1. INTRODUCTION

A captcha is a program that can generate and grade tests that: (A) most humans can pass, but (B) current computer programs can't pass. Such a program can be used to differentiate humans from computers and has many applications for practical security, including (but not limited to):

1.1 **Online Polls.** In November 1999, slashdot.com released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting “bots". MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote.

1.2 **Free Email Services**. Several companies (Yahoo!, Microsoft, etc.) offer free email services, most of which suffer from a specifc type of attack: “bots" that sign up for thousands of email accounts every minute. This situation can be improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a captcha of our design to prevent bots from registering for accounts. Their captcha asks users to read a distorted word such as the one shown below (current computer programs are not as good as humans at reading distorted text).

www.vtucs.com

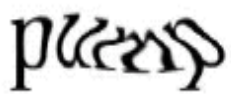

Fig. 1. The Yahoo! CAPTCHA.

1.3 **Search Engine Bots**. Some web sites don't want to be indexed by search engines. There is an html tag to prevent search engine bots from reading web pages, but the tag doesn't guarantee that bots won't read the pages; it only serves to say ”no bots, please”. Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly guarantee that bots won't enter a web site, captchas are needed.

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

1.5 **Preventing Dictionary Attacks**. Pinkas and Sander have suggested using captchas to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords. The goals of this paper are to lay a solid theoretical foundation for captchas, to introduce the concept to the cryptography community, and to present several novel constructions.

www.vtucs.com

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. We deliberately avoid using the term Reverse Turing Test (or even worse, RTT) because it can be misleading Reverse Turing Test has been used to refer to a form of the Turing Test in which both players pretend to be a computer.

www.vtucs.com

2. APPLICATIONS

Although the goal of the original Turing Test was to serve as a measure of progress for artificial intelligence—a computer would be said to be intelligent if it passed the Turing Test—making the judge be a computer allows CAPTCHAs to be useful for other practical applications. In November 1999, for example, the Web site slashdot.com released an online poll asking which was the best graduate school in computer science a dangerous question to ask over the Web. As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times: CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting "bots." MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote. Another application involves free email services. Several companies offer free email services that have suffered from a specific type of attack: "bots" that signed up for thousands of email accounts every minute. This situation has been improved by requiring users to prove they are human before they can get a free email account. Yahoo, for instance, uses a CAPTCHA of our design to prevent bots from registering for accounts.

Some Web sites don't want to be indexed by search engines. There is a HTML tag to prevent search engine bots from reading Web pages, but the tag doesn't guarantee that bots won't read the pages; it only serves to say "no bots, please." Search engine bots, since they usually belong to large companies, respect Web pages that don't want to allow them in. However, in order to truly guarantee bots won't enter a Web site, CAPTCHAs are needed. CAPTCHAs also offer a plausible solution against email worms and spam: only accept an email message if you know there is a human behind the other computer. A few companies, such as **www.spamarrest**. com are already marketing this idea. Pinkas and Sander have also suggested using CAPTCHAs to prevent dictionary attacks in password systems.

www.vtucs.com

The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

www.vtucs.com

3. EXAMPLES OF CAPTCHAS

CAPTCHAs further differ from the original Turing Test in that they can be based on a variety of sensory abilities. The original Turing Test was conversational —the judge was only allowed to ask questions over a text terminal. In the case of a CAPTCHA, the computer judge can ask any question that can be transmitted over a computer network.

3.1 GIMPY and OCR-based CAPTCHAs

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

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

GIMPY then presents a test to its user, which consists of the distorted image and the directions: "type three words appearing in the image." Given the types of distortions that GIMPY uses, most humans can read three words from the distorted image, but current computer programs can't. The majority of CAPTCHAs

used on the Web today are similar to GIMPY in that they rely on the difficulty of optical character recognition (the difficulty of reading distorted text).

3.2 Bongo.

Another example of a CAPTCHA is the program we call BONGO . BONGO is named after **M.M. Bongard**, who published a book of pattern recognition problems in the 1970's. BONGO asks the user to solve a visual pattern recognition problem. It displays two series of blocks, the left and the right. The blocks in the left series differ from those in the right, and the user must find the characteristic that sets them apart. A possible left and right series is shown in Figure 2. After seeing the two series of blocks, the user is presented with a single block and is asked to determine whether this block belongs to the left series or to the right. The user passes the test if he or she correctly determines the side to which the block belongs.

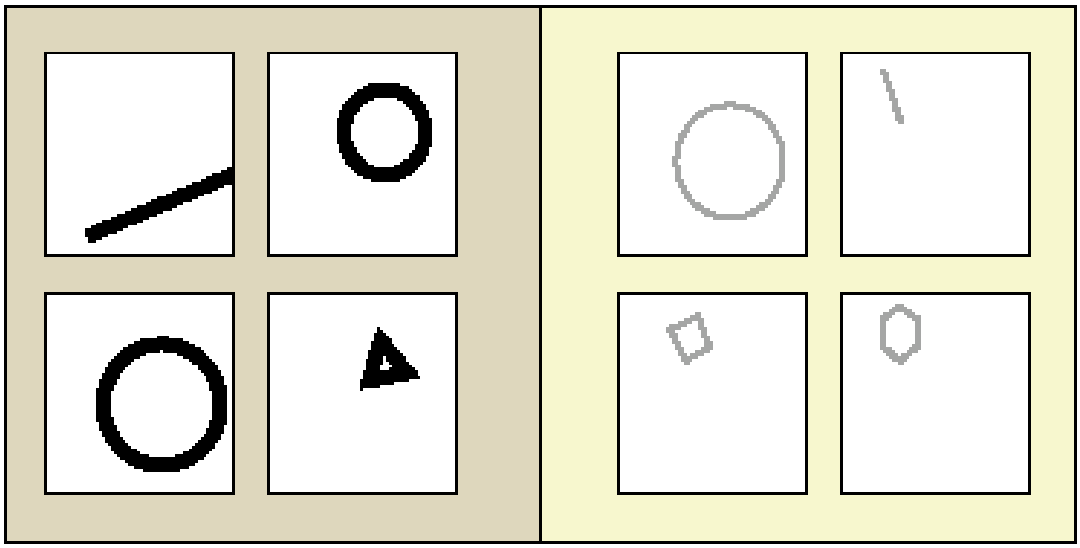

Figure 2. Everything on the left is drawn with thick lines, while everything on the right is drawn with thin lines.

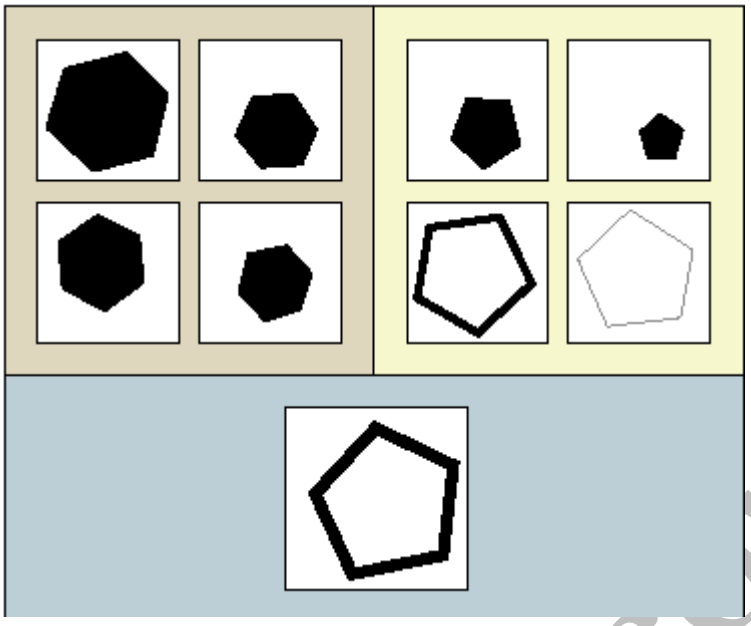

Figure 3. To which side does the block on the bottom belong?

3.3 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.

www.vtucs.com

3.4. Sound-based 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 .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.

www.vtucs.com

4. LAZY CRYPTOGRAPHERS DOING AI

Modern cryptography has shown that open or intractable problems in number theory can be useful: an adversary cannot act maliciously unless he can solve an open problem (like factor a very large number). Similarly, CAPTCHAs show that open problems in AI can be useful: adversaries cannot vote thousands of times in online polls or obtain millions of free email accounts unless they can solve an open problem in AI. In the case of ordinary cryptography, it is assumed (for example) that the adversary cannot factor 1024-bit integers in any reasonable amount of time. In our case, we assume the adversary cannot solve an artificial intelligence problem with higher accuracy than what's currently known to the AI community [1, 2, 5, 6, 8]. This approach has the beneficial side effect of inducing security researchers, as well as otherwise malicious programmers, to advance the field of AI (much like computational number theory has been advanced since the advent of modern cryptography).

This is how lazy cryptographers do AI. A good example of this process is the recent progress in reading distorted text images motivated by the CAPTCHA in use at Yahoo. In response to the challenge provided by this test, Malik and Mori have developed a program that can pass the test with over 80% accuracy. Malik and Mori's algorithm represents significant progress in the general area of textrecognition, and it is extremely encouraging to see such progress. A CAPTCHA implies a win-win situation: either the CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and a useful AI problem is solved.

www.vtucs.com

5. RELATED WORK

The first mention of ideas related to “Automated Turing Tests" seems to appear in an unpublished manuscript by Moni Naor . This excellent manuscript contains some of the crucial notions and intuitions, but gives no proposal for an Automated Turing Test, nor a formal definition. The first practical example of an Automated Turing Test was the system developed by Altavista to prevent ”bots" from automatically registering web pages. Their system was based on the difficulty of reading slightly distorted characters and worked well in practice, but was only meant to defeat off the-shelf **Optical Character Recognition (OCR) technology**.

In 2000, introduced the notion of a captcha as well as several practical proposals for Automated Turing Tests. This paper is the first to conduct a rigorous investigation of Automated Turing Tests and to address the issue of proving that it is difficult to write a computer program that can pass the tests. This, in turn, leads to a discussion of using AI problems for security purposes, which has never appeared in the literature. We also introduce the first Automated Turing Tests not based on the difficulty of Optical Character Recognition. A related general interest paper has been accepted by Communications of the ACM. That paper reports on our work, without formalizing the notions or providing security guarantees.

www.vtucs.com

6. DEFINITIONS AND NOTATION

Let C be a probability distribution. We use [C] to denote the support of C. If P(**.**) is a probabilistic program, we will denote P_r(**.**) by the deterministic program that results when P uses random coins r.

Let (P;V) be a pair of probabilistic interacting programs. We denote the output of V after the interaction between P and V with random coins u1 and u2, assuming this interaction terminates, by ($P_{U1,}V_{U2}$) (the subscripts are omitted in case the programs are deterministic). A program V is called a test if for all P and u1,u2, the interaction between P_{U1} and V_{U2} terminates and (P_{U1},V_{U2}) $\in$ {*accept; reject*}. We call V the *verifier or tester* and any P which interacts with V the *prover*.

5.1 Definition 1.

Define the success of an entity A over a test V by

$$\mathbf{Succ}_A^V = \Pr_{r,r'}[\langle A_r, V_{r'} \rangle = accept].$$

We assume that A can have precise knowledge of how V works; the only piece of information that A can't know is r0, the internal randomness of V.

CAPTCHA

Intuitively, a captcha is a test V over which most humans have success close to 1, and for which it is hard to write a computer program that has high success over V. We will say that it is hard to write a computer program that has high success over V if any program that has high success over V can be used to solve a hard AI problem.

www.tucs.com

5.2 Definition 2.

A test V is said to be (**α, β**) human executable if at least an **α** portion of the human population has success greater than **β** over V .

Notice that a statement of the form "V is (**α, β**) -human executable" can only be proven empirically. Also, the success of different groups of humans might depend on their origin language or sensory disabilities: color-blind individuals, for instance, might have low success on tests that require the differentiation of colors.

5.3 Definition 3.

An AI problem is a triple **ρ = (S;D; f),** where S is a set of problem instances, D is a probability distribution over the problem set S, and

$$f : S \rightarrow \{0,1\}^*$$

answers the instances. Let _ 2 (0; 1]. We require that for an α > 0 fraction of the humans H, $\Pr_{x \leftarrow D}[H(x) = f(x)] > \delta.$

5.4 Definition 4.

An AI problem **ρ** is said to be (**δ,τ**)-solved if there exists a program A, running in time at most **τ** on any input from S, such that

$$\Pr_{x \leftarrow D, r}[A_r(x) = f(x)] \geq \delta \ .$$

(A is said to be a (**δ,τ**) solution to **ρ**.) **ρ** is said to be a (**δ,τ**)-hard AI problem if no current program is a (**δ,τ**) solution to **ρ**, and the AI community agrees it is hard to find such a solution.

www.vtucs.com

5.5 Definition 5.

A (**α, β, η**)-captcha is a test V that is (**α,β**)-human executable, and which has the following property: There exists a (**δ,τ**)-hard AI problem **ρ** and a program A, such that if B has success greater than **η** over V then A^B is a (**δ,τ**) solution to ρ. (Here A^B is defined to take into account B's running time too.) We stress that V should be a program whose code is publicly available.

www.vtucs.com

6. REMARKS

- The definition of an AI problem as a triple (**S,D,f**) should not be inspected with a philosophical eye. We are not trying to capture all the problems that fall under the umbrella of Artifcial Intelligence. We want the definition to be easy to understand, we want some AI problems to be captured by it, and we want the AI community to agree that these are indeed hard AI problems. More complex definitions can be substituted for Definition 3 and the rest of the paper remains unaffected.

- A crucial characteristic of an AI problem is that a certain fraction of the human population be able to solve it. Notice that we don't impose a limit on how long it would take humans to solve the problem. All that we require is that some humans be able to solve it (even if we have to assume they will live hundreds of years to do so). The case is not the same for captchas. Although our definition says nothing about how long it should take a human to solve a captcha, it is preferable for humans to be able to solve captchas in a very short time. captchas which take a long time for humans to solve are probably useless for all practical purposes.

www.vtucs.com

7. AI PROBLEMS AS SECURITY PRIMITIVES

Notice that we define hard in terms of the consensus of a community: an AI problem is said to be hard if the people working on it agree that it's hard. This notion should not be surprising to cryptographers: the security of most modern cryptosystems is based on assumptions agreed upon by the community (e.g., we assume that 1024-bit integers can't be factored). The concept of a hard AI problem as a foundational assumption, of course, is more questionable than P 6= NP, since many people in the AI community agree that all hard AI problems are eventually going to be solved. However, hard AI problems may be a more reasonable assumption than the hardness of factoring, given the possibility of constructing a quantum computer.

Moreover, even if factoring is shown to be hard in an asymptotic sense, picking a concrete value for the security parameter usually means making an assumption about current factoring algorithms: we only assume that current factoring algorithms that run in current computers can't factor 1024-bit integers. In the same way that AI researchers believe that all AI problems will be solved eventually, we believe that at some point we will have the computational power and algorithmic ability to factor 1024-bit integers. (Shamir and Tromer , for instance, have proposed a machine that could factor 1024-bit integers; the machine would cost about ten million dollars in materials.) An important difference between popular cryptographic primitives and AI problems is the notion of a security parameter. If we believe that an adversary can factor 1024-bit integers, we can use 2048-bit integers instead. No such concept exists in hard AI problems.

AI problems, as we have defined them, do not dealwith asymptotics. However, as long as there is a small gap between human and computer ability with respect to some problem, this problem can potentially be used as a primitive for security: rather than asking the prover to solve the problem once, we can ask it to solve the problem twice. If the prover gets good at solving the problem twice, we can ask it to solve the problem three times, etc. There is an additional factor that

www.vtucs.com

simplifies the use of hard AI problems as security primitives. Most applications of captchas require the tests to be answered within a short time after they are presented. If a new program solves the hard AI problems that are currently used, then a different set of problems can be used, and the new program cannot affect the security of applications that were run before it was developed. Compare this to encryption schemes: in many applications the information that is encrypted must remain confidential for years, and therefore the underlying problem must be hard against programs that run for a long time, and against programs that will be developed in the future.

We also note that not all hard AI problems can be used to construct a captcha. In order for an AI problem to be useful for security purposes, there needs to be an automated way to generate problem instances along with their solution. The case is similar for computational problems: not all hard computational problems yield cryptographic primitives.

www.vtucs.com

8. WHO KNOWS WHAT?

Our definitions imply that an adversary attempting to write a program that has high success over a captcha knows exactly how the captcha works. The only piece of information that is hidden from the adversary is a small amount of randomness that the verifier uses in each interaction. This choice greatly affects the nature of our definitions and makes the problem of creating captchas more challenging.

Imagine an Automated Turing Test that owns a large secret book written in English and to test an entity A it either picks a paragraph from its secret book or generates a paragraph using the best known text-generation algorithm, and then asks A whether the paragraph makes sense (the best text-generation algorithms cannot produce an entire paragraph that would make sense to a human being). Such an Automated Turing Test might be able to distinguish humans from computers (it is usually the case that the best text-generation algorithms and the best algorithms that try to determine whether something makes sense are tightly related). However, this test cannot be a captcha: an adversary with knowledge of the secret book could achieve high success against this test without advancing the algorithmic state of the art. We do not allow captchas to base their security in the secrecy of a database or a piece of code.

www.vtucs.com

9. OTHER AI PROBLEM DOMAINS

The problems defined in this paper are both of a similar character, and deal with the advantage of humans in sensory processing. It is an open question whether captchas in other areas can be constructed. The construction of a captcha based on a text domain such as text understanding or generation is an important goal for the project (as captchas based on sensory abilities can't be used on sensory-impaired human beings). As mentioned earlier, the main obstacle to designing these tests seems to be the similar levels of program ability in text generation and understanding. Logic problems have also been suggested as a basis for captchas and these present similar difficulties, as generation seems to be difficult. One possible source of logic problems are those proposed by **Bongard** in the 70s; indeed presents a test based on this problem set. However, recent progress in AI has also yielded programs which solve these problems with very high success probability, exceeding that of humans.

www.vtucs.com

10. CONCLUSION

It's believed that the fields of cryptography and artificial intelligence have much to contribute to one another. Captchas represent a small example of this possible symbiosis. Reductions, as they are used in cryptography, can be extremely useful for the progress of algorithmic development. So, security researchers to create captchas based on different AI problems must be encouraged.

www.vtucs.com

11. REFERENCES

[1] http://www.captcha.net. 2000.

[2] http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf.

[3] http://www.cryptome.org/twirl.ps.gz

[4] http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps.

[5] 38th IEEE Symposium on Foundations of Computer Science (FOCS' 2001),

[6] Greg Mori and Jitendra Malik. Breaking a Visual CAPTCHA Unpublished Manuscript, 2002. Available electronically:

www.vtucs.com

CONTENTS

1.	INTRODUCTION		01
2.	APPLICATION	04	
3.	EXAMPLES OF CAPTCHAS	06	
	3.1. GIMPY AND OCR-BASED CAPTCHAS	06	
	3.2. BONGO.		07
	3.3. PIX.		08
	3.4. SOUND-BASED CAPTCHAS.	09	
4.	LAZY CRYPTOGRAPHERS DOING AI	10	
5.	DEFINITIONS AND NOTATION		12
6.	REMARKS		15
7.	AI PROBLEMS AS SECURITY PRIMITIVES	16	
8.	WHO KNOWS WHAT?	18	
9.	OTHER AI PROBLEM DOMAINS	19	
10.	CONCLUSION	20	
11.	REFERENCES		21

www.vtucs.com

ABSTRACT

We introduce **CAPTCHA**, an automated test that humans can pass, but current computer programs can't pass: any program that has high success over a captcha can be used to solve an unsolved Artifcial Intelligence **(AI)** problem. We provide several novel constructions of captcha**s**. Since captchas have many applications in practical security, our approach introduces a new class of hard problems that can be exploited for security purposes. Much like research in cryptography has had a positive impact on algorithms for factoring and discrete log, we hope that the use of hard **AI** problems for security purposes allows us to advance the field of Artifcial Intelligence. We introduce two families of **AI** problems that can be used to construct captchas and we show that solutions to such problems can be used for steganographic communication.. Captchas based on these **AI** problem families, then, imply a win-win situation: either the problems remain unsolved and there is alway to differentiate humans from computers, or the problems are solved and there is a way to communicate covertly on some channels.

www.vtucs.com