CAPTCHAs and Information Hiding

Neal R. Wagner[*]

The University of Texas at San Antonio

Department of Computer Science

San Antonio, Texas 78249 USA

wagner@cs.utsa.edu

Abstract. The goal of steganography is to hide information from humans that is accessible to machines. The so-called CAPTCHAs have the opposite goal: hide information from machines that is accessible to humans. This viewpoint suggests non-traditional applications, including instructions, for example online or in a battlefield, that a software agent will not be able to decode and may not even notice, but that a human or agent directly under human control will discover. Current proposals for CAPTCHAs suffer from the increasing ability of machines to discover the hidden information. The response of increasing the complexity of the CAPTCHAs makes them harder for humans to decode, negating their usefulness. This article presents three new approaches to the design of CAPTCHAs that may enlarge the gap between machine and human abilities.

1. Introduction and Motivation.

The viewpoint of this article is the idea of hiding information from software agents in such a way that a human will discover the information. This is the opposite of normal steganography, but it should have many useful applications. In fact, modern Internet providers and others often want to verify that they are dealing with a human being, rather than with a software agent. For example, services such as email or e-shopping are designed for humans but can be abused by automated software. Other automated software may search the entire web for information. To circumvent the automation and make sure one is dealing with a human, special automated tests are now popular—tests easy for a human to pass but hard for a machine. Manuel Blum and other early researchers in this area gave these tests the name CAPTCHA (see [Ahn2003]). Such software could also be useful in situations where one wants to signal to an online human, sending information that software agents will miss. Ideally, as with humans looking at steganography, the software will not even be aware of the hidden information.

A good CAPTCHA should be *public*, meaning that the complete code and all other data needed by the program are publicly available. Traditional CAPTCHAs focus on the need for a *test*, so they should then be able to generate arbitrarily many such tests. The tests are generated and evaluated automatically by computer, but it must require help from a human to pass a test. From the information hiding viewpoint, the CAPTCHA might just keep data intended for humans secret from software.

The most common type of CAPTCHA presents words in a distorted form, perhaps embedded in

[*]Thanks to Wayne Wagner for suggesting the first approach, to Bethany Callanan for drawing pictures for the third approach, and to John Wagner, Ian Callanan, Hugh Maynard, Konrad Hein, and Shouhuai Xu for discussions.

a background or written along with other words to make machine recognition harder, and asks the user to type several of the words. There are lots of variations, but that is the basic system. Researchers are developing software to identify these words, so the process is a continuing game that will surely end with machines better than humans at deciphering the words.

Different systems are needed, ones that will better utilize human pattern recognition abilities. The next section describes a simplified version of word recognition, using only five different possible shapes to identify, but burying the shape in as much random noise as possible. The two sections after that present systems based on images with errors or flaws in them. A human is asked to identify the errors. The first type of error is a distortion of a picture, while the second type of error consists of objects in an image that "don't belong" there.

Garfinkel [Gar2003] has described another aspect to the whole area of recognizing humans: unintended consequences of pursuing this research. Already it is sometimes annoying to have to take a test before proceeding with some activity, but as the programs get better with these tests, the tests will have to get harder and the whole process will take up more time. It is also possible to coerce (human) users into solving such tests or even employing a third world "sweatshop" of CAPTCHA solvers. These are serious issues, but the author hopes that the tests in this article will be easier to use and harder to break than current ones.

It would be relatively easy to create a single picture or message that will alert humans with special information, information not noticeable to software. From the information hiding point of view however, what is needed is a system that would be repeatedly used for such messages. One must assume that software programmers know the specifications of the system. Thus the full power and properties of a CAPTCHA would needed in this case.

- ❖ **Types:** *Circle, Square, Plus, Lines, Triangle.*
- ❖ **Thickness of lines:** From 4 to 12 pixels.
- ❖ **Size:** Random from about a sixth of the field to most of it.
- ❖ **Location:** Also random in the field.
- ❖ **Distortions:** Shapes distorted in both directions using sin curves with random start, period, and amplitude.
- ❖ **Randomization of Pixels:** Background pixels set randomly to black or white, while pixels in the lines are reversed according to the following table:

Percent Pixels Randomly Reversed

Line Thickness	**Difficulty**		
	Easy	**Medium**	**Hard**
4	20%	26%	32%
6	22%	28%	34%
8	24%	30%	36%
10	26%	32%	38%
12	28%	34%	40%

Table 1. *Parameters for Shapes.*

2. Five Random Shapes.

This CAPTCHA is inspired by the five symbols on cards that have long been used to test for ability with extrasensory perception (ESP): a circle, a square, a plus, three wavy lines, and a star. For the test, I turned the three wavy lines into three vertical lines, and turned the (five-pointed) star into a triangle with a vertical left side. Table 1 gives parameters for the shapes.

Thus the software draws a shape in undistorted form with line thickness from 4 to 12 pixels. Then the shape is distorted both vertically and horizontally using random sin curves. All white background pixels are changed randomly to black with probability 0.5. Finally, the pixels in the lines of the shape itself are changed to white with probabilities given in the table, and the shape

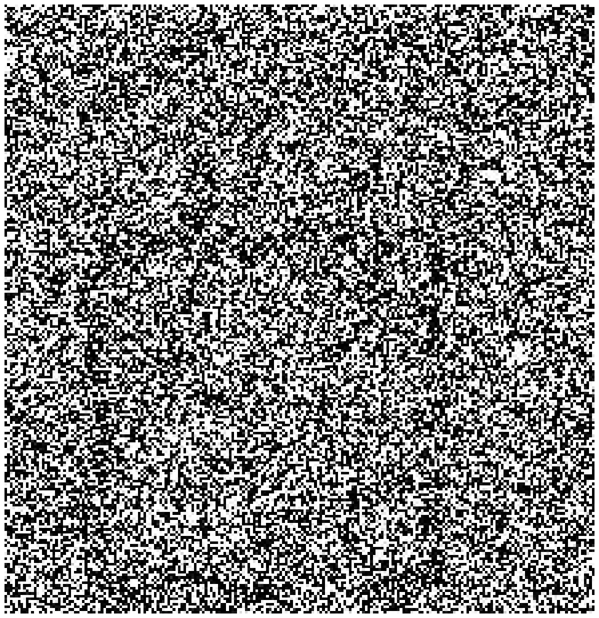

Figure 1. *Square Shape (dark).*

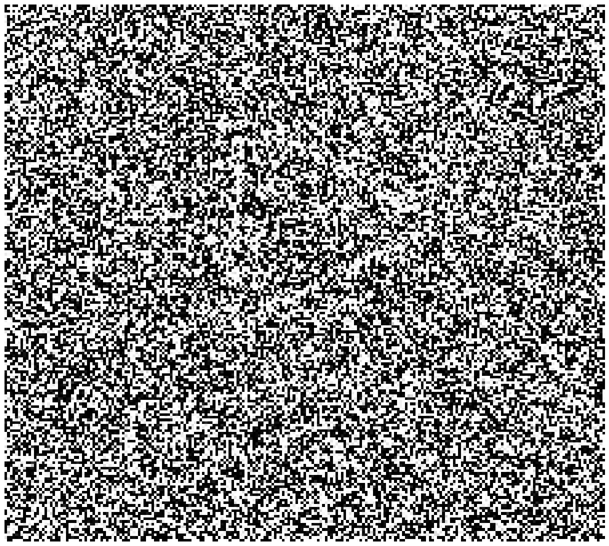

Figure 2. *Triangle Shape (light).*

drawing is black/white reversed with probability 0.5. In particular, for a shape with line thickness 12, and using the "hard" mode, the drawing itself is 40/60 white/black (or vice versa).

Right now the orientation of the shapes is not randomized, that is, there is only one orientation. It would also be possible to insert artifacts into the field, particularly additional portions of lines to confuse software analysis. Unlike many other CAPTCHAs, this one in its "hard" setting is right at the edge of human ability, so that any such additions would be equally confusing to humans, and one would need to decrease the numbers in Table 1 for humans to guess these shapes.

Software filtering techniques could try to find areas of the pictures with statistically significant variations of pixels from the average. Whether such software could actually guess the shape with reasonable accuracy in not known to me.

Figures 1 and 2 show pictures of two shapes as displayed in the "hard" setting of the CAPTCHA. I personally miss up to 5% of the shapes in this "hard" setting, although I can guess all shapes correctly in the other settings. See Section 5 for access to Java software to try out this and the other CAPTCHAs.

3. Distorted Pictures.

This and the next section treat pictures with errors or mistakes in them. To pass the test a human must find these errors. The approach in this section presents a picture with a region of distortion in it. The human user is supposed to click the mouse at the center of the distortion; he passes the test if he is fairly close to the center, perhaps in several successive pictures.

The specific approach here inserts a "fisheye" into a picture with enough regularity so that a human can find the region of distortion. For this application, a *fisheye* is a circular distortion that expands a picture at the center, contracts toward the edges, and smoothes out at the radius to end up the same as the original picture, without any sharp "kink" at that radius. The particular fisheye used here appears at a random location (though not too close to the boundary) and with a random radius (though not too small). The expansion and contraction from the center to the edge of the circle is given by the inverse of the equation:

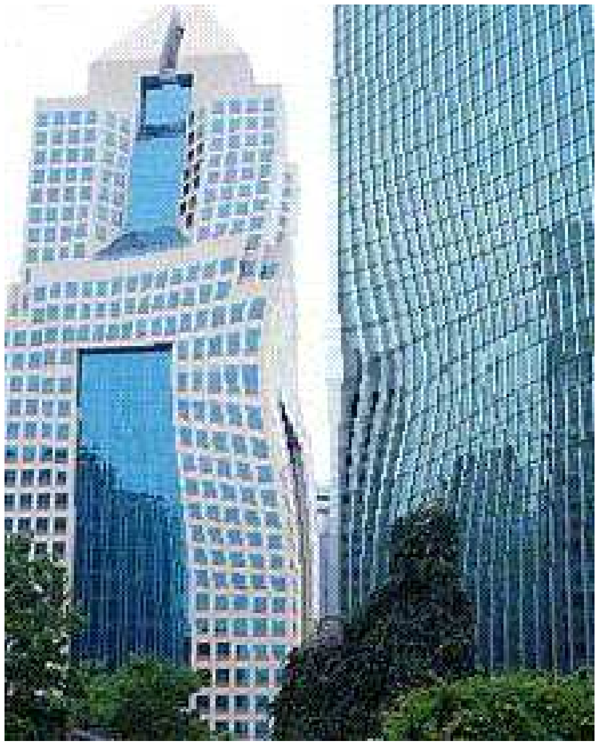

Figure 3. *Fisheye Picture.*

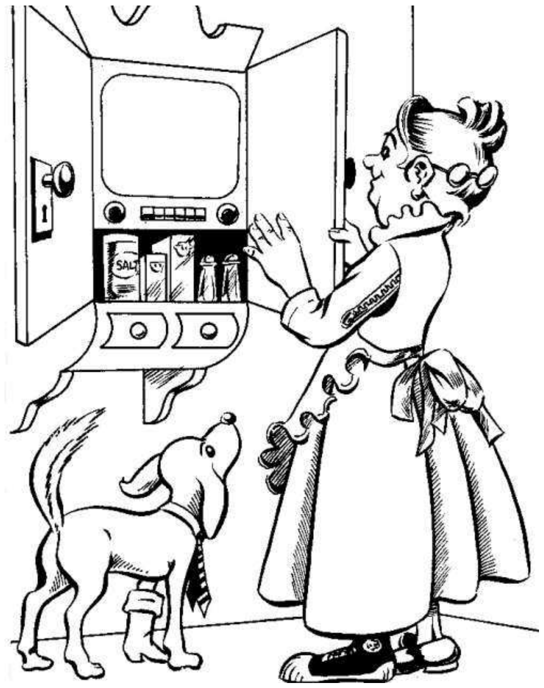

Figure 4. *Mother Hubbard*

$$g(s) = -\frac{3}{4}s^3 + \frac{3}{2}s^2 + \frac{1}{4}s, \;\; 0 \le s \le 1.$$

For this equation, I needed $g(0) = 0$, $g(1) = 1$, $g'(1) = 1$, and $g'(0)$ well below 1; in this case I chose the above simple function with $g'(0) = 1/4$.

Figure 3 contains a sample picture with such a fisheye, and Section 5 discusses a prototype implementation. In an actual system, one would want various additional distortions and randomizations, including cropping and distorting the original image, use of a random-shaped elliptical fisheye, and a final randomization of the resulting pixels.

4. What's Wrong With This Picture?

A common puzzle for children gives them a confusing picture with a number of items that "don't belong" in the picture. A picture of a fishbowl may have a lit candle in it, or a bedroom might have a fish on the wall. Identifying such "mistakes" is subjective and even dependent on cultural assumptions—hence potentially hard for machines to analyze and (if carefully constructed) easy of humans to spot. The underlying motivational idea is illustrated by Figure 4, which presents a common form of puzzle for children, asking the child to identify the "errors" in the picture. In this picture, some of the errors are obvious, but other less so; thus a CAPTCHA based on this idea should have unequivocal errors.

The particular system here starts with one of a number of generic background pictures, into which many objects would naturally fit. Pictures of quite a few of these objects are inserted, along with perhaps 3–5 objects that unquestionably do not belong in such a picture. More specifically, the CAPTCHA pictured in Figure 4 starts with a spider web, and adds various pictures of bugs. Then several other pictures of objects are added

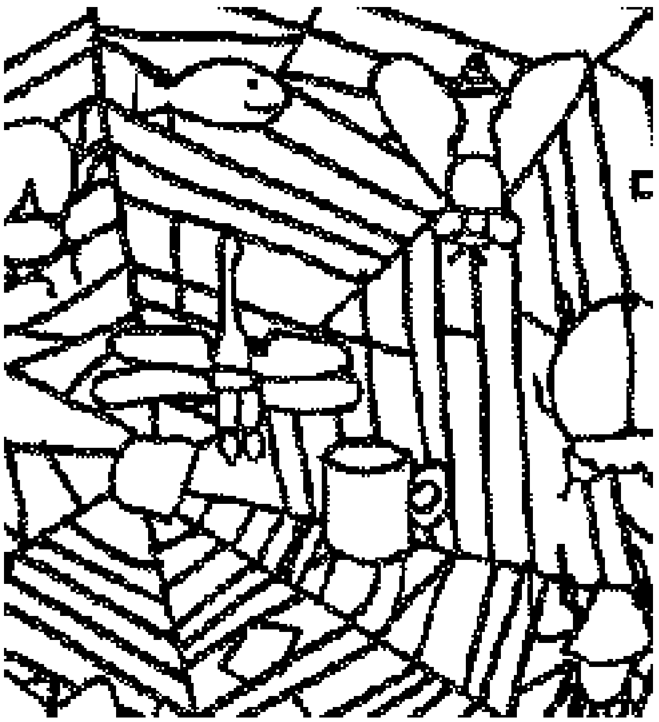

Figure 5. *Wrong Items (portion).*

that would be insane to find in a spider web. The user is asked to click the mouse at the rough center of any object not belonging. Figure 4 is a portion of a picture with 15 small additions altogether, and 5 that do not belong—here only the goldfish and coffee mug are wrong. Random software-generated clicks on objects would only get the correct 5 one time out of $C(15,5) = \frac{15!}{5!10!} = 2730$, that is, with probability 0.000366.

This system is quick and easy to use. It is also linguistically neutral, though it might be culturally dependent.

5. Prototype Implementations.

The author has prototype implementations of the three CAPTCHAs in the previous sections at:

`http://www.cs.utsa.edu/~wagner/captcha/`

These are just experimental implementations using Java applets. They are given to present the flavor of each of the approaches, but are not a final polished version. Each of these systems would be fairly easy for a reader to implement by hand.

The CAPTCHA consisting of 5 shapes in random noise is the closest to a complete system as presented. It is also the easiest to implement.

The second approach using a fisheye distortion needs the most work beyond what is presented online. Section 4 discussed some of the refinements that would be needed.

The third approach with errors in a picture consists at present of a single background picture (a spider web), and a limited collection of foreground objects (bugs, and other completely inappropriate objects). Even though one assumes that the complete database of images is public, it is still necessary to have a number of background pictures, and a large number of possible foreground objects. Of course different objects would be appropriate for different backgrounds. Each background should be clipped from a larger background and then distorted. Similarly the foreground objects need to be distorted and randomly rotated, as well as randomly placed as shown in the example applet. It was always clear that all the images here should be simple black and white line drawings, with thick lines. The author had such trouble finding appropriate drawings, that he finally commissioned his 12-year-old daughter to produce some, and that is what is shown in the applet.

References.

[Ahn2003] L. von Ahn, M. Blum, and J. Langford. “Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI,” *Comm. ACM* (to appear).

[Gar2003] S. Garfinkel, “Excuse Me, Are You Human?” *Technology Review* (June. 2003), p. 28.