem of labeling arbitrary images. In addition, we put greater emphasis on our method being fun because of the scale of the problem that we want to solve. We don’t expect volunteers to label all images on the Web for us: we expect all images to be labeled because people want to play our game.

General Description of The System

We call our system “the ESP game” for reasons that will become apparent as the description progresses. The game is played by two partners and is meant to be played online by a large number of pairs at once. Partners are randomly assigned from among all the people playing the game. Players are not told whom their partners are, nor are they allowed to communicate with their partners. The only thing partners have in common is an image that they can both see.

From the player’s perspective, the goal of the ESP game is to guess what their partner is typing on each image. Once both players have typed the same string, they move on to the next image (both player’s don’t have to type the string at the same time, but each must type the same string at some point while the image is on the screen). We call the process of typing the same string “agreeing on an image.”



Figure 2.1: The ESP Game. Players try to “agree” on as many images as they can in 2.5 minutes. The thermometer measures how many images partners have agreed on.

Partners have to agree on as many images as they can in 2.5 minutes (see Figure 2.1). Every time two partners agree on an image, they get a certain number of points. If they agree on 15 images they get a large number of bonus points. The thermometer at the bottom of the screen indicates the number of images that the partners have agreed on. By providing players with points for each image and bonus points for completing a set of images, we reinforce their incremental success in the game and thus encourage them to continue playing. Players can also choose to pass or opt out on difficult images. If a player clicks the pass button, a message is generated on their partner's screen; a pair cannot pass on an image until both have hit the pass button.

Since the players can't communicate and don't know anything about each other, the easiest way for both to type the same string is by typing something related to the common image. Notice, however, that the game doesn't ask the players to describe the image: all they know is that they have to "think like each other" and type the same string (thus the name "ESP"). It turns out that the string on which the two players agree is typically a good label for the image, as we will discuss in our evaluation section.

Taboo Words

A key element of the game is the use of taboo words associated with each image, or words that the players are not allowed to enter as a guess (see Figure 2.1). These words will usually be related to the image and make the game harder because they can be words that players commonly use as guesses.

Taboo words are obtained from the game itself. The first time an image is used in the game, it will have no taboo words. If the image is ever used again, it will have one taboo word: the word that resulted from the previous agreement. The next time the image is used, it will have two taboo words, and so on. (The current implementation of the game displays up to six different taboo words.)

Players are not allowed to type an image's taboo words, nor can they type singulars,

plurals or phrases containing the taboo words. The rationale behind taboo words is that often the initial labels agreed upon for an image are the most general ones (like “man” or “picture”), and by ruling those out the players will enter guesses that are more specific. Additionally, taboo words guarantee that each image will get many different labels associated with it.

Labels and Good Label Threshold

The words that we use as labels for images are the ones that players agree on. Although there is additional information that could be utilized (i.e., all other guesses that the players enter), for now such information will be ignored. We use only words that players agree on to ensure the quality of the labels: agreement by a pair of independent players implies that the label is probably meaningful. In fact, since these labels come from different people, they have the potential of being more robust and descriptive than labels that an individual indexer would have assigned [39].

To increase the probability that a label for a particular image is meaningful, we utilize a good label threshold. This means that before a label is attached to the image and used as a taboo word, it must have been agreed upon by at least X number of pairs, where X is the threshold. The threshold can be lenient and extremely low ($X=1$, one pair agreeing makes a label acceptable) or strict and high ($X=40$, forty pairs must have agreed on that label before it is attached to the image and made a taboo word). When is an Image “Done”?

As a particular image passes through the ESP game multiple times, it will accumulate several labels that people have agreed upon. The question is, at what point is an image considered to have been completely labeled and thus no longer used in the game? Our answer to this question is to remove an image from the game when it is no longer enjoyable to guess its contents with a partner. This will occur when a particular image has acquired an extensive list of taboo words, such that pairs are unable to agree on new labels and consistently ask their partners to pass on the image. Repeated passing notifies the system

that an image should no longer be used for the game at that point in time. (Repeated passing might also indicate that the image is too complex to be used in the game, in which case the image should also be removed.)

Fully labeled images are re-inserted into the game when several months have passed because the meaning of the images may have changed due to maturation effects. The English language changes over time, as do other languages [50]. We want to capture the labels appropriate to an image, and thus if the language referring to that image changes over time, so should our labels. In addition to changes in language, cultural changes may occur since a particular image has last been labeled. Thus a picture of something or someone that was labeled as “cool” or “great” six months prior may no longer be considered to be so. For example, an image of Michael Jackson twenty years ago might have been labeled as “superstar” whereas today it might be labeled as “criminal.”

Implementation and Other Details

The current version of the game is implemented as a Java applet and can be played at <http://www.espgame.org>. The applet connects to a centralized game server, which is responsible for the following: pairing up the players, providing each pair with a set of 15 different images and their corresponding taboo words, comparing the players’ guesses (currently, guesses can only be 13 characters long), and storing all the information. The game server starts a game every 30 seconds: when a new player logs in, it waits until the next 30-second boundary to pair them with another player and start their game. This is done to make sure that players get paired at random and cannot cheat by logging in at the same time as their friends.

The current implementation is complete except that only 350,000 images are available for playing (rather than all images on the Web). We currently use a good label threshold of $X=1$.

Pre-Recorded Game Play

Our implementation does not require two people to be playing at the same time: a single person can play with a pre-recorded set of actions as their “partner.” This set of actions is recorded from an earlier game session involving two people. For each image in the earlier session, every guess of each partner is recorded, along with timing information. We refer to the set of pre-recorded actions as the “bot.” Having pre-recorded game play is especially useful when the game is still gaining popularity. When there are few players, only a single person will usually be playing the game at a time.

Notice that pre-recorded game play does not necessarily stop the labeling process. If the single player and the bot agree on the label that was agreed on when the actions were recorded, we can increase our confidence regarding that label. If the single player and the bot match on another word, we get a brand new label.

Cheating

It is imperative that partners not be able to communicate with each other; otherwise agreeing on an image would be trivial. Similarly, players could cheat by being partnered with themselves or by agreeing on a unified strategy (for instance, a large group of players could agree to type “a” on every image; this could be achieved by posting this strategy on a popular website). The current implementation has several mechanisms in place to counter such cheating.

Notice first that no form of cheating is very likely: the game is meant to be played by hundreds, if not thousands, of people at once, most of which will be in distributed locations. Since players are randomly paired, they will have no information about who their partner is, and they will have no way to previously agree on a strategy. The probability of two cheaters using the same strategy being paired together should be low.

That being said, several additional steps are taken to minimize cheating. First, IP addresses of players are recorded and must be different from that of their partner to make

it difficult for players to be paired with themselves. Second, to counter global agreement of a strategy (e.g., “let’s all type ‘a’ for every image”), we use pre-recorded game-play. If a massive agreement strategy is detected, inserting a large number of bots acting out pre-recorded sets of actions will make cheating impossible. Once people realize that the massive agreement strategy doesn’t work, they should stop using it and we can lessen the use of pre-recorded game play. Massive global agreement of a strategy can be easily detected by measuring the average time in which players are agreeing on images: a sharp decrease in this average time should indicate massive agreement on a strategy. An alternative mechanism to prevent an agreement strategy is to enforce taboo words across an entire session. A pair’s answer to an image could become a taboo word for the duration of their session together. This, coupled with a good label threshold greater than one ($X > 1$) would also prevent global agreement of a strategy from corrupting our labels. If the strategy was to always type “a” for each image, it would only work for the first image in a session, as “a” would become a taboo word for the rest of the session. If the strategy was something more complicated, like “type ‘one’ for the first image, ‘two’ for the second, etc”, then the labels couldn’t be corrupted because of the good label threshold: in order for “one” to become a label for a certain image, the image would have to occur X times as the first image in games played by cheaters using the same strategy.

We also remark that some amount of cheating is acceptable for certain applications of our labels. In the case of image search, for instance, we expect to see an improvement over the current techniques even if some of the labels are meaningless. The current techniques, which associate most of the text on a website to each image, generate several inappropriate labels.

Selecting the Images

We believe that the choice of images used by the ESP game makes a difference in the player's experience. The game would be less entertaining if all the images were chosen from a single site and were all extremely similar.

A basic strategy for picking the images is to select them at random from the Web using a small amount of filtering. This is the strategy employed in the current implementation of the game, except for two minor differences. First, once an image is randomly chosen from the Web, we reintroduce it into the game several times until it is fully labeled. Second, rather than picking the images from the Web in an online fashion, we collected 350,000 images in advance and are waiting until those are fully labeled to start with the whole Web. The images were chosen using "Random Bounce Me" [42], a website that selects a page at random from the Google database [20]. "Random Bounce Me" was queried repeatedly, each time collecting all JPEG and GIF images in the random page, except for images that did not fit our criteria: blank images, images that consist of a single color, images that are smaller than 20 pixels on either dimension, and images with an aspect ratio greater than 4.5 or smaller than $1 / 4.5$. This process was repeated until 350,000 images were collected. The images were then rescaled to fit the game applet.

Spelling

The game server is equipped with a 73,000-word English dictionary that alerts players when they have misspelled a word. It does so by displaying the misspelled word in yellow rather than in white in the "Your Guesses" area. This is useful when one of the players doesn't know how to spell a word, or makes a typing mistake.

Extension: Context-Specific Labels

Presenting images randomly selected from the Web to a wide-ranging audience is likely to result in labels that are general. There might be more specific labels for some images, which could be obtained if the correct population of users was doing the labeling. For example, when presented with pictures of faculty members at a certain university, the average ESP game player might enter general words such as man, woman, person, etc. However, if the users playing the ESP game were all students at that university, they might input faculty member names.

In order to generate these kinds of specific labels for certain categories of images, we suggest the usage of “theme room” for the ESP game. These more specific theme rooms can be accessed by those who wish to play the ESP game using only certain types of images. Some players might want images from certain domains or with specific types of content (e.g., images of paintings). Images for these theme rooms can be obtained using either Web directories or the labels generated during the “general category” ESP game. The labels generated in such theme rooms are likely to be more specific and thus more appropriate for certain applications. In the “art” theme room, for instance, images of paintings could be labeled with the name of their creator, their title, and maybe even the year in which they were made.

Notice, however, that proper general labels will already provide a vast improvement for many applications. For the visually impaired, for example, knowing that an image has a man in it is better than not knowing anything about it. The current version of the game implements the “general category” ESP game.

Inappropriate Content

A small percentage of images on the Web are inappropriate for children (e.g., pornography). This means that the “general category” ESP game may also be inappropriate for children. Our suggested solution to this problem uses theme rooms as described above: children

would only be allowed to play the “children’s version” of the game. This version would obtain its images from the general category ESP game. Only images that have obtained a certain number of labels can be used in the children’s version; all of the labels for these images must be “safe.” To be more rigorous, we can combine this with text-based filtering. Images coming from web pages containing inappropriate words, etc., would not be allowed. We believe these strategies would prevent inappropriate images from reaching the children’s version. Notice also that the percentage of freely accessible images on the Web that are pornographic is small (the exact percentage of such images is hard to estimate, and varies depending on the source). Our game only displays images that are freely accessible.

Evaluation

We present data supporting our claims that people will want to play the ESP game and that the labels it produces are useful. In general it is difficult to predict if a game will become popular. One approach, which we followed early on, is to ask participants a series of questions regarding how much they enjoyed playing the game. Our data were extremely positive, but we follow a different approach here: we present usage statistics from arbitrary people playing our game on the Web. We also present evidence that the labels produced using the game are useful descriptions of the images. It’s not the case that players must input words describing the images: players are never asked to describe anything. We show, however, that players do input words describing the images. To do so, we present the results of searching for randomly chosen keywords and show that the proportion of appropriate images when searching using the labels generated by the game is extremely high. In addition, we present the results of a study that compares the labels generated using the game to labels generated by participants that were asked to describe the images.

Usage Statistics

For the past four months we have been running the ESP game over the Web, allowing independent users to sign up for accounts and play the game. The game was first posted on the website on August 9 of 2003 and the statistics here presented are for the four-month period ending on December 10. A total of 13,630 people played the game during this time, generating 1,271,451 labels for 293,760 different images. Over 80% of the people played on more than one occasion (i.e., more than 80% of the people played on multiple dates). Furthermore, 33 people played more than 1,000 games (this is over 50 hours of playing!). We believe these numbers provide evidence that the game is fun: almost 1.3 million labels were collected with only 13,630 players, some of whom spent over 50 hours playing the game!

Labeling Rate

The usage statistics also allowed us to determine the rate at which images are labeled using the game. The average number of labels collected per minute by a pair of individuals is 3.89 (std. dev. = 0.69). At this rate, 5,000 people playing the ESP game 24 hours a day would label all images on Google (425,000,000 images) in 31 days. This would only associate one word to each image. In 6 months, 6 words could be associated to every image. Notice that this is a perfectly reasonable estimate: on a recent weekday afternoon, the authors found 107,000 people playing in Yahoo! Games [54], 115,000 in MSN's The Zone [32] and 121,000 in Pogo.com [17]. A typical game on these sites averages well over 5,000 people playing at any one time. The time it takes players to agree on an image depends on the number of taboo words associated with the image. Our calculation of the labeling rate, however, is independent of the number of taboo words: every session of the game has roughly the same number of images with 0 taboo words, the same number of images with 1 taboo word, etc.

Quality of the Labels

We provide evidence that players input appropriate labels for the images, even though their primary goal is to maximize their score. We show the results of three distinct evaluations. The first is a measure of precision when using the labels as search queries. The second compares the labels generated using the game to labels generated by experimental participants asked to describe the images. The third consists of asking experimental participants whether the labels generated using the game were appropriate with respect to the images.

Search Precision

We performed an evaluation similar to that in [28]: we examined the results of searching for all images associated to particular labels. To do so, we chose 10 labels at random from the set of all labels collected using the game. We chose from labels that occurred in more than 8 images.

All of the images retrieved made sense with respect to the test labels. In more technical terms, the precision of searching for images using our labels is extremely high. This should be surprising, given that the labels were collected not by asking players to enter search terms, but by recording their answers as they tried to maximize their score in the ESP game.

Comparison to Labels Generated by Participants Asked to Describe the Images

To further determine whether the words that players agreed on were actually describing the image, we asked 15 participants to input word descriptions of images and we compared their descriptions to the labels generated by the game. The participants were between 20 and 25 years of age and had not played the game during the trial period.

Method. Twenty images were chosen at random out of the first 1,023 images that had more than 5 labels associated to them by the game (1,023 is the number of images that

had more than 5 labels associated to them at the time this experiment was performed). All 15 participants were presented with each of the 20 images in randomized order. For each image, the participant was asked to do the following:

Please type the six individual words that you feel best describe the contents of this image. Type one word per line below; words should be less than 13 characters.

Results. The results indicate that indeed players of the ESP game were generating descriptions of the images. For all (100%) of the 20 images, at least 5 (83%) of the 6 labels produced by the game were covered by the participants (i.e., each of these labels was entered by at least one participant). Moreover, for all (100%) of the images, the three most common words entered by participants were contained among the labels produced by the game.

Manual Assessment of the Labels

In addition to the previous evaluations, we had 15 participants rate the quality of the labels generated using the game. The participants were chosen as independent raters because they had not played the ESP game. None of the participants of this evaluation took part in the previous one and vice-versa. Participants were 20 to 25 years of age.

Method. Twenty images were chosen at random out of the first 1,023 images that had more than 5 labels associated to them by the game. All 15 participants were presented with each of the 20 images in randomized order. For each image the participant was shown the first six words that were agreed on for that image during the game. For each of the 20 image-word sets they were asked to answer the following questions:

1. How many of the words above would you use in describing this image to someone who couldn't see it.
2. How many of the words have nothing to do with the image (i.e., you don't understand why they are listed with this image)?

Results. For question 1, the mean was 5.105 words (std. dev. 1.0387), indicating that a majority (or 85%) of the words for each image would be useful in describing it. The mean for question 2 was 0.105 words (std. dev. 0.2529), indicating that for the most part subjects felt there were few (1.7%) if any labels that did not belong with each image.

Previous Techniques for Processing Images

To this point, we have presented a method for labeling images on the Web and we have presented evidence that it does indeed produce high-quality labels. There are a variety of other techniques for processing images on the Web, all of which are different in nature from ours. We now survey the different techniques and contrast them with our method.

Computer Vision

There has been considerable work in computer vision related to automatically labeling images. The most successful approaches learn from large databases of annotated images. Annotations typically refer to the contents of the image, and are fairly specific and comprehensive. Methods such as [6, 7] cluster image representations and annotations to produce a joint distribution linking images and words. These methods can predict words for a given image by computing the words that have a high posterior probability given the image. Other methods attempt to combine large semantic text models with annotated image structures [16]. Though impressive, such algorithms based on learning don't work very well in general settings and work only marginally well in restricted settings. For example, the work described in [16] only gave reasonable results for 80 out of their 371 vocabulary words (their evaluation consisted of searching for images using the vocabulary words, and only 80 of the words resulted in reasonable images).

A different line of research attempts to find specific objects in images. [46], for instance, introduced a method to locate human faces in still photographs. These algorithms are

typically accurate, but have not been developed for a wide range of objects. Additionally, combining algorithms for detecting specific objects into a single general-purpose classifier is a non-trivial task.

The ESP game provides a possible solution to the image-labeling problem, but having a computer program that can label images remains a more important goal. One application of the ESP game is in the creation of such a program: the limitations of the current computer vision techniques partly arise from the lack of large databases of annotated images. These databases could be constructed using methods similar to our game.

Image Search on the Web

Finding effective methods to search and retrieve images on the Web has been a prevalent line of research, both academically ([14, 28]) and in industry ([4, 20]). Text-based image retrieval systems such as [4] annotate images with text derived from the HTML documents that display them. The text can include the caption of the image, text surrounding the image, the entire text of the containing page, the filename of the containing HTML document, and the filename of the image itself. More recent proposals such as [20, 28] also make use of the link structure of the Web to assign "authority" values to the images. Images that come from more authoritative web pages (e.g., pages with higher PageRank [20]) are displayed before images coming from less authoritative pages. This improves the quality of the results by typically showing more relevant images first. Another possibility that has been explored involves combining text-based systems with computer vision techniques as in [14]. This approach allows different types of queries to be processed (e.g., similarity queries), but doesn't imply a significant improvement over the other approaches when it comes to standard text-based queries.

The fundamental limitation of current methods for image retrieval on the Web is the heavy use of text to determine the contents of images. Text adjacent to the images is often scarce, and can be misleading or hard to process [12]. Because of this, many queries return

inappropriate results. We argue that our game can improve the quality of image retrieval systems by providing meaningful labels that are independent of the text contained in the web pages.

Inappropriate Content Filters

Inappropriate content filters (e.g. [37]) attempt to block certain images from being displayed. Typically these filters try to block pornographic sites from reaching children at home or employees in the workplace. Since computer vision techniques for this purpose are not highly accurate [18], content filters usually analyze the text inside web pages to determine whether they should be blocked.

Most filters are reasonably accurate, but have several flaws. First, they only work for a few languages and in most cases only work for pages in English. Second, they work poorly when the pages don't contain any "incriminating" text: e.g., a page with a nude image and nothing else in it would not be correctly identified. For this reason, in order to ensure that inappropriate content does not get posted, dating services and websites that allow users to post images have to hire people to look over every single picture to be posted. Third, content filters have to be constantly updated: imagine what would happen when a new porn star named Thumbelina comes out; suddenly every search for "Thumbelina" would return some pornography.

Google Image Search [20] offers a content filter (called SafeSearch), which attempts to block all inappropriate images from being displayed in their search results. At the time of writing this proposal, a query for "interracial" returns several inappropriate images (and a more direct query like "wet tshirt" returns even more inappropriate results). We argue that having proper labels associated to each freely available image on the Web would improve content filtering technology.

Using Our Labels

The work thus far has been primarily concerned with obtaining appropriate labels for images, and not with how these labels should be used. In the case of image search, building the labels into the current systems is not difficult, since they can be thought of as HTML captions or text appearing right next to the image. This naive strategy would already signify an improvement over the current techniques, as these captions would provide more useful data to work with. More intelligent techniques could be conceived, such as assigning a higher weight to labels coming from the ESP game as opposed to regular HTML captions, or a numerical weight based on the "good label threshold". However, arriving at an optimal strategy for using the labels is left as future work.

In the case of providing textual descriptions for the visually impaired, using the labels is slightly less trivial. Our game produces labels, not explanatory sentences. While keyword labels are perfect for certain applications such as image search, other applications such as accessibility would benefit more from explanatory sentences. Nevertheless, having meaningful labels associated to images for accessibility purposes is certainly better than having nothing. Today's screen-reading programs for the visually impaired use only image filenames and HTML captions when attempting to describe images on the Web - the majority of images on the Web, however, have no captions or have non-descriptive filenames [33]. We propose that all the labels collected using the game be available for use with screen readers and that users determine themselves how many labels they want to hear for every image. Again, extensive tests are required to determine the optimal strategy.

Conclusion and Future Work

The ESP game is a novel interactive system that allows people to label images while enjoying themselves. We have presented evidence that people will play our game and that the labels it produces are meaningful. Our data also suggest that 5,000 people playing the game for

24 hours a day would enable us to label all images indexed by Google in a matter of weeks. This is striking because 5,000 is not a large number: most popular games on the Web have more than 5,000 players at any one time. Having proper labels associated to each image on the Web could allow for more accurate image retrieval, could improve the accessibility of sites, and could help users block inappropriate images.

Although the main application of the ESP game is to label images, our main contribution stems from the way in which we attack the labeling problem. Rather than developing a complicated algorithm, we have shown that it's conceivable that a large-scale problem can be solved with a method that uses people playing on the Web. We've turned tedious work into something people want to do.

We propose to attack other problems in a similar fashion. For instance, the ESP game can be used, with only minor modifications, to label sound or video clips (i.e., there is nothing inherent about images). Of course, the success of these variations of the ESP game depends on whether people will enjoy playing them. The same mechanism can also be used to attach labels to images in other languages. Other problems that could be solved by "stealing cycles from humans" include:

- **Monitoring security cameras.** One of the main stumbling blocks for installing more security cameras around the world is that it's extremely expensive to pay humans to watch the cameras 24 hours a day. What if people played a game that could alert somebody when illegal activity was going on? What if the output of security cameras was displayed as a screen-saver in millions of computers?
- **Locating specific objects in images.** The ESP Game can be used to determine whether an image contains a horse, for instance, but cannot be used to determine *where* in the image the horse is located. Such location information can be extremely useful as training data for computer vision algorithms. We are currently working with Manuela Veloso on a game that can be used to determine the location of ball in an

image — such information will be used by Manuela’s team as training data for the Robo Soccer agents.

- **Judging the Quality of Media.** Humans are extremely good at determining whether an image or a soundclip is “good”: is it not blurry or noisy, is it pleasing, are the objects in the image easy to see, etc. Such information can be extremely useful for search engines, which should return high quality results first. We are currently working on a game that allows us to determine the quality of an image, sound or video clip.
- **Similarity of Images.** Being able to decide whether two images are “similar” to each other has many applications. The system described in [26], for instance, assumes the existence of a large database of similarities between arbitrary images. Building such a large database is a difficult task, but turning the process into a game can solve the problem.

One could imagine many other applications and we hope that others may be inspired to develop systems similar in approach to the ESP game.

Chapter 3

The Flip Side of CAPTCHAs: Advancing AI and AI-Related Security*

We now return to CAPTCHA and show the flip side: breaking a CAPTCHA is as good and possibly even better than not being able to break it. CAPTCHAs are based on open problems in AI, and being able to break them implies progress in the field. In this Chapter we show families of CAPTCHAs with the following property: any program that breaks such CAPTCHAs can be used to do robust image-based steganography. This Chapter is more technical and formal than the previous ones, but for a good reason: we will *prove* that the programs that break the CAPTCHAs can be used for good; such proofs require a formalization of some of the concepts treated thus far.

We start by noting that, from a mechanistic point of view, there is no way to *prove* that a program cannot pass a test which a human can pass, since there is a program — the human brain — which passes the test. All we can do is to present evidence that it's

*Adapted from Hopper, N., Langford, J. and Von Ahn, L. Provably Secure Steganography, in *CRYPTO 2002* and von Ahn, L., Blum, M., Hopper, N. and Langford, J. CAPTCHA: Using Hard AI Problems for Security, in *EUROCRYPT 2003*.

hard to write a program that can pass the test. In this Chapter, we take an approach familiar to cryptographers: investigate state-of-the-art algorithmic developments having to do with some problem, assume that the adversary does not have algorithms for that problem that are much better than the state-of-the-art algorithms, and then prove a reduction between passing a test and exceeding the performance of state-of-the-art algorithms. In the case of ordinary cryptography, it is assumed (for example) that the adversary cannot factor 1024-bit integers in any reasonable amount of time. In our case, we assume that the adversary cannot solve an Artificial Intelligence problem with higher accuracy than what's currently known to the AI community. This approach, if it achieves widespread adoption, has the beneficial side effect of inducing security researchers, as well as otherwise malicious programmers, to advance the field of AI (much like computational number theory has been advanced since the advent of modern cryptography).

A CAPTCHA is a cryptographic protocol whose underlying hardness assumption is based on an AI problem.

An important component of the success of modern cryptography is the practice of stating, very precisely and clearly, the assumptions under which cryptographic protocols are secure. This allows the rest of the community to evaluate the assumptions and to attempt to break them. In the case of Artificial Intelligence, it's rare for problems to be precisely stated, but using them for security purposes forces protocol designers to do so. We believe that precisely stating unsolved AI problems can accelerate the development of Artificial Intelligence: most AI problems that have been precisely stated and publicized have eventually been solved (take chess as an example). For this reason it makes practical sense for AI problems that are used for security purposes to also be *useful*. If the underlying AI problem is useful, a CAPTCHA implies a win-win situation: either the CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and a useful AI problem is solved. Such is not the case for most other cryptographic assumptions: the

primary reason algorithms for factoring large numbers are useful is because factoring has applications in cryptanalysis.

Definitions and Notation

Let \mathcal{C} be a probability distribution. We use $[\mathcal{C}]$ to denote the support of \mathcal{C} . If $P(\cdot)$ is a probabilistic program, we will denote by $P_r(\cdot)$ the deterministic program that results when P uses random coins r .

Let (P, V) be a pair of probabilistic interacting programs. We denote the output of V after the interaction between P and V with random coins u_1 and u_2 , assuming this interaction terminates, by $\langle P_{u_1}, V_{u_2} \rangle$ (the subscripts are omitted in case the programs are deterministic). A program V is called a *test* if for all P and u_1, u_2 , the interaction between P_{u_1} and V_{u_2} terminates and $\langle P_{u_1}, V_{u_2} \rangle \in \{\text{accept}, \text{reject}\}$. We call V the *verifier* or *tester* and any P which interacts with V the *prover*.

Definition 1 Define the success of an entity A over a test V by

$$\mathbf{Succ}_A^V = \Pr_{r, r'}[\langle A_r, V_{r'} \rangle = \text{accept}].$$

We assume that A can have precise knowledge of how V works; the only piece of information that A can't know is r' , the internal randomness of V .

CAPTCHA

Intuitively, a CAPTCHA is a test V over which most humans have success close to 1, and for which *it is hard to write a computer program that has high success over V* . We will say that it is hard to write a computer program that has high success over V if *any program that has high success over V can be used to solve a hard AI problem*.

Definition 2 A test V is said to be (α, β) -human executable if at least an α portion of the

human population has success greater than β over V .

Notice that a statement of the form “ V is (α, β) -human executable” can only be proven empirically. Also, the success of different groups of humans might depend on their origin language or sensory disabilities: color-blind individuals, for instance, might have low success on tests that require the differentiation of colors.

Definition 3 An AI problem is a triple $\mathcal{P} = (S, D, f)$, where S is a set of problem instances, D is a probability distribution over the problem set S , and $f : S \rightarrow \{0, 1\}^*$ answers the instances. Let $\delta \in (0, 1]$. We require that for an $\alpha > 0$ fraction of the humans H , $\Pr_{x \leftarrow D}[H(x) = f(x)] > \delta$.

Definition 4 An AI problem \mathcal{P} is said to be (δ, τ) -solved if there exists a program A , running in time at most τ on any input from S , such that

$$\Pr_{x \leftarrow D, r} [A_r(x) = f(x)] \geq \delta .$$

(A is said to be a (δ, τ) solution to \mathcal{P} .) \mathcal{P} is said to be a (δ, τ) -hard AI problem if no current program is a (δ, τ) solution to \mathcal{P} , and the AI community agrees it is hard to find such a solution.

Definition 5 A (α, β, η) -CAPTCHA is a test V that is (α, β) -human executable, and which has the following property:

There exists a (δ, τ) -hard AI problem \mathcal{P} and a program A , such that if B has success greater than η over V then A^B is a (δ, τ) solution to \mathcal{P} . (Here A^B is defined to take into account B 's running time too.)

We stress that V should be a *program* whose code is publicly available.

Remarks

1. The definition of an AI problem as a triple (S, D, f) should not be inspected with a philosophical eye. We are not trying to capture all the problems that fall under the umbrella of Artificial Intelligence. We want the definition to be easy to understand, we want some AI problems to be captured by it, and we want the AI community to agree that these are indeed hard AI problems. More complex definitions can be substituted for Definition 3 and the rest of the work remains unaffected.
2. A crucial characteristic of an AI problem is that a certain fraction of the human population be able to solve it. Notice that we don't impose a limit on how long it would take humans to solve the problem. All that we require is that some humans be able to solve it (even if we have to assume they will live hundreds of years to do so). The case is not the same for CAPTCHAS. Although our definition says nothing about how long it should take a human to solve a CAPTCHA, it is preferable for humans to be able to solve CAPTCHAS in a very short time. CAPTCHAS which take a long time for humans to solve are probably useless for all practical purposes.

AI Problems as Security Primitives

Notice that we define *hard* in terms of the consensus of a community: an AI problem is said to be hard if the people working on it agree that it's hard. This notion should not be surprising to cryptographers: the security of most modern cryptosystems is based on assumptions agreed upon by the community (e.g., we *assume* that 1024-bit integers can't be factored). The concept of a hard AI problem as a foundational assumption, of course, is more questionable than $P \neq NP$, since many people in the AI community agree that all hard AI problems are eventually going to be solved. However, hard AI problems may be a more reasonable assumption than the hardness of factoring, given the possibility of constructing a quantum computer. Moreover, even if factoring is shown to be hard in an

asymptotic sense, picking a concrete value for the security parameter usually means making an assumption about current factoring algorithms: we only *assume* that current factoring algorithms that run in current computers can't factor 1024-bit integers. In the same way that AI researchers believe that all AI problems will be solved eventually, we believe that at some point we will have the computational power and algorithmic ability to factor 1024-bit integers. (Shamir and Tromer [45], for instance, have proposed a machine that could factor 1024-bit integers; the machine would cost about ten million dollars in materials.)

An important difference between popular cryptographic primitives and AI problems is the notion of a security parameter. If we believe that an adversary can factor 1024-bit integers, we can use 2048-bit integers instead. No such concept exists in hard AI problems. AI problems, as we have defined them, do not deal with asymptotics. However, as long as there is a small gap between human and computer ability with respect to some problem, this problem can potentially be used as a primitive for security: rather than asking the prover to solve the problem once, we can ask it to solve the problem twice. If the prover gets good at solving the problem twice, we can ask it to solve the problem three times, etc.

There is an additional factor that simplifies the use of hard AI problems as security primitives. Most applications of CAPTCHAs require the tests to be answered within a short time after they are presented. If a new program solves the hard AI problems that are currently used, then a different set of problems can be used, and the new program cannot affect the security of applications that were run before it was developed. Compare this to encryption schemes: in many applications the information that is encrypted must remain confidential for years, and therefore the underlying problem must be hard against programs that run for a long time, and against programs that will be developed in the future.[†]

We also note that not all hard AI problems can be used to construct a CAPTCHA. In order for an AI problem to be useful for security purposes, there needs to be an *automated* way to generate problem instances along with their solution. The case is similar for computational

[†]We thank one of our anonymous Eurocrypt reviewers for pointing this out.

problems: not all hard computational problems yield cryptographic primitives.

Two AI Problem Families

In this section we introduce two families of AI problems that can be used to construct CAPTCHAs. The section can be viewed as a precise statement of the kind of hardness assumptions that our cryptographic protocols are based on. We stress that solutions to the problems will also be shown to be useful.

For the purposes of this work, we define an *image* as an $h \times w$ matrix (h for height and w for width) whose entries are *pixels*. A pixel is defined as a triple of integers (R, G, B) , where $0 \leq R, G, B \leq M$ for some constant M . An *image transformation* is a function that takes as input an image and outputs another image (not necessarily of the same width and height). Examples of image transformations include: turning an image into its black-and-white version, changing its size, etc.

Let \mathcal{I} be a distribution on images and \mathcal{T} be a distribution on image transformations. We assume for simplicity that if $i, i' \in [\mathcal{I}]$ and $i \neq i'$ then $T(i) \neq T'(i')$ for any $T, T' \in [\mathcal{T}]$. (Recall that $[\mathcal{I}]$ denotes the support of \mathcal{I} .)

Problem Family 1 (P1) *Consider the following experiment: choose an image $i \leftarrow \mathcal{I}$ and choose a transformation $t \leftarrow \mathcal{T}$; output $t(i)$. $\mathcal{P}_{1, \mathcal{I}, \mathcal{T}}$ consists of writing a program that takes $t(i)$ as input and outputs i (we assume that the program has precise knowledge of \mathcal{T} and \mathcal{I}). More formally, let $S_{\mathcal{I}, \mathcal{T}} = \{t(i) : t \in [\mathcal{T}] \text{ and } i \in [\mathcal{I}]\}$, $D_{\mathcal{I}, \mathcal{T}}$ be the distribution on $S_{\mathcal{I}, \mathcal{T}}$ that results from performing the above experiment and $f_{\mathcal{I}, \mathcal{T}} : S_{\mathcal{I}, \mathcal{T}} \rightarrow [\mathcal{I}]$ be such that $f_{\mathcal{I}, \mathcal{T}}(t(i)) = i$. Then $\mathcal{P}_{1, \mathcal{I}, \mathcal{T}} = (S_{\mathcal{I}, \mathcal{T}}, D_{\mathcal{I}, \mathcal{T}}, f_{\mathcal{I}, \mathcal{T}})$.*

Problem Family 2 (P2) *In addition to the distributions \mathcal{I} and \mathcal{T} , let L be a finite set of “labels”. Let $\lambda : [\mathcal{I}] \rightarrow L$ compute the label of an image. The set of problem instances is $S_{\mathcal{I}, \mathcal{T}} = \{t(i) : t \in [\mathcal{T}] \text{ and } i \in [\mathcal{I}]\}$, and the distribution on instances $D_{\mathcal{I}, \mathcal{T}}$ is the one induced by choosing $i \leftarrow \mathcal{I}$ and $t \leftarrow \mathcal{T}$. Define $g_{\mathcal{I}, \mathcal{T}, \lambda}$ so that $g_{\mathcal{I}, \mathcal{T}, \lambda}(t(i)) = \lambda(i)$. Then*

$\mathcal{P}_{2,\mathcal{I},\mathcal{T},\lambda} = (S_{\mathcal{I},\mathcal{T}}, D_{\mathcal{I},\mathcal{T}}, g_{\mathcal{I},\mathcal{T},\lambda})$ consists of writing a program that takes $t(i)$ as input and outputs $\lambda(i)$.

Remarks

1. Note that a (δ, τ) solution A to an instance of $\mathcal{P}1$ also yields a $(\delta', \tau + \tau')$ solution to an instance of $\mathcal{P}2$ (where $\delta' \geq \delta$ and $\tau' \leq \log \|\mathcal{I}\|$ is the time that it takes to compute λ), specifically, computing $\lambda(A(x))$. However, this may be unsatisfactory for small δ , and we might hope to do better by restricting to a smaller set of labels. Conversely, $\mathcal{P}1$ can be seen as a special case of $\mathcal{P}2$ with λ the identity function and $L = \mathcal{I}$. Formally, problem families $\mathcal{P}1$ and $\mathcal{P}2$ can be shown to be isomorphic. Nonetheless, it is useful to make a distinction here because in some applications it appears unnatural to talk about labels.
2. We stress that in all the instantiations of $\mathcal{P}1$ and $\mathcal{P}2$ that we consider, \mathcal{I}, \mathcal{T} and, in the case of $\mathcal{P}2$, λ , will be such that humans have no problem solving $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ and $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$. That is, \mathcal{I} is a set of transformations that humans can easily undo in \mathcal{I} . Additionally, it has to be possible to perform all the transformations in \mathcal{I} using current computer programs.
3. There is nothing specific to *images* about these problem definitions; any other space of objects which humans recognize under reasonable transformations (e.g., organized sounds such as music or speech, animations, *et cetera*) could be substituted without changing our results.
4. It is easy to build a $(\delta_{\mathcal{I}}, \tau_{\mathcal{I}})$ -solution to $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$, where $\delta_{\mathcal{I}} = \max\{\Pr_{j \leftarrow \mathcal{I}}[j = i] : i \in \mathcal{I}\}$ and $\tau_{\mathcal{I}}$ is the time that it takes to describe an element of \mathcal{I} , by always guessing the image with the highest probability in \mathcal{I} . Similarly, it is easy to build a $(\delta_{\mathcal{I},\lambda}, \tau_{\mathcal{I},\lambda})$ -solution to $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$, where $\delta_{\mathcal{I},\lambda} = \max\{\Pr_{j \leftarrow \mathcal{I}}[\lambda(j) = \lambda(i)] : i \in \mathcal{I}\}$ and $\tau_{\mathcal{I},\lambda}$ is the time that it takes to describe a label in L . Therefore, we restrict our attention to

finding solutions to $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ where $\delta > \delta_{\mathcal{I}}$ and solutions to $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$ where $\delta > \delta_{\mathcal{I},\lambda}$.

Hard Problems in $\mathcal{P}1$ and $\mathcal{P}2$

We believe that $\mathcal{P}1$ and $\mathcal{P}2$ contain several hard problems. For example, the CAPTCHA shown in Section 1 (and other CAPTCHAs based on the difficulty of reading slightly distorted text) could be defeated using solutions to $\mathcal{P}2$. To see this, let W be a set of images of words in different fonts. All the images in W should be undistorted and contain exactly one word each. Let \mathcal{I}_W be a distribution on W , let \mathcal{T}_W be a distribution on image transformations, and let λ_W map an image to the word that is contained in it. A solution to $\mathcal{P}2_{\mathcal{I}_W,\mathcal{T}_W,\lambda_W}$ is a program that can defeat a CAPTCHA such as the one that Yahoo! uses (assuming \mathcal{T}_W is the same set of transformations they use). So the problem of determining the word in a distorted image is an instantiation of $\mathcal{P}2$ (it can be easily seen to be an instantiation of $\mathcal{P}1$ too). Reading slightly distorted text has been an open problem in machine vision for quite some time. (For a good overview of the difficulties of reading slightly distorted text, see [43].)

But $\mathcal{P}1$ and $\mathcal{P}2$ are much more general, and reading slightly distorted text is a somewhat easy instance of these problems. In general it will not be the case that the problem is reduced to matching $26 * 2 + 10$ different characters (upper and lowercase letters plus the digits).

The hardness of problems in $\mathcal{P}1$ and $\mathcal{P}2$ mostly relies on \mathcal{T} . In particular, it should be computationally infeasible to enumerate all of the elements of $[\mathcal{T}]$, since \mathcal{I} will normally be such that enumeration of $[\mathcal{I}]$ is feasible. Thus we are mainly interested in (δ, τ) solutions where $\tau \ll |[\mathcal{T}]|$, while $\tau > |[\mathcal{T}]|$ may sometimes be acceptable. In addition to the size of the transformation set, the character of the transformations is also important: it is necessary to defeat many simple checks such as color histogram comparisons, frequency domain checks, etc.

Since instantiations of $\mathcal{P}1$ and $\mathcal{P}2$ have never been precisely stated and published as challenges to the AI and security communities, there is no way to tell if they will withstand

the test of time. For now we refer the reader to *www.captcha.net* for examples of \mathcal{I} 's and \mathcal{T} 's which are believed to be good candidates. Any instantiation of $\mathcal{P}1$ and $\mathcal{P}2$ for security purposes requires that the precise \mathcal{I} and \mathcal{T} be published and thoroughly described.

Two Families of CAPTCHAs

We now describe two families of CAPTCHAs whose security is based on the hardness of problems in $\mathcal{P}1$ and $\mathcal{P}2$. Notice that if $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ is (δ, τ) -hard then $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ can be used to construct a CAPTCHA trivially: the verifier simply gives the prover $t(i)$ and asks the prover to output i . According to our definition, this would be a perfectly valid CAPTCHA. However, it would also be a very impractical one: if $[\mathcal{I}]$ is large, then humans would take a long time to answer. The CAPTCHAs we present in this section can be quickly answered by humans. The first family of CAPTCHAs, MATCHA, is somewhat impractical, but the second family, PIX, is very practical and in fact several instantiations of it are already in use.

MATCHA

A MATCHA instance is described by a triple $M = (\mathcal{I}, \mathcal{T}, \tau)$, where \mathcal{I} is a distribution on images and \mathcal{T} is a distribution on image transformations that can be easily computed using current computer programs. MATCHA is a CAPTCHA with the following property: *any program that has high success over $M = (\mathcal{I}, \mathcal{T})$ can be used to solve $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$.*

The MATCHA verifier starts by choosing a transformation $t \leftarrow \mathcal{T}$. It then flips a fair unbiased coin. If the result is heads, it picks $k \leftarrow \mathcal{I}$ and sets $(i, j) = (k, k)$. If the result is tails, it sets $j \leftarrow \mathcal{I}$ and $i \leftarrow U([\mathcal{I}] - \{j\})$ where $U(S)$ is the uniform distribution on the set S . The MATCHA verifier sends the prover $(i, t(j))$ and sets a timer to expire in time τ ; the prover responds with $res \in \{0, 1\}$. Informally, $res = 1$ means that $i = j$, while $res = 0$ means that $i \neq j$. If the verifier's timer expires before the prover responds, the verifier rejects. Otherwise, the verifier makes a decision based on the prover's response res and

whether i is equal to j :

- If $i = j$ and $res = 1$, then MATCHA accepts.
- If $i = j$ and $res = 0$, then MATCHA rejects.
- If $i \neq j$ and $res = 1$, then MATCHA rejects.
- If $i \neq j$ and $res = 0$, then MATCHA plays another round.

In the last case, MATCHA starts over (with a fresh new set of random coins): it flips another fair unbiased coin, picks another pair of images (i, j) depending on the outcome of the coin, etc.

Remarks

1. It is quite easy to write a computer program that has success probability $1/2$ over MATCHA by simply answering with $res = 1$. For most applications, a distinguishing probability of $1/2$ is unacceptable. In order for MATCHA to be of practical use, the test has to be repeated several times.
2. Our description of MATCHA contains an obvious asymmetry: when $i = j$, MATCHA presents the prover with $(i, t(i))$ for $i \leftarrow \mathcal{I}$, and when $i \neq j$ MATCHA presents $(i, t(j))$, where i is chosen *uniformly* from the set $[\mathcal{I}] - \{j\}$. This gives the prover a simple strategy to gain advantage over M : if i seems to come from \mathcal{I} , guess that $t(j)$ is a transformation of i ; otherwise guess that it isn't. The reason for the asymmetry is to make the proof of Lemma 1 easier to follow. We note that a stronger CAPTCHA can be built by choosing i from the distribution \mathcal{I} restricted to the set $[\mathcal{I}] - \{j\}$.
3. The intuition for why MATCHA plays another round when $i \neq j$ and $res = 0$ is that we are trying to convert high success against MATCHA into high success in solving $\mathcal{P}1$; a program that solves $\mathcal{P}1$ by comparing $t(j)$ to every image in $[\mathcal{I}]$ will encounter that most of the images in $[\mathcal{I}]$ are different from $t(j)$.

4. In the following Lemma we assume that a program with high success over M always terminates with a response in time at most τ . Any program which does not satisfy this requirement can be rewritten into one which does, by stopping after τ time and sending the response 1, which never decreases the success probability. We also assume that the unit of time that MATCHA uses is the same as one computational step.

Lemma 1 *Any program that has success greater than η over $M = (\mathcal{I}, \mathcal{T}, \tau)$ can be used to $(\delta, \tau|\mathcal{I}|)$ -solve $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$, where*

$$\delta \geq \frac{\eta}{1 + 2|\mathcal{I}|(1 - \eta)}.$$

Proof: Let B be a program that runs in time at most τ and has success $\sigma_B \geq \eta$ over M . Using B we construct a program A^B that is a $(\delta, \tau|\mathcal{I}|)$ -solution to $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$.

The input to A^B will be an image, and the output will be another image. On input j , A^B will loop over the entire database of images of M (i.e., the set $[\mathcal{I}]$), each time feeding B the pair of images (i, j) , where $i \in [\mathcal{I}]$. Afterwards, A^B collects all the images $i \in [\mathcal{I}]$ on which B returned 1 (i.e., all the images in $[\mathcal{I}]$ that B thinks j is a transformation of). Call the set of these images S . If S is empty, then A^B returns an element chosen uniformly from $[\mathcal{I}]$. Otherwise, it picks an element ℓ of S uniformly at random.

We show that A^B is a $(\delta, \tau|\mathcal{I}|)$ -solution to $\mathcal{P}1$. Let $p_0 = \Pr_{\mathcal{T}, \mathcal{I}, r}[B_r(i, t(i)) = 0]$ and let $p_1 = \Pr_{\mathcal{T}, j \leftarrow \mathcal{I}, i, r}[B_r(i, t(j)) = 1]$. Note that

$$\Pr_{r, r'}[\langle M_r, B_{r'} \rangle = \text{reject}] = 1 - \sigma_B = \frac{p_0}{2} + \frac{p_1 + (1 - p_1)(1 - \sigma_B)}{2} = \frac{p_0 + p_1}{1 + p_1},$$

which gives $\sigma_B \leq 1 - p_0$ and $p_1 \leq 2(1 - \sigma_B)$. Hence:

$$\Pr_{\mathcal{T}, \mathcal{I}, r} [A_r^B(t(j)) = j] \geq \sum_{s=1}^n \Pr_{\mathcal{T}, \mathcal{I}, r} [A_r^B(t(j)) = j | |S| = s] \Pr_{\mathcal{T}, \mathcal{I}} [|S| = s] \quad (3.1)$$

$$= \sum_{s=1}^n \frac{1 - p_0}{s} \Pr_{\mathcal{T}, \mathcal{I}} [|S| = s] \quad (3.2)$$

$$\geq \frac{1 - p_0}{E_{\mathcal{T}, \mathcal{I}} [|S|]} \quad (3.3)$$

$$\geq \frac{1 - p_0}{1 + |\mathcal{I}| p_1} \quad (3.4)$$

$$\geq \frac{\sigma_B}{1 + 2|\mathcal{I}|(1 - \sigma_B)} \quad (3.5)$$

(3.2) follows by the definition of the procedure A^B , (3.3) follows by Jensen's inequality and the fact that $f(x) = 1/x$ is concave, (3.4) follows because

$$E_{\mathcal{T}, \mathcal{I}} [|S|] \leq 1 + \sum_{i \neq j} \Pr_{\mathcal{I}, r} (B(i, t(j)) = 1)$$

and (3.5) follows by the inequalities for p_0 , p_1 and σ_B given above. This completes the proof.

Theorem 1 *If $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ is $(\delta, \tau|\mathcal{I}|)$ -hard and $M = (\mathcal{I}, \mathcal{T}, \tau)$ is (α, β) -human executable, then M is a $(\alpha, \beta, \frac{(2-|\mathcal{I}|)\delta}{2\delta-|\mathcal{I}|})$ -CAPTCHA.*

PIX

An instance $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can sometimes be used almost directly as a CAPTCHA. For instance, if \mathcal{I} is a distribution over images containing a single word and λ maps an image to the word contained in it, then $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can be used directly as a CAPTCHA. Similarly, if all the images in $[\mathcal{I}]$ are pictures of simple concrete objects and λ maps an image to the object that is contained in the image, then $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can be used as a CAPTCHA.

Formally, a PIX instance is a tuple $X = (\mathcal{I}, \mathcal{T}, L, \lambda, \tau)$. The PIX verifier works as follows. First, V draws $i \leftarrow \mathcal{I}$, and $t \leftarrow \mathcal{T}$. V then sends to P the message $(t(i), L)$, and sets a timer

for τ . P responds with a label $l \in L$. V accepts if $l = \lambda(i)$ and its timer has not expired, and rejects otherwise.

Theorem 2 *If $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$ is (δ, τ) -hard and $X = (\mathcal{I}, \mathcal{T}, L, \lambda, \tau)$ is (α, β) -human executable, then X is a (α, β, δ) -CAPTCHA.*

Various instantiations of PIX are in use at major internet portals, like Yahoo! and Hotmail. Other less conventional ones, like ANIMAL-PIX, can be found at www.captcha.net. ANIMAL-PIX presents the prover with a distorted picture of a common animal (like the one shown below) and asks it to choose between twenty different possibilities (monkey, horse, cow, *et cetera*).



Figure 3.1: ANIMAL-PIX.

CAPTCHA and Robust Steganography

We detail a useful application of (δ, τ) -solutions to instantiations of $\mathcal{P}1$ and $\mathcal{P}2$ (other than reading slightly distorted text, which was mentioned before). We hope to convey by this application that our problems were not chosen just because they can create CAPTCHAs but because they in fact have applications related to security. Our problems also serve to illustrate that there is a need for better AI in security as well. Areas such a Digital Rights Management, for instance, could benefit from better AI: a program that can find slightly distorted versions of original songs or images on the world wide web would be a very useful tool for copyright owners.

There are many applications of solutions to $\mathcal{P}1$ and $\mathcal{P}2$ that we don't mention here. $\mathcal{P}1$, for instance, is interesting in its own right and a solution for the instantiation when \mathcal{I} is a distribution on images of works of art would benefit museum curators, who often have to answer questions such as “what painting is this a photograph of?”

Robust Steganography is concerned with the problem of covertly communicating messages on a public channel which is subject to modification by a restricted adversary. For example, Alice may have some distribution on images which she is allowed to draw from and send to Bob; she may wish to communicate additional information with these pictures, in such a way that anyone observing her communications can not detect this additional information. The situation may be complicated by an adversary who transforms all transmitted images in an effort to remove any hidden information. In this section we will show how to use (δ, τ) -solutions to instantiations of $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ to implement a secure robust steganographic protocol for image channels with distribution \mathcal{I} , when the adversary chooses transformations from \mathcal{T} . Note that if we require security for arbitrary \mathcal{I}, \mathcal{T} , we will require a (δ, τ) -solution to $\mathcal{P}1$ for arbitrary \mathcal{I}, \mathcal{T} ; if no solution works for arbitrary $(\mathcal{I}, \mathcal{T})$ this implies the existence of specific \mathcal{I}, \mathcal{T} for which $\mathcal{P}1$ is still hard. Thus either our stegosystem can be implemented by computers for arbitrary image channels or there is a (non-constructive) hard AI problem that can be used to construct a CAPTCHA.

The results of this subsection can be seen as providing an implementation of the “supraliminal channel” postulated by Craver [15]. Indeed, Craver’s observation that the adversary’s transformations should be restricted to those which do not significantly impact human interpretation of the images (because the adversary should not unduly burden “innocent” correspondents) is what leads to the applicability of our hard AI problems.

Steganography Definitions

Fix a distribution over images \mathcal{I} , and a set of keys \mathcal{K} . A *steganographic protocol* or *stegosystem* for \mathcal{I} is a pair of efficient probabilistic algorithms (SE, SD) where $SE :$

$\mathcal{K} \times \{0,1\} \rightarrow [\mathcal{I}]^\ell$ and $SD : \mathcal{K} \times [\mathcal{I}]^\ell \rightarrow \{0,1\}$, which have the additional property that $\Pr_{K,r,r'}[SD_r(K, SE_{r'}(K, \sigma)) \neq \sigma]$ is negligible (in ℓ and $|K|$) for any $\sigma \in \{0,1\}$. We will describe a protocol for transmitting a single bit $\sigma \in \{0,1\}$, but it is straightforward to extend our protocol and proofs by serial composition to any message in $\{0,1\}^*$ with at most linear decrease in security.

Definition 6 *A stegosystem is steganographically secret for \mathcal{I} if the distributions $\{SE_r(K, \sigma) : K \leftarrow \mathcal{K}, r \leftarrow \{0,1\}^*\}$ and \mathcal{I}^ℓ are computationally indistinguishable for any $\sigma \in \{0,1\}$.*

Steganographic secrecy ensures that an eavesdropper cannot distinguish traffic produced by SE from \mathcal{I} . Alice, however, is worried about a somewhat malicious adversary who transforms the images she transmits to Bob. This adversary is restricted by the fact that he must transform the images transmitted between many pairs of correspondents, and may not transform them in ways so that they are unrecognizable to humans, since he may not disrupt the communications of legitimate correspondents. Thus the adversary's actions, on seeing the image i , are restricted to selecting some transformation t according to a distribution \mathcal{T} , and replacing i by $t(i)$. Denote by $t_{1\dots\ell} \leftarrow \mathcal{T}^\ell$ the action of independently selecting ℓ transformations according to \mathcal{T} , and denote by $t_{1\dots\ell}(i_{1\dots\ell})$ the action of element-wise applying ℓ transformations to ℓ images.

Definition 7 *A stegosystem (SE, SD) is steganographically robust against \mathcal{T} if it is steganographically secret and*

$$\Pr_{t_{1\dots\ell} \leftarrow \mathcal{T}^\ell, r, r', K} [SD_r(K, t_{1\dots\ell}(SE_r(K, \sigma))) \neq \sigma]$$

is negligible (in $|K|$ and ℓ) for any $\sigma \in \{0,1\}$.

Let $F : \mathcal{K} \times \{1, \dots, \ell\} \times L \rightarrow \{0,1\}$ be a pseudorandom function family. We assume that $\lambda : [\mathcal{I}] \rightarrow L$ is efficiently computable and $A : S_{\mathcal{P}2} \rightarrow L$ is a (δ, τ) -solution to $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ as

defined above (recall that $\mathcal{P}1$ is a special case of $\mathcal{P}2$), i.e. A operates in time τ and

$$\Pr_{t,i,r}[A_r(t(i)) = \lambda(i)] \geq \delta .$$

Let $c = \Pr_{i,j \leftarrow \mathcal{I}}[\lambda(i) = \lambda(j)]$. We *require* that $c < 1$ (that is, we require that there is enough variability in the labels of the images to be useful for communication). Notice that L can simply be equal to $[\mathcal{I}]$ and λ can be the identity function (in case the images in $[\mathcal{I}]$ have no labels as in $\mathcal{P}1$). We prove in [2] that the following construction is an efficient, robust stegosystem for \mathcal{T} .

Construction 1

Procedure SE :

Input: $K \in \mathcal{K}$, $\sigma \in \{0, 1\}$

for $j = 1 \dots \ell$ do

 draw $d_0 \leftarrow \mathcal{I}$, $d_1 \leftarrow \mathcal{I}$

 if $F_K(j, \lambda(d_0)) = \sigma$ then

 set $i_j = d_0$

 else

 set $i_j = d_1$

Output: i_1, i_2, \dots, i_ℓ

Procedure SD :

Input: $K \in \mathcal{K}$, $i'_1 \dots i'_\ell \in [\mathcal{I}]^\ell$

for $j = 1 \dots \ell$ do

 set $\sigma_j = F_K(j, A(i'_j))$

Output: *majority*($\sigma_1, \dots, \sigma_\ell$)

Proposition 1 *Construction 1 is steganographically secret and robust for \mathcal{I}, \mathcal{T} .*

The proof of the Proposition relies on the fact that when A returns the correct solution on received image i'_j , the recovered bit σ_j is equal to the intended bit σ with probability approximately $\frac{1}{2} + \frac{1}{4}(1 - c)$ and otherwise $\sigma_j \neq \sigma$ with probability $1/2$; therefore the probability that the majority of the σ_j are incorrect is negligible in ℓ . For details, see the Appendix.

Remarks

1. Better solutions to $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$ imply more efficient stegosystems: if δ is larger, then ℓ can be smaller and less images need to be transmitted to send a bit secretly and robustly.
2. Since we assume that $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$ (or, as it might be the case, $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$) is easy for humans, our protocol *could* be implemented as a cooperative effort between the human recipient and the decoding procedure (without the need for a solution to $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$). However, decoding each bit of the secret message will require classifying many images, so that a human would likely fail to complete the decoding well before any sizeable hidden message could be extracted (this is especially true in case we are dealing with $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ and a large set $[\mathcal{I}]$: a human would have to search the entire set $[\mathcal{I}]$ as many as ℓ times for each transmitted bit). Thus to be *practical*, a (δ, τ) -solution (for small τ) to $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I},\mathcal{T},\lambda}$ will be required.

Discussion and Closing Remarks

A primary goal of the CAPTCHA project is to serve as a challenge to the Artificial Intelligence community. We believe that having a well-specified set of goals will contribute greatly to the advancement of the field. A good example of this process is the recent progress in reading distorted text images driven by the CAPTCHA in use at Yahoo!. In response to the challenge provided by this test, Malik and Mori [35] have developed a program which can pass the test with probability roughly 0.8. Despite the fact that this CAPTCHA has no formal proof that a program which can pass it can read under other distributions of image transformations, Malik and Mori claim that their algorithm represents significant progress in the general area of text recognition; it is encouraging to see such progress. For this reason, it is important that even Automated Turing Tests without formal reductions attempt to test ability in general problem domains.

Acknowledgments

First and foremost I am grateful to Manuel Blum for his suggestions, advice, brilliant ideas and for his unconditional support. I am also grateful to my other coauthors for the hard work and great ideas: Laura Dabbish, Nicholas Hopper and John Langford. I would like to thank Aditya Akella, Sonya Allin, Josh Benaloh, Lenore Blum, Shuchi Chawla, Brighten Godfrey, Scott Hudson, Takeo Kanade Steven Katz, Lea Kissner, Tal Malkin, Udi Manber, Moni Naor, Lenore Ramm, Omer Reingold, Chuck Rosenberg, Roni Rosenfeld, Steven Rudich, David Stork, Manuela Veloso, and the anonymous reviewers from *CRYPTO 2002*, *Eurocrypt 2003*, *CHI 2004* and *Communications of the ACM* for helpful discussions and comments. Thanks also to Shingo Uchihashi and Andrew Bortz for allowing me to use their image crawlers. This work has been and will be partially supported by the National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982 (The Aladdin Center), and by a Microsoft Research Fellowship.

Bibliography

- [1] Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford. The CAPTCHA Web Page: <http://www.captcha.net>. 2000.
- [2] Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford. CAPTCHA: Using Hard AI Problems For Security. In EUROCRYPT 2003.
- [3] Luis von Ahn, Manuel Blum and John Langford. Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI. To appear in *Communications of the ACM*.
- [4] AltaVista Company. Altavista Search Home. <http://www.altavista.com/>
- [5] Ross J. Anderson and Fabien A. P. Petitcolas. *On The Limits of Steganography*. IEEE Journal of Selected Areas in Communications, 16(4). May 1998.
- [6] Barnard, K., Duygulu, P., and Forsyth, D. A. Clustering Art. IEEE conference on Computer Vision and Pattern Recognition, 2001, pages 434-441.
- [7] Barnard, K., and Forsyth, D. A. Learning the Semantics of Words and Pictures. International Conference of Computer Vision, 2001, pages 408-415.
- [8] Mihir Bellare, Russell Impagliazzo and Moni Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science (FOCS' 97)*, pages 374-383. IEEE Computer Society, 1997.

-
- [9] Mihir Bellare and Chanathip Namprempre. *Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm*. In: *Advances in Cryptology – Asiacrypt ’00*. December 2000.
- [10] Mikhail M. Bongard. *Pattern Recognition*. Spartan Books, Rochelle Park NJ, 1970.
- [11] C. Cachin. *An Information-Theoretic Model for Steganography*. In: *Information Hiding – Second International Workshop, Preproceedings*. April 1998.
- [12] Carson, C., and Ogle, V. E. Storage and Retrieval of Feature Data for a Very Large Online Image Collection. IEEE Computer Society Bulletin of the Technical Committee on Data Engineering, 1996, Vol. 19 No. 4.
- [13] A. L. Coates, H. S. Baird, and R. J. Fateman. Pessimial Print: A Reverse Turing Test. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR’ 01)*, pages 1154-1159. Seattle WA, 2001.
- [14] Columbia University Center for Telecommunications Research. webSEEK. www.ctr.columbia.edu/webseek/
- [15] Scott Craver. On Public-key Steganography in the Presence of an Active Warden. In *Proceedings of the Second International Information Hiding Workshop*, pages 355-368. Springer, 1998.
- [16] Duygulu, P., Barnard, K., de Freitas, N., and Forsyth, D. A. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. 7th European Conference on Computer Vision, 2002, pages 97-112.
- [17] Electronic Arts. Pogo.com. <http://www.pogo.com>
- [18] Fleck, M. M., Forsyth, D. A., and Bregler, C. Finding Naked People. ECCV, 1996, pages 593-602.

-
- [19] Oded Goldreich, Shafi Goldwasser and Silvio Micali. *How to Construct Random Functions*. Journal of the ACM, 33(4):792 – 807, 1986.
- [20] Google Inc. Google Web Search. <http://www.google.com>
- [21] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. *A pseudorandom generator from any one-way function*. SIAM Journal on Computing, 28(4):1364–1396, 1999.
- [22] Nicholas J. Hopper, John Langford and Luis Von Ahn. Provably Secure Steganography. In *Advances in Cryptology, CRYPTO' 02*, volume 2442 of *Lecture Notes in Computer Science*, pages 77-92. Santa Barbara, CA, 2002.
- [23] Russell Impagliazzo and Michael Luby. *One-way Functions are Essential for Complexity Based Cryptography*. In: 30th FOCS, November 1989.
- [24] G. Jagpal. *Steganography in Digital Images* Thesis, Cambridge University Computer Laboratory, May 1995.
- [25] D. Kahn. *The Code Breakers*. Macmillan 1967.
- [26] Takeo Kanade and Shingo Uchihashi. User Powered Content-Free Approach to Image Retrieval. *Proceedings of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society 2004 (DLKC04)*, March 2004. pages 24-32.
- [27] Stefan Katzenbeisser and Fabien A. P. Petitcolas. *Information hiding techniques for steganography and digital watermarking*. Artech House Books, 1999.
- [28] Lempel, R. and Soffer, A. PicASHOW: Pictorial Authority Search by Hyperlinks on the Web. WWW10.

- [29] M. D. Lillibridge, M. Adabi, K. Bharat, and A. Broder. Method for selectively restricting access to computer systems. Technical report, US Patent 6,195,698. Applied April 1998 and Approved February 2001.
- [30] Michael Luby and Charles Rackoff. *How to construct pseudorandom permutations from pseudorandom functions*. SIAM Journal on Computing, 17(2):373 – 386, 1988.
- [31] K. Matsui and K. Tanaka. *Video-steganography*. In: *IMA Intellectual Property Project Proceedings*, volume 1, pages 187–206, 1994.
- [32] Microsoft Corporation. MSN Gaming Zone. <http://zone.msn.com>
- [33] Milliman, R. E. Website Accessibility And The Private Sector. www.rit.edu/easi/itd/itdv08n2/milliman.html
- [34] T. Mittelholzer. *An Information-Theoretic Approach to Steganography and Watermarking* In: *Information Hiding – Third International Workshop*. 2000.
- [35] Greg Mori and Jitendra Malik. Breaking a Visual CAPTCHA. Unpublished Manuscript, 2002. Available electronically: <http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf>.
- [36] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [37] N2H2, Inc. N2H2 Filter. <http://www.n2h2.com>
- [38] Moni Naor. Verification of a human in the loop or Identification via the Turing Test. Unpublished Manuscript, 1997. Available electronically: <http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps>.
- [39] O'Connor, B. and O'Connor, M. Categories, Photographs and Predicaments: Exploratory Research on Representing Pictures for Access. Bulletin of the American Society for Information Science 25.6, 1999, page 17-20.

- [40] J. A. O'Sullivan, P. Moulin, and J. M. Ettinger. *Information theoretic analysis of Steganography*. In: *Proceedings ISIT '98*. 1998.
- [41] Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. In *Proceedings of the ACM Computer and Security Conference (CCS' 02)*, pages 161-170. ACM Press, November 2002.
- [42] Random Bounce Me Website. <http://random.bounceme.net/>
- [43] S. Rice, G. Nagy, and T. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Boston, 1999.
- [44] Phillip Rogaway, Mihir Bellare, John Black and Ted Krovetz. *OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption*. In: *Proceedings of the Eight ACM Conference on Computer and Communications Security (CCS-8)*. November 2001.
- [45] Adi Shamir and Eran Tromer. Factoring Large Numbers with the TWIRL Device. Unpublished Manuscript, 2003. Available electronically: <http://www.cryptome.org/twirl.ps.gz>.
- [46] Scheniderman, H. and Kanade, T. Object Detection Using the Statistics of Parts. *International Journal of Computer Vision*, 2002.
- [47] G.J. Simmons. *The Prisoner's Problem and the Subliminal Channel*. In: *Proceedings of CRYPTO '83*. 1984.
- [48] Stork, D. G. and Lam C. P. Open Mind Animals: Ensuring the quality of data openly contributed over the World Wide Web. *AAAI Workshop on Learning with Imbalanced Data Sets*, 2000, pages 4-9.
- [49] Stork, D. G. The Open Mind Initiative. *IEEE Intelligent Systems and Their Applications*, 14-3, 1999, pages 19-20.

- [50] Trask, R. L. *Language Change*. Routledge, 1994
- [51] Alan M. Turing. Computing Machinery and Intelligence. In: *Mind, Vol. 59, No. 236*, pp. 433-460. 1950.
- [52] A. Westfeld, G. Wolf. *Steganography in a Video Conferencing System*. In: *Information Hiding – Second International Workshop, Preproceedings*. April 1998.
- [53] J. Xu, R. Lipton and I. Essa. Hello, are you human. Technical Report GIT-CC-00-28, Georgia Institute of Technology, November 2000.
- [54] Yahoo!, Inc. Yahoo! Games. <http://games.yahoo.com>
- [55] J. Zollner, H.Federrath, H.Klimant, A.Pftizmann, R. Piotraschke, A.Westfeld, G.Wicke, G.Wolf. *Modeling the security of steganographic systems*. In: *Information Hiding – Second International Workshop, Preproceedings*. April 1998.