By Luis von Ahn, Manuel Blum, and John Langford

TELLING HUMANS AND COMPUTERS APART AUTOMATICALLY

How lazy cryptographers do AI.

You've probably seen them—colorful images with distorted text in them at the bottom of Web registration forms. CAPTCHAs are used by Yahoo, Hotmail, PayPal and many other popular Web sites to prevent automated registrations, and they work because no computer program can currently read distorted text as well as humans can. What you probably don't know is that a CAPTCHA is something

illustration by Jean-François Podevin

more than just an image with distorted text: it is a test, any test, that can be automatically generated, which most humans can pass, but that current computer programs cannot pass. Notice the paradox: a CAPTCHA is a program that can generate and grade tests that it itself cannot pass (much like some professors).

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 difficult for someone to write a computer program that can pass the tests generated by a CAPTCHA even if they know 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 [10]. 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. We deliberately avoid using 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.

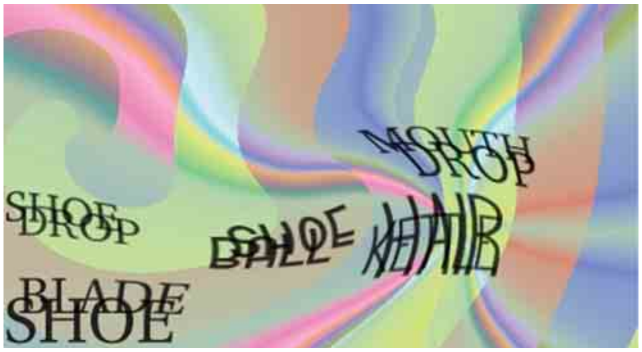

Figure 1. Can you read three words in this image?

Applications

Although the goal of the original Turing Test was to serve as a measure of progress for artificial intelligence—a computer would be said to be intelligent if it passed the Turing Test—making the judge be a computer allows CAPTCHAs to be useful for other practical applications.

In November 1999, for example, the Web site slashdot.com 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 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.

Another application involves free email services. Several companies offer free email services that have suffered from a specific type of attack: "bots" that signed up for thousands of email accounts every minute. This situation has been 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.

Some Web sites don't want to be indexed by search engines. There is a 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 bots won't enter a Web site, CAPTCHAs are needed.

CAPTCHAs also offer a plausible solution against email worms and spam: only accept an email message if you know there is a human behind the other computer. A few companies, such as www.spamarrest.com are already marketing this idea.

Pinkas and Sander [9] have also 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.

Examples of CAPTCHAs

CAPTCHAs further 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

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.**

computer judge can ask any question that can be transmitted over a computer network.

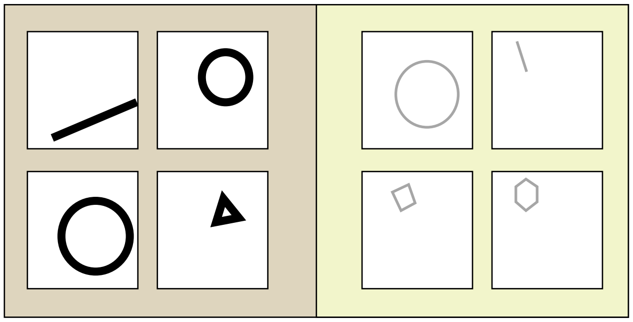

Figure 2. Everything on the left is drawn with thick lines, while everything on the right is drawn with thin lines.

GIMPY and OCR-based CAPTCHAs. GIMPY [2] is one of the many CAPTCHAs based on the difficulty of reading distorted text. GIMPY works by selecting seven words out of a dictionary and rendering a distorted image containing the words (as shown in Figure 1). 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 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 (the difficulty of reading distorted text).

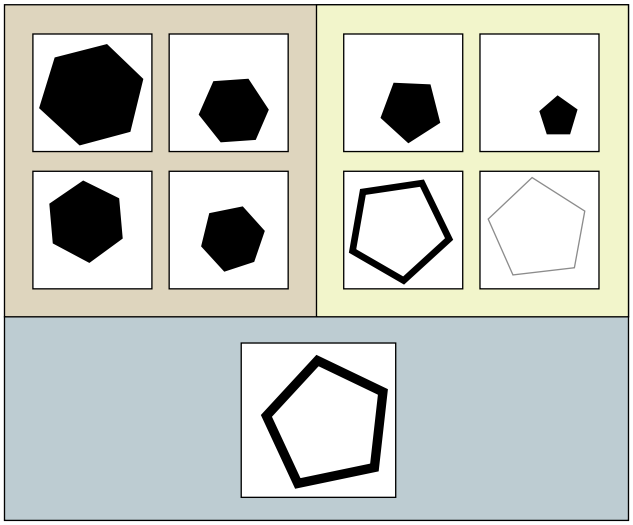

Figure 3. To which side does the block on the bottom belong?

Bongo. Another example of a CAPTCHA is the program we call BONGO [2]. BONGO is named after M.M. Bongard, who published a book of pattern recognition problems in the 1970s [3]. 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 is shown in Figure 2. 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 left series or to the right. The user passes the test if he or she correctly determines the side to which the block belongs. Try it yourself: to which side does the isolated block belong in Figure 3? (Answer: the right side.)

PIX. PIX [2] 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). The program picks an object at random, finds six 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 cannot easily search the database for the undistorted image.

Sound-based CAPTCHAs. The final example we offer is based on sound. The program picks a word

> THIS APPROACH HAS THE BENEFICIAL SIDE EFFECT OF INDUCING SECURITY RESEARCHERS, AS WELL AS **otherwise malicious programmers,** TO ADVANCE THE FIELD OF AI.

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 the user and asks users 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 [4].

It is extremely important to have CAPTCHAs based on a variety of sensory abilities. All CAPTCHAs presented here, except for the sound-based CAPTCHA, rely on the user being able to see an image. However, since there are many visually impaired people using the Web, CAPTCHAs based on sound are necessary for accessibility.

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.

Lazy Cryptographers Doing AI

Modern cryptography has shown that open or intractable problems in number theory can be useful: an adversary cannot act maliciously unless he can solve an open problem (like factor a very large number). Similarly, CAPTCHAs show that open problems in AI can be useful: adversaries cannot vote thousands of times in online polls or obtain millions of free email accounts unless they can solve an open problem in AI.

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 the adversary cannot solve an artificial intelligence problem with higher accuracy than what's currently known to the AI community [1, 2, 5, 6, 8]. This approach 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). This is how lazy cryptographers do AI.

A good example of this process is the recent progress in reading distorted text images motivated by the CAPTCHA in use at Yahoo. In response to the challenge provided by this test, Malik and Mori [7] have developed a program that can pass the test with over 80% accuracy. Malik and Mori's algorithm represents significant progress in the general area of text recognition, and it is extremely encouraging to see such progress. 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.

References

1. Ahn, L. von, Blum, M., Hopper, N.J., and Langford, J. CAPTCHA: Telling humans and computers apart. In *Advances in Cryptology, Eurocrypt '03*, volume 2656 of *Lecture Notes in Computer Science*, (2003), 294–311 .
2. Ahn, L. von, Blum, M., Hopper, N.J., and Langford, J. The CAPTCHA Web page; www.captcha.net.
3. Bongard, M.M. *Pattern Recognition*. Spartan Books, Rochelle Park, NJ, 1970.
4. Chan, N. Program Byan; drive.to/research.
5. Coates, A.L., Baird, H.S., and Fateman, R.J. Pessimal print: A Reverse Turing Test. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR '01)*, Seattle, WA, 2001, 1154–1159.
6. Lillibridge, M.D., Abadi, M., Bharat, K., and Broder, A. Method for selectively restricting access to computer systems. U.S. Patent 6,195,698.
7. Mori, G. and Malik, J. Recognizing objects in adversarial clutter—Breaking a visual CAPTCHA. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, June 2003.
8. Naor, M. Verification of a human in the loop or identification via the Turing Test; www.wisdom.weizmann.ac.il/\~naor/PAPERS/human.ps.
9. Pinkas, B. and Sander, T. Securing passwords against dictionary attacks. In *Proceedings of the ACM Computer and Security Conference (CCS '02)*, ACM Press, 161–170.
10. Turing, A.M. Computing machinery and intelligence. *Mind 59*, 236 (1950), 433–460.

Luis von Ahn (biglou@cs.cmu.edu) is a graduate student in the Department of Computer Science at Carnegie Mellon University.

Manuel Blum (mblum@cs.cmu.edu) is the Bruce Nelson Professor in the Department of Computer Science at Carnegie Mellon University.

John Langford (jl@tti-c.org) is a research associate in the Toyota Technological Institute at Chicago.