ON THE STRENGTH OF EGGLUE
and other Logic CAPTCHAs

Carlos Javier Hernandez-Castro, Arturo Ribagorda
IT&C Sec. Group (SETI), Comp. Science Dept., Univ. Carlos III, Leganés, Madrid, Spain
chernand@inf.uc3m.es, arturo@inf.uc3m.es

Julio Cesar Hernandez-Castro
School of Computing, Portsmouth University, United Kingdom
Julio.Hernandez-Castro@port.ac.uk

Keywords: CAPTCHA, HIP, common sense, AI, logic reasoning, semantic extraction, data mining, natural language modeling

Abstract: CAPTCHAs or HIPs are tests able to tell humans and computers apart, remotely and over an untrustworthy channel. They rely on abilities that are though to be hard for algorithms, yet easy for humans. General logic reasoning, based on common sense knowledge, is one of the areas that are still considered hard for AI. On the other hand, logic reasoning targeting very specific areas has achieved success in AI. In this article, we list current Semantic and Logic CAPTCHAs and examine how strong they are. We also discuss wether this model is suited or not for automatic challenge generation and grading.

1 Introduction

In a recent congress on Computer Security, one well known Computer Security expert asked me why just not use common sense questions to create strong CAPTCHAs. I tried to explain him all the associated theoretical problems, but I can tell that he was not very convinced, thinking that his idea was strong enough.

He is not the only one. Some CAPTCHA designers think alike, and also many programmers that design their own CAPTCHAs to protect their sites. There are several on-line discussions about CAPTCHA user-friendliness, and why not to use logic-CAPTCHAs to improve it[1]. In this article, we intend to present the current situation regarding Logic and Semantic CAPTCHAs: which ones are being used, how are they performing, specifically how strong they are against simple attacks.

1.1 Organization

The rest of the paper is divided into a very brief introduction to the state of the art of CAPTCHAs; an introduction to the state of the art of Logic and Semantic CAPTCHAs, along with a definition of them; a brief discussion on their security, as well as a more thorough discussion on one of the most used ones; a general discussion on Logic and Semantic CAPTCHAs; and a conclusion, summarizing our results and what we consider should (not) be future trends in Logic and Semantic CAPTCHA design.

[1]Some examples at `http://www.tylercruz.com/captcha-vs-human-logic/`, at `http://www.rubyflow.com/items/209` or at `http://www.w3.org/2004/Talks/0319-csun-m3m/slide15-0.html`.

2 Brief history of CAPTCHAs

Any active Internet user has already faced CAPTCHAs several times. If you have been asked to pick up images with a certain property from a set, to read a very distorted word, to hear and write part of an speech, to drag and drop one image from a set into a region, or to answer a common sense question, you have already faced a CAPTCHA challenge.

CAPTCHA history commences when engineers from Yahoo! met a research team at CMU to explain their problem with bots joining online chats and promoting web addresses. The CMU team came with an idea for a test for telling humans and computers apart, and a set of rules that *any* such a test should have. Their test (EZ-GIMPY) was put into production at Yahoo! with initial success. However, the first instance of a CAPTCHA in a production system was the one at the AltaVista search engine[2]. Although the first time that the idea was mentioned was even before (Naor, 1996).

[2]Used to prevent bots that automatically send URLs to the search engine for indexing (Broder, 2001).

The original problem that created the need for CAPTCHAs has aggravated in the recent years. Nowadays it is not only about preventing spam, but also about ticket scalping, search engine ranking manipulation (or SEO, Search Engine *Optimization*), dictionary attacks for retrieving account passwords, automatic registration (for phishing, social network information gathering, spam through SMS services, instant messaging, etc.), automatic voting in on-line polls, creation of on-line traps (as bride profiles in dating services), aggregation of information from different on-line sources, different scams (through sites that include reputation rankings, or fake job advertisements in on-line listings for mule recruiting), denial of service (DoS) through consumption of limited resources (stock booking), etc.

The initial success of CAPTCHAs did not last very long. Within a few years *most* text-based CAPTCHAs were broken. Concern over the strength of this kind of CAPTCHA impelled research in the field. But many of the new proposals had several drawbacks and resulted to be not as strong as intended by their authors. Examples of CAPTCHAs that were initially considered hard and resulted not to be so have been ASIRRA[3], IMAGINATION[4], Captcha2[5], reCAPTCHA[6], the Google text CAPTCHA, the original Hotmail CAPTCHA[7], Yahoo! CAPTCHAs (all three reportedly broken several times), different audio CAPTCHAs, etc. Several others resulted weak as well: the QRBGS CAPTCHA (Hernandez-Castro and Ribagorda, 2009a), PHP3BB CAPTCHAs [8], Megaupload [9] and other file-exchange services CAPTCHAs, etc.

The current situation is that just a few *'strong'* text-based CAPTCHAs are still on-line, all of them after several major changes in their design. At the same time, many on-line services are using particular and much weaker CAPTCHAs, that are broken as soon as there is enough interest. A new actor has also entered the market: the CAPTCHA breaking services based in the combination of low-wage workers and automatic solutions. None of the currently in use CAPTCHAs are able to counter these services, but there are a few proposals against them.

[3]Presented in (Elson et al., 2007), and broken in (Golle, 2009) and (Hernandez-Castro et al., 2009).

[4]Presented in (Datta et al., 2005) and broken in (Zhu et al., 2010)

[5]Available at `http://www.captcha2.com` as of the 8th of February of 2011, and broken at (Hernandez-Castro et al., 2010b)

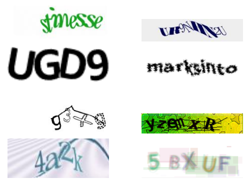
[6]Introduced in several articles, as (Ahn et al., 2008), and broken professionally, and academically by (Wilkins, 2010), currently still in use with variations.

[7]Hotmail is using reCAPTCHA as of January 2011.

[8]References to several ones can be found at `http://blog.phpbb.com/2008/08/28/captchas-in-phpbb/`, being the most common `http://www.codeproject.com/KB/web-security/PhpbbCaptcha.aspx`.

[9]With a Mozilla Firefox plug-in that bypasses it at `http://skipscreen.com/`.

Table 1: Text-based CAPTCHA examples.



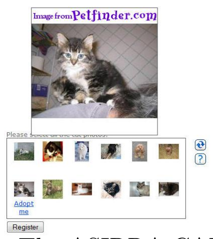


Figure 1: The ASIRRA CAPTCHA.

2.1 Current types of CAPTCHAs

There are literally hundreds, if not thousands, of different CAPTCHAs out in the Internet, if we sum all the different types and their variants. Thus, in this section we will try to present the most representative categories, citing some examples, so as to give an overview of the current state of the art.

Of course, evolved text-based CAPTCHAs are still in use (Table 1). They are typically based in older designs that have been evolved to prevent different attacks put into practice against them.

There are other ideas that can be considered also variations of text-based CAPTCHAs: using artificially generated *'handwritten-like'* text (Achint and Venu, 2008) (Rusu and Govindaraju, 2004), making the segmentation problem harder (Baird and Riopka, 2005). Breaking away from text-based CAPTCHAs, Chew and Tygar (Chew and Tygar, 2004) used a set of labeled images to generate CAPTCHAs challenges (using Google Images). Others (von Ahn and Dabbish, 2004) proposed other image-labeling mechanisms, and other authors have used on-line human-labeled sets of images (Elson et al., 2007) (Figure 1).

Other image-based CAPTCHAs rely on closed databases of labeled images (Warner, 2009) (Hernandez-Castro et al., 2010a) (Datta et al., 2005). All mentioned image-based CAPTCHAs have been broken (Hernandez-Castro et al., 2009) (Golle, 2009) (Zhu et al., 2010), or their image-base is considered too small to be secure. There are other proposals for image-based CAPTCHAs (Hoque et al., 2006), but their hardness has not properly been analyzed.

A logic evolution from image based CAPTCHAs has appeared recently, and is based in video (or animation) based CAPTCHAs (as in (Hernandez-Castro and Ribagorda, 2009b), or `http://www.nucaptcha.com/`, `http://www.kloover.com/video-captchas/`, etc.). One interesting proposal concerning moving images is called Emerging Images CAPTCHA (EI) (Mitra et al., 2009) (Figure 2). There is also another quite special proposal, the

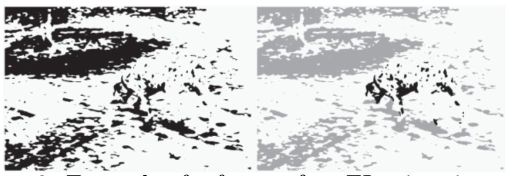


Figure 2: Example of a frame of an EI animation, with the original shape depicted to the right. Example from (Mitra et al., 2009).

Enhanced CAPTCHA (Athanasopoulos and Antonatos, 2006), which aims to prevent the relay problem[10].

Another *comprehensive* approach is based in trying to model the behavior of human users of a web-site, and use that model to tell them apart from computers. This scheme cannot be always called a CAPTCHA, as it might lack the *Public* characteristic. There are some examples of this kind of tests[11], although not yet very common. There are other types of CAPTCHA that typically mix more than just one approach. One example of these is the QRBGS (Stevanovic et al., 2008) CAPTCHA, that got wide media attention (Knight, 2007) (Vaughan-Nichols, 2008). And, of course, there are CAPTCHAs based on the supposedly unique ability of humans to understand *common sense* and solve *logic* questions.

3 Logic and Semantic CAPTCHAs

A Logic CAPTCHA is one in which challenges are based on a question that requires common knowledge, and sometimes some basic logic, to be answered. These are some examples:

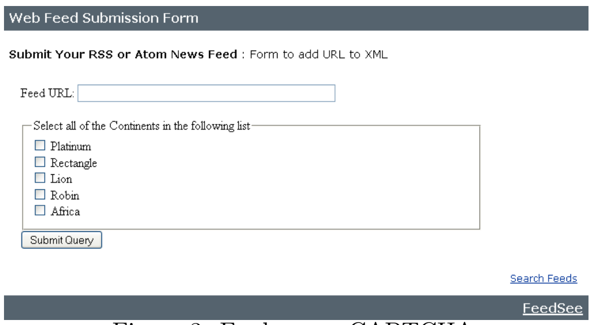
- Most people have ten fingers and . . . toes?
- What animal quacks? . . .
- What comes next? 'Monday Tuesday Wednesday . . . '
- Tomorrow is Saturday. If this is true, what day is today? . . .
- Choose the three that are artificial: moon, guitar, balloon, test, glass, snow

A Semantic CAPTCHA is the one in which the challenges urge you to complete it (or order it, etc.) in a way that *makes sense*, and also can be intended to involve some kind of *reasoning*. Of course, the definition of *to make sense* is not unique, and some answers that would be valid for some individuals might not be valid for others, including the CAPTCHA designer. Some examples of semantic CAPTCHA questions are:

- Complete with a verb: Stopwatches can . . . numbers

[10] `http://www.zdnet.com/blog/security/inside-indias-captcha-solving-economy/1835`, retrieved on the 1 of August 2010

[11] One provider is `http://www.pramana.com/`.


Web Feed Submission Form

Submit Your RSS or Atom News Feed : Form to add URL to XML

Feed URL:

Select all of the Continents in the following list

- ☐ Platinum
- ☐ Rectangle
- ☐ Lion
- ☐ Robin
- ☐ Africa

Figure 3: Feedse.com CAPTCHA.

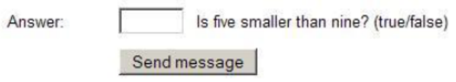

Answer: Is five smaller than nine? (true/false)

Figure 4: Kentico CMS CAPTCHA.

- Choose the one that makes more sense: 'A hummingbird is a . . . of bird' from: small, red, type, subconscious, stone

3.1 Current Logic and Semantic CAPTCHAs

Most current implementations of Logic and Semantic CAPTCHAs are quite basic. Let us provide here some examples:

1. The CAPTCHA used at `https://help.tenderapp.com/discussion/new`, which provides 8 pre-programmed questions.
2. The CAPTCHA used at Feedsee (`http://www.feedsee.com/submit.html`), which provides test-type questions (Figure 3). These tests require the user to select the one or several answers that are related to a certain category (numbers, shapes, birds, metal, mythical beings, continents, etc.). There are 5 words in each list, so 2^5 possible answers.
3. Logic CAPTCHA for Kentico CMS [12] (Figure 4). This CAPTCHA provides some logic questions, numerical questions, comparisons, pre-programmed statements (in file *statements.txt*) that can be *true* or *false*, etc.
4. `http://www.purple-dogfish.co.uk/free-stuff/accessible-captcha`, also allows just pre-programmed questions.
5. `http://www.vision.to/simple-accessible-logical-captcha.php`, also with pre-defined questions.
6. At xda-developers forum (a well-known site for development for HTC and Android phones, among others), they use not one CAPTCHA but two, a combination of reCAPTCHA and

[12] At `http://devnet.kentico.com/Blogs/Petr-Passinger/October-2010/New-and-Free-at-the-Marketplace--Logic-Captcha.aspx`.

NoSpam![13], a vBulletin[14] CAPTCHA plug-in that allows to add pre-programmed questions. In the case of the xda-developers forums, we have detected 22 questions pre-programmed. The usage of NoSpam! can be tracked in search engines looking for the string *NoSpam! Verification Question*, and a search engine as Google reports *approximately* 66.400 web-pages containing that string[15].

7. `http://www.vision.to/simple-accessible-logical-captcha.php`, with pre-programmed questions of the type *'Please enter the two first letters of: The lazy dog went to Chihuahua'*.

8. HumanAuth is a image labeling CAPTCHA, but because the classification it requests is not direct (i.e., it is not enough with naming what is in the picture), but instead requires to decide if the picture content is something natural or artificial, it can be considered a logic CAPTCHA too. In fact, HumanAuth has a version for vision-disabled people, in which every image is replaced for what the image represents, and the challenge has still the same hardness.

9. BrainBuster[16] (Figure 5) is a CAPTCHA programmed in Rail, which includes a module for storing questions and answers (thus, they have to be previously input).

10. TextCaptcha[17] creates different questions based on different question types, and an internal classification of certain words (*white* is a colour, etc.). Its author claims that it is able to generate 180,243,205 different questions[18]. It has been easily broken (Hernandez-Castro et al., 2011), and also by an a program accessible on-line[19], and even more easily using out of the box tools [20].

There is one a bit more elaborated. It relies on a knowledge base, and rules to create questions (and answers) based on it. It is egglue currently in broad use today, which we will discuss in further detail.

[13]More information on NoSpam! at `http://www.vbulletin.org/forum/showthread.php?t=124828`.

[14]A CMS system. More information at `http://www.vbulletin.org/forum/portal.php`.

[15]Result from Google when queried with *"nospam! verification question"*, the 29th of December of 2010.

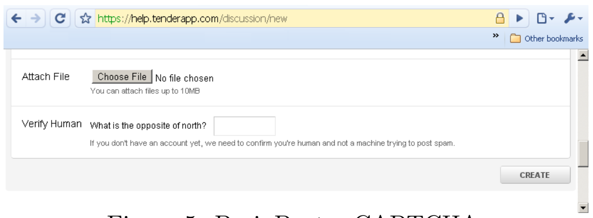
[16]The BrainBuster project page can be found at `https://github.com/rsanheim/brain_buster`.

[17]At `TextCaptcha.com`.

[18]Statements from `http://textcaptcha.com/how_it_works`, retrieved on the 17th of December of 2010

[19]It can be found at `http://textcaptchabreaker.appspot.com/`

[20]As in `http://joelvanhorn.com/2010/11/10/using-wolframalpha-to-hack-text-captcha/`, but with a lower success ratio.

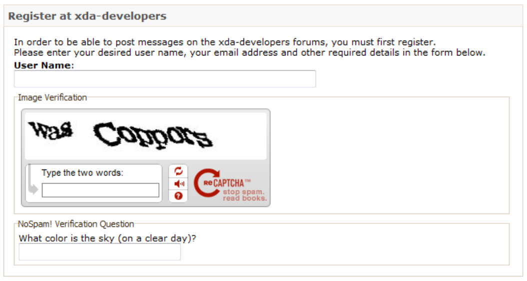


https://help.tenderapp.com/discussion/new

Attach File — Choose File — No file chosen
You can attach files up to 10MB

Verify Human — What is the opposite of north?
If you don't have an account yet, we need to confirm you're human and not a machine trying to post spam.

Figure 5: BrainBuster CAPTCHA.



Register at xda-developers

In order to be able to post messages on the xda-developers forums, you must first register.
Please enter your desired user name, your email address and other required details in the form below.
User Name:

Image Verification
Type the two words:

NoSpam! Verification Question
What color is the sky (on a clear day)?

Figure 6: xda-developers forum registration CAPTCHAs.

4 Egglue CAPTCHA

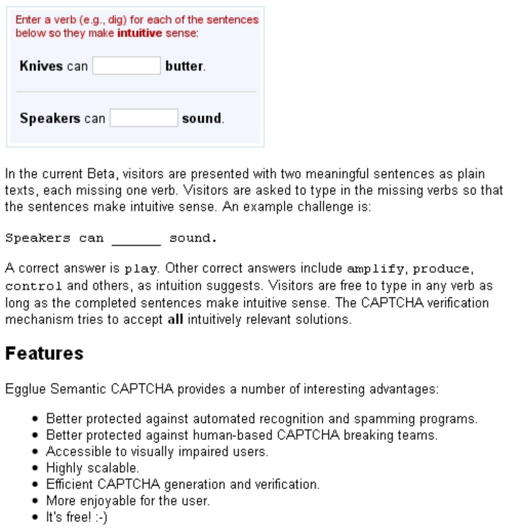
The Egglue CAPTCHA requests us to complete a sentence with a verb in a way that *makes sense*. For that, we will have to use our common sense, and general knowledge. The Egglue author explains in the web-page how it is designed (Figure 7).

To better grasp the capabilities of Egglue, some actual questions created by it in Figure 8.

We have analyzed the source code of the PHP module, written by Egglue's author, that interacts with the Egglue server. Other developers have created a Drupal[21] module for Egglue. Drupal provides a statistic of module usage, and the count of use for this module is currently 1,154 Drupal web-pages[22].

[21]Drupal is a very popular open-source CMS system. Drupal web site is at `http://drupal.org/`.

[22]All the figures in this section about Egglue usage have been collected on December 2010.



Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Knives can ______ **butter**.

Speakers can ______ **sound**.

In the current Beta, visitors are presented with two meaningful sentences as plain texts, each missing one verb. Visitors are asked to type in the missing verbs so that the sentences make intuitive sense. An example challenge is:

`Speakers can _____ sound.`

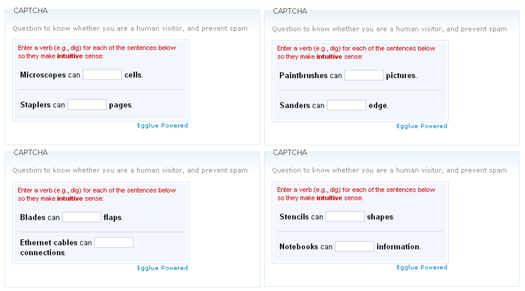
A correct answer is `play`. Other correct answers include `amplify`, `produce`, `control` and others, as intuition suggests. Visitors are free to type in any verb as long as the completed sentences make intuitive sense. The CAPTCHA verification mechanism tries to accept **all** intuitively relevant solutions.

Features

Egglue Semantic CAPTCHA provides a number of interesting advantages:

- Better protected against automated recognition and spamming programs.
- Better protected against human-based CAPTCHA breaking teams.
- Accessible to visually impaired users.
- Highly scalable.
- Efficient CAPTCHA generation and verification.
- More enjoyable for the user.
- It's free! :-)

Figure 7: Egglue description.


CAPTCHA

Question to know whether you are a human visitor, and prevent spam

Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Microscopes can ____ **cells**.

Staplers can ____ **pages**.

Egglue Powered

CAPTCHA

Question to know whether you are a human visitor, and prevent spam

Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Paintbrushes can ____ **pictures**.

Sanders can ____ **edge**.

Egglue Powered

CAPTCHA

Question to know whether you are a human visitor, and prevent spam

Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Blades can ____ **flaps**.

Ethernet cables can ____ **connections**.

Egglue Powered

CAPTCHA

Question to know whether you are a human visitor, and prevent spam

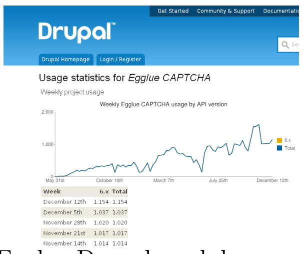
Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Stencils can ____ **shapes**.

Notebooks can ____ **information**.

Egglue Powered

Figure 8: Egglue challenge examples.



Usage statistics for *Egglue CAPTCHA*

Weekly project usage

Week	6.x	Total
December 12th	1,154	1,154
December 5th	1,037	1,037
November 28th	1,020	1,020
November 21st	1,017	1,017
November 14th	1,014	1,014

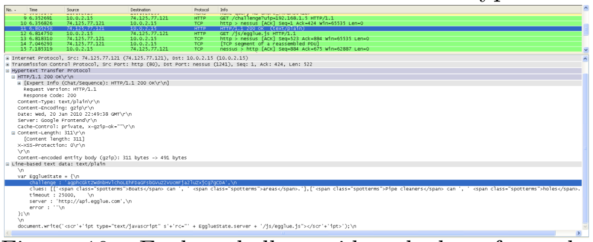
Figure 9: Egglue Drupal module usage statistics.

We can try to estimate also the usage of this module by querying Google with a string that is characteristic of this module: "egglue powered". This is a bad decision of the Drupal plug-in designer, as it easily renders a list of web-pages using this CAPTCHA and, worse, in order of relevance for a search engine. Anyway, processing this query in different search engines gives us an *estimated* count of 752 web-pages in Google, 43 web-pages in AltaVista and 866 web-pages in Yahoo!.

4.1 Egglue Protocol

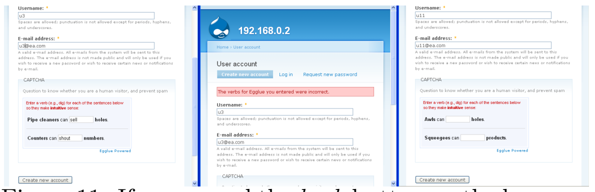
Egglue PHP source code connects to the Egglue server for challenge retrieval and answer evaluation. We have analyzed its communications using the well-known Wireshark tool. Egglue's communications run over HTTP, and are very straightforward. When a new challenge is requested from the Egglue server, its basic constituents are downloaded as an XML file. Among these, it is an *unique* identifier (ID), that will be sent again to the verification server with the proposed answers. Figure 10 shows an example of an Egglue capture in the moment that it answers with the challenge-id and the challenge *clues*.

One error in the implementation of Egglue is that it does not check for ID reutilization. That means that, once one correct solution is found, we can reuse the solution and the ID to bypass the


```
No. -  Time        Source          Destination     Protocol  Info
    9 6.352691    10.0.2.15       74.125.77.121   HTTP      GET /challenge?uip=192.168.1.5 HTTP/1.1
   10 6.356826    74.125.77.121   10.0.2.15       TCP       http > nessus [ACK] Seq=1 Ack=424 Win=65535 Len=0
   11 6.805250    74.125.77.121   10.0.2.15       HTTP      HTTP/1.1 200 OK  (text/plain)
   12 6.814750    10.0.2.15       74.125.77.121   HTTP      GET /js/egglue.js HTTP/1.1
   13 6.818310    74.125.77.121   10.0.2.15       TCP       http > nessus [ACK] Seq=523 Ack=884 Win=65535 Len=0
   14 7.046293    74.125.77.121   10.0.2.15       TCP       [TCP segment of a reassembled PDU]
   15 7.185319    10.0.2.15       74.125.77.121   TCP       nessus > http [ACK] Seq=884 Ack=675 Win=62887 Len=0

⊞ Internet Protocol, Src: 74.125.77.121 (74.125.77.121), Dst: 10.0.2.15 (10.0.2.15)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: nessus (1241), Seq: 1, Ack: 424, Len: 522
⊟ Hypertext Transfer Protocol
  ⊟ HTTP/1.1 200 OK\r\n
    ⊞ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Request Version: HTTP/1.1
      Response Code: 200
    Content-Type: text/plain\r\n
    Content-Encoding: gzip\r\n
    Date: Wed, 20 Jan 2010 22:49:38 GMT\r\n
    Server: Google Frontend\r\n
    Cache-Control: private, x-gzip-ok=""\r\n
  ⊟ Content-Length: 311\r\n
      [Content length: 311]
    X-XSS-Protection: 0\r\n
    \r\n
    Content-encoded entity body (gzip): 311 bytes -> 491 bytes
⊟ Line-based text data: text/plain
    \n
    var EggluеState = {\n
        challenge : 'agphcGktZWdnbHVlchoLEhFDaGFsbGVuZ2VUcmFja2luZxjCg7gCDA',\n
        clues: [['<span class="spotterms">Boats</span> can ', ' <span class="spotterms">areas</span>.'],['<span class="spotterms">Pipe cleaners</span> can ', ' <span class="spotterms">holes</span>.
        timeout : 25000,    \n
        server : 'http://api.egglue.com',\n
        error : ''\n
    };\n
    \n
    document.write('<scr'+'ipt type="text/javascript" s'+'rc="' + EggluеState.server + '/js/egglue.js"></scr'+'ipt>');\n
```

Figure 10: Egglue challenge id and clues from the Egglue server.


Username: *
u3
Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address: *
u3@ea.com
A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

CAPTCHA

Question to know whether you are a human visitor, and prevent spam

Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Pipe cleaners can sell **holes**.

Counters can shout **numbers**.

Egglue Powered

Create new account

192.168.0.2

Home › User account

User account

Create new account | Log in | Request new password

The verbs for Egglue you entered were incorrect.

Username: *
u3
Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address: *
u3@ea.com
A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

CAPTCHA

Username: *
u11
Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address: *
u11@ea.com
A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

CAPTCHA

Question to know whether you are a human visitor, and prevent spam

Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Awls can ____ **holes**.

Squeegees can ____ **products**.

Egglue Powered

Create new account

Figure 11: If we pressed the *back* button on the browser, the same ID was requested to Egglue, and we could try again to solve the same challenge.

CAPTCHA again. We can also try different solutions for the same ID till we find the correct one (Figure 11). Of course, this is a major failure in its implementation.

The user's answer is sent to `http://api.egglue.com/verify`, posted as a form, with the following parameters:

- remoteip: the IP of the client that is connecting to the PHP front-end to answer the challenge.
- egglue_challenge: the ID of the challenge.
- egglue_response1: the verb for the first challenge sentence.
- egglue_response2: the verb answered to the second challenge sentence.

The verification server sends back the string *true* if the answer is consider correct by the Egglue server.

4.2 Attack

Egglue poses us a question that *has to be answered* with just a verb. The question has always the same structure: *noun* can ... *noun*, with just one variant: *noun* can be also a full subject, like *"pair of pliers"*.

On a first look, many questions arise:

- How random are those questions?
- How many different subjects, direct objects and relations are in Egglue's database?
- Are there questions with more than one correct answer? (supposedly affirmative by Egglue's author) How many correct answers?
- How does Egglue deduct the relationship between the subject and the direct object?
- How does Egglue's scoring algorithm work?

The answers to these questions will help us know how strong is Egglue against attacks. For example, a low randomness in the subjects, or in the relations, will make a learning attack successful. Also, if Egglue's scoring algorithm is a simple *or* of both answers, it will be much weaker. We will address these questions after we analyze the results of our attacks, in Section 4.4.

The first, most straightforward strategy for debunking Egglue, would be to use some knowledge-base to answer its questions. Which knowledge base can be handy for that? There are different ones we can use:

- search engines: for example, for answering *noun1* can ... *noun2*, we can query a search engine with the string "*noun1* can" *noun2*, or just *noun1 noun2*, and explore the results for the regular expression *noun1* can * *noun2*, parsing whatever matches the * looking for verbs.
- English corpus on-line: there are a few of them, some even with the ability of matching up to four words substrings, which would be ideal for our case[23].
- create our own corpus, by parsing several on-line text catalogs as, for example, the Gutenberg Project.

One particular problem in English, that is not as bad in other languages, is being able to tell verbs from nouns, as many words can be both. As an example, the word *word* can be a noun, and also a verb. We took here a heuristic approach, consulting on-line dictionaries, and counting how many definition entries we found for the word as a noun, and as a verb. Of course this strategy is not fault-proof, but in general, it gave us a good rate of verb identification. We also tried with different alternatives, including lists of verbs, and querying simultaneously more than one on-line dictionries[24].

4.2.1 Type 1 attack : Search Engines

Our first approach for studying the *strength* of Egglue was just to use a search engine to find the correct answers. We tested several search engines, and found Google to be the best for this purpose, for two important reasons:

- It returned more, and more significant, results than the other search engines, and
- it allowed us to search using wildcards.

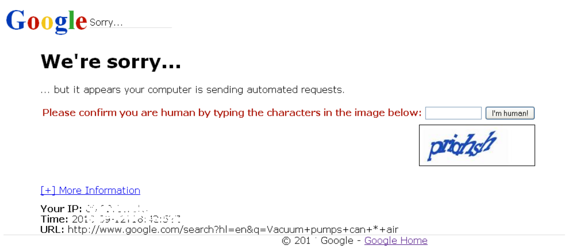
We created a program in Python that downloaded an Egglue challenge (consisting in two basic challenges), asked Google using queries related to each challenge, studied the results, and sent an answer back to Egglue.

We tested different ways of querying Google, being all of them combinations of the following basic queries:

- "*noun1* can" * *noun2*
- "*noun1* can * *noun2*"
- "*noun1* * *noun2*"
- *noun1 noun2*
- *noun1* can *noun2*

[23] As the British National Corpus, at `http://bncweb.info/`. Unluckily, we did not gain access to in on time.

[24] We particulary liked the results from `http://www.hyperdictionary.com/search.aspx?define=program`, `http://dictionary.cambridge.org/dictionary/british/program_1` and `http://www.onelook.com/?w=program\&ls=a`.



We're sorry...

... but it appears your computer is sending automated requests.

Please confirm you are human by typing the characters in the image below:

[+] More Information

Your IP: [illegible]
Time: 201[illegible]
URL: http://www.google.com/search?hl=en&q=Vacuum+pumps+can+*+air

© 201[illegible] Google - Google Home

Figure 12: Sometimes, Google required us to pass its CAPTCHA to continue getting results for the same query.

The more restrictive the query (as in: "*noun1* can * *noun2*") the more significative the results, but also the higher chance of not getting any result, so we added to the (possible) result of these queries, the results of less restrictive ones (as: *noun1* can *noun2*). If we found that a query had many results, we requested Google up to the first three pages of results (returning 20 results per page).

Google did not like much to answer these queries. Sometimes, it detected us as a bot and made us pass a CAPTCHA (Figure 12), and other times, it even did not give us this option.

There are other ways of querying Google apart from using its regular web interface. One possible way is using the Google Web Search API[25] (now deprecated) or the (newer) Custom Search API[26]. Both solutions were suboptimal, as the first one returned different results, in different order, and in less quantity, that the regular (web) search engine, and the second one is quota limited. That is why we stick to using Google's regular web interface, using HTTP Proxies and other tools and ideas to minimize the rejection rate of our queries.

4.2.2 Type 2 attack : Google + Common Verbs Combined

Sometimes, we were not able to get enough results from Google as to be able to select a verb, that would link both parts of the sentence. Initially, in those cases we sent the string *'???'* as an answer, expecting Egglue's to not validate it, and for us to have a mark in the log for that incident.

Of course, a natural idea would be that, in those cases, we can just send a random verb from the n most common, instead of a nonsense string. We tried this approach just as an easy way to marginally improve the results from the previous attacks. But our results were not exactly what we expected!.

4.2.3 Type 3 attack : Common Verbs

A simpler strategy would be to answer using random verbs. At first we did not even consider this basic strategy, but observing some weird results from

[25] `http://code.google.com/apis/websearch/`.

[26] `http://code.google.com/apis/customsearch/v1/overview.html`.

Egglue's validation mechanism, we decided to investigate into it.

Using all English verbs would be a poor strategy, but using the ones more common in English could be interesting. Given a list (taken from the Wikipedia[27]) of the most common verbs in English, we can answer selecting at random among the n most common.

4.3 Attack results

The attacks previously described were used to learn more from the Egglue CAPTCHA internals and strength. They were put in place in several rounds between June and October of 2010. We will discuss their results in this section.

After several series of the *type 1 attack*, we found that we did not get significantly different results with one type of query or the others. The only significant difference came when all our queries were too restrictive, so much as to not get a single verb as a valid result. Even though we sent the string *'???'* as an answer to Egglue in all these cases, not all of them were consider wrong. For example, in one of our test runs (composed of 500 challenges) we got the following (which include our mark *'???'*) as right:

- Allen wrenches can (???) shafts, Spanners can (loosen) nuts
- Screwdrivers can (open) locks, Pocket knives can (???) holes
- Carriers can (hold) cats, Paring knives can (???) holes
- Vises can (hold) rods, Razors can (???) holes
- ...

This lead us to conclude that the validation mechanism was clearly not an *and* of both sentences being correctly resolved, but maybe an *or*, or some kind of summation of correctness for both sentences that should pass a certain threshold.

In general, the *type 1 attack* gave a success ratio of 50.83% (calculated for our latest test run, with 214 correctly resolved challenges – according to Egglue – out of 421 challenges). If we use the combined *type 2 attack*, the success ratio raised to 68.66% (103 correctly answered out of 150). Looking into the logs of this result in more detail, we were astonished to see that it seemed that when we used the most common verbs in English, the success ratio seemed to increase. That is why we thought of the *type 3 attack*, just choosing randomly among

[27]From the article *Most common words in English*, retrieved the 2nd of June of 2010 (`http://en.wikipedia.org/w/index.php?title=Most_common_words_in_English\&oldid=365914607`). Note that this verb list is associated by *lemmas*, that is, the entry *be* contains within it the occurrences of *is*, *was*, *be*, *are*, etc. This information is referenced from the Oxford English Corpus (`http://www.oxforddictionaries.com/page/oec?view=uk`).

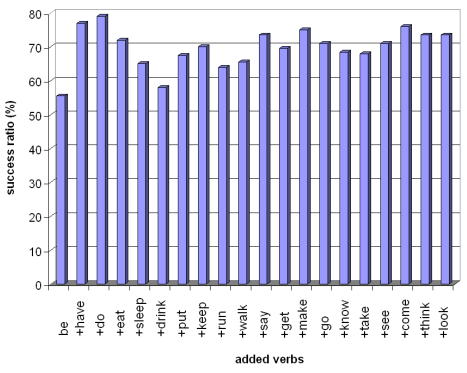


Figure 13: Success ratio if we answered using the n ($1 \leq n \leq 20$) most common English verbs.

the n most common verbs in English. Our first test included the first 20 most common English verbs according to the Wikipedia[28]. They were ordered in descending order of frequency of appearance, and we tested values of n between 1 (which means just always answering *be*) and 20 (using all the 20 most common verbs) – in increments of 1. Figure 13 represents how this strategy behaves for each value of n (with the detail of each verb added to the pool).

As we see, we never get results below 50%!. In fact, if we answer randomly using just the three most common verbs in English (*be*, *have* and *do*), we correctly guess the answers 79% of the time!. Of course, this is not possible in real life, so the validation mechanism of Egglue must be too lax to accept verbs when it should not. In fact, if we check the log corresponding to this situation (the three most common verbs), we see answers such as:

```
Choosing randomly among the 3 most common English verbs to answer
Answering strategy : 3
1 : ['Plungers can (?:have) water', 'Blades can (?:be) flaps']
0 : ['Electric mixers can (?:have) butter', 'Highlighters can (?:have) words']
1 : ['Paintbrushes can (?:have) colors', 'Pair of pliers can (?:have) tabs']
1 : ['Vacuum pumps can (?:have) air', 'Highlighters can (?:be) words']
1 : ['Scissors can (?:do) ends', 'Needles can (?:have) veins']
1 : ['Boats can (?:be) islands', 'Transformers can (?:do) impedances']
1 : ['Scales can (?:do) sizes', 'Trowels can (?:be) concrete']
1 : ['Forks can (?:have) eggs', 'Lenses can (?:do) images']
1 : ['Ladles can (?:have) fat', 'Awls can (?:do) holes']
1 : ['Meters can (?:have) voltage', 'Notebooks can (?:be) information']
1 : ['Chainsaws can (?:do) trees', 'Calculators can (?:have) answers']
1 : ['Pitchforks can (?:have) soils', 'Screwdrivers can (?:do) plates']
1 : ['Sextants can (?:be) altitudes', 'Pliers can (?:have) wires']
1 : ['Switches can (?:be) voltages', 'Wooden spoons can (?:do) rice']
0 : ['Spatulas can (?:do) layers', 'Pencils can (?:be) holes']
0 : ['Rakes can (?:be) areas', 'Toners can (?:be) skin']
(...)
```

It is very disputable that *Forks can have eggs* and that *Lenses can do images*, but this answer was marked as correct (*1*) by Egglue.

After analyzing the formerly discussed Figure 13, we thought that maybe, a reordering of the set of verbs, in which the ones that make the success ratio raise were used first, could even improve our success of bypassing Egglue. With the aim of being more precise at which verbs were more successful, we run a test in which we always answered with the same verb, for all these 20 most common verbs. We though Egglue might somehow detect it, and give a nearly 0% success rate in this scenario, but we were wrong.

[28]Retrieved the 2nd of June of 2010, at `http://en.wikipedia.org/w/index.php?title=`

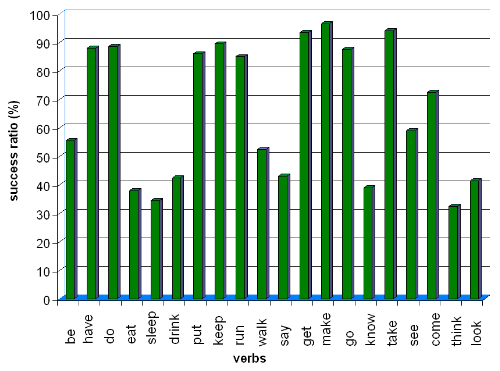


Figure 14: Success ratio of each of the most common English verbs.

Table 2: Egglue attacks and their success ratios.

Attack	Success ratio (%)
Search Engine	50.83
S.E. (restr.) & 20 most fr. verbs	68.66
20 most frequent verbs	73.5
3 most frequent verbs	79
make & *get* & *take*	95
make	96.5

Figure 14 shows the results from this test. Amazingly, if you always answer *make* to Egglue, you solve it 96.5% of the times. Two other verbs (*get* and *take*) can be used it to solve it over 90% of the time. Not only that, other six verbs (*have*, *do*, *put*, *keep*, *run*, and *go*) can be used to solve it more than 80% of the time. All verbs pass the CAPTCHA more than 30% of the time, being *think* the one with the lowest score, an still impressive 32.5%.

Given that it is considered that a success rate of 0.6% is enough to consider a CAPTCHA broken (Zhu et al., 2010) (others consider them broken with lower rates, such as 0.01%) the current implementation of the Egglue CAPTCHA can be considered completely broken, using a simple strategy: always answering *make*. Or, if you do not like being repetitive, you can answer randomly among *make*, *get* and *take*, and still score 95% of the time. A summary of our attack results can be seen in table 2.

4.4 Other Egglue Design Problems

Other Egglue versions, or derived CAPTCHAs might get better at scoring the *meaningfulness* of a sentence. If that is the case, we might need to resort to the original idea of Data Mining (in our case, no more than an advance search engine is needed) to bypass it. Even that might be problematic if the CAPTCHA is really well designed.

Unluckily with Egglue, this is not the case. When we tested our first basic automatic attacks against Egglue, we were surprised of some of our findings. We though that Egglue was able to do some kind of data mining for creating a knowledge base, based on which challenges were created. We expected this knowledge base to be broad enough,

`Most_common_words_in_English\&oldid=365914607`.

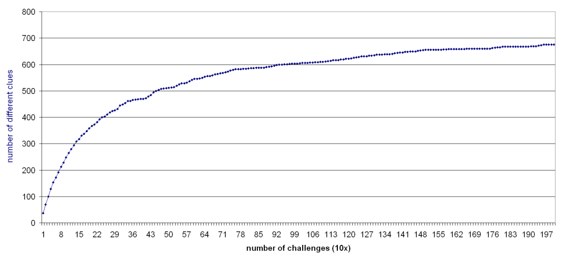


Figure 15: Number of new clues per every ten compound challenges (20 sentences).

Table 3: Egglue ratios of clue appearance.

25% most frequent			25% least frequent		
Clue	#	%	Clue	#	%
holes	550	2, 29	clinometers	1	0, 004
screws	312	1, 3	spring balances	1	0, 004
ends	282	1, 17	candles	1	0, 004
paintbrushes	249	1, 03	jointers	1	0, 004
areas	243	1, 01	vessels	1	0, 004
pair of pliers	228	0, 95	grounds	1	0, 004
images	215	0, 89	bars	1	0, 004
water	214	0, 89	sheep	1	0, 004
air	214	0, 89	lists	1	0, 004
nuts	201	0, 83	forces	1	0, 004
signals	199	0, 82	dimensions	1	0, 004
spatulas	185	0, 77	pushpins	1	0, 004
prisms	174	0, 72	hand drills	1	0, 004
hammers	167	0, 69	pulp	1	0, 004

and to cover different areas. Our findings were a bit different:

- Many of the objects used for creating the challenges were related to mechanics, electronics, and engineering in general. That is, items like "The Moon", or abstract ones like "Attraction", do not appear at all, or rarely.
- Many of the subjects are repeated. And worse than that,
- When subjects are repeated, they frequently are related to the same subset of direct objects.

Not only the subject of the clues is not wide and uniformly distributed: worse, the number of clues is not enough. If we count the appearance of the different *clues* (in Egglue's terminology, the beginning and ending of each sentence) in a test round in which we downloaded 6,000 compound challenges (14,000 sentences), we only found 710 different *clues*. As an example, in another test run of 2,000 challenges (that would be a maximum number of 8,000 different clues) we found just 676 different clues (Figure 15). The total number of categories is going to be well under 1500.

Another problem is the frequency in which these clues appeared. In table 3 you can see the first 15 more frequent clues, and the least 15 frequent ones. Just the 10 most frequent clues account for 11, 28% of appearances. Learning how to answer to those with a fair success rate will also bypass this CAPTCHA.

Such a big deviation from a standard distribution makes Egglue weaker than necessary.

4.4.1 Egglue Scoring Algorithm

Apart from that, we discovered that the scoring algorithm was not very precise. Sentences that clearly make sense were scored as wrong, wether sentences

Table 4: Examples of Egglue wrongly scored answers.

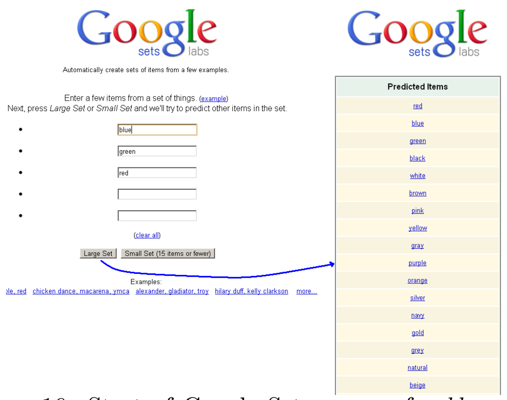
Wrongly scored as correct.
Allen wrenches can (???) shafts Spanners can (loosen) nuts
Knives can (bit) bits Micrometers can (selected) diameter
Screwdrivers can (open) locks Pocket knives can (???) holes
Lasers can (remove) tissues Soldering irons can (buy) wires
Stopwatches can (unlimited) numbers Magnetic compasses can (indicate) directions
Awls can (make) holes Wrenches can (cleco) valves
Socket wrenches can (bolts) screws Screwdrivers can (remove) plates
Forks can (contain) eggs Hatchets can (get) wood
(...) (...)
Wrongly scored as wrong.
Cranes can (reach) trees Heating pads can (relieve) pain
Notebooks can (?:get[29]) information Rakes can (seed) seeds
Smokers can (?:eat) bees Tape measures can (measure) lengths
Can openers can (punch) holes Atlases can (map) maps
Notebooks can (provide) information[30] Spanners can (tighten) nuts
(...) (...)

that were clearly wrong were scored as right. For example, during our first basic attacks, we used the mark *???* for those times in which our (initial) algorithm did not find a verb suitable for answering. Even though, some of these answers containing *???* were scored as correct!. It also happened than sometimes our algorithms incorrectly classified a word as a verb, and used it for answering the challenge . . . and some of these challenges answers were reported as correct (!). A few of the answers that were scored as correct, when they clearly should have not, and correct ones classified as wrong, are shown in table 4.

5 Other Considerations

The logic and semantic CAPTCHAs will always have their advocates, as it is simple to think of improved schemes that will filter typical bots. The problem is that people designing those schemes will typically not consider how easy it would be to circumvent them for a bot, if they became popular enough, or if they are used to protect an interesting site. Let us give the following example: on-line journals that allow comments on news, as well as blogs that allow comments on new posts, are typical sites for bots to wander and publish their spam. One can argue that, as every entry in these systems should be done by an human (the reporter, or the blog author), then it would be easy for them to add one or two logic questions derived from the text. For example, if in my blog about *Charles Darwin* I create a new entry about his school years, then I can easily add the following two questions for anyone trying to post a comment:

- How many siblings did Charles Robert Darwin have?


Automatically create sets of items from a few examples.

Enter a few items from a set of things. (example)
Next, press *Large Set* or *Small Set* and we'll try to predict other items in the set.

- blue
- green
- red

(clear all)

Large Set | Small Set (15 items or fewer)

Examples:
ple, red chicken dance, macarena, ymca alexander, gladiator, troy hilary duff, kelly clarkson more...

Predicted Items
red
blue
green
black
white
brown
pink
yellow
gray
purple
orange
silver
navy
gold
grey
natural
beige

Figure 16: Start of Google Sets answer for *blue green red.*

- With which brother did he attended Edinburgh University?

Do you consider a general CAPTCHA scheme like this a good idea? It is a very bad idea: both questions can be answered with one single word. Even worse: it is a word that will be in the article!. Even if we select more complex questions, as *At Edinburgh University, what could Charles not bear?* (being correct answers *the sight of blood or suffering*), a substring of just three consecutive words from the text can be used to correctly answer.

Also, the field of data mining is quickly improving. As an example, IBM has created an application, based on data mining and parallel computing, that is able to beat human players when playing the popular TV game *Jeopardy*[31]. It would be hard to base a CAPTCHA on a *private* data mining algorithm (thus, a CATCHA, without the *P*) that would prevail long enough. If instead we use a corpus of private knowledge, what is going to assure us that there is not a public corpus which intersects with ours in a non null set?

Other possibilities arise. We can think of a new logic CAPTCHA which questions can be based on telling the word, in a list, that does not correspond to the rest. For example: *Which word does not belong to the list: blue green cat red?*[32]. A quick access to *Google Sets*[33] would be enough to answer it (Figure 16).

6 Conclusions and further work

In this article we have presented the state of the art of logic and semantic CAPTCHAs. We have seen that, despite their current limitations and weaknesses, there is a certain amount of hype and misunderstanding of their capability in telling humans apart from computers.

[31]More information on this subject can be found at `http://www-943.ibm.com/innovation/us/watson/research-team/index.html`.

[32]Example taken from the *word list CAPTCHA* at `http://drupal.org/project/captcha_pack`.

[33]Google Sets can be found at `http://labs.google.com/sets`.

We have analyzed in more detail one particular CAPTCHA that is currently being used by hundreds of web-sites, called *Egglue*. We have found that the strength claims made by its author are unfortunately not withstanding. We have seen that other popular logic CAPTCHAs (as the *TextCaptcha*, the Feddsee.com CAPTCHA, BrainBuster, etc.), presented in this article, share a similar fate.

We conclude that there is not a strong enough logic or semantic CAPTCHA to be put into production. Also, designers willing to create new logic and semantic CAPTCHAs should be aware of the common pitfalls done when estimating their strength.

We do not recommend to pursue research on this line of CAPTCHA design. We consider at least uncertain that a possible logic CAPTCHA is anything else than a chimera. Instead, we recommend future CAPTCHA designers to explore other ways of creating a CAPTCHA that have not to do with logic, semantic, or data mining, because of the many possibilities for CAPTCHAs based on those principles to be flawed.

REFERENCES

Achint, T. and Venu, G. (2008). Generation and performance evaluation of synthetic handwritten captchas. In *Proceedings of the First International Conference on Frontiers in Handwriting Recognition, ICFHR.*

Ahn, L. V., Maurer, B., Mcmillen, C., Abraham, D., and Blum, M. (2008). recaptcha – human-based character recognition via web security measures.

Athanasopoulos, E. and Antonatos, S. (2006). Enhanced captchas – using animation to tell humans and computers apart. In Leitold, H. and Markatos, E., editors, *Communications and Multimedia Security*, volume 4237 of *Lecture Notes in Computer Science*, pages 97–108. Springer Berlin – Heidelberg.

Baird, H. S. and Riopka, T. (2005). Scattertype: a reading captcha resistant to segmentation attack. In *Proc., IS&T/SPIE Document Recognition and Retrieval Conf*, pages 16–20.

Broder, A. (2001). US Patent no. 6,195,698.

Chew, M. and Tygar, J. D. (2004). Image recognition captchas. In *Image Recognition CAPTCHAs*, pages 268–279. Springer.

Datta, R., Li, J., and Wang, J. Z. (2005). Imagination: a robust image-based captcha generation system. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 331–334, New York, NY, USA. ACM.

Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2007). Asirra: a captcha that exploits interest-aligned manual image categorization. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 366–374, New York, NY, USA.

Golle, P. (2009). Machine learning attacks against the asirra captcha. In *Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS 2009, Mountain View, California, USA, July 15-17, 2009*, ACM International Conference Proceeding Series. ACM.

Hernandez-Castro, C. J. and Ribagorda, A. (2009a). Pitfalls in captcha design and implementation: the math captcha, a case study. *Computers & Security.*

Hernandez-Castro, C. J. and Ribagorda, A. (2009b). Videocaptchas. In *Proceedings of the 5th International Conference on Security and Protection of Information, Brno.*

Hernandez-Castro, C. J., Ribagorda, A., and Hernandez-Castro, J. C. (2011). Do current logic and semantic captchas make sense?

Hernandez-Castro, C. J., Ribagorda, A., and Saez, Y. (2009). Side-channel attack on labeling captchas. http://arxiv.org/abs/0908.1185.

Hernandez-Castro, C. J., Ribagorda Garnacho, A., and Saez, Y. (2010a). Side-channel attack on the humanauth captcha. In *Proceedings of the International Conference on Security and Cryptography.*

Hernandez-Castro, C. J., Stainton-Ellis, J. D., Ribagorda, A., and Hernandez-Castro, J. C. (2010b). Shortcomings in captcha design and implementation – captcha2, a commercial proposal. In *Proceedings of the Eighth International Network Conference (INC2010).*

Hoque, M. E., Russomanno, D. J., and Yeasin, M. (2006). 2d captchas from 3d models. In *Proceedings of the IEEE SoutheastCon.*

Knight, W. (2007). Newscientist entry on the qrbgs captcha. Retrieved on the 14th of August, 2010.

Mitra, N. J., Chu, H.-K., Lee, T.-Y., Wolf, L., Yeshurun, H., and Cohen-Or, D. (2009). Emerging images. *ACM Transactions on Graphics*, 28(5).

Naor, M. (1996). Verification of a human in the loop or identification via the turing test.

Rusu, A. and Govindaraju, V. (2004). Handwritten captcha – using the difference in the abilities of humans and machines in reading handwritten words. *Frontiers in Handwriting Recognition, International Workshop on*, 0:226–231.

Stevanovic, R., Topic, G., Skala, K., Stipcevic, M., and Rogina, B. (2008). Quantum random bit generator service for monte carlo and other stochastic simulations. In Heidelberg, S.-V. B., editor, *Lecture Notes in Computer Science*, volume 4818, pages 508–515.

Vaughan-Nichols, S. J. (2008). Techworld.com article on fallen captchas. Retrieved on the 14th of August, 2010.

von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In Press, A., editor, *CHI '04: Proceedings of the 2004 conference on Human factors in computing systems*, pages 319–326. ACM Press.

Warner, O. (2009). Kittenauth. http://www.thepcspy.com/kittenauth.

Wilkins, J. (2010). Strong captcha guidelines. http://bitland.net/captcha.pdf.

Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., and Cai, K. (2010). Attacks and design of image recognition captchas. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 187–200, New York, NY, USA. ACM.