CAPTCHA
S. SAI RAMA PALLAVI
B-Tech 2nd yr. CSE
Swami Vivekananda Institute of Technology

ABSTRACT
Every web surfer, in the course of his or her browsing, has been forced to stop and perform this weird little task: look at a picture of some wavy, distorted letters and type them into a box. That string of letters is called a CAPTCHA. And it’s a test. For you, the test should be simple and straightforward. But for a computer, the test should be almost impossible to solve. And by transcribing it, you have proved to computer that you are a human being. A CAPTCHA is a program that can generate and grade tests that humans can pass but current computer programs cannot. For example, humans can read distorted text or can choose the odd ones from a given list, but current computer programs cannot.

This paper includes the definition, history and need to develop a CAPTCHA, various types and development of CAPTCHA, the algorithms, applications and advantages of CAPTCHA.

KEYWORDS
CAPTCHA, Turing Test, Bots, Email, Spam, Algorithm, Image Processing.

INTRODUCTION
With an increasing number of free services on the internet, we find a pronounced need to protect these services from abuse. Automated programs (often referred to as bots) have been designed to attack a variety of services. To thwart automated attacks the CAPTCHA was developed. 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. 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.
A BRIEF HISTORY

In November 1999, for example, the Web site slashdot.com released an online poll asking which was the best graduate school in computer science— 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. Students from the Carnegie Mellon University and the Massachusetts Institute of Technology created bots that repeatedly voted for their respective colleges. This incident created the urge to use CAPTCHAs for such online polls to ensure that only human users are able to take part in the polls. Moni Naor was the first person to theorize a list of ways to verify that a request comes from a human and not a bot. Primitive CAPTCHAs seem to have been developed in 1997 by Andrei Broder, Martin Abadi, Krishna Bharat, and Mark Lillibridge to prevent bots from adding URLs to their search engine. In order to make the images resistant to OCR (Optical Character Recognition), the team simulated situations that scanner manuals claimed resulted in bad OCR. In 2000, Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford coined the term ‘CAPTCHA’, improved and publicized the notion, which included any program that can distinguish humans from computers. They invented multiple examples of CAPTCHAs, including the first CAPTCHAs to be widely used, which were those adopted by Yahoo!

APPLICATIONS

CAPTCHAs have several applications for practical security, including:

1. Preventing Comment Spam in Blogs. Most bloggers are familiar with programs that submit bogus comments, usually for the purpose of raising search engine ranks of some website (e.g., "buy penny stocks here"). This is called comment spam. By using a CAPTCHA, only humans can enter comments on a blog.

2. Protecting Website Registration. Several companies (Yahoo!, Microsoft, etc.) offer free email services. Up until a few years ago, most of these services suffered from a specific type of attack: "bots" that would sign up for thousands of email accounts every minute. The solution to this problem was to use CAPTCHAs to ensure that only humans obtain free accounts. In general, free services should be protected with a CAPTCHA in order to prevent abuse by bots.

3. Protecting Email Addresses From Scrapers. Spammers crawl the Web in search of email addresses posted in clear text. CAPTCHAs provide an effective mechanism to hide your email address from Web scrapers. The idea is to require users to solve a CAPTCHA before showing your email address.

4. Online Polls. Can the result of any online poll be trusted? Not unless the poll ensures that only humans can vote. Hence is the need for CAPTCHA.
5. Preventing Dictionary Attacks. CAPTCHAs can also be used 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 it to solve a CAPTCHA after a certain number of unsuccessful logins. This is better than the classic approach of locking an account after a sequence of unsuccessful logins, since doing so allows an attacker to lock accounts at will.

6. Search Engine Bots. It is sometimes desirable to keep webpages unindexed to prevent others from finding them easily. There is an html tag to prevent search engine bots from reading web pages. The tag, however, doesn't guarantee that bots won't read a 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 them in. However, in order to truly guarantee that bots won't enter a web site, CAPTCHAs are needed.

7. Worms and Spam. CAPTCHAs also offer a plausible solution against email worms and spam.

**TYPES OF CAPTCHA**

CAPTCHAs are classified based on what is distorted and presented as a challenge to the user.

1. **Text CAPTCHAs:**
   These are simple to implement. The simplest yet novel approach is to present the user with some questions which only a human user can solve. Examples of such questions are:
   1. What is twenty minus three?
   2. What is the third letter in UNIVERSITY?
   3. Which of Yellow, Thursday and Richard is a colour?
   4. If yesterday was a Sunday, what is today?

   Other CAPTCHAs include:

1.a. **Gimpy:**
   Gimpy is a very reliable text CAPTCHA built by CMU in collaboration with Yahoo for their Messenger service. Gimpy is based on the human ability to read extremely distorted text and the inability of computer programs to do the same. Gimpy works by choosing ten CAPTCHA words randomly from a dictionary, and displaying them in a distorted and overlapped manner. Gimpy then asks the users to enter a subset of the words in the image. The human user is capable of identifying the words correctly, whereas a computer program cannot.
1.b. Ez – Gimpy:
This is a simplified version of the Gimpy CAPTCHA, adopted by Yahoo in their signup page. Ez – Gimpy randomly picks a single word from a dictionary and applies distortion to the text. The user is then asked to identify the text correctly.

![Fig. Yahoo’s Ez – Gimpy CAPTCHA](image)

1.c. BaffleText:
This was developed by Henry Baird at University of California at Berkeley. This is a variation of the Gimpy. This doesn’t contain dictionary words, but it picks up random alphabets to create a nonsense but pronounceable text. Distortions are then added to this text and the user is challenged to guess the right word. This technique overcomes the drawback of Gimpy CAPTCHA because, Gimpy uses dictionary words and hence, clever bots could be designed to check the dictionary for the matching word by brute-force.

![finans](image)

1.d. MSN Captcha:
Microsoft uses a different CAPTCHA for services provided under MSN umbrella. These are popularly called MSN Passport CAPTCHAs. They use eight characters (upper case) and digits. Foreground is dark blue, and background is grey. Warping is used to distort the characters, to produce a ripple effect, which makes computer recognition very difficult.

![XTNM5YRE](image)

2. Graphic CAPTCHAs:
Graphic CAPTCHAs are challenges that involve pictures or objects that have some sort of similarity that the users have to guess. They are visual puzzles, similar to Mensa tests. Computer generates the puzzles and grades the answers, but is itself unable to solve it.

2.a. Bongo:
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 below.
These two sets are different because everything on the left is drawn with thick lines and those on the right are in thin lines. After seeing the two blocks, the user is presented with a set of four single blocks and is asked to determine to which group the each block belongs to. User passes the test if s/he determines correctly to which set the blocks belong to. We have to be careful to see that the user is not confused by a large number of choices.

2.b. PIX:
PIX is a program that has a large database of labeled images. 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?” 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.

3. Audio CAPTCHAs:
The final example we offer 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 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. The idea is that a human is able to efficiently disregard the distortion and interpret the characters being read out while software would struggle with the distortion being applied, and need to be effective at speech to text translation in order to be successful. This is a crude way to filter humans and it is not so popular because the user has to understand the language and the accent in which the sound clip is recorded.

GUIDELINES

If your website needs protection from abuse, it is recommended that you use a CAPTCHA. There are many CAPTCHA implementations, some better than others. The following guidelines are strongly recommended for any CAPTCHA code:

1. Accessibility. CAPTCHAs must be accessible. CAPTCHAs based solely on reading text — or other visual-perception tasks — prevent visually impaired users from accessing the protected
resource. Such CAPTCHAs may make a site incompatible with Section 508 in the United States. Any implementation of a CAPTCHA should allow blind users to get around the barrier, for example, by permitting users to opt for an audio or sound CAPTCHA.

2. **Image Security.** CAPTCHA images of text should be distorted randomly before being presented to the user. Many implementations of CAPTCHAs use undistorted text, or text with only minor distortions. These implementations are vulnerable to simple automated attacks.

3. **Script Security.** Building a secure CAPTCHA code is not easy. In addition to making the images unreadable by computers, the system should ensure that there are no easy ways around it at the script level. Common examples of insecurities in this respect include: (1) Systems that pass the answer to the CAPTCHA in plain text as part of the web form. (2) Systems where a solution to the same CAPTCHA can be used multiple times (this makes the CAPTCHA vulnerable to so-called "replay attacks"). Most CAPTCHA scripts found freely on the Web are vulnerable to these types of attacks.

4. **Security Even After Wide-Spread Adoption.** There are various "CAPTCHAs" that would be insecure if a significant number of sites started using them. An example of such a puzzle is asking text-based questions, such as a mathematical question ("what is 1+1"). Since a parser could easily be written that would allow bots to bypass this test, such "CAPTCHAs" rely on the fact that few sites use them, and thus that a bot author has no incentive to program their bot to solve that challenge. True CAPTCHAs should be secure even after a significant number of websites adopt them.

5. **Should I Make My Own CAPTCHA?** In general, making your own CAPTCHA script (e.g., using PHP, Perl or .Net) is a bad idea, as there are many failure modes.

**BREAKING CAPTCHA**

The challenge in breaking a CAPTCHA isn't figuring out what a message says -- the really hard task is teaching a computer how to process information in a way similar to how humans think.

Another vulnerability that most CAPTCHA scripts have is again in their use of sessions; if we're on an insecure shared server, any user on that server may have access to everyone else's session files, so even if our site is totally secure, a vulnerability on any other website hosted on that machine can lead to a compromise of the session data, and hence, the CAPTCHA script. One workaround is by storing only a hash of the CAPTCHA word in the session, thus even if someone can read the session files, they can't find out what the CAPTCHA word is.

1. **Breaking CAPTCHAs without OCR:**
Most CAPTCHAs don't destroy the session when the correct phrase is entered. So by reusing the session id of a known CAPTCHA image, it is possible to automate requests to a CAPTCHA-protected page.

1.a. **Manual steps:**
Connect to CAPTCHA page
Record session ID and CAPTCHA plaintext

1.b. Automated steps:
Resend session ID and CAPTCHA plaintext any number of times, changing the user data. The other user data can change on each request. We can then automate hundreds, if not thousands of requests, until the session expires, at which point we just repeat the manual steps and then reconnect with a new session ID and CAPTCHA text.

Traditional CAPTCHA-breaking software involves using image recognition routines to decode CAPTCHA images. This approach bypasses the need to do any of that, making it easy to hack CAPTCHA images.

2. Breaking a visual CAPTCHA:
Greg Mori and Jitendra Malik of University of California at Berkeley's Computer Vision Group evaluate image based CAPTCHAs for reliability. They test whether the CAPTCHA can withstand bots who masquerade as humans.

Algorithm for breaking visual CAPTCHA consists of 3 main steps:

2.a. Locate possible (candidate) letters at various locations: The first step is to hypothesize a set of candidate letters in the image. This is done using our shape matching techniques. The method essentially looks at a bunch of points in the image at random, and compares these points to points on each of the 26 letters. The comparison is done in a way that is very robust to background clutter and deformation of the letters. The process usually results in 3-5 candidate letters per actual letter in the image. In the example shown, the "p" of profit matches well to both an "o" or a "p", the border between the "p" and the "r" look a bit like a "u", and so forth. At this stage we keep many candidates, to be sure we don't miss anything for later steps.

![Fig. Breaking of Visual CAPTCHA](image)

2.b. Construct graph of consistent letters: Next, we analyze pairs of letters to see whether or not they are "consistent", or can be used consecutively to form a word.

2.c. Look for plausible words in the graph: There are many possible paths through the graph of letters constructed in the previous step. However, most of them do not form real words. We select out the real words in the graph, and assign scores to them based on how well their individual letters match the image. Similar algorithms are also devised by Mori and Malik to evaluate other image based CAPTCHAs.

3. Breaking an audio CAPTCHA:
Recent research is suggesting that Google's audio capture is the latest in a string of CAPTCHA's to have been defeated by software. It has been theorized that one cost-effective means of breaking audio captures and image captures that have not yet had automated systems developed is to use a mechanical turk and pay low rates for per-CAPTCHA reading by humans, or provide another form of motivation such as access to popular sites for reading the CAPTCHA. However, it always required a
significant level of resources to achieve. The development of software to automatically interpret CAPTCHAs brings up a number of problems for site operators. The problem, as discovered by Wintercore Labs and published at the start of March is that there are repeatable patterns evident in the audio file and by applying a set of complex but straightforward processes, a library can be built of the basic signal for each possible character that can appear in the CAPTCHA. Wintercore point to other audio CAPTCHAs that could be easily reversed using this technique, including the one for Facebook. The wider impact of this work might take some time to appear, but it provides an interesting proof of breaking audio CAPTCHAs. At the least, it shows that both of Google’s CAPTCHA tools have now been defeated by software and it should only be a matter of time until the same can be said for Microsoft and Yahoo!'s offerings. Even with an effectiveness of only 90%, any failed CAPTCHA can easily be reloaded for a second try.

4. Social Engineering used to break CAPTCHAs:
Spammers often use social engineering to outwit gullible Web users to serve their purpose. Security firm, Trend Micro warns of a Trojan called TROJ_CAPTCHAR, which masquerades as a strip tease game. At each stage of the game, the user is asked to solve a CAPTCHA. The result is relayed to a remote server where a malicious user is waiting for them. The strip-tease game is a ploy by spammers to identify and match solutions for ambiguous CAPTCHAs from legitimate sites, using the unsuspecting user as the decoder of the said images.

CONCLUSION
CAPTCHA can be a great way to limit the amount of successful, unwanted HTTP POST requests in your application, CAPTCHAs are by definition fully automated, requiring little human maintenance or intervention in administering the test. This has obvious benefits in cost and reliability. There would be a lot of work required to make this foolproof, and also take time to establish a trustworthy community.

REFERENCES

2. What’s Up CAPTCHA? A CAPTCHA Based On Image Orientation from WWW'09by Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja

