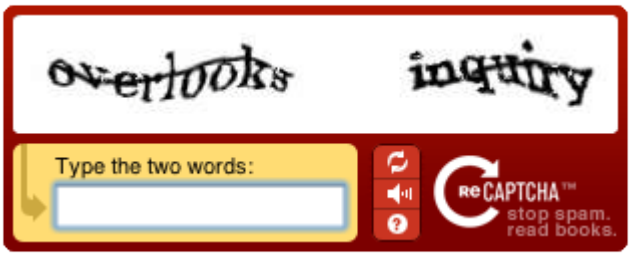
CAPTCHA



Navin kumar

970100

KIIT School of Computer Application

KIIT UNIVERSITY

Bhubaneswar, Odisha, India

content **Page No**

1.ABSTRACT

One common application of CAPTCHA is for verifying online polls. In fact, a former Slashdot poll serves as an example of what can go wrong if pollsters don't implement filters on their surveys. In 1999, Slashdot published a poll that asked visitors to choose the graduate school that had the best program in computer science. Students from two universities -- Carnegie Mellon and MIT -- created automated programs called **bots** to vote repeatedly for their respective schools. While those two schools received thousands of votes, the other schools only had a few hundred each. If it's possible to create a program that can vote in a poll, how can we trust online poll results at all? A CAPTCHA form can help prevent programmers from taking advantage of the polling system.

Registration forms on Web sites often use CAPTCHAs. For example, free Web-based e-mail services like Hotmail, yahoo mail or gmailallow people to create an e-mail account free of charge. Usually, users must provide some personal information when creating an account, but the services typically don't verify this information. They use CAPTCHAs to try to prevent spammers from using bots to generate hundreds of spam mail accounts.

HowStuffWorks Yahoo uses alphanumeric strings rather than words as CAPTCHAs when you sign up for a Yahoo! account. Ticket brokers like TicketMaster also use CAPTCHA applications. These applications help prevent ticket scalpers from bombarding the service with massive ticket purchases for big events. Without some sort of filter, it's possible for a scalper to use a bot to place hundreds or thousands of ticket orders in a matter of seconds. Legitimate customers become victims as events sell out minutes after tickets become available. Scalpers then try to sell the tickets above face value.

While CAPTCHA applications don't prevent scalping, they do make it more difficult to scalp tickets on a large scale.

Some Web pages have message boards or contact forms that allow visitors to either post messages to the site or send them directly to the Web administrators. To prevent an avalanche of spam, many of these sites have a CAPTCHA program to filter out the noise. A CAPTCHA won't stop someone who is determined to post a rude message or harass an administrator, but it will help prevent bots from posting messages automatically.

The most common form of CAPTCHA requires visitors to type in a word or series of letters and numbers that the application has distorted in some way. Some CAPTCHA creators came up with a way to increase the value of such an application: digitizing books. An application called reCAPTCHA harnesses users responses in CAPTCHA fields to verify the contents of a scanned piece of paper. Because computers aren't always able to identify words from a digital scan, humans have to verify what a printed page says. Then it's possible for search engines to search and index the contents of a scanned document.

Here's how it works: First, the administrator of the reCAPTCHA program digitally scans a book. Then, the reCAPTCHA program selects two words from the digitized image. The application already recognizes one of the words. If the visitor types that word into a field correctly, the application assumes the second word the user types is also correct. That second word goes into a pool of words that the application will present to other users. As each user types in a word, the application compares the word to the original answer. Eventually, the application receives enough responses to verify the word with a high degree of certainty. That word can then go into the verified pool.

It sounds time consuming, but remember that in this case the CAPTCHA is pulling double duty. Not only is it verifying the contents of a digitized book, it's also verifying that the people filling out the form are actually people. In turn, those people are gaining access to a service they want to use

2. INTRODUCTION

what is captcha?

If you try to get a new email account at Yahoo, you'll 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. But how do you prove to a computer that you'rea human?

Proving that you're a human to another human can be done using an idea from the 1950s: theTuring Test. A human judge asks you a bunch of questions and decides, depending on your answers, whether he's talking to a human or a computer. Proving that you're a human to a computer is another matter. It requires a test (or a set of tests) that computers can grade, humans can pass, but paradoxically, computers can't pass. In our lingo, it requires a captcha.so

A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot. For example, humans can read distorted text as the one shown below, but current computer programs can't:

The term CAPTCHA (for Completely Automated Public Turing Test To Tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University.
It is a contrived acronym based on the word "capture" and standing for "**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part". Carnegie Mellon University attempted to trademark the term,but the trademark application was abandoned on 21 April 2008.

Why they are used

CAPTCHAs are used to prevent robots from submitting forms and creating accounts, spamming and various other things. In some cases robots can cause some problems. Take for example a robot signing up for thousands of Gmail accounts. While it might not cause much stress on Gmail's servers it would create lots of email accounts that could be used for spamming people. Another case is spammers creating accounts on forums and then spam the forum. CAPTCHAs help prevent robots from using websites and webapps.

Luis von Ahn of Carnegie Mellon University is one of the inventors of

CAPTCHA, In the year of 2006.
Captcha is some times called reverse turing test.
Reverse turing test:

CAPTCHA technology has its foundation in an experiment called the **Turing Test**. Alan Turing, sometimes called the father of modern computing, proposed the test as a way to examine whether or not machines can think -- or appear to think -- like humans. The classic test is a game of imitation. In this game, an interrogator asks two participants a series of questions. One of the participants is a machine and the other is a human. The interrogator can't see or hear the participants and has no way of knowing which is which. If the interrogator is unable to figure out which participant is a machine based on the responses, the machine passes the Turing Test.

Of course, with a CAPTCHA, the goal is to create a test that humans can pass easily but machines can't. It's also important that the CAPTCHA application is able to present different CAPTCHAs to different users. If a visual CAPTCHA presented a static image that was the same for every user, it wouldn't take long before a spammer spotted the form, deciphered the letters, and programmed an application to type in the correct answer automatically.

Characteristics

A CAPTCHA is a means of automatically generating challenges which intends to:

- Provide a problem easy enough for all humans to solve.
- Prevent standard automated software from filling out a form, unless it is specially designed to circumvent specific CAPTCHA systems.

A check box in a form that reads "check this box please" is the simplest (and perhaps least effective) form of a CAPTCHA. CAPTCHAs do not have to rely on difficult problems in artificial intelligence, although they can.

This has the benefit of distinguishing humans from computers. It also creates incentive to further develop artificial intelligence of computers

Type of captcha:

There are two main type of captcha:

1.visual

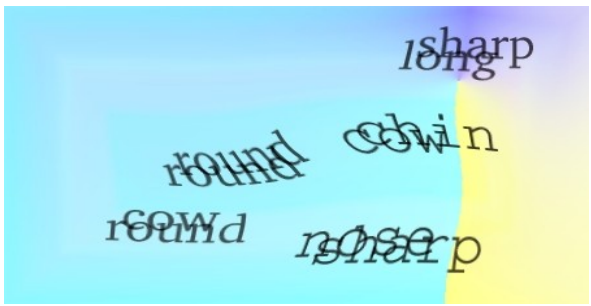
3.graphics

3.audio(it is mainly used for visual imapaired people)

1.Gimpy :

Gimpy(it is text based captcha). gimpy works as follows: it picks seven words out of a dictionary, and renders a
distorted image containing the words. 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 can't.



2.Bongo:

Bongo(it is graphics captcha). 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.

3.Pix:

Pix(it is graphics captcha). Yet another example of a captcha is a program that has a large database of labeled

images (such databases can be found in [6, 5]). All of these images should be pictures of
concrete objects (a horse, a table, a house, a ower, etc). The program picks an object at
random, _nds 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.
But actually, pix, as stated, is not a captcha: 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 _nd their label. One way for
pix to become a captcha is to randomly distort the images before presenting them to the
user, in such a way that searching the database for the images is hard for current computer

programs.

4.eco:

Eco is a sound-based captcha. 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 the contents of the sound clip. eco is based on the gap in ability between humans and computers in recognizing spoken language. Nancy Chan of the City University in Hong Kong has also implemented a sound-based system of this variety .

Progresses in captchas:

Type 1.early captchas

- **Generated by the EZ-Gimpy program;**

  Used previously on Yahoo!

Type2.: Improved CAPTCHA.

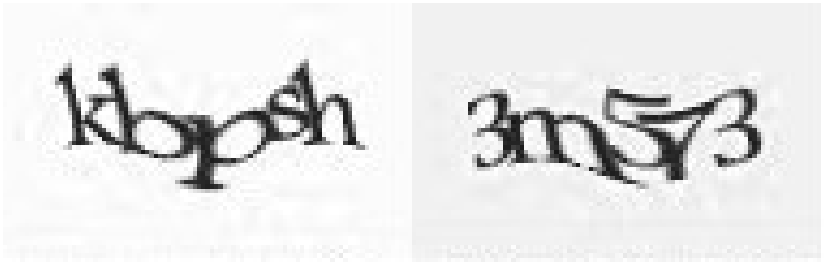
- **high contrast for human readability;**
- **medium, per-character perturbation;**
- **random fonts per character;**
- **low background noise;**

3Type: A modern CAPTCHA.

- **rather than attempting to create a distorted background and high levels of warping on the text;**
- **focus on making segmentation difficult by adding an angled line;**

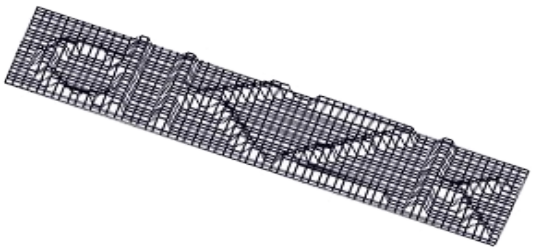
- **another way to make segmentation difficult is to crowd symbols together;**

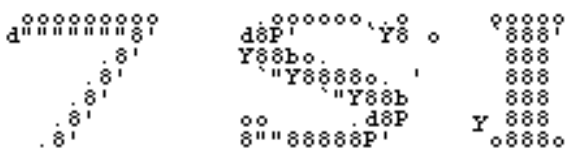
  this can be read by humans but cannot be



Other Types of CAPTCHA

- Animated CAPTCHAs
- 3D CAPTCHA
- ASCII art
- Reverse CAPTCHA "Leave this field blank"



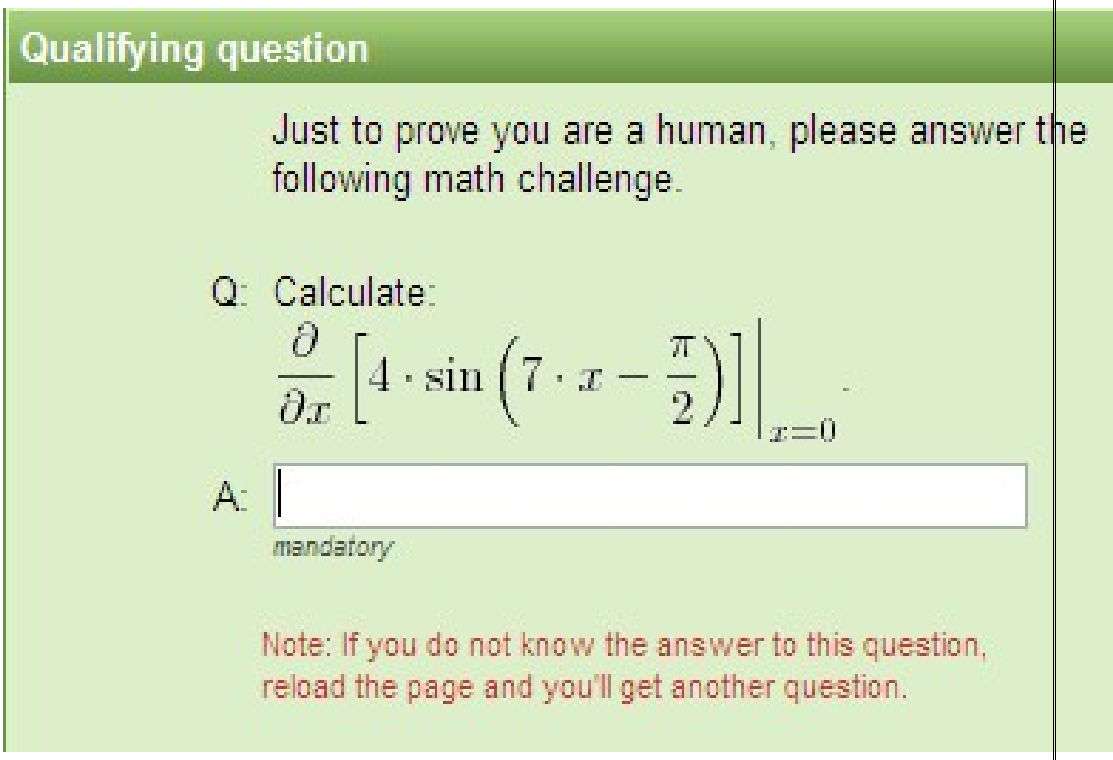


Other: Cognitive Puzzles

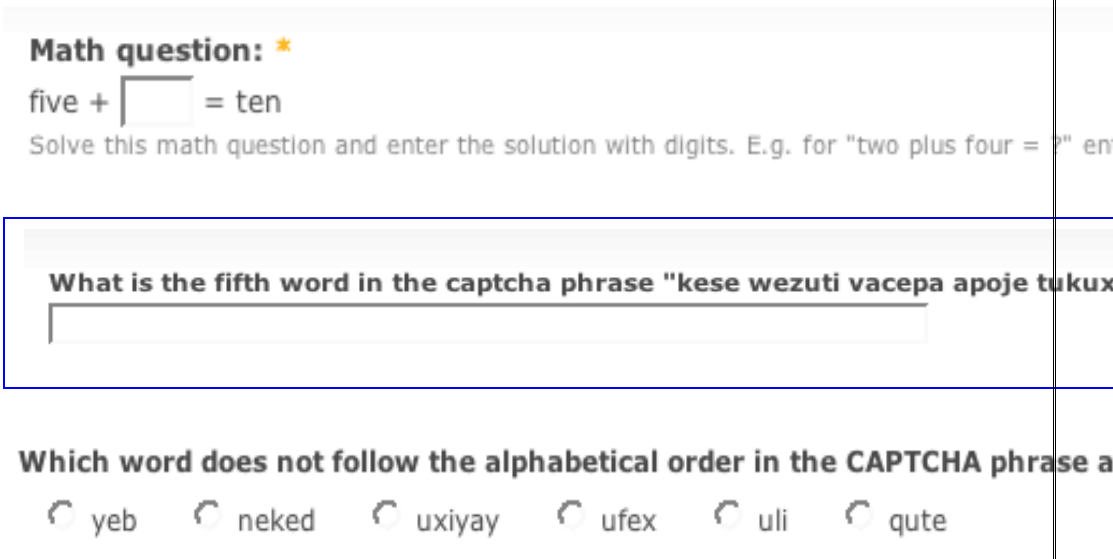
- **Distinguish pictures of dogs from cats**
- **Choose a word that relates to all the images**
- **Trivia questions**

- **Math and word problems**
- **3D Object CAPTCHA**
- **Solve failed OCR inputs**

Other: Mathematical CAPTCHA



Other: Drupal Examples.



Family of captcha:

Two Families of captchas

We now describe two families of captchas whose security is based on the hardness of problems in P1 and P2. 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 = (I; T ; t), where I is a distri-
bution on images and 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 = (I; T) can be used to solve $P1_{I;T}$

PIX:

An instance $P2_{I;T;t}$ can sometimes be used almost directly as a captcha. For
instance, if I is a distribution over images containing a single word and _ maps an image to the word contained in it, then $P2_{I;T;t}$ can be used directly as a captcha. Similarly, if all the images in [I] are pictures of simple concrete objects and _ maps an image to the object that is contained in the image, then $P2_{I;T;\_}$ can be used as a captcha.

Circumvention

There are several approaches available to defeating CAPTCHAs:

- exploiting bugs in the implementation that allow the attacker to completely bypass the CAPTCHA,
- improving character recognition software, or
- using cheap human labor to process the tests

Insecure implementation

Like any security system, design flaws in a system implementation can prevent the theoretical security from being realized. Many CAPTCHA implementations, especially those which have not been designed and reviewed by experts in the fields of security, are prone to common attacks.

Some CAPTCHA protection systems can be bypassed without using OCR simply by re-using the session ID of a known CAPTCHA image. A correctly designed CAPTCHA does not allow multiple solution attempts at one CAPTCHA. This prevents the reuse of a correct CAPTCHA solution or making a second guess after an incorrect OCR attempt.[Other CAPTCHA implementations use a hash (such as an MD5 hash) of the solution as a key passed to the client to validate the CAPTCHA. Often the CAPTCHA is of small enough size that this hash could be cracked. Further, the hash could assist an OCR based attempt. A more secure scheme would use an HMAC . Finally, some implementations use only a small fixed pool of CAPTCHA images. Eventually, when enough CAPTCHA image solutions have been collected by an attacker over a period of time, the CAPTCHA can be broken by simply looking up solutions in a table, based on a hash of the challenge image.

Computer character recognition

A number of research projects have attempted (often with success) to beat visual CAPTCHAs by creating programs that contain the following functionality:

1. Pre-processing: Removal of background clutter and noise.
2. Segmentation: Splitting the image into regions which each contain a single character.
3. Classification: Identifying the character in each region.

Steps 1 and 3 are easy tasks for computers. The only step where humans still outperform computers is segmentation. If the background clutter consists of shapes similar to letter shapes, and the letters are connected by this clutter, the segmentation becomes nearly impossible with current software. Hence, an effective CAPTCHA should focus on the segmentation.

Several research projects have broken real world CAPTCHAs, including one of Yahoo's early CAPTCHAs called "EZ-Gimpy" and the CAPTCHA used by popular sites such as PayPal,[9] LiveJournal, phpBB, and other services. In January 2008 Network Security Research released their program for automated Yahoo! CAPTCHA recognition. Windows Live Hotmail and Gmail, the other two major free email providers, were cracked shortly after.

In February 2008 it was reported that spammers had achieved a success rate of 30% to 35%, using a bot, in responding to CAPTCHAs for Microsoft's Live Mail service[and a success rate of 20% against Google's Gmail CAPTCHA. A Newcastle University research team has defeated the segmentation part of Microsoft's CAPTCHA with a 90% success rate, and claim that this could lead to a complete crack with a greater than 60% rate.

Human solvers

CAPTCHA is vulnerable to a relay attack that uses humans to solve the puzzles. One approach involves relaying the puzzles to a group of human operators who can solve CAPTCHAs. In this scheme, a computer fills out a form and when it reaches a CAPTCHA, it gives the CAPTCHA to the human operator to solve.

Spammers pay about $0.80 to $1.20 for each 1,000 solved CAPTCHAs to companies employing human solvers in Bangladesh, China, India, and many other developing nations. Other sources cite a price tag of as low as $0.50 for each 1,000 solved.

Another approach involves copying the CAPTCHA images and using them as CAPTCHAs for a high-traffic site owned by the attacker. With enough traffic, the attacker can get a solution to the CAPTCHA puzzle in time to relay it back to the target site.In October 2007, a piece of malware appeared in the wild which enticed users to solve CAPTCHAs in order to see progressively further into a series of striptease images. A more recent view is that this is unlikely to work due to unavailability of high-traffic sites and competition by similar sites.

These methods have been used by spammers to set up thousands of accounts on free email services such as Gmail and Yahoo!.Since Gmail and Yahoo! are unlikely to be blacklisted by anti-spam systems, spam sent through these compromised accounts is less likely to be blocked.

Breaking a CAPTCHA:

The challenge in breaking a CAPTCHA isn't figuring out what a message says -- after all, humans should have at least an 80 percent success rate. The really hard task is teaching a computer how to process information in a way similar to how humans think. In many cases, people who break CAPTCHAs concentrate not on making computers smarter, but reducing the complexity of the problem posed by the CAPTCHA.

Let's assume you've protected an online form using a CAPTCHA that displays English words. The application warps the font slightly, stretching and bending the letters in unpredictable ways. In addition, the CAPTCHA includes a randomly generated background behind the word.

A programmer wishing to break this CAPTCHA could approach the problem in phases. He or she would need to write an **algorithm** -- a set of instructions that directs a machine to follow a certain series of steps. In this scenario, one step might be to convert the image in grayscale. That means the application removes all the color from the image, taking away one of the levels of obfuscation the CAPTCHA employs.

Next, the algorithm might tell the computer to detect patterns in the black and white image. The program compares each pattern to a normal letter, looking for matches. If the program can only match a few of the letters, it might cross reference those letters with a database of English words. Then it would plug in likely candidates into the submit field. This approach can be surprisingly effective. It might not work 100 percent of the time, but it can work often enough to be worthwhile to spammers.

What about more complex CAPTCHAs? The **Gimpy** CAPTCHA displays 10 English words with warped fonts across an irregular background. The CAPTCHA arranges the words in pairs and the words of each pair overlap one another. Users have to type in three correct words in order to move forward. How reliable is this approach?

As it turns out, with the right CAPTCHA-cracking algorithm, it's not terribly reliable. Greg Mori and Jitendra Malik published a paper detailing their approach to cracking the Gimpy version of CAPTCHA. One thing that helped them was that the Gimpy approach uses actual words rather than random strings of letters and numbers. With this in mind, Mori and Malik designed an algorithm that tried to identify words by examining the beginning and end of the string of letters. They also used the Gimpy's 500-word dictionary.

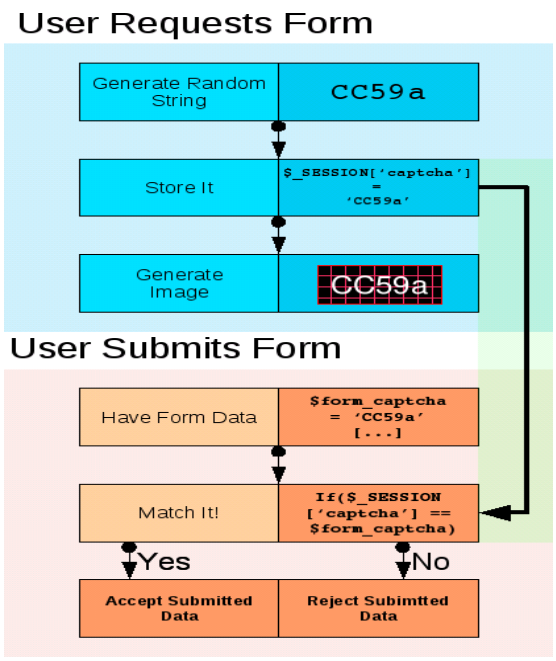
Mori and Malik ran a series of tests using their algorithm. They found that their algorithm could correctly identify the words in a Gimpy CAPTCHA 33 percent of the time [source: Mori and Malik]. While that's far from perfect, it's also significant. Spammers can afford to have only one-third of their attempts succeed if they set bots to break CAPTCHAs several hundred times every minute

How captcha works

Basically CAPTCHA works in the following manner:

1. Create Random Value: Some random string is generated, random values are often hard to guess and predict.
2. Generate an Image: Images are used as these are generally a lot harder to read for computers while being nice and readable to humans. This is also the most important step as simple text in images can be read (and CAPTCHA cracked) quite easily. To make it difficult for them, developers employ different techniques so that the text in the image becomes hard to read for computers. Some create zig-zag lines for background while others twist-and-turn individual characters in the image. Possibilities are many and new techniques are being developed all the time as crackers are always into finding ways to break them.
3. Store it: The random string generated (which is also in the image) is stored for matching the user input. The easiest way to do so is to use the *Session variables*.
4. Matching: After the above step, the CAPTCHA image is generated and shown on some form which we want to protect from being abused. The users fills in the form along with the CAPTCHA text and submits it. Now we have the following:
    a. All submitted form data.
    b. CAPTCHA string (from form), input by user.
    c. CAPTCHA string (real one, generated by us), from session variable. Session variable is generally used as it can keep stored values across page requests. Here, we needed to preserve stored values from one page (form page) to another (action page-that receives form data).
5. If both match, it's okay otherwise not, in that case we can give the user a message that the CAPTCHA they had entered was wrong and their form could not be submitted. You could also ask them to verify it again.

The following image might illustrates this better:



Applications of CAPTCHAs:

CAPTCHAs have several applications for practical security, including (but not limited to):

i **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 automated scripts.

a **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. A free and secure implementation that uses CAPTCHAs to obfuscate an email address can be found at reCAPTCHA MailHide.

. **Online Polls.** In November 1999, *http://www.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 using 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 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 ensures that only humans can vote.

u **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.

u **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.

g **Worms and Spam.** CAPTCHAs also offer a plausible solution against email worms and spam: "I will only accept an email if I know there is a human behind the other computer." A few companies are already marketing this idea

Guidelines:

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

- **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.

Attempts at more accessible CAPTCHAs:

Even audio and visual CAPTCHAs will require manual intervention for some users, such as those who have disabilities. There have been various attempts at creating more accessible CAPTCHAs, including the use of JavaScript, mathematical questions ("how much is 1+1") and common sense questions ("what colour is the sky on a clear day"). However, these types of CAPTCHAs do not meet the criteria for a successful CAPTCHA. They are not automatically generated and they do not present a new problem or test for each attack

- **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.

- **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.
- **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.

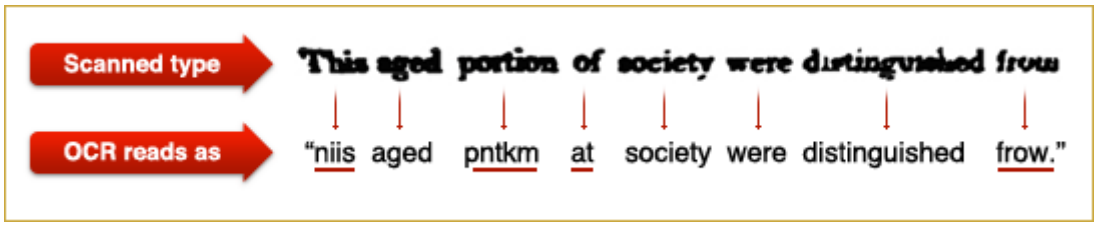
Should I Make My Own CAPTCHA?:

In general, making your own CAPTCHA script (e.g., using PHP, Perl or .Net) is a bad idea, as there are many failure modes. We recommend that you use a well-tested implementation such as reCAPTCHA.

WHAT IS reCAPTCHA? (channeling the effort spent solving CAPTCHAs online into "reading" books.)

reCAPTCHA is a free CAPTCHA service that helps to digitize books, newspapers and old time radio shows.

About 200 million CAPTCHAs are solved by humans around the world every day. In each case, roughly ten seconds of human time are being spent. Individually, that's not a lot of time, but in aggregate these little puzzles consume more than 150,000 hours of work each day. What if we could make positive use of this human effort? reCAPTCHA does exactly that by channeling the effort spent solving CAPTCHAs online into "reading" books.To archive human knowledge and to make information more accessible to the world, multiple projects are currently digitizing physical books that were written before the computer age. The book pages are being photographically scanned, and then transformed into text using "Optical Character Recognition" (OCR). The transformation into text is useful because scanning a book produces images, which are difficult to store on small devices, expensive to download, and cannot be searched. The problem is that OCR is not perfect.



reCAPTCHA improves the process of digitizing books by sending words that cannot be read by computers to the Web in the form of CAPTCHAs for humans to decipher. More specifically, each word that cannot be read correctly by OCR is placed on an image and used as a CAPTCHA. This is possible because most OCR programs alert you when a word cannot be read correctly.

But if a computer can't read such a CAPTCHA, how does the system know the correct answer to the puzzle? Here's how: Each new word that cannot be read correctly by OCR is given to a user in conjunction with another word for which the answer is already known. The user is then asked to read both words. If they solve the one for which the answer is known, the system assumes their answer is correct for the new one. The system then gives the new image to a number of other people to determine, with higher confidence, whether the original answer was correct.

The "Pornography Attack" is Not a Concern

It is sometimes rumored that spammers are using pornographic sites to solve CAPTCHAs: the CAPTCHA images are sent to a porn site, and the porn site users are asked to solve the CAPTCHA before being able to see a

pornographic image. **This is not a security concern for CAPTCHAs**. While it might be the case that some spammers use porn sites to attack CAPTCHAs, the amount of damage this can inflict is tiny (so tiny that we haven't even noticed a dent!). Whereas it is trivial to write a bot that abuses an unprotected site millions of times a day, redirecting CAPTCHAs to be solved by humans viewing pornography would only allow spammers to abuse systems a few thousand times per day. The economics of this attack just don't add up: every time a porn site shows a CAPTCHA before a porn image, they risk losing a customer to another site that doesn't do this.

Captcha and Artificial Intelligence

CAPTCHA tests are based on open problems in artificial intelligence (AI): decoding images of distorted text, for instance, is well beyond the capabilities of modern computers. Therefore, CAPTCHAs also offer well-defined challenges for the AI community, and induce security researchers, as well as otherwise malicious programmers, to work on advancing the field of AI. CAPTCHAs are thus a win-win situation: either a CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and an AI problem is solved.

Luis von Ahn of Carnegie Mellon University is one of the inventors of CAPTCHA. In a 2006 lecture, von Ahn talked about the relationship between things like CAPTCHA and the field of artificial intelligence (AI). Because CAPTCHA is a barrier between spammers or hackers and their goal, these people have dedicated time and energy toward breaking CAPTCHAs. Their successes mean that machines are getting more sophisticated. Every time someone figures out how to teach a machine to defeat a CAPTCHA, we move one step closer to artificial intelligence.

Spammers Advance AI Technology to Trick CAPTCHA

Optical Character Recognition (OCR) software is used to allow a computer to "read" text and numbers. People who use spam bots to automatically post links to Internet forums are typically foiled by CAPTCHA images, since the images of slightly distorted text and numbers are decipherable by humans, but difficult for an OCR. As a result, spam programmers are working to achieve an AI program that can recognize the images used in CAPTCHA boxes.

OCR software interprets an image, and compares it to other images in a database. Early OCR software required training for each font, and could only recognize a limited amount of text. Current OCR programs are much more sophisticated, and can recognize letters and text that contain differences, within a certain threshold. Handwriting recognition is also available in OCR software, but is more difficult, and therefore, more expensive.

In the case of a forum spam bot, the bot uses the information gleaned from this comparison to determine the correct threads in which to post responses. The

bot contains information to enable it to automatically post a brief message and a link in the response portion of a thread, but the link will only be effective if the link is pertinent to the thread topic. Once the OCR has found a thread that matches its internal criteria, it posts the message and marketing links to the thread, increasing website traffic without the need for an actual marketing team. CAPTCHA boxes are placed in the forum submission area, requiring a "human" response for a forum posting. This prevents a spam bot from completing the process, due to the bot's inability to translate distorted images.

-

CAPTCHA Programming Advances

Currently, programmers that work on CAPTCHA boxes are firmly on one side or the other- either working to make the CAPTCHA boxes more difficult to beat, or trying to find better ways to beat them. In order to beat the CAPTCHA box, programmers will have to find a way for the AI programs to learn to recognize distorted images, which will provide many other benefits for AI technology

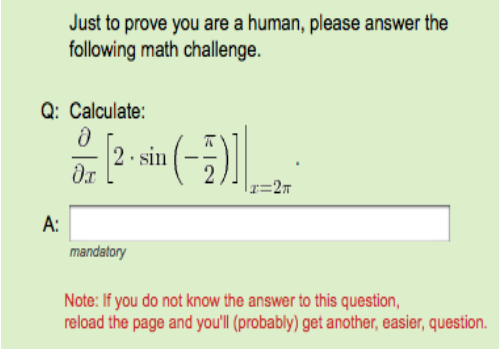
The Bad:

There are some downsides to using CAPTCHAs that I have already alluded to. CAPTCHAs aren't always 100 percent effective against robots. There are tools out there that allow for robots to read and extract the text of images then rendering a CAPTCHA useless.

We also have to consider folks who are blind or have cognitive disabilities. Now you may say that the audio CAPTCHAs solve the blind user dilemma and it just might, but have you ever tired to listen to one of them? I've tired and

never been quite able to understand them, and I like to think I have pretty good hearing. As for the folks with cognitive disabilities, they may have problems understanding what it is you are asking them to do with a CAPTCHA.

So let us consider an elderly person with poor vision and poor hearing. Using a CAPTCHA, even with an audio component, will effectively eliminate them as a user. CAPTCHAs can be hard for people with 20/20 vision to understand.



There is another fundamental problem with CAPTCHAs and the crux of this article. CAPTCHAs assume by their very nature that all users are computers unless the user proves otherwise. Why should I have to prove to a computer that I'm a human? That seems a little backwards, shouldn't computers assume that their users are human until the user does something that is indicative of a computer?

Electronic Ears

Audio CAPTCHAs aren't foolproof either. In the spring of 2008, there were reports that hackers figured out a way to beat Google's audio CAPTCHA system. To crack an audio CAPTCHA, you have to create a library of sounds representing each character in the CAPTCHA's database. Keep in mind that depending on the distortion, there might be several sounds for the same character. After categorizing each sound, the spammer uses a variation of voice-recognition software to interpret the audio CAPTCHA [source: Networkworld].

Stealing Cycles From Humans:

Now, captchas aren't always a good solution the the Shopping Agent problem. Why? Because if
an online store displays a captcha test before allowing a user to see the prices, then a shopping
agent can simply ask the human behind the computer to solve the captcha test.
This same strategy can be used to defeat captchas in many cases. We call it stealing cycles from
humans. Consider web porn companies, for instance. These companies can still obtain thousands
of free email accounts even if the email provider uses a 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 thousand users

seeing pornographic pictures, and one of them is told: \please solve the following test before we can
show you the next picture". Voila. Solving this problem is still a largely open question.
Notice how stealing cycles from humans works: humans are used in order to expand the compu-
tational abilities of bots. This leads to an interesting digression: in general, how can we use humans
to expand our computational abilities? Currently, there are many things that humans can do but
8
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? Such issues have been partially
considered by the Open Mind project [9], which attempts to make computers \smarter" by using
people in the internet to \teach" computers simple concepts.
Imagine a program that tells you which is the best haircut for you. It seems nearly impossible to
write such a program given the current state of technology. But we can steal cycles from humans!
If you visit http://www.hotornot.com, you'll get a picture of a person (man or woman) and you're
supposed to rate how attractive you think the person is (on a scale from 1 to 10). After you rate the
person you learn what other users have said about them and you also get to rank another person.
This site is very popular: sometimes entire o_ces of graduate students are found rating these
pictures. The \Best Haircut" program should now be obvious: you give the program a picture of
yourself and it posts several pictures of you, all with di_erent haircuts, on http://www.hotornot.com.
Whichever picture is rated the highest is the \best" haircut for you.
Notice that captchas are the ultimate tool for stealing cycles from intelligent people: malicious,
intelligent programmers (writing programs that get thousands of free email accounts, etc.) can be

put to work on hard unsolved AI problems.

Alternative to captcha: saptcha:

SAPTCHA stands for Semi Automatic Public Turing Test to Tell Computers and Humans Apart.

The key concept is same as with CAPTCHA: user is presented with test

question or instructions and must give correct answer to use resource. Main difference is that computer does not try to automatically generate "unique" test questions on each query; only verification of answer is automatic. Instead, unique test question and answer[s] are set by moderator or owner when SAPTCHA is installed, and are changed every time spamming happens.

SAPTCHA is proposed as more accessible alternative to CAPTCHA that may replace CAPTCHA in services such as most blogs and forums. SAPTCHA works as lightweight CAPTCHA.
The concept follows from observation that there are many cases where automated generation of unique test question or image does not add any security - spammer do not need to pass test more than once on same forum or blog. Often, there's no human spammer interacting with website at all [Indeed, every blog or site owner would love to believe that his site is so important that it is spammed personally, in a very weird way whereby rather than just reading the image with eyes, spammer would write visual recognition software, but that's in fact not happening :-)]. In such cases, static question can't be worse at stopping bot than dynamic.
Human generated questions have much broader diversity and are thus harder for computer to answer. Parsing arbitrary sentences is an unsolved problem, *unlike distorted letter recognition which is a solved problem*. It must be also noted that CAPTCHA itself is not really "completely automatic" - human has to write and maintain CAPTCHA software, and update it every time it is broken.

Example questions: User is given instruction like "write [no i'm not a computer!] in this text field" or "write 'i'm human' in reverse" or "write[or copy-paste] web address of this page there" (please don't use too similar things. No default questions and answers. Think up something yourself. Don't try to be clever. It should be not more complex to understand and do than rest of registration, and thus shouldn't decrease website's accessibility(!). It's better if answer is more than 1 character long.)
Bots can try to understand text written by human in normal language (very hard problem in AI) or try to guess (some delay can make it pointless) or try some common test answers if any (then, common test questions and answers will disappear)
Spammer have to manually answer the question to start spamming. This is exactly same problem as with CAPTCHA at registration. Similarly to CAPTCHA at registration, human invervention is necessary to stop spam. - account must be banned and for SAPCHA question must be changed.

In a way, SAPTCHA can be viewed as light weight disposable CAPTCHA test that is cheap to replace when it get compromised.