ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

CAPTCHA As Graphical Passwords Online Poll Verify New Cryptography Techniques Used Hard Problems

K.Raju, Bullarao Domathoti, P.Nageswara Rao

M. Tech Student, Dept. of CSE, SITS, JNTUA University, Tirupati, India

Assistant Professor, Dept. of. CSE, SITS, JNTUA University, Tirupati, India

Assistant Professor, Dept. of. CSE, SITS, JNTUA University, Tirupati, India

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. Many security primitives are based on hard mathematical problems. Using hard AI problems for security is emerging as an exciting new paradigm,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. In this paper, we present a new security primitive based on hard AI problems, namely, a novel family of graphical password systems built on top of Captcha technology, which we call Captcha as graphical passwords (CaRP). CaRP is both a Captcha and a graphical password scheme. CaRP addresses a number of security problems altogether, such as online guessing attacks, relay attacks, and, if combined with dual-view technologies, shoulder-surfing attacks.

KEYWORDS: Local binary pattern; Gaussian filters; images; data set; robustness;

I. INTRODUCTION

You're trying to sign up for a free email service offered by Gmail or Yahoo. Before you can submit your application, you first have to pass a test. It's not a hard test -- in fact, that's the point . For you, the test should be simple and straightforward. But for a computer, the test should be almost impossible to solve. This sort of test is a **CAPTCHA**. They're also known as a type of **HumanInteraction Proof** (**HIP**). You've probably seen CAPTCHA tests on lots of Web sites. The most common form of CAPTCHA is an image of several distorted letters.It's your job to type the correct series of letters into a form. If your letters match the ones in the distorted image, you pass the test.CAPTCHAs are short for **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part. The term "CAPTCHA" was coined in 2000 by Luis Von Ahn, Manuel Blum, Nicholas J. Hopper (all of Carnegie Mellon University, and John Langford (then of IBM). They are challenge-response tests to ensure that the users are indeed human. The purpose of a CAPTCHA is to block form submissions from spam bots – automated scripts that harvest email addresses from publicly available web forms. A common kind of CAPTCHA used on most websites requires the users to enter the string of characters that appear in a distorted form on the screen.

CAPTCHAs are used because of the fact that it is difficult for the computers to extract the text from such a distorted image, whereas it is relatively easy for a human to understand the text hidden behind the distortions. Therefore, the correct response to a CAPTCHA challenge is assumed to come from a human and the user is permitted into the website. Why would anyone need to create a test that can tell humans and computers apart? It's because of people trying to **game** the system -- they want to exploit weaknesses in the computers running the site. While these individuals probably make up a minority of all the people on the Internet, their actions can affect millions of users and Web sites. For example, a free e-mail service might find itself bombarded by account requests from an automated program. That

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

automated program could be part of a larger attempt to send out spam mail to millions of people. The CAPTCHA test helps identify which users are real human beings and which ones are computer programs. Spammers are constantly trying to build algorithms that read the distorted text correctly. So strong CAPTCHAs have to be designed and built so that the efforts of the spammers are thwarted.

A. Motivation:

The proliferation of the publicly available services on the Web is a boon for the community at large. But unfortunately it has invited new and novel abuses. Programs (bots and spiders) are being created to steal services and to conduct fraudulent transactions. Some examples:

- Free online accounts are being registered automatically many times and are being used to distribute stolen or copyrighted material.
- Recommendation systems are vulnerable to artificial inflation or deflation of rankings. For example, EBay, a famous auction website allows users to rate a product. Abusers can easily create bots that could increase or decrease the rating of a specific product, possibly changing people's perception towards the product.
- Spammers register themselves with free email accounts such as those provided by Gmail or Hotmail and use their bots to send unsolicited mails to other users of that email service.
- Online polls are attacked by bots and are susceptible to ballot stuffing.
- This gives unfair mileage to those that benefit from it.

In light of the above listed abuses and much more, a need was felt for a facility that checks users and allows access to services to only human users. It was in this direction that such a tool like CAPTCHA was created.

II. EXPERIMENTAL BACKGROUND

The need for CAPTCHAs rose to keep out the website/search engine abuse by bots. In 1997, AltaVista sought ways to block and discourage the automatic submissions of URLs into their search engines. Andrei Broder, Chief Scientist of AltaVista, and his colleagues developed a filter. Their method was to generate a printed text randomly that only humans could read and not machine readers. Their approach was so effective that in a year, "spam-add-ons'" were reduced by 95% and a patent was issued in 2001. In 2000, **Yahoo**'s popular **Messenger** chat service was hit by bots which pointed advertising links to annoying human users of chat rooms. Yahoo, along with Carnegie Mellon University, developed a CAPTCHA called EZ-GIMPY, which chose a dictionary word randomly and distorted it with a wide variety of image occlusions and asked the user to input the distorted word. In November 1999, s*lashdot.com* released a poll to vote for the best CS College in the US. 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.

*1. CAPTCHAs and the 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. Most, but not all, CAPTCHAs rely on a visual test. Computers lack the sophistication that human beings have when it comes to processing visual data. We can look at an image and pick out patterns more easily than a computer. The human mind sometimes perceives patterns even when none exist, a quirk we call pareidolia. Ever see a shape in the clouds or a face on the moon? That's your brain trying to associate random information into patterns and shapes.But not all CAPTCHAs rely on visual patterns. In fact, it's important to have an alternative to a visual CAPTCHA. Otherwise, the Web site administrator runs the risk of disenfranchising any Web user who has a visual impairment. One alternative to a visual test is an audible one. An audio CAPTCHA usually

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

presents the user with a series of spoken letters or numbers. It's not unusual for the program to distort the speaker's voice, and it's also common for the program to include background noise in the recording. This helps thwart voice recognition programs.

Another option is to create a CAPTCHA that asks the reader to interpret a short passage of text. A contextual CAPTCHA quizzes the reader and tests comprehension skills. While computer programs can pick out key words in text passages, they aren't very good at understanding what those words actually mean.

2. 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 are 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?

Such questions are very easy for a human user to solve, but it’s very difficult to program a computer to solve them. These are also friendly to people with visual disability – such as those with colour blindness. Other text CAPTCHAs involves text distortions and the user is asked to identify the
text hidden. The various implementations are:

3. 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 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.

Fig 2.1 *Gimpy CAPTCHA*

4. 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

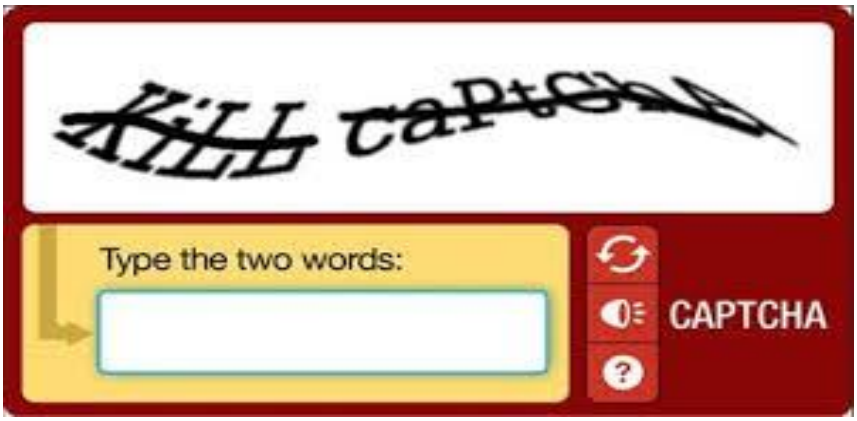

Fig 2.2 *Yahoo's Ez – Gimpy CAPTCHA*

5. *Baffle Text*:

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 ourses

Fig 2.3 *Baffle Text examples*

6. *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.

Fig 2.4 *MSN Passport CAPTCHA*

7. *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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

8. PIX:

PIX 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.

9. 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.

10. reCAPTCHA and book digitization:

To counter various drawbacks of the existing implementations, researchers at CMU developed a redesigned CAPTCHA aptly called the reCAPTCHA. 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.

11. Improve Artificial Intelligence (AI) technology:

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. As people find new ways to get around CAPTCHA, computer scientists like von Ahn develop CAPTCHAs that address other challenges in the field of AI. A step backward for CAPTCHA is still a step forward for AI – "every defeat is also a victory".

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

III. RELATED EXPERIMENTAL WORK

Things to know:

The first step to create a CAPTCHA is to look at different ways humans and machines process information. Machines follow sets of instructions. If something falls outside the realm of those instructions, the machines aren't able to compensate. A CAPTCHA designer has to take this into account when creating a test. For example, it's easy to build a program that looks at **metadata** – the information on the Web that's invisible to humans but machines can read. If you create a visual CAPTCHA and the images' metadata includes the solution, your CAPTCHA will be broken in no time. Similarly, it's unwise to build a CAPTCHA that doesn't distort letters and numbers in some way. An undistorted series of characters isn't very secure. Many computer programs can scan an image and recognize simple shapes like letters and numbers.

One way to create a CAPTCHA is to pre-determine the images and solutions it will use. This approach requires a database that includes all the CAPTCHA solutions, which can compromise the reliability of the test. According to Microsoft Research experts Kumar Chellapilla and Patrice Simard, humans should have an 80 percent success rate at solving any particular CAPTCHA, but machines should only have a 0.01 percent success rate [source: Chellapilla and Simard]. If a spammer managed tofind a list of all CAPTCHA solutions, he or she could create an application that bombards the CAPTCHA with every possible answer in a **brute-force** attack. The database would need more than 10,000 possible CAPTCHAs to meet the qualifications of a good CAPTCHA.

ClickAnimal

Captcha Zoo [32] is a Captcha scheme which uses 3D models of horse and dog to generate 2D animals with different textures, colors, lightings and poses, and arranges them on a cluttered background. A user clicks all the horses in a challenge image to pass the test. Fig. 3 shows a sample challenge wherein all the horses are circled red.

ClickAnimal is a recognition-based CaRP scheme built on top of Captcha Zoo [32], with an alphabet of similar animals such as dog, horse, pig, etc. Its password is a sequence of animal names such as ρ = "Turkey, Cat, Horse, Dog,…." For each animal, one or more 3D models are built. The Captcha generation process is applied to generate ClickAnimal images: 3D models are used to generate 2D animals by applying different views, textures, colors, lightning effects, and optionally distortions. The resulting 2D animals are then arranged on a cluttered background such as grassland. Some animals may be occluded by other animals in the image, but their core parts are not occluded in order for humans to identify each of them. Fig. 4 shows a ClickAnimal image with an alphabet of 10 animals. Note that different views applied in mapping 3D models to 2D animals, together with occlusion in the following step, produce many different shapes for the same animal's instantiations in the generated images. Combined with the additional anti-recognition mechanisms applied in the mapping step, these make it hard for computers to recognize animals in the generated image, yet humans can easily identify different instantiations of animals.

IV. IMPLEMENTATION

Embeddable CAPTCHAs: The easiest implementation of a CAPTCHA to a Website would be to insert a few lines of CAPTCHA code into the Website's HTML code, from an open source CAPTCHA builder, which will provide the authentication services remotely. Most such services are free. Popular among them is the service provided by www.captcha.net's reCAPTCHA project. *Custom CAPTCHAs:* These are less popular because of the extra work needed to create a secure implementation. Anyway, these are popular among researchers who verify existing CAPTCHAs and suggest alternative implementations.

There are advantages in building custom CAPTCHAs:

- ✓ A custom CAPTCHA can fit exactly into the design and theme of your site. It will not look like some alien element that does not belong there.
- ✓ We want to take away the perception of a CAPTCHA as an annoyance, and make it convenient for the user.
- ✓ Because a custom CAPTCHA, unlike the major CAPTCHA mechanisms, obscure you as a target for spammers. Spammers have little interest in cracking a niche implementation.
- ✓ Because we want to learn how they work, so it is best to build one ourselves.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

CAPTCHA Logic:

- ✓ The CAPTCHA image (or question) is generated. There are different ways to do this. The classic approach is to generate some random text, apply some random effects to it and convert it into an image.
- ✓ Step 2 is not really sequential. During step 1, the original text (prealtered) is persisted somewhere, as this is the correct answer to the question. There are different ways to persist the answer, as a server-side session variable, cookie, file, or database entry.
- ✓ The generated CAPTCHA is presented to the user, who is prompted to answer it.
- ✓ The back-end script checks the answer supplied by the user by comparing it with the persisted (correct) answer. If the value is empty or incorrect, we go back to step 1: *a new CAPTCHA is generated.* Users should never get a second shot at answering the same CAPTCHA.
- ✓ If the answer supplied by the user is correct, the form post is successful and processing can continue. If applicable, the generated CAPTCHA image is deleted.

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 disability access rules in most countries. 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.

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. Beating spammers is not a hard science; you need to have multiple layers of defense. What you see above is in fact the most successful strategy against spammers: *obscurity*. Simply because we have a *CUSTOM CAPTCHA*, makes us less of a target. Spammers go for mass targets, as their success rate is typically extremely low. Even if we would have a single animal image with just two answers, and the answers would be in the same order each time, we would drastically reduce spam submissions.

An example of this effect is the site www.codinghorror.com/blog; it has a custom CAPTCHA check that simply lets you enter the word "Orange" to post a comment. Orange. Each time. Nothing is random, and there is only one answer. Still, it has drastically reduced the spam on that blog, and it's a big blog. Obscurity works. Not for security, but against spammers. 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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

Breaking CAPTCHAs

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.

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.

- *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 in Fig 5.1, 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.
- *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.
- *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.

V. **CONCLUSIONS**

We believe that the fields of cryptography and artificial intelligence have much to contribute to one another. Captchas represent a small example of these possible symbiosis reductions, as they are used in cryptography, can be extremely useful for the progress of algorithmic development, they are crucial to preventing bot attacks. We encourage security researchers to create captchas based on different AI problems, hopefully they will become more user-friendly to people with disabilities (visual/mental). CAPTCHA's are mainly produced from Asynchronous Java-script And XML **(**AJAX) & using a bit of Hypertext Preprocessor (PHP) technology, various algorithms are present.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

REFERENCES

[1]. Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford. The CAPTCHA Web Page: http://www.captcha.net. 2000.
[2].Luis von Ahn, Manuel Blum and John Langford. Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI. To appear in Communications of the ACM.
[3]. Mihir Bellare, Russell Impagliazzo and Moni Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In 38th IEEE Symposium on Foundations of Computer Science (FOCS' 97), pages 374-383. IEEE Computer Society, 1997.
[4]. Mikhail M. Bongard. Pattern Recognition. Spartan Books, Rochelle Park NJ, 1970.
[5]. A. L. Coates, H. S. Baird, and R. J. Fateman. Pessimal Print: A Reverse Turing Test. In Proceedings of the International Conference on Document Analysis and Recognition (ICDAR' 01), pages 1154-1159. Seattle WA, 2001.
[6]. Scott Craver. On Public-key Steganography in the Presence of an Active Warden. In Proceedings of the Second International Information Hiding Workshop, pages 355-368. Springer, 1998.
[7]. Nicholas J. Hopper, John Langford and Luis von Ahn. Provably Secure Steganography. In Advances in Cryptology, CRYPTO' 02, volume 2442 of Lecture Notes in Computer Science, pages 77-92. Santa Barbara, CA, 2002.
[8]. M. D. Lillibridge, M. Abadi, K. Bharat, and A. Broder. Method for selectively restricting access to computer systems.USPatent 6,195,698. Applied April 1998 and Approved February 2001.
[9]. Greg Mori and Jitendra Malik. Breaking a Visual CAPTCHA. Unpublished Manuscript, 2002. Available electronically: *http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf*.
[10]. Moni Naor. Veri_cation of a human in the loop or Identi_cation via the Turing Test. Unpublished Manuscript, 1997.Available electronically: *http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps*.