Shortcomings in CAPTCHA Design and Implementation: Captcha2, a Commercial Proposal

Carlos Javier Hernandez-Castro[1], Jonathan D. Stainton-Ellis[2], Arturo Ribagorda[1] and Julio Cesar Hernandez-Castro[2]

[1]IT&C Sec. Group (SETI), Comp. Science Dept., Univ. Carlos III, Madrid, Spain
[2]School of Computing, Portsmouth University, United Kingdom
chernand@inf.uc3m.es, jonathan.stainton-ellis@myport.ac.uk,
arturo@inf.uc3m.es, Julio.Hernandez-Castro@port.ac.uk

Abstract: Many CAPTCHA proposals have shortcomings in their design or implementation that make them much weaker than intended. In this paper we study Captcha2, a commercial algorithm, as a means of showing typical flaws that make many CAPTCHAs prone to successful low-cost attacks. The attack we present makes no use of any AI techniques, not affecting the resilience of the original AI problem this CAPTCHA is (supposedly) based upon. That's why it can be considered a pure side-channel attack. We conclude with some tips for improving this CAPTCHA, which can be also used as general guidelines for avoiding a certain family of very common flaws.

1 Introduction

Nowadays we find an increasing number of opportunities for social and business interaction through the Internet. There are sound economic reasons to abuse many of these services. This typically involves automated actions [GD05, Her02]. The main approach to preventing such automated abuse has been to develop the ability to tell humans and computers apart, remotely and through an untrustworthy channel. With this aim, many tests have been developed, generically called CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIPs (Human Interactive Proof), relying on capacities inherent to the human mind but supposedly difficult for computers to mimic; that is, problems that remain wide open for Artificial Intelligence (AI) researchers.

1.1 Brief History of CAPTCHAs

Moni Naor seems to have been the first to propose theoretical methods of remotely telling computers and humans apart [Nao96]. The first known use of a CAPTCHA was in 1997, in the AltaVista web-search engine [Bro01]. In 2000, Udi Manber of Yahoo! described their "chat room problem" to researchers at C.M.U. Professors Manuel Blum, Luis von Ahn and John Langford described some desirable properties for any test used to detect them, and developed a CAPTCHA, named GIMPY [vABHL03].

1.2 Types of CAPTCHAs

Text-based CAPTCHAs were, and still are, very popular [HCR09a], although they can be attacked [MM03, CLSC05]. The trend with text-based CAPTCHAs has been to evolve them away from their known vulnerabilities, by changing a parameter (typeface, distance between characters, overlapping, colors, background, etc.) or introducing a new distortion. One example is reCAPTCHA [Wil10], as recently they introduced a new distortion (based on an idea found in BaffleText [CB03]). Other similar cases include Google, Yahoo! and Hotmail. The last decade has seen the publication of many new techniques enabling the breaking of text-based CAPTCHAs [HGH08, YA08a, YA08b, Wil10].

1.3 Evolution of CAPTCHAs

The strength of the text-based scheme is of increasing concern. As general vision seems to be a harder problem, more designs have focused on using pictures. Others are still based on text, but including a graphic "approach" (graphic distortions, backgrounds, click instead of type, 3D projections, etc). Some examples are Teabag, the old version of Rapidshare (a difficult CAPTCHAs for humans!), the new version of reCAPTCHA (all in fig. 1), and the one this article focuses on, Captcha2, a commercial proposal.

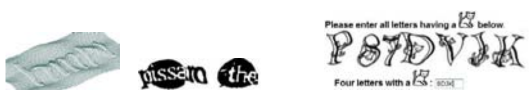
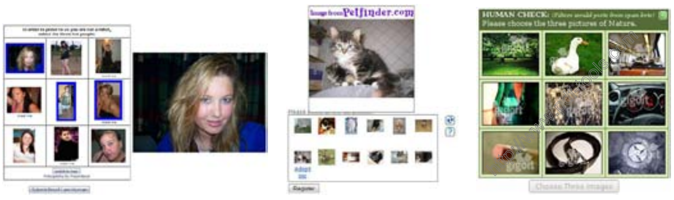


Figure 1: Teabag v1.2, reCaptcha and RapidShare CAPTCHAs.

Chew and Tygar [CT04] were the first to create a CAPTCHA based on Google Images. The problem of referencing names and words with two or more meanings is present. Ahn and Dabbish [vAD04] created the "ESP game" to feed the PIX CAPTCHA database. The site HotCaptcha.com (fig. 2(a)) was the first to propose using a large-scale, human-labeled database (HotOrNot.com web-site). Oli Warner had the idea of using photos of kittens to tell computers and humans apart [War09]. Its database of pictures is too small ($<$ 100). The HumanAuth CAPTCHA (fig. 2(c)) asks the user to distinguish between pictures depicting either a nature-related image, or a human-generated one [HCRS09]. ASIRRA (fig. 2(b)) uses a similar approach, but uses a large database "of more that 3 million photos from Petfinder.com" [EDHS07], but it has been broken [Gol09, HCRS09]. Other research papers propose many different image-based CAPTCHAs (fig. 3).



(a) HotCAPTCHA (b) ASIRRA (c) HumanAuth

Figure 2: Three image-based CAPTCHAs.

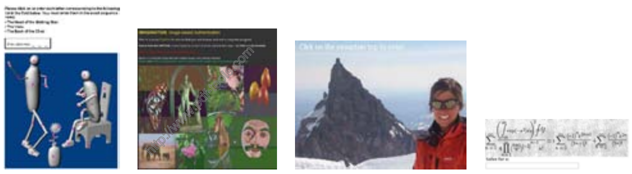


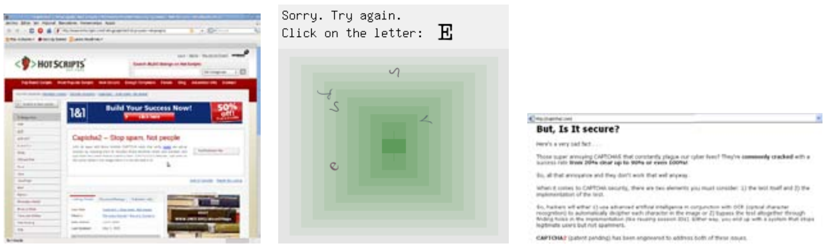
Figure 3: Other CAPTCHA proposals.

1.4 Motivation

Today we do not yet understand if a particular problem, used in a particular design, is really hard enough. Even if this were the case, is a particular CAPTCHA open to a side-channel attack? Is the hardness of the underlying AI problem fully transmitted to its design? Recent CAPTCHA design history is filled with mistakes which have made them much weaker than intended [HCR09a, HCRS09, Gol09, HCR09b]. We find it very interesting to try to find shortcomings in current CAPTCHA design, to get to a point where well-known and tested assumptions and methodologies give a base for more secure CAPTCHAs.

2 Captcha2

The Captcha2 test, available on the Captcha2 web-site (http://www.captcha2.com), is a text- and image-based CAPTCHA, comprising a colored background, upon which are superimposed a selection of characters. It is being marketed as a commercial product (fig. 4(a)). To pass a Captcha2 test, a user must click with the mouse on a specified, single, alphabetic character, and must pass two consecutive challenges. The background of the lower part of the image challenge is filled with shades of a single color. The graduation pattern varies. Captcha2 uses a variable number of characters, (some of which are not letters of any alphabet) of various sizes, rotated and randomly placed.



Sorry. Try again.
Click on the letter: E

But, Is It secure?

Here's a very sad fact . . .

Those super annoying CAPTCHAS that constantly plague our cyber lives? They're **commonly cracked** with a success rate **from 20% clear up to 90% or even 100%!**

So, all that annoyance and they don't work that well anyway.

When it comes to CAPTCHA security, there are two elements you must consider: 1) the test itself and 2) the implementation of the test.

So, hackers will either 1) use advanced artificial intelligence in conjunction with OCR (optical character recognition) to automatically decipher each character in the image or 2) bypass the test altogether through finding holes in the implementation (like reusing session IDs). Either way, you end up with a system that stops legitimate users but not spammers.

CAPTCHA2 (patent pending) has been engineered to address both of these issues.

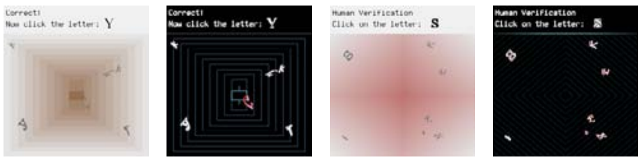
(a) Captcha2 referenced in other sites (b) Captcha2 challenge example (c) Captcha2 security claims

Figure 4: Captcha2 distribution, example and claims.

2.1 Captcha2 Design Goals

Captcha2 is designed to provide a level of security against some known attacks as good as the best, current, text-based CAPTCHAs, while at the same time being easier and more friendly to use. Ease-of-use comes from the user not having to leave the mouse to solve the CAPTCHA, making it as simple as a game. One idea used in Captcha2 for improved robustness is the requirement of identifying a single letter twice (problem description and image). Because the answer space seems quite broad, one can have the naïve impression that Captcha2 is strong against brute force attacks. The clickable image is 260x221 px, and each letter uses a circle of around 9 pixels radius. That involves that the success space is $pi * 9^2 = 254$ pixels over 57,460 pixels, which is a 0.44% chance, very high.

The background image of a challenge comprises a gradient of color. Why? It could be to render it "resistant" to edge analysis. In fact, using different edge-analysis routines (Sobel, Differential, etc.), we obtain results like the ones in fig. 5.



Correct!
Now click the letter: Y

Correct!
Now click the letter: Y

Human Verification
Click on the letter: S

Human Verification
Click on the letter: S

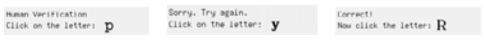
Figure 5: Edge-detection examples.

2.2 Captcha2 Security

The author has included details of the security features of the Captcha2 system on his product's web-site. After stating that "We don't believe in security by obscurity. Peer review is important", the author claims that the clickable bounding-box corresponding to the answer is placed at random, the character shown in the clickable test box is displayed in a different font and the opposite case to the one shown, the background of the image is a gradient that changes direction, shape and color at random, the color, shade, alignment and font of the test character all change at random, the coordinates of the test box are only stored on the server-side, the single-use validation ID is valid for no longer than ten seconds, and the originating IP address is banned from further action for ten minutes after ten consecutive failures, classifying it as a bot. According to the author, its benefits are its friendliness and ease of use; the author believing it more fun to solve than a traditional, text-based CAPTCHA. There is no need to type, the user just has to click with a mouse. This concept may translate well to other environments where keyboards are awkward to use or are unavailable, e.g. mobile phones, web-kiosks. Also, a graphical analysis or OCR-based attack would need to use two analysis, and in the graphical area, every candidate character is anti-aliased with the background, giving a less pronounced change.

3 Analysis

Before the attack, we analyzed how Captcha2 internals work using Wireshark. We confirmed our analysis prototyping some interfacing functions in Python, using httplib. The server creates a new JavaScript script for each challenge. The challenge image is downloaded from a URL that includes another ID referenced in the previously mentioned JavaScript (for example, http://captcha2.com/[...]?evid=aa56f9b39[...]a83). Also related to this ID is the URL called to validate the coordinates on which the user clicks, of the form http://captcha2.com/sys/validate.php?evid=aa56f9b[...]25a83&x='+x+'&y='+y, x and y being the coordinates on which the user has clicked.



Human Verification
Click on the letter: p

Sorry. Try again.
Click on the letter: y

Correct!
Now click the letter: R

Figure 6: Possible message-embedded messages.

Upon receiving the client answer, the server returns three possible answers: "1", when the user needs to pass another challenge, a string of 32 characters, indicating that the test has been passed successfully, or the string "Error", when the test has been failed.

```
image = load_CAPTCHA_challenge_image()
most_frequent_color = most_frequent_color_of(image)
color_list = []
for each pixel p = (x,y) in image:
  color = color_of_pixel(p)
  if color not in color_list:
    cl_color = closest_color_in_RGB_space(color, color_list)
    adjacent_colors = colors_of_adjacent_pixels(p, image)
    if max_RGB_difference(color, cl_color) <= MAX_DIF_PER_CHANNEL
       and cl_color in adjacent_colors:
      append color to color_list
% by this point, all colors in color_list
% are considered to be background
image = change_color (image, color_list, white)
```

Table 1: Pseudocode for removing the background

3.1 Insights

After a little experimenting with different Captcha2 challenges and some further testing, we discovered the following facts, which can be employed to attack the CAPTCHA:

1. The colors used for the background are (typically) not used for the letters.
2. The colors used for a background area are close (in RGB coordinates) to its contiguous background areas.
3. The background areas are big (in pixel count).
4. The correct letter is, most of the time, rendered in a bigger font size.
5. The correct letter is, many times, rendered in bold.
6. There is a very small chance of overlapping letters.

The three first observations make it possible to erase the background of the challenges (algorithm in table 1). Being able to remove the background also breaks a tenet of CAPTCHA design, that the best possible scenario would be for all answers to be equally likely. We must be careful choosing how close to black a pixel must be for it to count as if it were black, because it will influence in pixel counting. We can do better if we recall the typeface characteristics mentioned earlier: the right answer is often the character with the largest dimensions, and the correct letter is regularly emboldened. By following the procedure outlined in table 2, we conduct a pixel count of each contiguous area. We wondered how effective an attack based on these methods of removing the background and classifying the characters by pixel count would be. Many factors could affect the success of this method: non-contiguous elements, use of the background color for letters, etc. We also wondered about the effectiveness of the IP-based protection.

4 Attack

We have created a Python program to connect to the Captcha2 web-site, download and process the challenges, and send back to the server the estimated coordinates that corre-

```
region_list = []
for each p = (x,y) black pixel in image:
  image, number_of_pixels, centre_of_area =
    flood_fill(p, image, background_color)
  add (number_of_pixels, center_of_area) to region_list
max_center = maximun_number_of_pixels(region_list)
```

Table 2: Pseudocode for selecting the answer

spond to the requested letter. As Captcha2 is protected against a successive number of failures coming from the same IP address, we programmed a routine to detect when our IP was banned and then reset the public IP address of our router. There are some parameters that affect the processing of each challenge. Some of the more relevant are:

1. the difference per RGB channel that is allowed for considering a contiguous color also a background color
2. after background removal, which is the cutoff level for black
3. which pixels should be considered adjacent (typically, either $4px.$ or $8px.$)

To adjust those parameters, we manually downloaded a series of small set of challenges $(50+)$ and tried different values. It was very easy to adjust these parameters for values that gave good results (typically 15, 95% and $8px.$). We could have used an algorithmic search for the best candidates, but it was clearly not necessary. For the definitive statistics, we did a series consisting in 250-challenge tests. We find this number sufficiently representative for statistical purposes. A higher number would probably draw the attention of the server administrators and, it being a black-box attack, we wanted not to appear suspicious.

4.1 Attack Results

After 10 failures in a row Captcha2 classifies the originating IP address as a bot, which is also considered a failure. We had no such failures in our result set. From an image standpoint, we correctly solved 219 of the 250 challenges (31 failed), giving 87.6% accuracy. As a Captcha2-bypassing method, we had 100% accuracy – never being rejected as a bot.

Failed images	31	12.4%
Failed tests	0	0%
Correct images	219	87.6%
Correct tests	101	100%

Two successes in a row are needed to pass the test. We passed the test 101 times, so we needed 2.47 challenges to pass the test – a mere 23.5% overhead. The images that our program fail to solve are the more interesting ones. They typically used a small font for the correct letter and/or big symbols that did not represent the correct answer. Figure 7 shows some of the 31 challenges that our attack did not pass. The correctly-answered challenges are less interesting. One of the 219 images solved is shown in fig. 8, along with the major analysis steps.

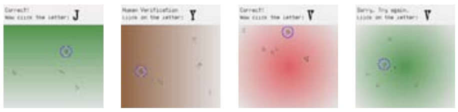
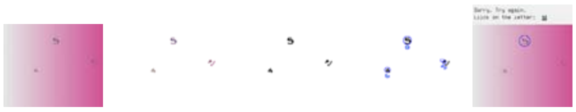


Figure 7: Wrongly solved challenges.



(a) (b) (c) (d) (e)

Figure 8: Example of a correctly solve challenge, and the main steps involved.

5 Improvements

This section suggests areas for improvement in the implementation of Captcha2 with a view to avoiding the attack detailed here and others. The first set of recommendations concerns flaws in the design and the second more general issues.

Related to design Flaws:

1. Not all background pixels of nearby regions should be of similar colors. The histogram of colors of the whole image should be as uniform as possible, and this should apply in any dimension.
2. A range of colors should be used in the character pixels.
3. More incorrect answers should be included, some overlapping, and characters should be further distorted to prevent easy discovery of adjacent zones (even after dilation).

General Issues:

1. The correct and incorrect answers should be statistically similar, in any way. When either answer can be characterized the system becomes vulnerable to side-channel attacks exploiting this characteristic.
2. The attacker should not be able to determine that submissions from their IP address are being rejected, because it is a trivial matter to change that address. The response should indicate solely that the test has been failed.

3. It should not be easy for an attacker to know if the test has been passed or not, requiring another level of processing to determine the outcome.
4. It should not be assumed that a design aimed at avoiding a particular type of attack (e.g. resist edge-detection) will be resilient to other attacks.
5. The target character should not always be in the opposite case to the character shown to the user. It should vary randomly between upper and lower cases, thus giving no information to an attacker.

6 Conclusions and Further Work

Currently there is a lack of methodological analysis and design in the CAPTCHA schemes being produced. These are being designed and deployed “in-house” or offered, sometimes commercially, as libraries without further testing. Most of these designs exhibit shortcomings and cannot resist proper analysis and attack. This is not a developmental model that will lead to the production of resilient designs, and it is likely to sully the reputation of CAPTCHA security. Captcha2 is a good example of a CAPTCHA that cannot withstand a thorough analysis and attack; although based on interesting and original ideas, it is not a reliable mechanism to tell humans and computers apart, due to a number of major flaws.

We have shown that a CAPTCHA design in which some of the variables that affect its creation (colors, background, typeface, size, etc.) are far from uniformly random is prone to attacks based on analyzing and taking advantage of these non-uniformities. We consider that this is a problem common to various CAPTCHAs [HCR09a, HCRS09, HCR09b, Wil10] and which has to be addressed in their design. Based on these ideas, we are doing further research towards a methodology for CAPTCHA design and testing, that can be subject to public discussion. We also recommend that all CAPTCHA designs are put under public scrutiny before being put into production.

References

[Bro01] Andrei Broder. http://www.freepatentsonline.com/6195698.html, 2001. US Patent no. 6,195,698.

[CB03] M. Chew and H. S Baird. BaffleText: a human interactive proof. In *Proceedings of 10th IS&T/SPIE Doc. Recognition & Retrieval Conf.*, San Jose, CA, USA, 2003.

[CLSC05] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In *Proceedings of the 2nd Conference on Email and Anti-Spam*, 2005.

[CT04] Monica Chew and J. D. Tygar. Image Recognition CAPTCHAs. In *Image Recognition CAPTCHAs*, pages 268–279. Springer, 2004.

[EDHS07] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. 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, 2007.

[GD05] Philippe Golle and Nicolas Ducheneaut. Preventing bots from playing online games. *Comput. Entertain.*, 3(3):3, 2005.

[Gol09] Philippe Golle. 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, 2009.

[HCR09a] Carlos J. Hernandez-Castro and Arturo Ribagorda. Pitfalls in CAPTCHA design and implementation: the Math CAPTCHA, a case study. *Computers & Security*, July 2009.

[HCR09b] Carlos Javier Hernandez-Castro and Arturo Ribagorda. Remotely telling humans and computers apart: an unsolved problem. In *Proc. of the iNetSec 2009, IFIP AICT 309*, 2009.

[HCRS09] Carlos Javier Hernandez-Castro, Arturo Ribagorda, and Yago Saez. Side-channel attack on labeling CAPTCHAs. http://arxiv.org/abs/0908.1185, 2009.

[Her02] Julio Cesar Hernandez. Compulsive voting. In *Proceedings IEEE 36th Annual 2002 International Carnahan Conference on Security Technology (Cat. No.02CH37348)*, pages 124–133, Atlantic City, USA, 2002. IEEE.

[HGH08] Abram Hindle, Michael W. Godfrey, and Richard C. Holt. Reverse Engineering CAPTCHAs, 2008.

[MM03] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In IEEE, editor, *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–134–I–141 vol.1, 2003.

[Nao96] M. Naor. Verification of a human in the loop or Identification via the Turing Test, 1996.

[vABHL03] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using Hard AI Problems For Security. In *In Proceedings of Eurocrypt*, volume 2656, pages 294–311, 2003.

[vAD04] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In ACM Press, editor, *CHI '04: Proceedings of the 2004 conference on Human factors in computing systems*, pages 319–326. ACM Press, 2004.

[War09] Oli Warner. Kittenauth. http://www.thepcspy.com/kittenauth, 2009.

[Wil10] Jonathan Wilkins. Strong CAPTCHA Guidelines. http://bitland.net/captcha.pdf, 2010.

[YA08a] Jeff Yan and Ahmad Salah El Ahmad. Breaking Visual CAPTCHAs with Nave Pattern Recognition Algorithms, 2008.

[YA08b] Jeff Yan and Ahmad Salah El Ahmad. Low-cost Attack on a Microsoft CAPTCHA, 2008.