Remotely telling humans and computers apart: an unsolved problem

Carlos Javier Hernandez-Castro, Arturo Ribagorda

Security Group, Department of Computer Science, Carlos III University, 28911 Leganes, Madrid, Spain
{chernand, arturo}@inf.uc3m.es

Abstract. The ability to tell humans and computers apart is imperative to protect many services from misuse and abuse. For this purpose, tests called CAPTCHAs[1] or HIPs[2] have been designed and put into production. Recent history shows that most (if not all) can be broken given enough time and commercial interest: CAPTCHA design seems to be a much more difficult problem than previously thought. The assumption that difficult-AI problems can be easily converted into valid CAPTCHAs is misleading. There are also some extrinsic problems that do not help, especially the big number of in-house designs that are put into production without any prior public critique. In this paper we present a state-of-the-art survey of current HIPs, including proposals that are now into production. We classify them regarding their basic design ideas. We discuss current attacks as well as future attack paths, and we also present common errors in design, and how many implementation flaws can transform a not necessarily bad idea into a weak CAPTCHA. We present examples of these flaws, using specific well-known CAPTCHAs. In a more theoretical way, we discuss the threat model: confronted risks and countermeasures. Finally, we introduce and discuss some desirable properties that new HIPs should have, concluding with some proposals for future work, including methodologies for design, implementation and security assessment.

Key words: HIP, CAPTCHA, design, implementation, flaw, methodologies, security assessment

1 Introduction

Herein we discuss some aspects of current CAPTCHA design and implementation. We are interested in both CAPTCHAs in use at major web-sites, and other commercial and/or academic proposals, some of which are still not broadly deployed, but present interesting aspects. After that, we will show the publicly-known state-of-the-art in attacks. We will introduce some examples to discuss common pitfalls in their design and implementation. We will also discuss the different threat models they assume. Then we will discuss CAPTCHAs in a theoretical way. We will try to address why they have failed so frequently - put it other way, why they have remained unbroken for so brief time. We will discuss the reasons that make their design a much harder

[1] Completely Automated Public Turing test to tell Computers and Humans Apart
[2] Human Interactive Proof

problem than previously thought. Also, there are more ambitious proposals that have tried to cover different types of attacks, including human farming attacks. We will conclude recommending some good properties that new designs should meet; properties that can be used as a starting point for a design methodology and for their security assessment.

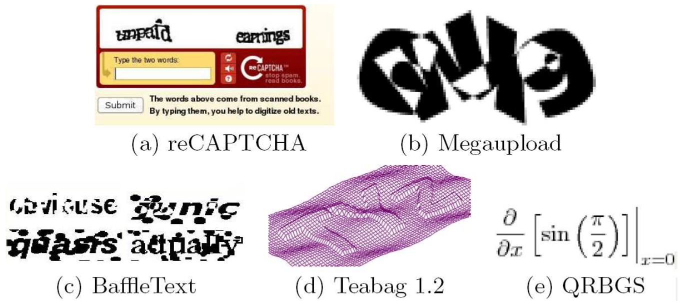
1.1 Motivation: why to study and break CAPTCHAs

Today we do not fully understand if the problems that form the basis of the current CAPTCHAs designs are hard enough. Even worse, we do not have a way to tell if the precise method employed by a particular design uses the full strength of the underlying, supposedly hard, AI problem. We do not have a clear methodology to check if a design and its implementation are flaw-free. It is then interesting to try to break current CAPTCHAs and find pitfalls in their design to make the state-of-the-art advance and get to a point where better known and tested assumptions give rise to more secure CAPTCHAs.

1.2 CAPTCHAs' Evolution

CAPTCHAs are somewhat recent. Moni Naor was the first to propose theoretical methods of remotely telling apart computers from humans [19]. The first known use of a CAPTCHA on-line is credited to the AltaVista web search-engine in 1997 [22]. Andrei Broder developed a filter, randomly generating an image of printed text, and transforming it so that machine character recognition systems (OCRs) could not read it, but humans still could. In 2000, Udi Manber of Yahoo! described their "chat room problem" to researchers at Carnegie-Mellon University: 'bots' (automatic computer scripts) were joining on-line chat rooms, and pointing their users to advertising sites. Professors Manuel Blum, Luis A. von Ahn, and John Langford, from Carnegie-Mellon University, described some desirable properties that a test to exclude bots should have. These properties are mostly theoretical, and do not deal with specific design or implementation issues (i.e., nothing is said about whether challenge generation is better in the client or server side). The Carnegie-Mellon University team developed a 'hard' GIMPY, which picked English words at random, and rendered them as images of printed text, under a wide variety of shape deformations and image distortions[17], with an easier version put to work at Yahoo!.

Text-based CAPTCHAs In the following years, OCR/text-based CAPTCHAs became increasingly popular. For awhile, attackers preferred those sites not protected by CAPTCHAs as still there were many services that did not use them, so advertising/spam/phishing/voting abuse etc. were still possible by avoiding CAPTCHAs. Even after graphic distortion and degradation, some approaches have been able to "read" them automatically around 92% of the time [18], specially when it is possible to divide the rendered text image into its constituent letters or characters, a procedure called "segmentation". After some more attacks on OCR/text-based CAPTCHAs were made public, the outcome was not the creation of new paradigms, but the redesign of OCR/text-based proposals in a way to try to prevent those attacks [25]. Newer attacks appeared [32].


unpaid
earnings
Type the two words:
reCAPTCHA™
stop spam.
read books.
Submit
The words above come from scanned books.
By typing them, you help to digitize old texts.

$$\frac{\partial}{\partial x}\left[\sin\left(\frac{\pi}{2}\right)\right]\Big|_{x=0}$$

(a) reCAPTCHA (b) Megaupload

(c) BaffleText (d) Teabag 1.2 (e) QRBGS

Fig. 1. Some current OCR/text based CAPTCHAs.

Although now the commercial and academic proposals offer a wide variety of types to choose from, most deployed CAPTCHA belong to the OCR/text-based type of challenge. Among them, there are some interesting ones, as reCAPTCHA [13], which is an interesting mix of a normal OCR/text and database based CAPTCHA: it is based on extracts of texts that had not been correctly recognized by OCR software. That reason is why the test presents two words: one comes from the failed-to-OCR set, the other is one which reading is known: either an automatically generated and distorted one, or a failed-to-OCR one already read by humans. In this sense, reCAPTCHA is an *incremental CAPTCHA*. reCAPTCHA also includes an audio version, further explained in the audio section of this paper. The current Megaupload [3] is interesting because uses a strong overlapping model (trying to make segmentation much harder). This approach is sound, as computers are even better than humans at recognizing isolated characters [33], so the most difficult problem for a machine is segmentation. BaffleText uses a somewhat related approach [28], using "occlusion or interference by random shapes", against which, they state, "image restoration cannot replace missing parts without prior knowledge of the occluding shapes", humans being "remarkably good at recognizing an entire shape or picture in spite of incomplete, sparse, or fragmentary information". Another atypical text-based scheme is the one proposed to protect the "Quantum Random Bit Generator Service" [5]. They ask to solve or compute a not-so-simple math expression to prove that you are human (and good enough in Math). To prevent automatic OCR reading, they do not rely on transformations, but on the low quality of the depicted text, using small fonts and antialiasing.

Another interesting and atypical OCR/text CAPTCHA is Teabag [6], currently in production, which depicts the text in 3D, also transforming in 3D space. Other proposals based on the OCR/text reading challenge are the commercial Captcha2[7], providing an interesting mix of an OCR/text-based and an image-based: the user

[3] http://www.megaupload.com/. Their previous CAPTCHA has reportedly been broken during January, 2009 [4]. http://userscripts.org/scripts/show/38736

[5] http://random.irb.hr/, Center for Informatics and Computing of the Rud̄er Boškovíc Institute (Zagreb, Croatia)

[6] OCR Research Team, Kharkov (Ukraine), at http://ocr-research.org.ua/ and http://barafranca.com/game-register.php

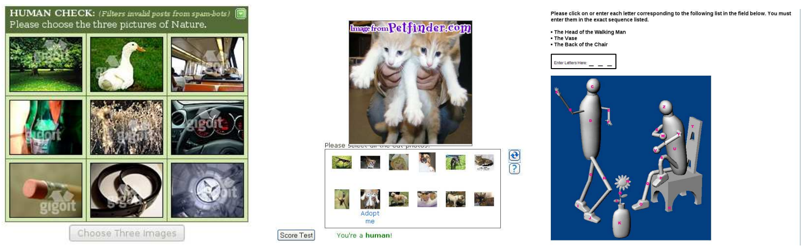
[7] http://captcha2.com/

must read the character presented after the message "click on the letter", and select the same letter by clicking close to it in the box displayed below the caption. There are also some OCR/text CAPTCHAs that rely on a broad range of parameters for text depicting, thus looking as if they were composite versions of simple ones, as JCaptcha[8], deployed in many sites[9], Securimage[10], an open-source free PHP CAPTCHA script, or BotDetect[11], a commercial proposal. There is also a proposal [23, 10] that tries to mimic the human handwriting in a synthetic form for creating challenges that are more difficult for OCRs.

Picture-based CAPTCHAs Many text-based CAPTCHAs were recently broken [18, 30, 31, 35, 36, 32, 33], so there is increasing concern about their overall strength and their accessibility for humans (as they became harder, for countering these attacks). Possibly influenced by the idea that the general vision problem seems to be a harder problem than character recognition, more designs lately focus on using pictures instead. Those CAPTCHAs do not really rely on a "general vision recognition problem", but in a downsized version of it, typically consisting in labeling images, finding their center, etc. Chew and Tygar[20] were the first to use a set of labeled images to produce CAPTCHA challenges, using Google Image Search. Google relates a picture to its description, and to its surroundings in the document that contains it. Ahn and Dabbish [21] proposed a new way to label images, embedding the task as a game, thus creating the PIX CAPTCHA database. HotCaptcha.com proposed a new way to use a large-scale human labeled database provided by the HotOrNot.com web-site [12], a site that invites users to post photos of themselves and rate others' in a sex appeal numerical scale. Oli Warner came with the idea of using photos of kittens to tell computers and humans apart [41]. KittenAuth features nine pictures of cute little animals, only three of which are feline. The database of pictures is small enough (< 100) to manually classify them, and this limitation seems troublesome even if we apply new methods involving image distortion. Another proposal, known as the HumanAuth CAPTCHA (fig. 2(a)), asks the user to distinguish between pictures depicting Nature or an artificial thing (like a clock, or a vase, for instance). In addition, the CAPTCHA has also an alternative version for the visually challenged, as they can switch to text-based descriptions of the pictures.

The main problem with HumanAuth is the picture database size, which is too small, even though the authors include an algorithm to mix it with a logo for watermarking and avoiding hash-function-based image classification (that will not affect text descriptions). ASIRRA [16] (fig. 2(b)) uses a similar approach, but using a database of more that 3 million photos from Petfinder.com. ASIRRA asks for identifying images under two categories, cats and dogs. VidoopSECURE [13] is a company that provides security solutions, and gives access free of charge to their VidoopCAPTCHA, composed of 12 images, each one labeled with a letter, asking the user to enter the

[8] http://jcaptcha.sourceforge.net/
[9] Including a major airline web-site, (http://www.iberia.com
[10] http://www.phpcaptcha.org/captcha-gallery/
[11] http://captcha.biz/captcha.html
[12] No longer active, as of January 2009.
[13] http://drupal.org/project/vidoopcaptcha


HUMAN CHECK: *(Filters invalid posts from spam-bots)*
Please choose the three pictures of Nature.

Please click on or enter each letter corresponding to the following list in the field below. You must enter them in the exact sequence listed.

- The Head of the Walking Man
- The Vase
- The Back of the Chair

Enter Letters Here: _ _ _

(a) HumanAuth (b) ASIRRA (c) 3D CAPTCHA

Fig. 2. Some current picture based CAPTCHAs.

letters in those images that refer to three given categories. IMAGINATION [26] is a CAPTCHA divided in two stages: the system shows the user an image composed of 8 images tiled and asks the user to click ”near” the geometric center of one of the images, then that image is transformed and presented again, and this time the user has to choose the word (from a set) most related to the image being showed. 3D CAPTCHA [14] does not exactly fall into the picture database-based category. Instead, it uses pre-generated 3D models that are rendered in a randomized scene in 2D. For each part of each model, there is a textual description. The computer selects random objects from the set of available ones and sets them up in a scene, rendered also from a random point of view and with random light sources, and finally attaches letters to the different parts of the different models, asking the user to enter them in the sequence specified by a list of given parts of models (figure 2(c)). The idea of generating 2D CAPTCHAs from 3D models is not new, as [27] proposes a simpler scheme of 2D challenge generation from 3D object models.

Other visual types We propose in [2] some video CAPTCHAs schemes based in real videos, analyzed and then transformed. The user has to answer whether them had been transformed or not, and the way. The videos need to have a former analysis to avoid easy-to-solve specimens. With some extra care, it is also possible to use on-line video repositories as the input source for some of our CAPTCHA schemes. Using animation, [3] proposes a general way of preventing relay and human-solving farms attacks: they include a time component in their CAPTCHAs, so that the same challenge has a time dependant answer, and could not be easily display exactly as-is in a remote computer. A long chimera of many proposals is to be almost transparent, not noticeable by the user. One surreptitious CAPTCHAs that does not alter navigation is [29], which consists of replacing many of the links of the web-site for images with captions to tell where to click. Metadata [15] is supposed to be another type of this scheme, but no information is public to date.

[14] http://spamfizzle.com/CAPTCHA.aspx
[15] (http://www.forbes.com/2008/11/25/captcha-pramana-bots-tech-identity08-cx_ag_1125captcha.html)

Audio CAPTCHAs Visual CAPTCHAs put in production need a counterpart to provide access to the visually impaired[15]. Audio CAPTCHAs can be hard to solve for humans [4]. The GMail and Microsoft Live! audio CAPTCHAs are based on voice samples of people reading decimal digits (0 to 9), with some background noise. They have already been broken with simple algorithms [40, 39]. A different type of audio CAPTCHA is the new[16] one provided by reCAPTCHA. Using the same idea on which their visual CAPTCHA is based, they provide sound recordings (from radio broadcasts) that have been partially recognized by their software. The user has to correctly recognize at least the part that the software audio recognizes. Note that this scheme is weaker than the visual one, as here the test is either a previously human-solved test, or one to be solved (and not a mix of both). If it is one to be solved, then the current best software for audio recognition will be able to break the CAPTCHA.

2 Attacks

To date, publicly known attacks on CAPTCHAs are mostly the work of amateur programmers or researchers. Some schemes have been known to change without a public notice of its breakdown, so there is a chance that some attacks have been and will remain unpublished.

2.1 Text recognition

OCR/text-based CAPTCHAs have been the first type to be deployed, and are still the most widely used type. It is logical that they have attracted so far most of the attention. The first published attack against an OCR/text CAPTCHA is devoted to Mori and Malik [18]. They use shape-matching (in a two steps approach) to find candidate letters, and then filter the candidate words. In another approach they try to find the whole words directly. Another attack [30], based on distortion estimation, succeeds 99% of the time. For simpler OCR/text CAPTCHAs, it is possible to use simpler algorithms (as pixel counting [31], letter derotation [35], etc.). There are open source algorithms able to decode the most simple OCR/text based [36]. More recent attacks [32] allow to break most OCR/text based CAPTCHAS (including Yahoo, GMail, etc.). It has been shown [33] that typical text transforms do not increase security, as computers recognize isolated characters better than humans.

These examples are just current attack possibilities, but there are many others to explore. During our preliminary research, we have found that an attack which focusses on regular shape detection (using adapted Hough transforms for segments, ellipses, etc.) is of use against Baffletext, and also Megaupload. Captcha2's background is not a major problem; it follows some clear color-shift patterns. Also, the transformations used to depict the letters are fixed, and uses the same set of transformed letters for depicting the character that has to be clicked on in the box. Thus, a simple pattern matching algorithm would be able to solve it. Teabag has many possible analysis, notably those based on the non-uniformity of the challenge (see later in 2.7).

[16] Introduced in 2009 to replace their old, digit based, model.

2.2 Audio recognition

Typical audio CAPTCHAs are very basic, and excluding very few exceptions, most current implementations can be considered defeated. Current proposals are so simple than no complex analysis is required, just a slight filtering and pattern matching [40, 39]. More sophisticated attacks have been studied [12], rendering even better success ratios. The new version (as of 2009) of the reCAPTCHA audio CAPTCHA is one of the few that remains unsolved, but as we stated in the previous section, it is not as strong as the visual one, and we are exploring the possibility that unsolved challenges can be used to bypass it.

2.3 Side-channel attacks

These type of attacks are based on deviations from randomness that allow for a correlation among the challenges and their answers. In this section we present a detailed example (against HumanAuth), and we briefly describe other attack against ASIRRA [5]. The ASIRRA Public Corpus, that the developers of ASIRRA have created to help researchers, is composed of around 25.000 images classified as cats or dogs, a half in each class. We analyzed all of the files using the ENT[17] tool, producing a formatted output (in ARFF format, so to be used with Weka [6]). These were later processed by a classifier which was able to distinguish cats and dogs pictures with a nearly 60% accuracy, without using any kind of image recognition technique. The simplest decision tree [7], based on only the **size** of a jpeg file, is able of distinguishing between cats and dogs with an accuracy over 58%, significantly better than random. Following an identical approach, we used the ENT tool with the images in the HumanAuth source code, producing an ARFF file. The best classifier (in this case RandomForest) was able to show an accuracy rate of 77.8761%, significantly better than the $\frac{68}{68+45} = 60.177\%$ that a trivial classifier (that always predicts the larger class) will do. To prevent easy image library indexing, the authors of HumanAuth decided to merge a PNG image with the random JPG image taken from the library: they put the watermark PNG in a random position into the JPG canvas and merge both using a level of transparency. This suggests that the initial small set of images, when used with the scheme proposed by the HumanAuth authors of merging with a watermark, may not be of use against this type of attack, even though might be enough to prevent hash-function indexing. Choosing a different watermark that alters more the original image could be beneficial, but that would be also at the expense of human recognition.

2.4 Feature-based attacks

In Philippe Golle's work [8], he presents the strongest attack against ASIRRA as of April-2009. This attack is based in image processing, as it divides the photographs into NxN cells of color and texture (gray-scale) information, and use that to feed two support-vector machine (SVM) classifiers that, when used together, are capable of classifying with around a 83% accuracy, thus allowing them to solve the 12-photos challenge with a 10.3% probability. They found that color-presence features are more

[17] http://www.fourmilab.ch/random/

accurate for classifying cats and dogs than color-histogram features. With this scheme, the authors reach a 77.1% accurate classification rate. With texture processing, they reach an 80.4% successful classification. Combining both methods, they reach an 82.7% accurate classification rate.

2.5 Attacks against database-based CAPTCHAs

When a CAPTCHA is based on a database of knowledge (i.e., labeled pictures), and especially if that database is public, there are some possible attacks against its database that could thwart its reliability. Basically, those are:

1. Database poisoning attacks: if the database is public, and not protected, we can upload information in a way that assures that when later confronted with challenges created using that uploaded information we will solve those challenges.
2. Database indexing attacks: if the database is small enough and/or the bandwidth is high enough, we can download (maybe partially) the database contents and get enough information to solve or greatly simplify the CAPTCHA.

2.6 Implementation flaws

Some CAPTCHA protection systems can be bypassed without using OCR techniques at all, simply by reusing the session ID of a known CAPTCHA image [44]. That is the result of a very bad implementation, but it was not uncommon some years ago. A common error, even today [18], is that some codify the answer to the challenges in the URL address or in the value of a form field. Knowing this, we can request as many challenges as we like with the same solution. We can calculate medium values of those challenges, and thus launch a mean attack [45]. Other implementations use a hash (such as an MD5 hash) of the solution as a key passed to the client. If the number of possible answers is finite, or their distribution is not uniform, solutions hashes could be learned in a proportion enough to solve it. Another straightforward implementation flaw consists on using only a small fixed pool of challenges. For example, HumanAuth uses less than one hundred images, and even masking them with logos, we cannot prevent them from being indexed or characterized [5]. As another example, QRBGS challenges are not created on-demand, but repeated [43]. If the answer to the challenges could be any (i.e., floats, in a broad range) and it followed an uniform distribution, repetition of challenges would not be so deleterious, but in nearly all other cases is fatal. Some additional design flaws in HumanAuth (commented later) imply there is a big chance than the challenge answer is 0, or, alternatively, a small integer. This makes possible to use another, very successful, kind of attack: if answering 0 fails, then we will answer with a succession of integers that will run through N, starting with the smallest absolute values. As a last and typical implementation flaw, we have to note that some algorithms use a quite unwise way of communicating with the CAPTCHA server, that is easy to attack [14].

[18] See, for example, the CAPTCHAS currently used by Fotolog, Uol, Conduit, MetroFlog and others.

2.7 Design flaws

In this section we introduce some common design flaws that have rendered some known CAPTCHAs much less secure than intended.

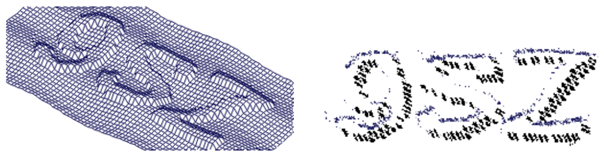
Biased answer distribution One obvious mistake, but quite common to find, is selecting a clearly non-uniformly distributed subset of the possible answers as correct answers. One example is QRBGS (MathCAPTCHA), whose designers use one-digit figures in all their arithmetic operations. That makes it likely that the answers will be small integers. Additionally, they use derivatives of trigonometric functions when x has a typical value of $\frac{\pi}{2}$ or π and they do not expect floats as answers (possibly to avoid precision problems), so low-integers and specially zero are very good candidates for successful blind answers (up to 93% success rate, depending on problem subtype) [11].

Another example is phase one of IMAGINATION, in which the user has to click on the "center" of any of the images. Clearly this center distribution is not a uniform distribution. We find other examples in OCR/text CAPTCHAs that do not use the full set of possible characters. For example, the Megaupload CAPTCHA avoids values O, I, J, 0 to prevent user confusion. Worse, it always uses the scheme of three letters followed by a digit. That makes it more user friendly, but also much weaker. Teabag uses only three characters, and are strongly not uniformly distributed [34], possibly to avoid characters that are difficult to distinguish in the 3D projections they use. In a sample of 100 challenges, characters 'S', 'Z', '3', 'P', 'b', 'w', 'M', 't' and 'd' appeared more than 3% of the time (peak: 4.3%), while a large set of other 34 characters appeared none, including '1', '0' (perhaps to avoid coincidence with 'I' and 'O'), etc. The chi-square value for this distribution with 61 degrees of freedom gives us 262.08, corresponding to a two-tailed p value of less than 0.0001. As the challenge answer does not differentiate among upper and lowercase [19], the situation is even worse. In this case, the most common character is letter 'm' (6%), and there are 4 characters that appear more than a 3% of the times, while other 18 do not appear at all. That means that just blindly answering the string 'mmm' will be a good guess. Depending on the scheme, uniformity in the answer distribution can be considered a theoretical aim, but we should always know how far we are from it, as being too far can render our CAPTCHA useless.

Biased challenge distribution Any bias from randomness in the characteristics of the challenges can allow for challenge analysis, which may lead to side-channel attacks (if they allow correlation with the answers), or simpler-than-intended challenge categorization and/or analysis. One example is Teabag. In it, the frontal borders of the characters can be selected by area size, because this distribution is far from uniform: so size has a meaning. Additionally, thanks to the non-uniformity of the image (for example, the areas at the borders), it is easy to tell the angles of the image: flood-filling the corners and studying the lines that they create, we can de-rotate and

[19] Teabag page does not specify this detail, but the implementation at http://barafranca.com/game-register.php does not make any difference.

de-shear the image to a normal 2D perspective. There is also a correlation among pixels that allow for (probabilistic) back border detection, using very simple algorithms, like growing the background areas by pixel continuity. In a fair percentage of the challenges generated, the non-character part of the image can be completely or almost completely removed [34].



(a) challenge (b) w/o background

Fig. 3. Teabag, original and first phase of process.

Another example is the Megaupload CAPTCHA: the letters and the digit are always printed using the same font type, similar if not equal to Antique Olive (as identified by Identifont [20]). They are rotated, but at a precise angle, either clockwise or counter-clockwise, being the first letter rotated clockwise, the second counter-clockwise, etc. Additionally, it prevents more than two characters from overlapping (this and the former allow for the creation of overlapping maps with all combinations, although they are not necessary to break the CAPTCHA) [9].

Correlation between Challenge and Answer This is a worse scenario than the ones presented before: here, the challenge somehow provides (non intending, that is "leaked" or "side-channel") information on the answer. When the answer is binary, then the problem is similar to the previous biased challenge distribution problem. The already explained side-channel attack against HumanAuth and ASIRRA clearly exemplifies the dangers of this scenario.

Limiting the size of the set of correct answers Is a straightforward weak point that takes place much more commonly than what could be thought. Clear examples are MegaUpload, asking always for 3 letters and a digit (in that order) and not distinguishing case (also avoids some characters (I, J, L, O, 0), possibly to avoid confusions), or Teabag, asking for only 3 characters (and not distinguishing case, either). Also, many OCR/text base CAPTCHAs limit the number of characters of the answer.

Categorization of the answer There is not always the need to make it easy for a program to know if the guessed answer for the CAPTCHA was right or wrong.

[20] http://www.identifont.com

Interaction with the remote system (typically, web-site) can go on as usual, being the actual data results different, etc. If possible, it should be avoided the immediate knowledge of whether an answer to the challenge is correct or wrong, or any other way to know if it is close to being a correct answer. We can communicate this information using some intermediary communication mechanism (like email accounts, which will also need to be controlled not to use the same more than n times), or present that information to the user in a way difficult to distinguish automatically.

User dependance Make the CAPTCHA, in any way, dependant on the challenger, is in general a very bad idea, specially worse if this dependance is, or could be, known. One example of this problem is ASIRRA that, to further increase the possibilities of adoptions of the pets showed in the CAPTCHA, will show the pets in Petfinder that are close to the challenger (using IP geolocation). This weakness is egregious, as it renders easier to many types of attacks, including database poisoning (the number of poisoned records to bypass the CAPTCHA is significatively lower), database indexing, etc.

2.8 Human solving

CAPTCHAs are designed to be solved by humans, but there is an ongoing market of CAPTCHA solving services [38] [21] (typically located in an area where relatively cheap labor can be found) and relay attacks, that present the CAPTCHA challenge to a human user that receives some benefit from solving it [37]. This is part of the CAPTCHA economics. There is still place for larger abnormalities, i.e., one client willing to break a CAPTCHA can actually sell a CAPTCHA service to other sites, and use them as relay agents to solve it. We have already mentioned a CAPTCHA that tries to prevent human relay attacks.

3 The general CAPTCHA design problem

In this section we try to build upon the previous sections that showed different kinds of CAPTCHAs; the AI problems they try to be based on, some of their design problems and implementation flaws; and the currently known attacks against them; to discuss the general problem of CAPTCHA design, and finally, their current strength and limits.

3.1 Threat model

The original purpose of a CAPTCHA was determining a human from a machine remotely. If that determination is possible, we can protect services from automatic/massive requests. With the evolution of the attacks on some web services, it has been more evident that the ability to tell a human from a machine may not be enough. Is a

[21] Some examples of human-solving services are (http://captchaocr.com/ad.html) and (http://decaptcher.com).

human in a CAPTCHA solving farm something we should avoid? Should a human surfing through another web site, or presented with another program GUI, not be eligible for solving our CAPTCHAs? Is a human assisted computer program still an automatic attack? There are different types of cryptanalytic techniques depending on the threat model we choose: chosen plaintext, known plaintext, known cyphertext, related key, side-channel, etc. Analogously, there are different threats that a CAPTCHA can choose or not to prevent, from the most basic to the most comprehensive:

1. be able to distinguish humans from computers, by measuring an "human" quality, ability or behavior.
2. be able to prevent 'magnifying/human-assisted' attacks: distinguish humans from human-assisted algorithms, i.e., algorithms that with some human intervention can solve the CAPTCHA many times. This assisted algorithms have a big ratio of CAPTCHAs solved per human intervention [22].
3. be able to prevent relay attacks: distinguish humans presented with the CAPTCHA in the original CAPTCHA site, to those presented with the CAPTCHA in another site/interface [23] [3].
4. be able to prevent 'human farms' attacks: including methods to inhibit or make more difficult solving the CAPTCHA using 'farms' of solvers [24].

3.2 AI hardness not transmitted

Most CAPTCHAs have been broken [18, 8, 5, 9, 11, 14, 42, 40, 39, 32, 31, 30, 34] due to one of these problems:

1. The "hard-AI" underlaying problem they are based on is not the original one intended, but a much more specific and weaker one. For example, most OCR/text CAPTCHAs require to solve a problem far easier than the "general OCR" problem [32, 9, 34]. In this aspect, reCAPTCHA tries to approach the real problem more realistically.
2. Design and implementation flaws make them much more easy to bypass using a procedure that, analyzing the challenges in a non detailed manner, permits to guess an answer with a high percentage of success. Thus, these procedures can be called side-channel attacks, as they do not try to solve the original problem that the CAPTCHA designers intended attackers to solve, but one which is much easier [11, 5]. That is the difference between the intended solving path and the actual solving path. The intended path is the one based on a hard problem (purportedly representing an AI-hard problem), the real one is the one based on any design and/or implementation flaw that avoids the harder way.

The difficulty of an AI-unsolved problem is thus not easily transmitted to a CAPTCHA design. One of the reasons is because we do not know how to categorize AI-hardness, so we do not know how to tell if a particular subset (generated by

[22] An example could be a variant of the Megaupload with multiple typefaces: solved with lines and shapes analysis (using the Hough transform) and human intervention (classifying the new schemes as corresponding letters).
[23] Sites/programs that, upon solving the CAPTCHA, reward the user in some way.
[24] One possible way of doing this is with in-site "hidden" CAPTCHAs [29].

a CAPTCHA) is hard enough. Also, an AI-problem can be difficult to solve correctly more than, i.e., 15%, but we typically need a much smaller figure to be able to attack the CAPTCHA. Regarding the image recognition problem, used for many modern CAPTCHA designs, we face the same obstacles. "Image recognition/understanding" is not typically the same problem that the CAPTCHA designers finally implement. Human-solved database-based CAPTCHAs can approach the hardness of the original AI problem if we filter out easy specimens (which is what reCAPTCHA does), although they impose the limitations of a database-based CAPTCHA design, including the need of a constant input stream of unsolved challenges.

4 Related problems

Designers face additional problems that have not to do with their resistance to attack. As many CAPTCHAs protect free services, or commercial services with competence, CAPTCHAs have to be easy to solve and appealing. Ideally, if the service is worldwide, they have to be language-independent (i.e., reCAPTCHA is harder for non-English speakers, with a 97% vs. 92% success ratio). The problem of accessibility is also of greater concern, and few visual CAPTCHAs are fully already prepared for it (i.e., HumanAuth). If possible, they have to be also possible to operate through other channels (i.e., mobile phones). This requirements have made some researchers invent new forms of reusing known CAPTCHA schemes, as for example, converting text CAPTCHAs into clickable images [1]. An additional problem is the tendency to overlook the difficulty of their design, specially when designing new in-house developed CAPTCHAs, typically by groups of programmers that are not specialists, which designs are put right into production [11] [25].

5 Where to go

Recapitulating on the previous sections, we present a brief corpus of conclusions, learned from the previously studied problems. We describe good properties that a CAPTCHA design should have, ways to theoretically study the assurance level of a design, and to empirically test for good properties regarding to randomness in different aspects of the CAPTCHA.

5.1 Good properties CAPTCHAs should have

Here we present some good properties that new CAPTCHA designs should have:

1. Randomness, uniform distribution, in all parameters. For example, for a text CAPTCHA: variable number of characters, uniform numbers of pixels/areas/lines/... with certain properties (color, group, size of group, etc.), different type faces used, size of images, etc.

[25] As an example, CAPTCHAs offered for Wordpress at (http://wordpress.org/extend/plugins/tags/captcha)

2. No easier challenges: subtypes or alternatives should have the same strength (audio and visual CAPTCHAs).
3. Problem posed should be as broad, and as close as possible, to the AI problem that inspires the test.
4. Design should incorporate features to prevent relay attacks, and to detect automatic bypass easier.
5. Challenges should be independent and uniformly distributed (this excludes dependance from the user). Also the answers should be random, uniformly distributed. There should not be any statistical correlation among challenges and answers.
6. Make it difficult for a program to tell if its answer was or not correct.

As a good practice, we recommend, for any new CAPTCHA design, to be put into production in a test web-page, without other protections (to focus on the hardness of the CAPTCHA), for a period long enough to allow study.

5.2 Assurance level of CAPTCHAs

In this section we propose some basic methodologies to analyze the security offered by a CAPTCHA design. This analysis is theoretical, and studies the behavior of the CAPTCHA design under unfavorable assumptions. For knowing better the security offered by a CAPTCHA design, we propose to theoretically analyze the CAPTCHA under the following assumptions:

1. Answer repetition: if an attacker is able to collect a finite sample of challenges with the same answers, in any quantity, for any number of different answers she wants, confirm that she will not able to create a better answer, for a challenge that pertains to one of those answers, than a random one (that is, that there is no better attack than trial and error).
2. Challenge repetition: if our CAPTCHA has only n different challenges, and we do not know their answer, there should be no better strategy to solve them than trial and error, which success ratio should be low. This should apply also when $n = 1$.
3. Non categorization: if our CAPTCHA is composed of different types of challenges, there should be no way to automatically distinguish them. There should also be no way of categorizing the difficulty of different challenges.

5.3 Security assessment

In this section we propose a practical test that can be done over any CAPTCHA design and implementation, and that should be added to the ones aforementioned. This test measures the random distribution of answers to the challenges.

For this test, we create a large enough set of elements (T = test, A = answer) of tests with their correct answers. For the designers, creating this set should be straightforward. Then, using general randomness and statistical analysis tools[5], we search for non-uniformities in this distribution, that is:

1. Non-uniformities in the distribution of A (that would allow for a blind attack)

2. Correlations among T and A (that would allow for a side-channel attack)
3. Non-uniformities in the distribution of T (that could allow for type-of-challenge categorization, and challenge analysis)

These test can be done for T as is (as a bit stream) or for some simple properties of T, which, depending on the type of challenge, can be color histogram, areas' sizes histogram, distances between similar areas, bit correlation with given vectors (i.e., trained by GA), maximum and minimum for a block of bytes, etc. This can be used as a very general analysis tool to realistically estimate the security parameters of any CAPTCHA proposal, and we believe it will be advisable to use it in the future before any similar systems are launched to have adequate, well-reasoned, and founded security parameters and realistic estimations. One of their main advantages is that it does not depend on the underlying format (image, sound, video, etc.) or problem, and that it could be useful for avoiding pitfalls such as the existence of some trivial and irrelevant parameter values leaking too much information [5, 11]. For the study of correlations and other predictive possibilities we, apart from the classical statistical tools, strongly recommend the use of Machine Learning algorithms such as those found on the free Weka [6] tool.

6 Conclusions and further work

It is possible to assume broader threat models and design CAPTCHAs that also prevent human farming, human relay attacks, and are able to detect when they are under computer attack, and when they have been bypassed. Hardness assumptions relying on the base AI problem underlying a particular CAPTCHA are often used as arguments, and given as facts. In reality, the AI problem proposed by the CAPTCHA is always a small subset, typically much easier to solve than the original. Worse, typically CAPTCHA designs have pitfalls that make them even easier to pass. We expect this work to be a good starting point: both for designers of new CAPTCHAs, to avoid some common design flaws that could render their ideas useless, and for the creation of additional methodologies for security assessment, and evaluations of the assurance level they provide.

References

1. R. Chow, P. Golle, M. Jakobsson, L. Wang and X. Wang. "Making CAPTCHAs Clickable". Proc. of HotMobile 2008.
2. Hernandez-Castro, Carlos Javier and Ribagorda, Arturo. "VideoCAPTCHAs". To be puslished in the Proceedings of the 5th International Conference on Security and Protection of Information, Brno, 2009.
3. Athanasopoulos, Elias and Antonatos, Spiros. "Enhanced CAPTCHAs: Using Animation to Tell Humans and Computers Apart". IFIP International Federation for Information Processing, 2006.
4. Jeffrey P. Bigham and Anna C. Cavender. "Evaluating Existing Audio CAPTCHAs and an Interface Optimized for Non-Visual Use". CHI 2009.
5. Hernandez-Castro, Carlos Javier, Ribagorda, Arturo and Saez, Yago Saez. "Side-channel attack on labeling CAPTCHAs". Submitted for publication.

6. G. Holmes, A. Donkin, and I.H. Witten. "Weka: A machine learning workbench". In Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane, Australia, 1994.
7. Ross Quinlan. "Machine Learning". Morgan Kaufmann Pub., CA.
8. Phillipe Golle. "Machine Learning Attacks Against the Asirra CAPTCHA". ACM CCS 2008.
9. Hernandez-Castro, C.J. and Ribagorda, A. "Preliminary analysis on the Megaupload CAPTCHA". Submitted for publication.
10. Achint, Thomas and Venu, Govindaraju. "Generation and Performance Evaluation of Synthetic Handwritten CAPTCHAs". ICFHR 2008.
11. Hernandez-Castro, Carlos J, and Ribagorda, Arturo. "Pitfalls in CAPTCHA design and implementation: the Math CAPTCHA, a case study". Submitted for publication.
12. Tam, Jennifer, Simsa, Jiri, Hyde, Sean and Von Ahn, Luis. "Breaking Audio CAPTCHAs". (http://www.captcha.net/Breaking_Audio_CAPTCHAs.pdf)
13. von Ahn, Luis, Maurer, Benjamin, McMillen, Colin, Abraham, David and Blum, Manuel. "reCAPTCHA: Human-Based Character Recognition via Web Security Measures". Science Magazine, 2008, Vol. 321. no. 5895, pp. 1465 - 1468.
14. Caine. A. and Hengartner, U. "The AI Hardness of CAPTCHAs does not imply Robust Network Security". IFIP, v. 238, Trust Management, pp. 367-382.
15. W3C Working Draft, Techniques for WCAG 2.0. G144: *Ensuring Web Page contains another CAPTCHA serving same purpose using a different modality.*
16. J. Elson, J.R. Douceur, J. Howell, J. Saul. "Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization". Proccedings og the 14th ACM CCS, 2007.
17. L. von Ahn, M. Blum, N.J. Hopper, and J. Langford. "CAPTCHA: Using hard AI problems for security". In EUROCRYPT 2003, Warsaw, Poland, v. 2656 of LNCS, pp. 294–311. Springer, 2003.
18. G. Mori and J. Malik. "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA". In Proc. of CVPR03, pp. 134–144. IEEE, 2003.
19. M. Naor. "Verification of human in the loop or Identification via Turing Test". www.wisdom.weizmann.ac.il/ naor/PAPERS/human.ps. Retrieved 2009-01-01
20. M. Chew and J. D. Tygar. "Image recognition CAPTCHAs". In Proc. of ISC 2004, pp. 268-279. A longer version as UC Berkeley Computer Science Division technical report UCB/CSD-04-1333.
21. L. von Ahn and L. Dabbish. "Labeling Images with a Computer Game". ACM Conference on Human Factors in Computing Systems, CHI 2004. pp 319-326.
22. US Patent no. 6,195,698, "Method for selectively restricting access to computer systems". http://www.freepatentsonline.com/6195698.html
23. Rusu A., and Govindaraju V. "Handwritten CAPTCHA: using the difference in the abilities of humans and machines in reading handwritten words", in Proceedings of the IWFHR-9, 2004, Tokyo, Japan, pp. 226–231.
24. R. Stevanovic, et al. "Quantum Random Bit Generator Service for Monte Carlo and Other Stochastic Simulations," in LNCS, vol. 4818, 2008, pp. 508-515.
25. Baird H.S. and Riopka T. "ScatterType: a Reading CAPTCHA Resistant to Segmentation Attack", in Proc. of the IS & T/SPIE Document Recognition & Retrieval Conference, 2005, 197-207.
26. R. Datta et al. "IMAGINATION: A Robust Image-based CAPTCHA Generation System", in Proc. of ACM Multimedia Conf., pp. 331-334, 2005.
27. M. E. Hoque, D. J. Russomanno and M. Yeasin. 2D Captchas from 3D Models, in Proceedings of the IEEE SoutheastCon 2006, Memphis, April 2006.
28. Chew M. and Baird H. S.: BaffleText: a Human Interactive Proof. Proceedings of the 10th SPIE/IS&T Doc. Recog. Retr. Conf. (DRR2003), 2003.

29. Baird H. S. and Bentley J. L.. Implicit CAPTCHAs, in Proc. SPIE/IS&T Conf. on Document Recognition and Retrieval XII (DR & R 2005), 2005, 191-196.
30. G. Moy et al. ”Distortion Estimation Techniques in Solving Visual CAPTCHAs”, in Proc. of the CVPR’04, v. 2, 2004.
31. Jeff Yan, Ahmad Salah El Ahmad. ”Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms”. Proceedings of the ACSAC, 2007.
32. K. Chellapilla, P.Y. Simard. ”Using Machine Learning to Break Visual HIPs”, in Proc. of the Conf. on Neural Information Processing Systems (NIPS) 2004.
33. K. Chellapilla, K. Larson, P.Y. Simard, M. Czerwinski. ”Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs)”. Procs. of CEAS, 2005.
34. Hernandez-Castro, C.J. and Ribagorda, A. ”Analysis of the Teabag CAPTCHA version 1.2”. Submitted for publication.
35. DarkSEO programming: letter derotation. http://www.darkseoprogramming.com/2008/04/05/letter-derotation/.
36. PWNtcha. http://caca.zoy.org/wiki/PWNtcha
37. TROJ_CAPTCHAR.A Trojan horse to relay CAPTCHAs at TrendMicro. Article at http://blog.trendmicro.com/captcha-wish-your-girlfriend-was-hot-like-me/
38. D. Danchev. Inside India’s CAPTCHA solving economy. http://blogs.zdnet.com/security/?p=1835
39. Van der Vorm, J. ”Defeating audio (voice) CATPCHAs.” http://vorm.net/captchas
40. R. Santamarta. ”Breaking GMail’s audio CAPTCHA”. http://blog.wintercore.com/?p=11
41. Oli Warner. Kittenauth. http://www.thepcspy.com/kittenauth
42. Techworld.com article on fallen CAPTCHAS. http://www.techworld.com/security/features/index.cfm?featureid=41
43. NewScientist on the QRBGS CAPTCHA. http://www.newscientist.com/blog/technology/2007/08/prove-youre-human-do-math.html
44. Horward Yeen. Breaking CAPTCHAs without using OCR. (http://www.puremango.co.uk/2005/11/breaking_captcha_115/)
45. Wolfgang Wieser. Captcha recognition via averaging. (http://www.triplespark.net/misc/captcha/)