

available at www.sciencedirect.comjournal homepage: www.elsevier.com/locate/cose
**Computers
&
Security**


Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study

Carlos Javier Hernandez-Castro*, Arturo Ribagorda

Security Group, Department of Computer Science, Carlos III University, Avd Universidad 30, 28911 Leganes, Madrid, Spain

ARTICLE INFO

Article history:

Received 11 March 2009

Received in revised form

1 June 2009

Accepted 27 June 2009

Keywords:

CAPTCHA

QRBGs

Math

Design flaw

Implementation flaw

Design-pitfall

Side-channel

OCR

ABSTRACT

We present a black-box attack against an already deployed CAPTCHA that aims to protect a free service delivered using the Internet. This CAPTCHA, referred to as “Math CAPTCHA” or “QRBGs CAPTCHA”, requests the user to solve a mathematical problem in order to prove human. We study significant problems both in its design and its implementation, and how those flaws can be used to completely solve this CAPTCHA using a low-cost attack. This attack requires no development in Artificial Intelligence or automatic character recognition, the intended path, thus becoming a side-channel attack, based on the previously mentioned CAPTCHAs flaws. We relate these flaws to common flaws found in other CAPTCHA proposals. We conclude with some tips for enhancing this CAPTCHA that can be considered as general guidelines.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

The last decade has seen increasing interest in abusing some of the services provided by the Internet, mainly for economical reasons. There has been misuse of on-line services: e-mail account creation (for spam sending and phishing), automatic anonymous posting (Wikipedia, blogs comments, news sites, job listing sites, etc.) for adding links for commercial promotion, or for harassment or vandalism, abuse of remote voting mechanisms (Hernandez-Castro et al., 2002), automatic site wandering for resource consumption, automatic anonymous abuse of on-line games or chat rooms for commercial promotion, etc. There are plenty of sound economical reasons to abuse services provided through the Internet.

The main trend to prevent this automatic abuse has been to develop the ability to tell humans and computers apart – remotely and through an untrustworthy channel. Many tests – generically called CAPTCHAs¹ or HIPs² – developed with that aim. These tests rely on capacities inherent to the human mind but supposedly difficult to mimic for computers. That is, problems traditionally hard to solve with computers (as problems that still remain wide open for Artificial Intelligence researchers).

1.1. Brief history of CAPTCHAs

Moni Naor (Naor, 1996) is the first person that mentioned some theoretical methods for telling apart computers from

* Corresponding author. Tel.: +34017508327.

E-mail addresses: chernand@inf.uc3m.es (C.J. Hernandez-Castro), arturo@inf.uc3m.es (A. Ribagorda).

¹ Completely Automated Public Turing test to tell Computers and Humans Apart.

² Human Interactive Proof.

0167-4048/\$ – see front matter © 2009 Elsevier Ltd. All rights reserved.

doi:10.1016/j.cose.2009.06.006



Fig. 1 – The EZ-Gimpy CAPTCHA.

humans remotely. He did it looking for ways to prevent the abuse of web services.

The first known use of a CAPTCHA test was done in the Alta-Vista web-search engine. Early in 1997 they faced the problem of the automatic submission of URLs to their search engine. This free “add-URL” service broadens its search coverage, but some users were abusing it by automating the submission of large numbers of URLs, in an effort to manipulate AltaVista’s importance ranking algorithms (that same kind of attacks have been known against other search engines, like Google).

Andrei Broder at AltaVista and his colleagues developed a filter. Their method is to generate an image of printed text randomly so that machine character recognition systems (OCR) cannot read it but humans still can. Five years later, Broder claimed that the system had been in use for “over a year” and had reduced the number of “spam add-URL” by “over 95%”. A U.S. patent was issued in April 2001 (US Patent no).

In 2000, Udi Manber of Yahoo! described their “chat room problem” to researchers at Carnegie-Mellon University: ‘bots’ (automatic computer scripts – a kind of program) were joining on-line chat rooms and pointing the chat-room users to advertising sites. They wanted to find a way to prevent ‘bots’ from joining the chat rooms.

Professors Manual Blum, Luis A. von Ahn, and John Langford, from Carnegie-Mellon University, described some desirable properties of such a test:

1. The test’s challenges can be automatically generated and graded, that is, a computer program should be able to







In order to prove to us you are not a robot, select the three hot people:		
 meet me	 meet me	 meet me
 meet me	 meet me	 meet me
 meet me	 meet me	 meet me
<div>switch to men</div> <div>hotcaptcha by frozenbear</div>		
<div>Submit Proof I am Human</div>		

Fig. 2 – The HotCAPTCHA.

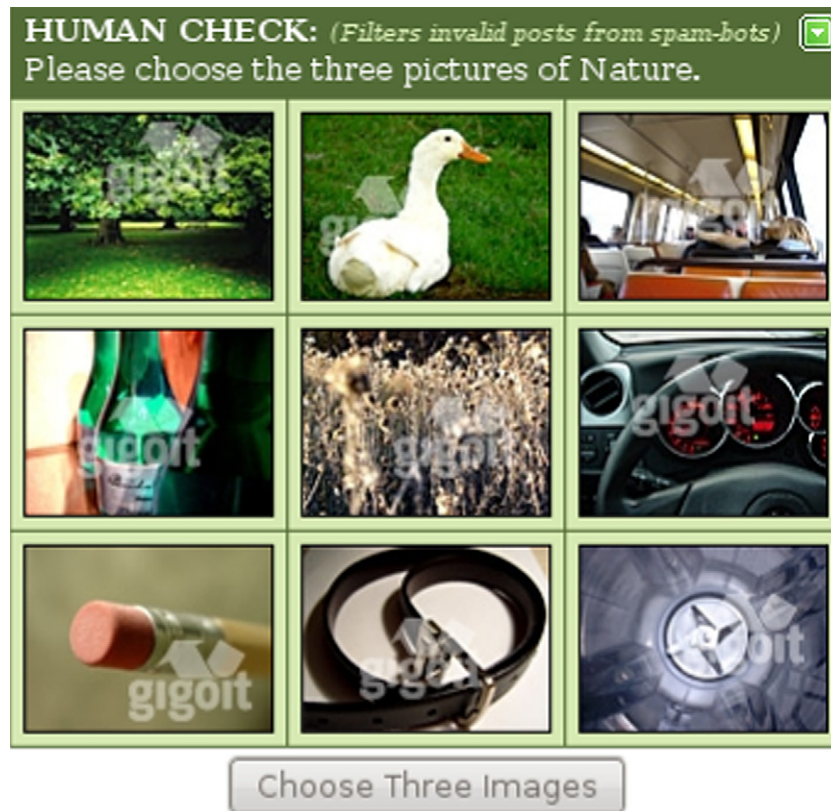


Fig. 3 – The HumanAuth CAPTCHA.



Fig. 4 – The ASIRRA CAPTCHA.

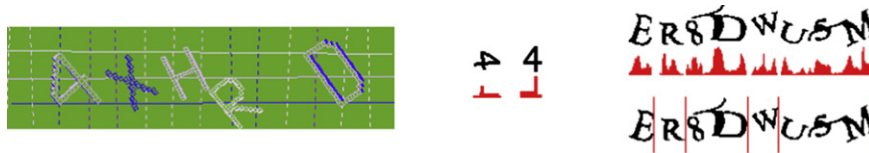


Fig. 5 – The phpBB3 CAPTCHA, letter de-rotation, and vert.-histogram for segmentation.

generate the challenges and mark the answers as qualifying or not.

2. The test can be taken quickly and easily by human users.
3. The test will accept virtually all human users with high reliability.
4. Low false negatives (“insults”): the test will reject very few human users.
5. Low false positives (“frauds”): the test will reject virtually all machine users.
6. The test should resist automatic attack for many years even as technology advances.

The Carnegie-Mellon University team developed a ‘hard’ GIMPY CAPTCHA which picked English words at random and rendered them as images of printed text under a wide variety of shape deformations and image distortions, including the images of different words overlapping. The user was asked to transcribe some minimum number of those words correctly. A

simplified version of GIMPY (EZ GIMPY, Fig. 1), using only one word-image at a time, was installed by Yahoo!. The term CAPTCHA (for Completely Automated Turing Test To Tell Computers and Humans Apart) was coined in 2000 by this CMU team (Ahn et al., 2003).

1.2. Types of CAPTCHAs

Text-based CAPTCHAs are popular because with as little as 5 characters (case-insensitive letters and digits) you have $36^5 \approx 60$ million possible combinations. However, even after graphic distortion and degradation, some approaches have been able to “read” them and thus solve the test automatically around 92% of the time (Mori and Malik, 2003). More so when it is possible to divide the graphic into its constituent letters. Some approaches have focussed on making this division harder, typically at the expense of making it also harder to the human challenger.

Quantum Random Bit Generator Service

News

- [2008-09-22 T 13:12+01] Due to regular server maintenance QRBG service was down for an hour. Now it's back online. Updated. :-)
- [2008-03-14 T 19:30+01] Mr. Javier Ruiz from the Escuela Técnica Superior de Ingeniería Informática de la Universidad de la Laguna en Tenerife organized a **Pi Day** celebration today at his university. Here is a short [blog post](#) about it (in Spanish). In short, they played approx. Pi minutes of number Pi, with digits downloaded from our service. :-)
- Happy Pi Day everyone! :-)**
- [2008-01-14 T 20:00+01] The service is up again! Unfortunately, the QRBG device we had died last week. Until we buy new a generator, the service will be serving you from its 1 TiB disk cache. For that reason, we're starting to (temporarily) enforce per-user download quota of 1 GiB per day.
- [2008-01-08 T 12:30+01] The service will be down for some time, due to technical difficulties with the QRBG hardware. We'll try to resolve these problems as soon as possible; until then, we apologize for the inconvenience...
- [2007-11-22] The service (including the web) will be down in period: [2007-11-26T14:00+01, 2007-11-26T20:00+01], due to server maintenance.
- [2007-10-15] I'm running some Monte Carlo simulations that require over 1 TiB of random data. So, I apologize for any slow-down you may experience. Simulations will take a couple of days. Enjoy the quantum randomness! :-)
- [2007-08-22] For the last several days the QRBG service/server was unavailable, due to a disk failure caused by a heavy thunderstorm last week. The service is now back online. We managed to recover all data, including the users database, so you can continue using the service as usual. Thank you for your understanding and patience.
- [2007-07-20] "The crowd went wild" :-)
- Yesterday, over 1 million hits. Over 60,000 visits. Over 2800 registered users. Over 5GiB of random data downloaded. Over 20,000 user requests served...
- Keep on coming. :-)
- [2007-07-19] I was thrilled to see that the QRBGS project has been SlashDotted. And Dugg (or however you spell it :D). Unfortunately, the original text that these sites refer to has been written by the PR department, misinterpreting some facts in the process, and forgetting to run the final version by me. In spite of that, the article inspired quite a bit of interest. I would like to apologise to all the readers who are now busily generating their pseudo-random number databases by rolling dice. :-)

Introduction

The work on QRBG Service has been motivated by scientific necessity (primarily of local scientific community) of running various simulations (in cluster/Grid environments), whose results are often greatly affected by quality (distribution, nondeterminism, entropy, etc.) of used random numbers. Since **true random numbers** are impossible to generate with a finite state machine (such as today's computers), scientists are forced to either use specialized expensive hardware number generators, or, more frequently, to content themselves with suboptimal solutions (like

Login

Username:

Password:

☒ Remember me on this computer

Don't have a login? [Register!](#)

Server status

Device: connected
Server: running

- uptime: 5d 22h 29m 28s
- total clients (requests): 28678
 - now being served: 0
 - successfully served: 28659
 - failed to serve: 19
- total data requested: 3.52 GiB
- total data served: 3.52 GiB

Available from: <http://random.irb.hr/8000/>

Cumulative server statistics

	Connections	Data
Served	2,724,328	1.25 TiB
Rejected	5,431	5.81 TiB
> Quota limit	775	5.96 GiB
> Auth failed	4,656	5.80 TiB
Σ	2,729,759	7.07 TiB

Total users registered: 9,358

Since: Thursday, June 14, 2007 18:08:31.
*Cumulative statistics is updated once per hour.

Press Kit

Fig. 6 – QRGBS homepage.

Many text-based CAPTCHAs were known to be broken during these earlier years, so the general strength of them is an area of increasing concern. That is the main reason why researchers have begun looking for other types of CAPTCHAs. General vision seems to be a harder problem than character recognition, so more designs have focused on using pictures instead – even though most of those CAPTCHAs do not really rely on a “general vision problem”, but in a downsized version, consisting in categorizing images.

Chew and Tygar (2004) were the first to use a set of labeled images to produce CAPTCHA challenges. For that purpose, they used the label associated with images in Google Image Search. This technique is not well suited for CAPTCHAs, as Google relates a picture to its description and its surroundings, so the word ‘bicycle’ could refer to a bicycle or to a music band name.

Ahn and Dabbish (2004) proposed a new way to label images by embedding the task as a game, called the “ESP game”, that gives points to teams who pick the same label for a random image. Those images constitute the PIX CAPTCHA database. However, it has a fixed number of object classes (70) and the image database seems not large enough.

The site HotCaptcha.com (Fig. 2) proposed a way to use a large-scale human labeled database provided by the HotOrNot.com web-site, a site that invites users to post photos of themselves and rate others in a numerical scale of “hotness”. This proposal is no longer active, and as of Jan-2009 the site HotCaptcha.com is down.

Warner created the idea of using photos of kittens to tell computers and humans apart. KittenAuth features nine pictures of cute little animals, only three of which are feline. The problem with this system is that 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. 3), asks the user to distinguish between pictures depicting Nature or a human-generated thing (like a clock, or a museum, for instance). The CAPTCHA is also available to blind people, as they can switch to text-based descriptions of the pictures, and still be able to answer the CAPTCHA correctly. As with KittenAuth, the main problem with this CAPTCHA is the picture database 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.

ASIRRA (Elson et al., 2007) (Fig. 4) uses a similar approach but using a giant database of more than 3 million photos from Petfinder.com, a web-site devoted to finding homes for homeless pets, with a daily addition of around 10,000 more pics. Asirra displays 12 images from the database (mostly composed of dogs or cats images) and asks the user to select the cats in it.

There are other research papers proposing different image-based CAPTCHAs, including synthetic 3D computer generated images, pictures with labeled regions, etc.

1.3. Motivation

Today researchers do not fully understand yet if the problems chosen for the design of current

CAPTCHAs are really hard enough. Even if this is the case, there are two questions that remain to be solved:

1. We do not have a clear methodology to check if a particular CAPTCHA design and implementation is open to a side-channel attack. A side-channel attack is one that solves the CAPTCHA but not the AI problem it is based on, therefore not improving the state of the art on AI. It is named side-channel, thus, as it solves the problem using a method that does not follow the intended attacking path.
2. We lack a method to check if the particular problem that a CAPTCHA challenge poses is as hard as the underlying AI problem – that is, if the AI hardness has been fully transmitted to the CAPTCHA design.

CAPTCHA recent history is filled of mistakes, many of which have made them easier to break. There are many examples, we will present just three. In the phpBB3 CAPTCHA (Fig. 5), their authors try to avoid identification of single letters. For that, they depict them in a way that a normal flood-fill will fail. This is easy to circumvent: we can flood-fill and count the size of the areas. The smaller ones are going to be part of letters, thus, letter identification is still easy. In many text-based CAPTCHAs, letter rotation is a crucial part to try to prevent letter identification. But it is easy to de-rotate a letter, because most characters have vertical lines, and some, horizontal lines, so we just need to rotate the letter in small angles and create a histogram of non-background pixels per column (possibly helped by other with pixels per line). The histograms with biggest spikes are the ones with biggest chances of corresponding to the original characters. Also common in many text CAPTCHA designs is to rotate and arrange characters in a way that it is harder for programs to segment the characters (separate them individually). Vertical histograms (of non-background pixel count) can be easily used to solve this problem, too.

This is just an example of typical known flaws. In the recent work of our group, we have focused on analyzing the security of different CAPTCHA schemes. By analyzing those CAPTCHA proposals, we can find and classify different flaws in their design and implementation, and start proposing an initial methodology for avoiding such pitfalls. In our recent work, we have analyzed some CAPTCHAs, and found that they have related flaws (further on Section 6). We consider this particular case, the Math CAPTCHA, (Figs. 6 and 7), to be a very good example of those flaws, and also shows how some of those flaws can contribute together to make a CAPTCHA vulnerable to a side-channel attack. In this way, we find that the lessons learned from breaking this CAPTCHA can be used for breaking others and, more importantly, for avoiding common mistakes when designing new ones.

We think that it is then very interesting to try to break current CAPTCHAs and find pitfalls in their design, to advance the state-of-the-art of CAPTCHA design,

Table 1 – Most visited web-sites.^{4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}

web-site	subject	challenge	challenge	challenge
MySpace ⁴	social			
Facebook ⁵	social			
Microsoft Live/Passport ⁶	email social			
Ebay ⁷	thrift			
Megaupload ⁸	file exchange			
AOL ⁹	email ISP			
Hulu ¹⁰	video			
Tagged ¹¹	social			
Friendster ¹²	social			
Youtube ¹³	video			
Orkut ¹⁴	social			
Fotolog ¹⁵	pic. share			
Skyrock ¹⁶	social			
Wikipedia ¹⁷	culture			
Blogger ¹⁸	blogs			
Yahoo! ¹⁹	email games			
Vkontakte ²⁰	social			
Photobucket ²¹	pic. share			
Uol ²²	email ISP			
Conduit ²³	groups			
Dailymotion ²⁴	video			
Globo ²⁵	email news			
Rediff ²⁶	news storage			
Mininova ²⁷	torrents			
LiveJournal ²⁸	blog			
MetroFlog ²⁹	pics. social			
Netlog ³⁰	social			

and get to a point where well known and tested assumptions and methodologies give base for more secure CAPTCHAs.

1.4. CAPTCHAs in use

One of the characteristics of the CAPTCHA we analyze in this paper is that it has been designed and implemented by people who are not Security experts on CAPTCHAs. Their designers are the same people offering a computing service using the Internet. The design and implementation of CAPTCHAs by non experts is typically more weak, because they are not aware of the current methods of CAPTCHA solving, and of the typical flaws present in CAPTCHA design.

It is reasonable to wonder if this in-house development scenario is a very uncommon one, or maybe common only among small, scarcely visited web-sites. In a survey conducted by our team, we searched the 100 most visited sites³. After ruling out sites that offer no registration, blogging or updating of any kind (9), sites in languages our team does not understand (11 in Chinese, 3 in Japanese, 4 in Russian, one in Polish), and local repetitions of the same sites, it remained 23 services that used e-mail for registration and 27 that used CAPTCHAs (Table 1).

All of those CAPTCHAs are text based, thus relying (as our Math CAPTCHA) in the impossibility of common OCR programs to read them. With the exception of Google services and 4 others that use reCAPTCHA³¹, the rest (Ebay, AOL, Megaupload, Friendster, Fotolog, etc.) have been developed in

house, or by groups of programmers with no previous experience in Computer Security, and offered with no prior Security analysis. All those developments can be considered weak to very weak by current standards: some of them (Fotolog, Uol, Conduit, MetroFlog, etc.) are even vulnerable to a repetition and mean calculation attack (Wieser).

As we see, the scenario exemplified by our case is much more frequent than what could be thought.

1.5. Organization

The rest of the paper is organized as follows: In the next section, we introduce the QRBGS CAPTCHA in greater detail. After this, in Section 3 we describe the results of the analysis performed in this CAPTCHA, including the design and implementation flaws discovered. Then, in Section 4 we create an analysis tool (and attack tool) to quantify how severely the flaws affect the capacity of the QRBGS CAPTCHA to tell humans and computers apart. In Section 4.2 we explain how to use that information to launch very simple and effective attacks against this CAPTCHA, and more difficult versions of this CAPTCHA.

In Section 5 we discuss some possible improvements, some ways to harden this CAPTCHA, that can be applied in general for every CAPTCHA. In Section 6, we relate the flaws discovered in this CAPTCHA to other current CAPTCHA designs. Finally, in Section 7 we extract some conclusions and propose future research lines.

2. The “Math CAPTCHA”

The Center for Informatics and Computing of the Ruder Bošković Institute (Zagreb, Croatia) provides a “Quantum Random Bit Generator Service” delivered by its web address.³² In Stevanovic et al. (2008), they describe the functioning of their random number generation service. It relies on a stand-alone hardware number generator driven by photonic emission in semiconductors. Once registered at their web-page, you can use one of the many software client libraries to connect to their server and download the numbers provided by them when needed.

Their service is free, they only require to sign up for access. In order to protect this service from vandalism or denial of service, they have developed an innovative CAPTCHA that has been at the center of some media attention (Techworld; NewScientist). There are other designs and implementations of mathematical-problem-based CAPTCHAs around the web, but they are much simpler than this. Most of them rely on basic arithmetic operations (+, *, –), and some even do not need OCRs, as they are implemented using regular characters.

This media attention is at least partially due to the originality of their proposal: to solve or compute a not-so-simple math expression to prove that you are human (and good enough in Math). As the main user of their service is expected to have some level of mathematical knowledge, they were not very worried of them not being able to solve the expressions. But, just in case, they provide different

³ According to Alexa http://www.alexa.com/site/ds/top_500 and Netcraft <http://toolbar.netcraft.com/stats/topsites>.

⁴ <http://www.recaptcha.com>.

⁵ <http://signups.myspace.com/index.cfm>.

⁶ <http://www.facebook.com/>.

⁷ <https://signup.live.com/signup.aspx>.

⁸ <https://scgi.ebay.com/ws/eBayISAPI.dll>.

⁹ <http://www.megaupload.com/?c=signup>.

¹⁰ <https://new.aol.com/freeaolweb/>.

¹¹ <https://secure.hulu.com/signup>.

¹² <http://www.tagged.com/>.

¹³ <http://www.friendster.com/>.

¹⁴ <http://www.youtube.com/signup>.

¹⁵ <https://www.google.com/accounts/NewAccount>.

¹⁶ <http://account.fotolog.com/register>.

¹⁷ <http://www.skyrock.com/m/account/subscribe.php>.

¹⁸ <http://en.wikipedia.org/w/index.php?title=Special:UserLogin&type=signup&returnto=MainPage>

¹⁹ <https://www.google.com/accounts/NewAccount?service=blogger>.

²⁰ <https://edit.europe.yahoo.com/registration>.

²¹ <http://vkontakte.ru/reg0>.

²² <http://register.photobucket.com/step2>.

²³ <https://visitante.cadastro.uol.com.br/index-bol.html>.

²⁴ <http://accounts.conduit.com/Wizard/>.

²⁵ <http://www.dailymotion.com/register>.

²⁶ <http://cadastrofree.globo.com/cadastro/1948>.

²⁷ <http://register.rediff.com/register/register.php>.

²⁸ <http://www.mininova.org/register>.

²⁹ <https://www.livejournal.com/create.bml>.

³⁰ <http://www.metroflog.com/signup.php>.

³¹ <http://www.recaptcha.com>.

³² <http://random.irb.hr/>

Quantum Random Bit Generator Service: Sign up

Login credentials

Username:
from 4 to 100 alphanumeric characters or symbols

Password:
from 5 to 100 ascii alphanumeric / punctuation characters

Personal information

Name:
first middle last

E-mail:
shall be kept confidential

Country:
where do you live in?

Affiliation

Organization:
preferably

Intended use

Use:
optional

Qualifying question

Just to prove you are a human, please answer the following math challenge.

Q: Calculate:

$$\frac{\partial}{\partial x} \left[5 \cdot \sin \left(7 \cdot x + \frac{\pi}{2} \right) + 2 \cdot \cos \left(5 \cdot x + \frac{\pi}{2} \right) \right] \Big|_{x=0}$$

A:
mandatory

Note: If you do not know the answer to this question, reload the page and you'll (probably) get another, easier, question.

Terms of use

By registering, you accept:

Fig. 7 – Initial screen of the QRBGS CAPTCHA.

types of mathematical challenges. The easiest, being a simple arithmetic operation. They allow the user to reload the page and thus change the challenge to find an easier one.

In Table 2 we can see some examples of challenges that we may find while trying to log in to the QRBG Service.

When we fail the math challenge, we are shown a web-page (Fig. 8), and we can go back to the log-in page (reloading it) and thus be provided with a different math challenge. The only nuisance is that we have to fill in the web-form again, but there is no limitation on the number of trials per IP, or per browser (using cookies), etc.

Table 2 – Examples of challenges.

Calculate $\partial/\partial x[\sin(\pi/2)]|_{x=0}$
 Find the smallest
 zero of $p(x) = (x+1)(x-0)(x-1)$
 Find the smallest
 zero of $p(x) = (x-5)(x-4)(x+6)(x+2)(x-5)$
 Find the smallest
 zero of $p(x) = (x^2 - 4x + 4)$
 Solve $-2 + 4 \times 2 + 5 \times 3 = ?$

When we finally succeed in passing the math test we are provided with a different web-page (Fig. 9) from where we can go on, log in, etc.

Thus, we do not need a valid e-mail to know if the CAPTCHA has been successfully solved or not, we can learn about the solution right away just by watching (for a machine, by parsing) the HTML page sent as a response to the web-form post.

“Playing” with this CAPTCHA we noted some important design flaws. We wondered if the QRBGS CAPTCHA designers already knew about them or they just overlooked them. We also wondered how successful they would render a low-cost attack based upon them, an attack not encompassing any other special analysis or complex algorithm. Just using standard tools out-of-the-box, with no further training and no customizing at all, and some simple strategies.

2.1. The black-box approach

We have studied this CAPTCHA in a black-box way. That means that, during its analysis, we have not contacted its authors for more information. We have done the full analysis counting only

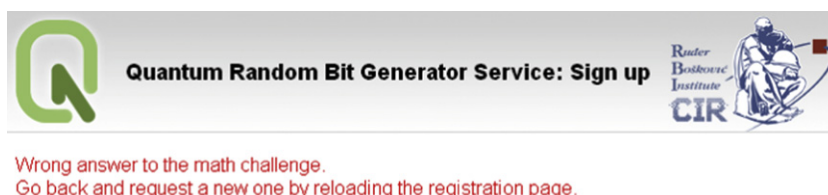


Fig. 8 – Failure!.

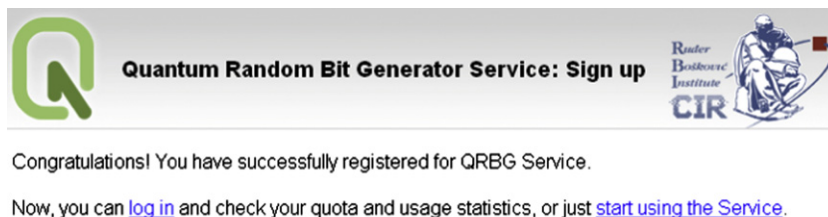


Fig. 9 – Success!.

with the least information possible regarding this CAPTCHA internals. The reason for that is that we wanted to reproduce an attack done by an anonymous attacker, that will learn only what anyone can learn by visiting this CAPTCHA web-site.

If we had used some “Social Engineering” methods, it would have been easier to attack the CAPTCHA, as we would have had some information before we find it out for ourselves, and maybe some more information to further improve the attack. Our aim was to attack this CAPTCHA in the way anyone wanting to leave little trace (just an IP address) would have done, with the least information possible, and using only the flaws that we could discover.

3. Analysis

Before the attack, we analyzed the CAPTCHA. After some digging it is easy to spot that the CAPTCHA always proposes one of four kinds of different challenges:

1. Solving a derivative on the variable x of the form $\partial/\partial x$, in a formula with trigonometric functions, one-digit coefficients, etc.
2. Telling the smallest zero of a polynomial with the form $p(x) = (x - a)(x - b)(x - c)(x - d)(x - e)\dots$
3. Telling the smallest zero of a polynomial with the form $p(x) = \dots + dx^3 + cx^2 + bx + a$

4. Solving a simple arithmetic operation

The probability distribution of the type of challenge is easily spotted as non uniform (we will learn more about it in the attack/analysis results section).

The authors of the CAPTCHA seem concerned about the mathematical abilities of the users of their service. Thus, they provide the possibility of reloading the page and being challenged with a, maybe, easier challenge – with no limits at all. That means there are no limits on the number of reloads they admit from the same IP address (we have tried up to 25,000 without problems) or the same browser (they use cookies, but do not use them to control the number of reloads).

The authors of the CAPTCHA also seem to be worried about float number precision in their results. Thus, if you examine some of the CAPTCHAs, you realize that a great amount of the answers – if not all – are integers.

3.1. General OCR results

The authors of the CAPTCHA try to prevent OCR scanning of their images. They can be quite confident with the formulas that use derivatives, as currently there seems to be no commercial or free OCR capable of automatically translates them into the corresponding formula without lots of errors.

For the other three cases (the two kinds of polynomials and the arithmetic challenge), they rely on using a very small font

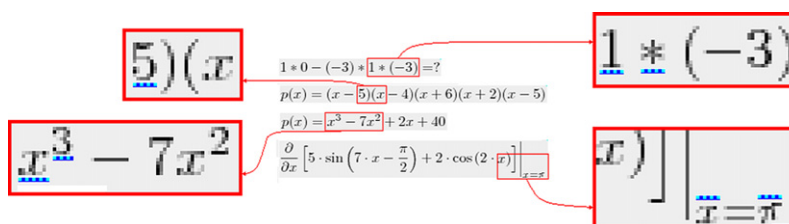


Fig. 10 – Different challenges -some of their parts parts zoomed.

Table 3 – Different types of challenges showed.

Problem type	Challenges	OCR result
$\partial/\partial x$	$\partial/\partial x[3 \sin(7x + \pi/2)] _{x=2\pi}$	$A[3-.m(7-I+gmph$
$\partial/\partial x$	$\partial/\partial x[7+4 \cos(4x + \pi/2)] _{x=\pi}$	$A[7+4-wA \gg (.1-A \gg +g)]LW$
$\partial/\partial x$	$\partial/\partial x[\sin(\pi/2)] _{x=0}$	$; [(2)]L,$
$\partial/\partial x$	$\partial/\partial x[\sin(x - \pi/2)] _{x=0}$	$\langle -A; \rangle L,$
$\partial/\partial x$	$\partial/\partial x[5 \sin(7x - \pi/2) + 2 \cos(2x)] _{x=\pi}$	$A[s-.m(7-;A > g) + 2 - A > o < oA > (2 - JM$
$(x-a)^*$	$p(x) = (x+1)(x-0)(x-1)$	$p(:) (: r+1); (rA''0); (rA''1)$
$(x-a)^*$	$p(x) = (x-4)(x-1)(x-4)$	$1A > o < (r); (r 4 > (r 1)(r 4)$
$(x-a)^*$	$p(x) = (x-5)(x+5)(x-2)$	$p(:) (: rA''5); (r+5); (rA''2)$
$(x-a)^*$	$p(x) = (x-2)(x-0)(x+5)$	$p(:) (: rA''2); (rA''0); (r+5)$
$(x-a)^*$	$p(x) = (x-5)(x-4)(x+6)(x+2)(x-5)$	$p(:) (: rA''5); (rA''4); (r+6); (r+2); (rA''5)$
$ax^3 + bx^2$	$p(x) = x^2 + 5x + 6$	$pp); H + sm + 6$
$ax^3 + bx^2$	$p(x) = x^3 - 7x^2 + 2x + 40$	$pp); HA \gg nz + 21 + 40$
$ax^3 + bx^2$	$p(x) = x^2 + 4x + 4$	$p(:) : HA''4 : +4$
$ax^3 + bx^2$	$p(x) = x^2 - 2x - 3$	$141); ; * + 21A''1$
$ax^3 + bx^2$	$p(x) = x^2 + 5x + 4$	$pp); H + sm + 4$
arithm. calc	$7 - 4 + 7 = ?$	$7A''A1 + 7; !$
arithm. calc	$-2 + 4 \times 2 + 5 \times 3 = ?$	$A''2 + 4A = 2 + 5 > v3: A!$
arithm. calc	$1 \times 0 - 3(-3) \times 1 \times (-3) = ?$	$1_0A''(3) _ i _ (r3); ?$
arithm. calc	$0 - 1 + 3 = ?$	$0A''1 + 3; !$
arithm. calc	$1 - 4 + 1 \times 2 \times 3 = ?$	$1A''4 + 1 > v2 > v3: ''!$

size and aliasing (Fig. 10), thus making it very difficult for commercial OCRs to read them as text. OCR programs in general are quite sensitive to scanned resolution – scan at too high or too low a resolution and you would not get a good result. For many OCRs, around 300 DPI is usually near the right resolution. The resolution that the images created by the QRBGS CAPTCHA provide is far less, making much more difficult a normal OCR analysis.

Even though, we wanted to know how good a general purpose OCR program would perform when trying to read this CAPTCHA challenge images. We have created a Python program able to capture the cookies sent from the QRBGS CAPTCHA site, download the image containing the mathematical challenge (a GIF image), analyze it, send back a complete web-form including an answer to the mathematical challenge, and parse the result as a failure or a success.

For the image analysis, we have used an image analysis and conversion library, ImageMagick.³³ It has a *convert* program, which allows us to convert the downloaded images from GIF to TIFF format in 8 bits per pixel, as a deeper gray-scale resolution is not admitted by some OCR programs.

We have tried different open-source OCR programs and finally relied in the one which is considered to be among the best performing – not only for open source programs, but also for commercial – the Tesseract OCR program.³⁴ Tesseract admits further training, so the OCR scanning results could possibly be better than ours (even though provided with such low-res images for the challenges), but we wanted to try a very low-cost attack to the QRBGS CAPTCHA to test how the design and implementation flaws really affect the ability to tell humans and computers apart.

The results we obtained from the OCR program were the expected ones, but for the $(x-a)^*$... polynomials they were not so bad (Table 3).

3.2. Tailored OCR strategies

Of course, better approaches involving more work can be tested. Better than using standard OCRs techniques, we can use different pattern-matching and/or shape-recognition algorithms to tell the different characters and mathematical symbols used in the formulas.

Character-based CAPTCHAs difficulties typically reside in the problem of segmentation (Chellapilla et al., 2005). In our case, this problem is non existent, as there is no overlapping of characters. Also, individual characters are not transformed from their originals. The only problem is that some of them are rendered with a very low resolution (DPI), which makes them difficult for OCRs. Another problem is that formulas are also rendered using symbols, and those symbols are relatively positioned to each other in a non trivial way (as normal text typically is).

We can use different well known methods to find the individual characters and symbols constituents of the formula image we are analyzing. Two well known ones are shape-recognition algorithms, and pixel (pattern) matching. Shape-recognition algorithms typically work with shapes that are depicted using a big number of pixels, so it is easy to tell the characteristic parts of the shape (edges, joints, angles, etc.). In our case, we can use it for big-enough symbols, as brackets and parentheses, i.e., those which use enough non-background pixels in their depiction. The little x symbol may not be a good candidate for these algorithms.

On the other side, pixel (pattern) matching (using Hamming distances) has been successfully applied before to detect similar

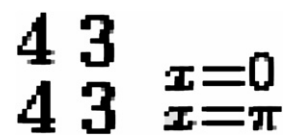


Fig. 11 – Pixel matching has to be adapted to gray-scale.

³³ <http://www.imagemagick.org>.

³⁴ <http://code.google.com/p/tesseract-ocr/>.

$$\frac{\partial}{\partial x} \left[3 \cdot \sin \left(4 \cdot x + \frac{\pi}{2} \right) \right] \Big|_{x=0} \quad [()]$$

$$\frac{\partial}{\partial x} [4 \cdot \sin (3 \cdot x)] \Big|_{x=\pi} \quad [()]$$

Fig. 12 – Different sizes of the same symbols.

characters. The only problem is, the smaller the character we are looking for, the longer the process will take. A possible solution would be to search before for bigger symbols. Also, as characters are depicted using aliasing, we may convert them to black & white before pixel matching, or apply an algorithm that is prepared to deal with gray-scale distances (Fig. 11).

We can solve the problem of variably-sized symbols (Fig. 12), such as symbols wrapping expressions that can be of a different height. Pixel matching will not work well with those if sizes are not fixed, but shape recognition can, given they are depicted with enough resolution.

Once basic characters and symbols are detected, we should relate them, by size and position, so as to know what is “inside” each symbol range (Fig. 13).

If in case of doubt between two different possibilities, it is possible to build the graphical version of both formulas and match them against the image of the challenge. This can be done incrementally, to find the whole formula from its constituent parts, somewhat reverse-engineering the formula construction process.

Of course, these attack approaches proposed here are much more costly. If proved successful, they would only show that one of the design decisions of this CAPTCHA (showing the math formulas right away without any distortion, just low-res graphics) is not well suited to prevent automatic scanning.

We do not focus on these types of attacks in this paper, because they are attacks that follow the intended path. The main motivation of this paper is to show common flaws in CAPTCHA design and implementation, which are applicable also to many other CAPTCHAs, and how these flaws can collaborate to make a side-channel attack successful. Those attacks are generally much easier to perform, and much more efficient. Against the Math CAPTCHA scheme, this approach has an outstanding success ratio.

3.3. Design flaws

For the coming analysis, we consider a design characteristic one that is related to the basic decisions involving this CAPTCHA construction, which in our example are, basically, two. First, the AI problem chosen, which we assume is formula recognition (as formula solving can be done using standard mathematical

Table 4 – Algorithm for the discovery of the type of challenge.

If the height of the GIF image downloaded is:

1. 16: can we find the string “Find the least real zero of the polynomial:” in the HTML page downloaded?
 - (a) Yes: it is a “find the least zero of” polynomial $p(x) = (x - a)(x - b)(x - c)(x - d)(x - e) \dots$ -type challenge
 - (b) No: it is an arithmetic challenge
2. 41: it is a $\partial/\partial x$ -type challenge
3. 18: it is a “find the least zero of” polynomial $\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$ -type challenge
4. Else: it is an arithmetic challenge

software packages). Second, how randomly/uniformly distributed are the parameters (in this case: figures and functions used, distribution of the correct answers, etc.) chosen to create each particular challenge, studying how they influence challenge construction. For example, we consider the use of trigonometric functions, embedded into derivatives, to be a design decision: it affects many aspects of the CAPTCHA, i.e.: the correct answer distribution, the chances of an OCR attack, the difficulty of the CAPTCHA for human beings, etc. We consider implementation characteristics the ones related to more specific details, like how the challenge generator works. An implementation characteristic should be easily modifiable without basically affecting the rest of the CAPTCHA. Examples are: the number of challenges created (if they are not created on line), the way the challenge is transmitted and presented (in our case, as an image with a fixed set of predetermined heights, depending on the type of challenge), the way the solution is represented and sent back to the server, the typeface chosen, etc.

One example of a design flaw is the fact that the QRBGS CAPTCHA designers use one-digit figures in all their arithmetic operations, what makes it likely that the correct answers will be small integers.

The fore-mentioned fact, the fact that they use derivatives of trigonometric functions when x has a value typically equal to $\pi/2$ or π , and the fact that they do not expect many (if any) floats as answers (possibly in a measure to avoid precision problems), makes low-integers, and specially 0, possible candidates for successful blind answers.

The possibility of reloading the page to get an easier challenge—for a human—comes in this particular case with a major drawback: analyzing the web-pages and images wrapping the challenges, it is not difficult to see that it is, for a machine, easy to tell with which type of challenge we are facing.

We have two instruments that help us with that: the image size and the strings that appear in the web-page just before the challenge (Table 4).

Thus, for a machine it is possible to reload the page to arrive at an easier challenge!

Fig. 13 – Relating different characters and symbols.

3.4. Implementation flaws

While we were doing basic testing and analysis of the QRBGS CAPTCHA we discovered another flaw, this time not regarding its design, but an implementation decision: the repetition of challenges.

If the answer to the challenges could be floats, and we did not have any clues about answers, then the repetition of challenges would not be so bad.

But from the previously discussed design flaws, we know that there is a big chance that the answer to a challenge could be 0, or if not, a small integer.

Also, when a challenge is repeated, the GIF image containing the challenge is repeated exactly. Actually, it even uses the same filename. Therefore, it is simple to correlate “different” challenges to the same one.

All this made us disguise another kind of attack: if answering 0 fails, then let us answer a succession of integers that will run through N , starting with the smallest absolute values.

4. Attack

After the discussed analysis, we programmed a very simple attack to the QRBGS CAPTCHA.

Please keep in mind that we did not program the most successful attack, as we also wanted to use this script as an analysis tool. This analysis program, as an attack, is much more restrictive than the best possible attack, because we did

4.1. Attack results

The results of these three different strategies of answers we discussed above may vary according to these factors:

1. How probable is the answer 0
2. How many repetitions of challenges we have in that subset of problems
3. How well does the OCR-strategy perform (for the polynomial $p(x) = (x - a) \dots$ challenge)

In a set of 10,000 tests requested to the CAPTCHA web-site, of each kind of problem, we find:

Problem type	Different challenges	Total challenges
$\partial/\partial x$	394	5713
Polyn. $\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$	50	848
Polyn. of type $(x - a) \dots$	118	1493
Arithmetic calc	157	1946

In total, we find only 719 different challenges for 10,000 challenges requested to the QRBGS CAPTCHA web-site. You get the impression that the CAPTCHAs are based on a pre-built set of challenges, encompassing around 725–750 challenges, and picked-up randomly on each challenge request. As the original set of challenges is not uniformly distributed (there is not an equal number of challenges for every type) so is not the probability of appearance of each type of CAPTCHA:

Problem type	Different challenges	Solved	%	Total challenges	Solved	%
$\partial/\partial x$	394	227	57.61	5713	3196	55.94
Polyn. $\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$	50	49	98.00	848	592	69.81
Polyn. $(x - a) \dots$	118	109	92.37	1493	1391	93.16
Arithmetic calc	157	66	42.03	1946	507	26.05

not use page reloads to get an easier challenge (we discuss later the success of a very easy attack like that).

This analysis/attack does the following:

1. Classify the type of challenge we are facing
2. Depending on the different type of challenge:
 - (a) If it is to find the smallest zero of a polynomial of the form $(x - n_1)(x - n_2) \dots$, scan it using the OCR, enhance the result using heuristics, and answer with $-\max(n_i)$
 - (b) If it is another type of challenge
- i. If we already know the solution to the challenge because we already saw it and solved it properly, just repeat the same solution
- ii. If we do not know the solution
 - A. Send 0 (as the most probable answer)
 - B. If we already sent 0, then sent the next in the series $s(0) = 1$, $s(n) = s(n-1) \cdot (-1) + ((s(n-1) < 0) ? 0:1)$ (that is, 1, -1, 2, -2, 3, ...).

In Fig. 14 we learn that the first factor (how probable is the answer 0) is very dependant on the type of challenge we face, and in the case of the $\partial/\partial x$ challenge it is (as we hypothesized) very probable (43% of the 394 different $\partial/\partial x$ challenges we were showed, as can be also checked in Fig. 15), being also very probable in the other types – but not to the same extent.

After 10,000 requests and challenges, we learn that we have been shown less than 750 different challenges. For the repeated ones, we have tried different integral values and, for most of them, we have found the solution in this number of trials.

As we see, the ratio of solved problems is higher if we consider them as single problems with many trials (repetitions) – that is due to our strategy of trying out the series 1, -1, 2, -2, 3 ... Once one of this challenges is solved, it remains solved for the next time we face it, but until it is solved it accounts for failures.

It must be noted that, once this knowledge-base is built, which in our case only requires a few thousands trials, the

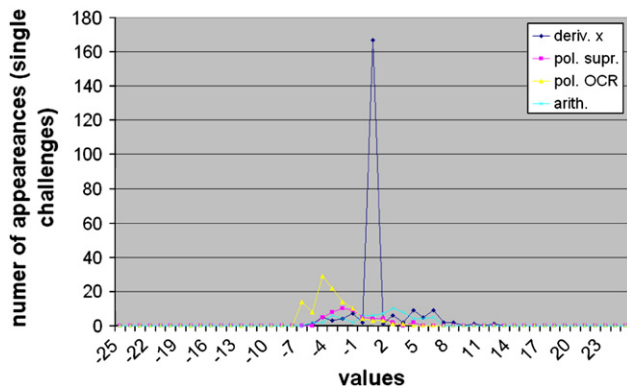


Fig. 14 – Histograms of solved challenges by value, for each different challenge type.

solving ratios are the first percentages stated in the 2nd table in Section 4.1, as the pool of challenges seems to be a static set. In this circumstance, the least success ratio we find is 42.03% (which can be improved with additional trials) and in general, we solve the CAPTCHA $451/719 = 62\%$ of the times – without the need to reload for an easier challenge!

The non-uniformity on 0 as a good answer explains the different results obtained by the different answering strategies, depending on the type of challenge.

In Fig. 16 we find how successful the two answering strategies evolve with time for the $\partial/\partial x$ challenge. As we see, the 0-strategy performs always with the same level of efficiency, which rounds up to 42%. The $s(n)$ -strategy performs better with time, in a clear growth curve that should reach a 100%, as the set of challenges is limited. For this Fig. 16 and for the rest of this kind that we are going to analyze, it should be noted that the ratio of successful answers to each of the challenges is (clearly) the sum of both values.

In Fig. 17 we see the efficiencies of the same two answering strategies, evolving with the number of challenges faced, for the $\dots + n_3 \cdot x^3 + n_2 \cdot x^2 + n_1 \cdot x + n_0$ challenge. As we see, the 0-

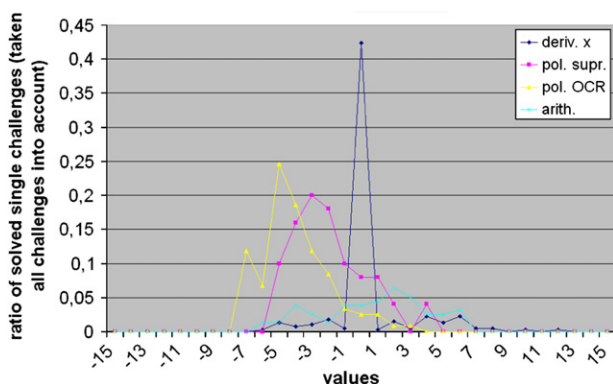


Fig. 15 – Histograms of solved challenges by value, for each different challenge type, percentage (compared to the number of challenges, not to the number of solved challenges), detail.

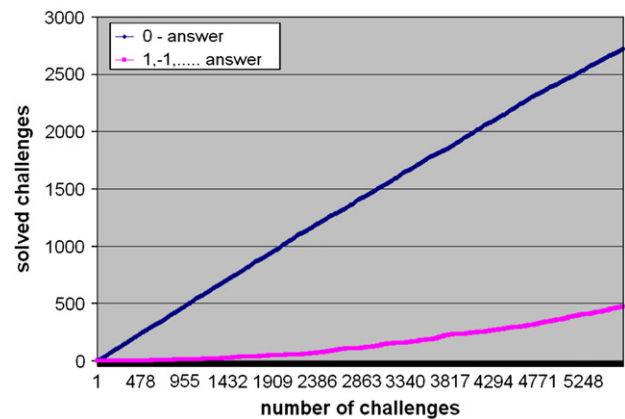


Fig. 16 – Success rate as a function of the number of challenges and strategy type (x-derivatives challenge).

strategy performs worse this time, and also with the same level of efficiency throughout all the testing. The $s(n)$ -strategy performs better with time, early beating the 0-strategy, and reaching almost a 100% success ratio at the end of the figure. This is due to the very limited number of different polynomials of this type found (50).

In Fig. 18 we see how the OCR + heuristics interpretation of the challenge performs. The efficiency rate is the same all the time, which is around 93%. We could improve this ratio by remembering the correct answers and trying the $s(n)$ -strategy with the incorrect ones, but we just wonder how a simple OCR + heuristics approach would perform with this particular type of challenge.

In Fig. 19 we see that the 0-strategy performs not so well, and also with the same level of efficiency throughout all the tests. The $s(n)$ -strategy performs better with time, not far from reaching 100% at the end of the figure, even though we need more challenges to solve them all.

In Fig. 20 we see all the results data merged together. The derivatives of the curves and lines are the success ratio, being a 1-value (45% slope) a 100% efficiency, here represented by

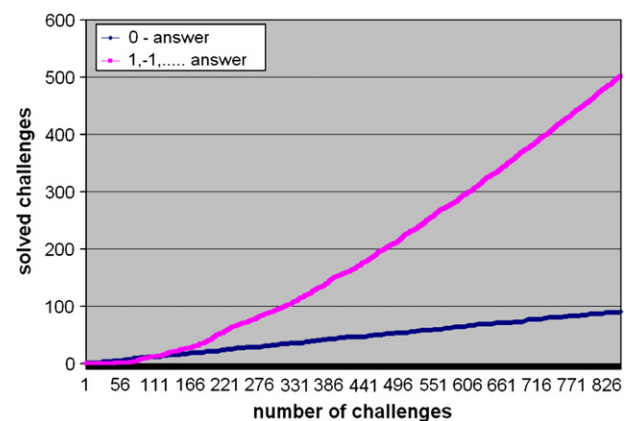


Fig. 17 – Success rate as a function of the number of challenges and strategy type (polynomial-superindex challenge).

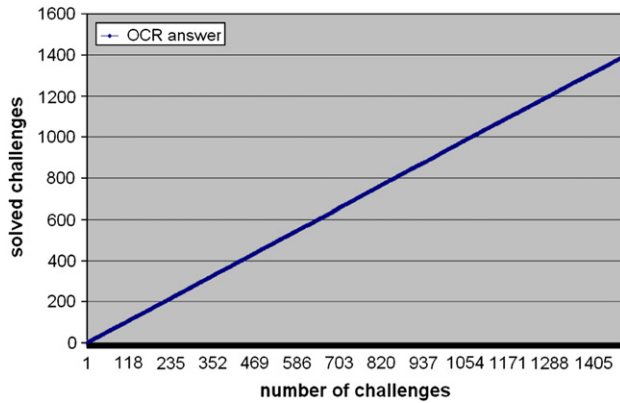


Fig. 18 – Success rate as a function of the number of challenges and strategy type (polynomial = $(x - a)(x - b)(x - c)(x - d)(x - e) \dots$ challenge).

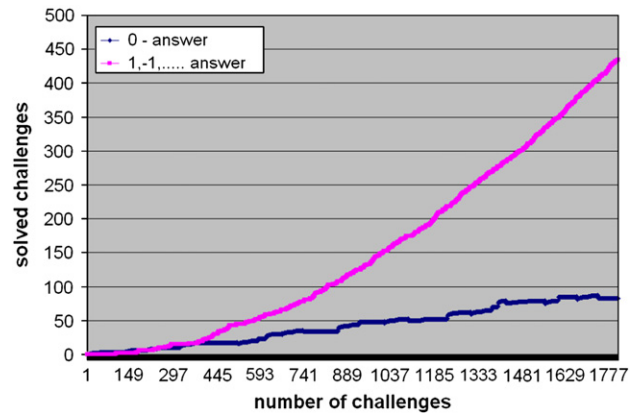


Fig. 19 – Success rate as a function of the number of challenges and strategy type (arithmetic challenge).

the OCR + heuristics answer and also by the $s(n)$ -strategy for the $\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$ polynomials.

4.2. Better (and simpler) attacks

The OCR-strategy for polynomials of type $(x - a)(x - b) \dots$ has a success ratio of 92.37%. If we had used page reloads to get an easier challenge – the most favorable for us would be this kind of polynomials – which would typically require around 6 reloads (as this kind of problem appears around 15% of the times) we could easily have a ratio of solved challenges over this figure.

Even if the authors take out this particular type of challenge, we still can solve the complex derivatives on x formulas with a 99% ratio if we wait long enough to automatically solve all the single challenges. However, if we do not want to wait to do so, just 5700 of this challenges requested gives us a 55% solving ratio.

If we do not want to construct a database of solved challenges, or if the authors repeat the generation process of the challenges for every CAPTCHA request with a similar probability distribution, our best strategy would be to answer by OCR to the polynomials of type $(x - a)(x - b) \dots$ and to answer 0 in any other case. Which would be the ratios in that situation? The big chance of appearance of 0 as a common solution is enough to consider this CAPTCHA solved even if those improvements are taken into action: for another test run of 10,000 request with this two strategies (0/OCR) we obtain the following results:

Problem type	Total challenges	Solved	%
$\partial/\partial x$	5749	2949	51.29
Polyn. $\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$	842	84	9.97
Polyn. $(x - a) \dots$	1463	1335	91.25
Arithmetic calc	1946	86	4.42
0.5			
Total	10,000	4454	44.54

As we would still be able to differentiate between types of challenges, the CAPTCHA remains solved 100% of the time. But even if the authors correct this so no longer we can differentiate the type of challenge, we would be able to solve the CAPTCHA in a striking 44.54% of the times.

5. Improvements

We have already discussed the design and implementation flaws, now we are going to discuss some mechanisms to avoid them that can be applied to this CAPTCHA but also to any design and implementation of a new CAPTCHA.

1. The possibility of refreshing the page to get an easier CAPTCHA is very dangerous in general, for two main reasons:
 - (a) It makes it possible for a program to reload the page till it knows how to answer the challenge, making a 1% successful attack just 100 times slower, but also a 100% efficient attack.

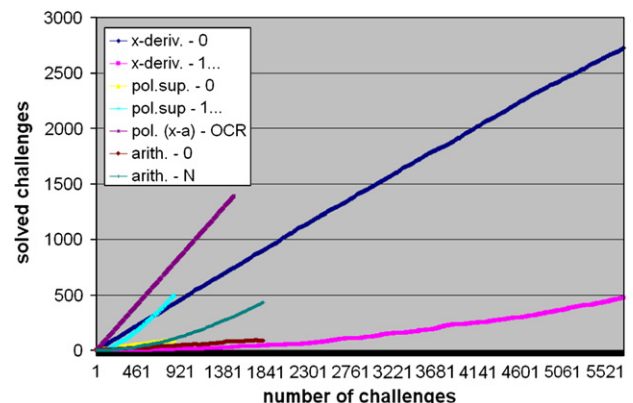


Fig. 20 – Success rate as a function of the number of challenges and strategy type.



Fig. 21 – The answers to the first phase of the IMAGINATION CAPTCHA are far from random.

- (b) It is more dangerous if there is an automatic way to tell which type of challenge (a normal or an easier one) we are facing each time. This way, it would be also possible for a program to refresh the page till it gets a challenge it knows how to answer. That way, even if the number of errors is limited, the attack can still be promoted from negligible effectiveness to 100% effectiveness.
2. For the same reason, the possibility of reloading the page after a bad answer has been given to the CAPTCHA has to be controlled and limited.
3. It should be avoided the immediate knowledge of the correctness of an answer to the CAPTCHA challenge. To prevent giving this information, we can use some intermediary communication mechanism (like e-mail accounts, which will also need to be controlled not to use the same more than n times) that makes it more costly for a bot to know if the answer was correct. It would be even better to present that information to the user in a way difficult to distinguish automatically.
4. It is important to avoid any non-uniformity in the distribution of correct answers, even more when their range is limited.
5. It should be avoided the limitation on the range of (correct and incorrect) answers.
6. It is important to avoid the repetition of challenges. If the challenges are based on a challenge database, then they should be hardly masked and “disguised” in order to prevent automatic detection of repetitions.
7. It is important to have an uniform distribution on the different parts that compose a challenge, and that those parts do not affect the whole challenge (so an answer in particular, like 0, is not more probable than the rest).
5. Do not tell if the CAPTCHA answer was correct or wrong. For example, send an e-mail notifying it to the user and asking for account activation (so just trying to log in is not an automatic way of telling if it was successful or not).
6. Instead of using integers, use random floats with at least two-digit precision and a range big enough, of at least 10,000 different values, per figure used.
7. Instead of asking for a numerical value, ask the user to tick the correct answer among a set (random + 1 correct) given. To avoid a blind attack for being successful, we can present the user with several columns of n -rows of ticks, each column containing a part of the answers, being only one (per column) the correct one. Thus, if we have 10 rows and 5 columns, the (blind) chance of success would be $1/10^5$. By inserting more rows and/or columns, we can make it as low as necessary.
8. Study the CAPTCHA answer distribution for any non-uniformity using random-number test suites.
9. Study the CAPTCHA OCR results, and distort more the challenges when needed.
10. On-line creation of different challenges to avoid repetition.
11. Create an alternative for visually impaired users.

6. Other CAPTCHAs affected

An ideal CAPTCHA could be considered as a function which input parameters take random values, and which produces a challenge and an answer, in a way that they cannot be related automatically, but is easy for a human being to relate them:

$$f(\text{random}) \Rightarrow (\text{challenge}, \text{answer})$$

In this case study, we have seen that just a heavily non-uniform distribution of the answers makes a CAPTCHA suitable for side-channel attacks. How common is this scenario? In the first phase of the IMAGINATION CAPTCHA (Datta et al., 2005) (Fig. 21), the user has to click in the

To improve the QRBGS CAPTCHA, we could do the following:

1. Limit the number of retries per IP address.
2. Limit the number of retries per browser (using cookies).
3. Limit the number of retries per e-mail address.
4. Limit the number of practical retries per time, creating an artificial delay from trial to trial.

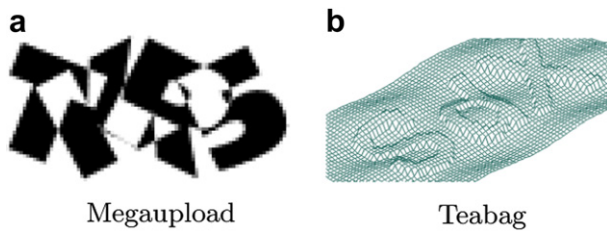


Fig. 22 – Megaupload and Teabag CAPTCHAs.

geometrical center of any of the images that form the image (with some error tolerance, as humans are not so precise). But, as sizes of images are non random at all, there are certain zones in which it is much more likely that we find a geometrical center.

Another examples are character-based CAPTCHAs which prevent some characters from appearing because they can be confusing (taking into account the graphical transformations they use). For example, the Megaupload CAPTCHA³⁵ (Fig. 22a) 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. The Teabag CAPTCHA version 1.2³⁶ (Fig. 22b) uses only three characters, and are strongly not uniformly distributed, 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 never appeared, including '1', '0' (perhaps to avoid coincidence with 'l' 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, 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 with the string "mmm" will be a fairly good guess.

Also, a certain correlation on the challenges and the answers can be dramatic in the ability to bypass a CAPTCHA. In our case of study, we have found that some parameters of the challenges are enough to tell the type of challenge faced – and, with this information, we did not need to go any further. In other cases, we can tell much more information. In an analysis of the HumanAuth CAPTCHA (Hernandez-Castro et al., 2009) (Fig. 3), we show that, studying just eight parameters of the image files that compose a CAPTCHA challenge, it is possible to tell (with better than 90% accuracy) their classification, and thus, to pass the CAPTCHA. The same classification for ASIRRA gives us a 58% accuracy, better than the one achieved by ASIRRA authors using basic image analysis.

Golle (2008) tweaks a bit the same approximation used by ASIRRA authors to obtain a 83% classification accuracy. His approach consists in basic image analysis, using color presence, texture distance, and classifiers. His result, basically,

depicts a clear correlation in colors and textures of the challenge pictures and the two possible classifications (cat or dog). That result is achieved after ASIRRA authors considered that the cat/dog classification problem was among the most difficult ones in picture recognition.

7. Conclusions and further work

Surprisingly, due to design and implementation flaws, two of the types of problems that typically could be considered to be the most difficult – the derivatives over $x \partial/\partial x$ and the zeros of polynomials with no super-indexes, represented as $(x - a)^*$... – are the easiest to solve. But even if the QRBGS CAPTCHA authors restraint to their two more resistant problems against this approach, and generate a new random problem upon each request, their CAPTCHA is not ready to be put into production, as a CAPTCHA that offers an automatic solving ratio of more than 5% just by using a simple answer: 0 – and even more than 15% (arithmetic challenge) and 35% ($\dots + n_2 \cdot x^2 + n_1 \cdot x + n_0$ polynomials) just trying three different answers.

The CAPTCHA scene suffers a lack of methodological analysis and design of the schemes placed into production. Now, many “private” designs and implementations are used – designed and put into production “in house”, or sold/given as program libraries, without further testing.

This is not the way to do it, as their designers are not using any guideline nor aware of some of the typical pitfalls. Thus, most of them do not resist a proper analysis and attack. This is a good example: the ideas behind the QRBGS CAPTCHA are interesting and original, but due to a number of important flaws, it is not useful as a proper mechanism to tell humans and computers apart.

A good line of further research would be to improve the OCR-strategy using a different kind of analysis taking ideas from shape recognition. We are confident that OCR-like analysis of the one-line challenges of this CAPTCHA is not a serious problem. With more complex equations like the $\partial/\partial x$, the challenge is not to recognize the characters, but to properly relate them.

We have seen that the problems related to the non-uniform distribution of all the parameters involved in a CAPTCHA challenge creation (input parameters, functions chosen, distribution of correct answers, distribution of different challenge characteristics, etc.) poses a serious risk to their security. This risk is even worse if there is an easy to find correlation between challenge characteristics and answers – which happens surprisingly often.

We expect this work to be a good starting point for designers of new CAPTCHAs, to avoid some common design flaws that could render their ideas useless, and to pass their designs through some basic randomness tests that can find some early flaws in them.

REFERENCES

- Ahn Luis von, Dabbish Laura. Labeling Images with a Computer Game. ACM Conference on Human Factors in Computing Systems. CHI; 2004. p. 319–26.

³⁵ <http://www.megaupload.com>.

³⁶ <http://ocr-research.org.ua/tb/getimage.php5>.

- Ahn Luis von, Blum Manuel, Hopper Nicholas J, Langford John. CAPTCHA: using hard AI problems for security. In: Advances in cryptology – EUROCRYPT 2003, international conference on the theory and applications of cryptographic techniques, Warsaw, Poland, May 4–8, 2003. Springer; 2003. p. 294–311. Proceedings, volume 2656 of Lecture Notes in Computer Science.
- Chellapilla K, Larson K, Simard PY, Czerwinski M. Computers beat humans at single character recognition in reading based Human Interaction Proofs (HIPs). In: Proceedings of the Conference on Email and Anti-Spam (CEAS); 2005.
- Chew M, Tygar JD. Image recognition CAPTCHAs. In: Proceedings of the 7th international information security conference (ISC 2004). Springer; September 2004. p. 268–79. A longer version appeared as UC Berkeley Computer Science Division technical report UCB/CSD-04-1333, June 2004.
- Datta Ritendra, Li Jia, Wang James Z. IMAGINATION: a robust image-based CAPTCHA generation system. In: Proceedings of ACM Multimedia Conference; 2005. p. 331–4.
- Elson Jeremy, Douceur John R, Howell Jon, Saul Jared. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: Proceedings of 14th ACM conference on Computer and Communications Security (CCS). Association for Computing Machinery, Inc.; Oct. 2007.
- Golle Phillipe. Machine learning attacks against the asirra CAPTCHA. In: Proceedings of the ACM computer and communications security conference; 2008.
- Hernandez-Castro Julio Cesar, Sierra J.M., Hernandez-Castro Carlos Javier, Ribagorda Arturo. Compulsive voting. Proceedings of the 36th annual 2002 international carnahan conference on security technology; 2002. pp. 124–133.
- Hernandez-Castro Carlos Javier, Ribagorda Arturo, Saez Yago. Side-channel attack on labeling CAPTCHAs. Unpublished manuscript; 2009. Available electronically at: <http://arxiv.org/abs/0908.1185>.
- Mori Greg, Malik Jitendra. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: Conference on Computer Vision and Pattern Recognition (CVPR 03). IEEE Computer Society; 2003. p. 134–41.
- Naor Moni. Verification of a human in the loop or identification via the turing test, <http://www.wisdom.weizmann.ac.il/naor/PAPERS/human.ps>; July, 1996 [accessed 01.01.09].
- NewScientist entry on the QRBGS CAPTCHA, <http://www.newscientist.com/blog/technology/2007/08/prove-youre-human-do-math.html>.
- Stevanovic R, Topic G, Skala K, Stipcevic M, Rogina BM. Quantum random bit generator service for Monte Carlo and other stochastic simulations. In: Lirkov I, Margenov S, Wasniewski J, editors. Lecture Notes in Computer Science, vol. 4818. Berlin Heidelberg: Springer-Verlag; 2008. p. 508–15.
- Techworld.com article on fallen CAPTCHAs, <http://www.techworld.com/security/features/index.cfm?featureid=4168>, https://www.techworld.com.au/article/253015/how_captcha_got_trashed?pp=3.
- US Patent no. 6,195,698, method for selectively restricting access to computer systems, <http://www.freepatentsonline.com/6195698.html>.
- Warner Oli. Kittenauth, <http://www.thepecspy.com/kittenauth>.
- Wieser Wolfgang. Captcha recognition via averaging, <http://www.triplespark.net/misc/captcha/>.
- Carlos Javier Hernandez-Castro** got his degree in Computer Science at Complutense University of Madrid in 1996. Between 2001 and 2003 he did a MSc. in Computer Security at the same University, and a second one in Computer Security & E-Commerce at UNED University. He spent a year at the Consejo Superior de Investigaciones Científicas (CSIC) with a formative research grant. He then was a Lecturer at Complutense University, while working in IT. Currently, he is an Assistant Professor, lecturing in Information Security, at the Computer Science Department of Carlos III University, where he also is pursuing a Ph.D. degree. His main research interests are CAPTCHAs, specially breaking and designing new ones, and Machine Learning and Cryptology related subjects.
- Arturo Ribagorda Garnacho** got a PhD. in Computer Science at the Politecnico University of Madrid. He then joined the Computer Science Department of Carlos III University of Madrid, where he is a full Professor and a Former Dean of the School. He currently leads the Computer Security Group, comprising approximately 25 members, and has more than 100 papers published in prestigious International Journals and Conferences.