Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI

Luis von Ahn* Manuel Blum* John Langford*

If you try to get a new email account at Yahoo, you will be asked to prove that you're a human and not a computer. Why? Because a single computer program can get thousands of free email accounts per second. And that's bad for Yahoo. In this article, we introduce the technology that Yahoo uses to prevent automatic registration.

The ideas and the theory were developed at Carnegie Mellon University in the fall of 2000, when Udi Manber, chief scientist of Yahoo started working in collaboration with the cryptography and complexity theory group at CMU. Here, then, we present a theoretician's solution to a practical problem.

We introduce the concept of a CAPTCHA: a program that can differentiate between humans and computers based only on interactions over a computer network. CAPTCHAs have been around for quite some time (though not under that name), as "quick and dirty" ways to keep computer agents from entering web sites, but no theory had been developed up to this point.

Gotcha!

Intuitively, a CAPTCHA is a program that can generate and grade tests that:

- most humans can pass, but
- current computer programs cannot pass.[1]

CAPTCHA stands for "Completely Automated Public Turing Test to Tell Computers and Humans Apart". Notice how a CAPTCHA is like an *automated* version of the Turing Test [6], one in which the judge is a computer.

The P in CAPTCHA stands for Public: the code for the program should be commented and easy to understand; additionally, the commented code and all the data used by a CAPTCHA should be public. Thus a program that can generate and grade tests that distinguish humans from computers but whose code or data are private is *not* a CAPTCHA.

Examples

Let's look at few examples of CAPTCHAs before moving on. When reading, think about why each of these is a valid example of a CAPTCHA. The code and demos for all the programs described in this section can be found online at *www.captcha.net*.

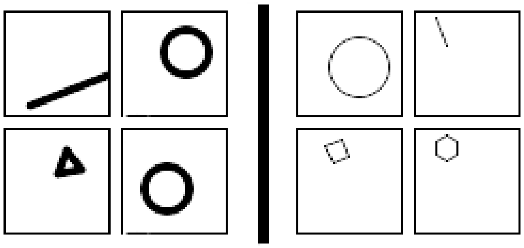
- **Gimpy.** Consider the following program, which we will refer to as GIMPY. GIMPY works as follows: it first picks seven words out of a dictionary and renders the words into a distorted image (as shown in the figure below). GIMPY then presents a test to its user, which consists of the distorted image and the directions: *"type three words appearing in the image"*. Given the types of deformations that GIMPY uses, most humans can read three words from the distorted image, while current computer programs cannot. (We still haven't *proved* that GIMPY is a CAPTCHA;

[1] Computer Science Dept., Carnegie Mellon University, Pittsburgh PA 15213. {biglou,mblum,jcl}@cs.cmu.edu

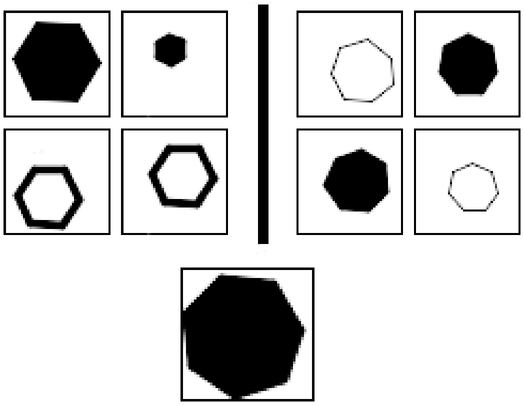
[1] The words "Most" and "Current" can be precisely defined, but the intuitive meaning is good for the purposes of this article.

the issues revolving around such a proof will be discussed later.)

- **Bongo.** Another example of a CAPTCHA is the program we call BONGO[2]. BONGO asks the user to solve a visual pattern recognition problem. In particular, 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 below:

  (These two series are different because everything in the LEFT is drawn with thick lines, while everything in the RIGHT is drawn with thin lines.) 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 the figure below?

- **Pix.** Yet another example of a CAPTCHA is a program that has a large database of labeled images (such databases can be found in [4, 3]). All of these images should be 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.

  The reader might notice that this example, as stated, is not a CAPTCHA. Why? Because it is very easy to write a program that can answer the question "what are these images 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. In order for this program to actually be a CAPTCHA, the images need to be randomly distorted before being presented to the user, in such a way that searching the database for the images is hard for computer programs.

- **Eco.** The final example we will mention can be thought of as a sound version of GIMPY. The program picks a word or a sequence of numbers at random, renders the word or the numbers into

[2] BONGO is named after M.M. Bongard, who published a book of pattern recognition problems ([2]) in 1951.

a sound clip and distorts the sound clip. It then presents the distorted sound clip to its user and asks the user to type in the contents of the sound clip.

The creation of CAPTCHAs is an interesting endeavor in and of itself and later we will see why it is good to have a wide variety of CAPTCHAs. We end this section by remarking that it is still an open problem to create a text-based CAPTCHA (one which doesn't use any images or sound clips).

The Enemy

We will consider CAPTCHAs from the perspective of cryptography, so we need the notion of an enemy. In our case, THE ENEMY will be the person trying to write a program that can pass the tests generated by a CAPTCHA. Intuitively, we will assume that THE ENEMY is an expert programmer with:

- Complete knowledge of the code and data used by the CAPTCHA
- Full access to the web
- A super computer
- Lots of free time

We want to prove or give evidence that it is hard for THE ENEMY to write a program that can pass the tests generated by a CAPTCHA.

Reductions to Hard AI Problems

How do we *prove* that computer programs will never pass the tests generated by a CAPTCHA? Unfortunately, we believe there is no way to prove such a strong statement: since humans can pass the tests, proving that computer programs cannot pass them would show that there exist things that humans can do but computers can't. We (the authors) believe that some day computers will be as good as humans (and possibly better) in every respect.

The best we can do is to present evidence that, given the current state of technology, it is hard to write a program that can pass the tests generated by a CAPTCHA. We can do this by proving the following statement: *Any program that passes the tests generated by a* CAPTCHA *can be used as an oracle to solve a hard unsolved Artificial Intelligence (AI) problem.*

The previous statement requires explanation and justification. (Hopefully, we will be able to convince the reader that a reduction to a hard AI problem is the right notion of security for CAPTCHAs.)

- First, by "[a] program that passes the tests generated by a CAPTCHA", we mean a program (possibly probabilistic) that has some (not so small, but precisely defined) probability of passing a given random test generated by the CAPTCHA.

- What is a hard unsolved AI problem? Intuitively, it is an AI problem that remains unsolved despite the fact that the AI community has been working on it for several years. For the proof of this theorem to go through, an AI problem needs to be *precisely* defined. It is a good exercise for the reader to try to define AI problems in precise terms.

- Why does this give evidence that it is hard to pass the tests generated by a CAPTCHA? If an AI problem remains unsolved after many intelligent people have been working on it for a long time, then it is hard to write a program that can solve this problem. Our statement says that it is at least as hard to write a program that can pass the tests generated by a CAPTCHA.

Notice that classical cryptography is laid on a foundation similar to ours: cryptographers only *assume* factoring large numbers is hard, as nobody has found a way to do it.

P Stands for Public. Why?

Several people (especially those outside the cryptography community) have objected to CAPTCHAs having publicly available code and data. They claim that CAPTCHAs would be more secure if the code was hidden. Here, we will state why we want all the code and data used by a CAPTCHA to be publicly available.

- **Hackers**. If a CAPTCHA requires some private data in order to be secure, then this CAPTCHA is very vulnerable to hacker attacks: once a hacker breaks into the system and finds this secret data, the CAPTCHA seizes to be secure forever. We want CAPTCHAs to generate tests that are hard to pass for all computers; not that are hard to pass for computers that don't have a particular secret.

- **Reverse Engineering**. If the CAPTCHA weren't public, THE ENEMY can still learn a lot about the way a CAPTCHA works and all the data it uses by simply taking many of the tests generated by a CAPTCHA.

- **Analysis**. Moreover, there is no general way to know exactly how much THE ENEMY can learn about how a CAPTCHA works. Hence, we may as well assume that THE ENEMY can learn *everything* about how the CAPTCHA works and then prove that *even in this case* it is hard for THE ENEMY to write a program that can pass the tests generated by the CAPTCHA

- **We're doing AI**. We *want* our CAPTCHAs to be broken! Why? Because if a CAPTCHA is broken, then a previously unsolved AI problem is solved. This is how lazy cryptographers do AI! We help THE ENEMY as much as we can.

Applications

CAPTCHAs have several applications. Here we mention just a few.

- **Online Polls.** Almost two years ago, *slashdot.com* released an online poll asking for the best school in computer science (a dangerous question to ask over the web!). As it 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 figured out a way to stuff the ballots and wrote programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote *their* own program and the poll had become a contest between voting "bots". MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000.

  The point here is that any online poll suffers from this weakness against bots. But CAPTCHAs offer a solution: voters should "prove" they are human before being allowed to vote.

- **Free Email Services.** Several companies offer free email services over the web. Most of these suffer from a specific type of attack: "bots" that sign up for thousands of email accounts a minute. In particular, web pornography companies use such tactics to obtain free email accounts from which they can send advertisements. A solution to this problem is to have users solve a CAPTCHA test before they can get an account. Yahoo has adopted this solution, and, in fact, uses a variation of our program GIMPY.

- **Search Engine Bots.** You may not want to have your web page indexed. It is sometimes surprising how much others can learn about you by just typing in your name in Google or Altavista. There is an *html* tag to prevent bots from reading your web page. The tag, however, doesn't guarantee that bots will not read your web page; 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 bots. However, if you want to *guarantee* that bots will not enter your web-site, you need to have your viewers prove that they are indeed human before letting them in.

- **Shopping Agents.** Many web sites offer free shopping to their users, along with extremely low prices. Part of the reason these companies can offer such a service is because of advertisements. However, there are several "shopping bots" available, which visit many shopping sites, retrieve the best prices from all of them and present them to the user. These shopping agents are bad for the shopping sites because users don't get to see the advertisements. A solution to this problem is for the shopping sites to put a CAPTCHA test before revealing their prices, to ensure that only humans visit their site.

Stealing Cycles From Humans

Some readers might notice that CAPTCHAs really aren't a good solution the the Shopping Agent problem. Why? Because if shopping sites display CAPTCHA tests before allowing their users to see the prices, then shopping agents can simply ask the human behind the computer to solve the CAPTCHA test.

And, in fact, this same strategy can be used to defeat CAPTCHAs in many cases. We call it "stealing cycles from humans". Consider free web porn companies, for instance. These companies can still obtain thousands of free email accounts even if the email provider uses CAPTCHA. How? The porn companies can have bots that try to get free email accounts, and as soon as the bots encounter a CAPTCHA test, they simply send the test to the porn site. Back at the porn site, there are several thousands of users seeing free pornographic pictures, and one of them is told: "please solve the following test before we can show you the next picture". *Voila!*

This leads to an interesting digression. Notice how stealing cycles from humans works: humans are used in order to expand the computational abilities of the porn company bots. Here we digress and ponder: in general, how can we use humans as oracles to expand our computational abilities? Currently, there are several things that humans can do but computers can't: recognize facial expressions from any angle, understand language, read distorted text, recognize which animal is in a picture. Can we use humans in some way to make computers solve these problems in a distributed manner? Can we have a kind of SETI@home project in which web users help to classify all books in the library of congress?

Can you imagine a program that can tell you which is the best haircut for you? Such a program can be written by stealing cycles from humans. If you visit *www.hotornot.com*, you will get a picture of a person (man or woman) and you are supposed to rate how attractive you think the person is (on a scale from 1 to 10). After you rate the person you get to learn what other users have said about this person and you also get to see another person that you can rank. What's interesting about this site is that it is very popular and catchy: the authors have seen entire offices of graduate students rating pictures. A program that can tell you which is the best haircut for you works as follows: you give it a picture of yourself and it sends several copies of this picture to *www.hotornot.com*, all with different hair styles. Whichever gets the highest ranking is the best.

Epilogue

The little story about CAPTCHAs ends here. As a last point, notice how the whole idea of a CAPTCHA can be seen under the light of "stealing cycles". CAPTCHAs are the ultimate tools for stealing cycles from intelligent people: malicious, intelligent programmers (writing programs for getting thousands of free email accounts, etc.) can be put to work on hard unsolved AI problems.

References

[1] Luis von Ahn, Manuel Blum and John Langford. The Captcha Project *http://www.captcha.net*.

[2] Mikhail M. Bongard. (1970). *Pattern Recognition.* Rochelle Park, N.J.: Hayden Book Co., Spartan Books.

[3] Corbis. *http://www.corbis.com.*

[4] The Getty Images. *http://www.gettyimages.com.*

[5] Moni Naor. Verification of a human in the loop or Identification via the Turing Test. Unpublished.

[6] Alan M. Turing. Computing Machinery and Intelligence. In: *Mind, Vol. 59, No. 236, pp. 433-460.* 1950.