Novel CAPTCHA schemes

Ville Saalo
Helsinki University of Technology
Ville.Saalo@iki.fi

Abstract

A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a program that can generate and grade tests that most humans are able to solve, yet current computer programs are not. They are used to protect various kinds of online services from advertising spam, brute force attacks and denial of service by automatic computer programs. The development of CAPTCHAs has been an evolutionary process where new CAPTCHAs are designed to resist the current attacks and attacks are developed to break the new CAPTCHAs. This same process has been present in other security related software such as spam, viruses and cryptography as well. This paper studies an attack against the Microsoft CAPTCHA and applies the lessons learned into the design of two novel CAPTCHA schemes.

1 Introduction

Three main types of CAPTCHAs have been identified: text-based (a list of examples can be found from [11]), sound-based (such as [9, 17]; evaluation of existing systems in [1]) and image-based [6, 8, 16, 19, 20, 24] schemes. Text-based schemes typically rely on distorting text images, hopefully rendering them unrecognizable to computer programs but still recognizable for humans. Sound-based schemes typically require the users to solve a speech recognition task, while the image-based schemes require the users to perform an image recognition task. Of these classes the text-based CAPTCHAs are the most used.

CAPTCHAs are technically a class of HIPs: Human Interactive Proofs (or Human Interaction Proofs [2]). The difference is that the data and algorithms of a CAPTCHA should, by definition, be publicly available. An example of a HIP that was not a CAPTCHA would be an image-based test which would have a secret database of images. In [7] it is suggested that HIPs should be public (i.e. CAPTCHAs) to encourage scientists to work on the problem of artificial intelligence.

This paper focuses on text-based CAPTCHAs.[1] Text-based CAPTCHAs are comprised of a segmentation problem and one or more recognition problems. The segmentation problem means that before being able to recognize individual characters the characters must be isolated from the background, separated from one another and placed into the correct order. The recognition problem is about recognizing the characters. Usually the characters are somehow scaled, warped, rotated or otherwise distorted to make this part harder. It has been shown that computers are better at solving the recognition problem than the segmentation problem, and that when solving a recognition problem they can even beat humans [2].

Figure 1: An example of a Microsoft CAPTCHA [13]

The rest of this paper is organized as follows. Section 2 presents the Microsoft CAPTCHA and an attack that effectively broke the scheme. Section 3 presents two new CAPTCHA schemes to which ideas have been drawn from the attacks against the Microsoft CAPTCHA. Section 4 discusses the general design principles of CAPTCHAs and applies them to the new schemes. Section 5 contains further discussion on parametrizing and attacking these new schemes, and finally Section 6 summarizes the conclusions.

2 The Microsoft CAPTCHA

As character recognition has been proven to be an easy task for computers, the Microsoft CAPTCHA was designed specifically to rely on the segmentation challenge, i.e. the fact that segmentation is difficult for computers [13]. In [3] the authors found out that using thick, non-intersecting arcs around the characters would make their CAPTCHA difficult for computers yet easy for humans to solve. An example of the Microsoft CAPTCHA is presented in Figure 1.

For automatic solving of their CAPTCHA they proposed a segmentor which would have had a 1/10,870 to 1/10,518,300 chance of a correct segmentation [4]. This, especially when combined with the uncertainly of the recognition part, would have been well beyond their established maximum acceptable success rate of automatic scripts, 1/10,000. In the same paper they also demonstrated the breaking of several other commercial CAPTCHAs, achieving success rates that were orders of magnitude better than 1/10,000.

The Microsoft CAPTCHA was effectively broken by a segmentation attack presented in [23]. They reported a segmentation success rate of higher than 90% and that segmenting one challenge took approximately 80 milliseconds on an

[1] For the purposes of this paper it is not important to distinguish CAPTCHAs and HIPs so all HIPs will be discussed as CAPTCHAs from now on.

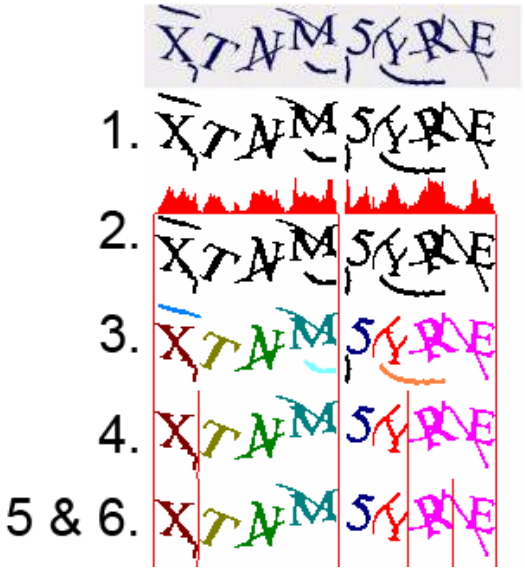

Figure 2: An example of how the attack against the Microsoft CAPTCHA works [23].

ordinary desktop computer. The attack has the following stages, demonstrated in Figure 2:

1. *Pre-processing*
   At this stage the CAPTCHA image is reduced into a black and white image with just one bit of color information per pixel. Pixels with a high intensity are converted to white, those with a lower intensity to black.
2. *Vertical segmentation*
   The image is segmented into *chunks* at those pixel columns where there are no foreground (i.e. black) pixels.
3. *Color filling segmentation*
   At this point connected components inside each chunk are detected by applying a flood-fill coloring algorithm into all black components until none are left.
4. *Thick arc removal*
   Thick arcs are removed based on their pixel count, location on the image and shape.
5. *Locating connected characters*
   At this point the algorithm locates those connected characters that escaped the vertical segmentation and color filling phases. This relies on the observation that every challenge has exactly 8 characters and works based on the assumption that wide components contain more than one character.
6. *Segmenting connected characters*
   Finally, the connected characters are segmented by dividing their chunks evenly.

3 New schemes

In this section two new CAPTCHA schemes, motivated by the aforementioned attack on the Microsoft CAPTCHA, are presented. The key objective of designing these CAPTCHA schemes was that they should be resistant to the parts of the attack on the Microsoft CAPTCHA. Discussion on these schemes is presented in Sections 4 and 5.

Figure 3: Transparent letter CAPTCHA. The random string "GHAERY" is being displayed.

3.1 The transparent letter scheme

The first scheme we have dubbed the *transparent letter scheme*. This is because the letters were made to overlap each other so much that they had to be made transparent to be readable. The transparency is reflected in the fact that where two grey letters overlap, a dark-grey area is shown. An example of this scheme is presented in Figure 3.

There is also clutter that is designed to confuse computer programs. Two kinds of clutter exists in the scheme: large and fine. Large clutter consists of shapes of familiar objects, such as animals, that are roughly of the same size as the letters. Fine clutter is more like noise and forms no recognizable shapes, serving to introduce areas of various shades of grey to places where there are no letters or where there is no overlap. The fine clutter also serves to break the solid white background and solid grey foreground surfaces.

The pre-processing stage should be more difficult in this scheme than in the Microsoft CAPTCHA as this image cannot be reduced to a one bit per pixel image without losing a lot of information: it is important to distinguish the background from at least two levels of foreground – letters and overlapping letters. The vertical segmentation phase would also have only limited use with this scheme as the characters are connected throughout the entire text and there are no empty vertical pixel columns. Only the intensities of pixel columns could be counted and even that approach should not yield much information due to the clutter.

The color filling algorithm is also what should be especially confused with this scheme. Firstly, the darkest areas are parts of multiple objects, and secondly, there are actually no well-defined surfaces to fill because of the fine clutter.

The thick arc removal phase is designed specifically against the Microsoft scheme but with the transparent letter scheme the equivalent of this phase would be the recognition of large clutter objects. The attack on Microsoft CAPTCHA relied here heavily on the heuristic that the arcs had a low pixel count compared to letters. Here, some of the large clutter objects may even be larger than the smallest letters. Another heuristic the attack on the Microsoft scheme used was that the arcs had no "loops" or "holes": they were just lines. Letters such as A, R and B have holes in them. With

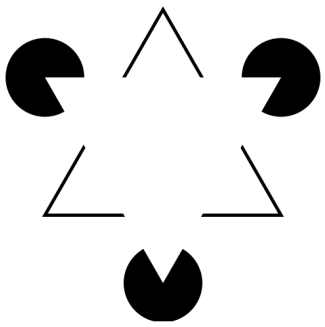

Figure 4: The Kanizsa triangle [22]

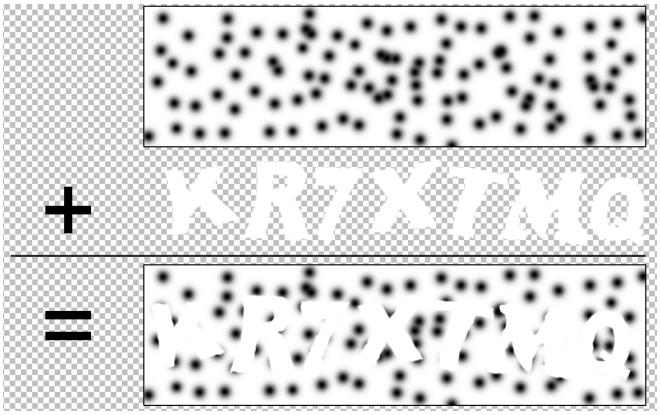

Figure 5: An example of a Kanizsa CAPTCHA challenge. To create this kind of a challenge the computer first creates a random background (top). Then it overlays it with some white text (middle) to get the final result (bottom).

the transparent letter scheme this heuristic is thwarted by the addition of random holes into the large clutter objects. Additionally, the holes are of realistic shapes and sizes and could be found from the letters as well. Finally, the arcs were removed based on their location in the challenge: objects close to the image edges were classified as arcs. To neutralize this attack the positions of large clutter objects are not limited in that sense: in the example image (Figure 3), for example, the letter Y is the rightmost object, but the cat shape is the leftmost. There is even a shape of a hand in the middle of the text, disconnecting letters A and E altogether.

3.2 Kanizsa CAPTCHA

The next scheme, called the Kanizsa CAPTCHA, is based on the *Kanizsa triangle* optical illusion where a triangle is perceived even though one is not actually drawn [21]. A Kanizsa triangle is presented in Figure 4. The creation of this kind of CAPTCHA challenge is demonstrated in Figure 5. The CAPTCHA challenge consists of a random background, round blobs in this case, overlaid with white text.

A similar approach was taken with the WaterCap CAPTCHA [15] but it was quickly cracked [14]. However, our scheme is more secure because the background is randomly created and the fonts are randomly chosen and distorted, while WaterCap used fixed image sprites with some easy to remove noise.

The vertical segmentation phase of the attack on the Microsoft CAPTCHA could probably be used to identify the locations of the letters to some accuracy. However, the next phase, the *color filling segmentation* (CFS), would be useless with the Kanizsa CAPTCHA scheme, as trying to color white objects on white background would result in a colored background with no new information revealed.

The closest equivalent to the thick arc removal phase would be the removal of those objects that are not a part of the outline of any character. This leaves only those objects that help outline the characters, but this can be countered by using random clutter as the background. This does, however, make the CAPTCHA significantly more difficult to humans as well because random clutter provides little hints on where the character contours lie. As a result the amount of background paint has to be increased, though increasing it too much would result in plain white text on black background, which would not be a very practical CAPTCHA challenge. Clearly a balance between the shapes and amount of background objects or clutter needs to be found by experimental studies.

The next phase of the attack on the Microsoft CAPTCHA, locating connected characters, is not applicable to the Kanizsa CAPTCHA. Firstly, the characters are white just like the background, and secondly, cramming the letters too close to each other would probably make the CAPTCHA too hard to read for humans.

The final segmentation phase would then have to recognize which background objects belong to which characters. How this phase is done probably highly depends on the scheme that was used to create the background. If the objects are random but there are many of them, the characters could be segmented based on where the largest white areas are. If the objects are of some given shape and there are less of them, the segmenter would probably have to rely on some other tactic, such as machine learning.

4 Design principles

These new schemes, the transparent letter scheme and the Kanizsa CAPTCHA, were designed to be resistant to the type of attack that was used to break the Microsoft CAPTCHA. However, CAPTCHAs usually should follow some other design principles as well. In [7] the authors of the Drag and Drop CAPTCHA also identified the following objectives that could be applied to virtually any CAPTCHA:

1. *Simplicity of operation*
   It is clear that this point is satisfied since both of the schemes only require the user to be able to recognize and input letters, i.e. no special expertise is needed.

2. *Test must be easy for humans*
   It seems to us that these tests are not more difficult to humans than the existing text-based CAPTCHAs. The difficulty can also be adjusted with various parameters. However, the true difficulty of these schemes with various parameters is to be found out in experiments with

humans.

3. *Test must be difficult for current computer programs*
   Both of the schemes were designed with the attack on the Microsoft CAPTCHA in mind so they should be more resistant against the types of techniques used against the Microsoft CAPTCHA. Again, the schemes would have to be evaluated with machine learning experts for more insight into this matter.

4. *Higher safety with lower bandwidth consumption*
   We found out that the CAPTCHA images Microsoft is using today[12] are from five to seven kilobytes in size and saved in the JPEG format. The CAPTCHA schemes presented in this paper seem to match that size easily when a high enough JPEG compression is used. They may need to be presented slightly larger than the current Microsoft CAPTCHAs which are 218×24 pixels but that does not have a significant impact on the file sizes.

5. *Easy to implement and maintain*
   We believe that especially the Kanizsa CAPTCHA would be easy to implement. The transparent letter scheme is, however, more complex and will be more difficult to implement. We estimate that the implementation would still not be significantly more difficult than implementing, for example, the Microsoft CAPTCHA.

With these high level objectives in mind one can then make some lower level design choices on the implementation of the CAPTCHA. Six choices were identified in [4]:

1. *Character set*: The character set to be used in the challenge.
   Clearly some characters should be left out because they would be too easy to confuse with similar characters. This would be the case with l, I and 1, for example. Also the Kanizsa CAPTCHA examples in this paper feature Q as the final letter, which could be easily confused with the letter O. Choosing upper case letters and digits and removing those that may be confused with each other we end up with roughly 30 possible characters[2]. If we allow the text length to vary from 8 to 10 characters, this would leave us with $30^8 + 30^9 + 30^{10}$ or approximately 6.1×10^{14} unique challenge strings.

2. *Affine transformations*: Translation, rotation and scaling of characters.
   Especially translation and scaling should be used with the transparent letter scheme. Some rotation could be applied to both schemes.

3. *Adversarial clutter*: Random noise or shapes that intersect with the characters and themselves.
   With the transparent letter scheme the clutter has already been defined in Section 3.1. The Kanizsa CAPTCHA could also be strengthened by adding white to areas where there are actually no letters.

4. *Image warp*: Elastic deformations of the challenge image.
   Image warp makes it more difficult to separate the background and foreground objects and textures from each other in the pre-processing state. After the segmentation phase it also makes the recognition problem more difficult. The text in Figure 5 demonstrates an elastic deformation, "rippling", in all the characters, especially well seen in the characters 7 and M in the bottom image. The letters H, A and E as well as the hand-shaped large clutter object in Figure 3 demonstrate a global warp effect. There are also some local ripple-like deformations present, most notably at the upper left corner of the letter R. Clearly both schemes can use and would benefit from using some image warp. However, it should be noted that it was found out in [4] that the human performance decreases considerably when high levels of local or global warp are applied to the text.

5. *Background and foreground textures*: Textures that are used to form a colored image based on the grayscale image generated by using the previously mentioned choices.
   Colored textures do not apply well to either schema presented in this paper. However, the fine clutter in the transparent letter scheme is already a texture designed to confuse computer programs.

6. *Language model*: Whether the challenges should use random strings or words from a dictionary.
   Using real words instead of random strings makes it easier to solve CAPTCHA challenges as words provide contextual clues on what the individual letters might be. Using real words instead of random strings, however, has the inherent drawback that the selection of available challenge texts decrease dramatically. Observing lists of English words of five, six, seven and eight letters [5] we can extrapolate the numbers of nine and ten letter words to about 37000 and 44000 respectively. There could then be only about 111000 unique CAPTCHA challenge strings that would be from eight to ten letters long. However, the words could also be longer than this as it is typically faster to type real words than random character strings. Increasing the allowed length of the words would also increase their amount.

   As the backgrounds are random and texts can be distorted, it seems to the authors that these numbers could well be enough, especially if only used for the first time a person or a bot tries to solve the challenge. If the first try is incorrect the subsequent challenges could be random strings. This vastly expands the solution space and decreases computer bots' chances of solving the challenge correctly. This kind of extra protection system would be somewhat like the two systems presented in [8]: the token bucket scheme and the Partial Credit Algorithm (PCA). The token bucket scheme works so that if the bot answers wrong multiple times, it then has to answer correctly to two consequent challenges to pass the test. The PCA system on the other hand accepts solutions that are almost correct if there are two consecutive solutions like that. Both of these schemes could

[2]For example, the personal identity number used in Finland has a check character that can be one of 31 different characters.[18]

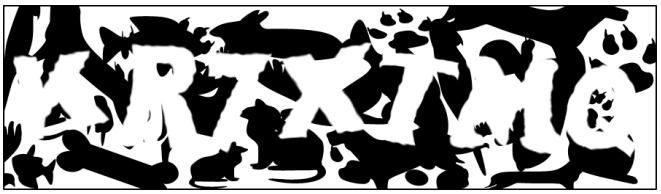

Figure 6: A Kanizsa CAPTCHA challenge using shapes of animals, paw prints and such as background objects.

be used with both the transparent letter scheme and the Kanizsa CAPTCHA as well.

5 Discussion

Both of the schemes presented in this paper have parameters that could be adjusted. They could both use different fonts even for different letters in the same challenge. This would confuse automatic programs trying to solve the challenges as all characters could be represented in several, slightly different ways. An example of different fonts can be seen in Figure 3 where the letter A has very round features but the letter Y has sharp corners and straight lines. The font used for the letter H is somewhere in between these two.

An interesting parameter in both schemes is the clutter. Figure 3, for example, has shapes of animals. There is nothing to prevent these shapes to be changed into, say, shapes of cars for a web site about cars. However, the best protection against bots is achieved when the clutter object selection is as large as possible. The objects could even be just random shapes but then one has to take care that the pixel counts and shapes do not give them away as was the case with the attack against the Microsoft CAPTCHA. Another risk in using completely random shapes is that they may end up resembling too much like real characters which would hinder the ability of humans to solve the challenges as well.

The Kanizsa CAPTCHA also does not need to have a background of round blobs: it could use the same kind of clutter objects as the transparent letter scheme uses. A website about pets could have the background created out of silhouettes of pets, such as in Figure 6.

The length of the challenge strings is an obvious parameter. The Microsoft CAPTCHA was broken partly because the strings were always eight characters long, so it is recommended that the texts in the new schemes be of random lengths.

The attack on the Microsoft CAPTCHA, presented in Section 2, is a practical example of the general CAPTCHA solving framework later presented in [10]. In that paper the authors also described another way of solving CAPTCHAs besides character recognition and segmentation: cloning. Solving CAPTCHAs by cloning is easiest if the CAPTCHA source code is available or if it is easy to create CAPTCHA challenges that are identical to the one being targeted. The basic idea is to generate challenges that all begin with different characters, compare them with the challenge that is being solved, keep the best matches and expand those challenges that were kept, character by character, until the entire CAPTCHA is matched. We believe that this might well be a viable tactic against the Kanizsa CAPTCHA, but given all the variation in the transparent letter scheme the search space would probably be too large for this approach.

6 Conclusions

This paper studied an attack on the Microsoft CAPTCHA and learned from the weaknesses that made the attack possible. As a result two new CAPTCHA schemes were developed and presented: the transparent letter scheme, which is based on heavily overlapping letters and large clutter objects, and the Kanizsa CAPTCHA, which relies on an optical illusion of outlines of letters that really do not exist. Preliminary analysis of the proposed schemes shows that they are simple for humans to operate yet difficult to be attacked using the mechanisms described in [23]. They are also versatile in that they can be customized with various parameters.

As to the future work, both of the new CAPTCHA schemes still need to be implemented as real programs. The challenges then have to be evaluated with users and their parameters tweaked according to the experiences gained. The schemes also need to be reviewed with machine learning experts for some insight into the strengths and weaknesses of the schemes.

References

[1] J. P. Bigham and A. C. Cavender. Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1829–1838, New York, NY, USA, 2009. ACM.

[2] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs). In *In 2nd Conference on Email and Anti-Spam*, 2005.

[3] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing Human Friendly Human Interaction Proofs (HIPs). In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720, New York, NY, USA, 2005. ACM.

[4] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski. Building Segmentation Based Human-friendly Human Interaction Proofs (HIPs. In *In Proceedings of the Second International Workshop on Human Interactive Proofs*, pages 1–26. Springer-Verlag, 2005.

[5] J. Chew. John Chew's Scrabble®Lists. `http://www.math.toronto.edu/jjchew/scrabble/lists/`. Online; accessed 17-October-2010.

[6] R. Datta, J. Li, and J. Z. Wang. 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, 2005. ACM.

[7] A. Desai and P. Patadia. Drag and Drop: A Better Approach to CAPTCHA. pages 1 –4, dec. 2009.

[8] J. Elson, J. R. Douceur, J. Howell, and J. 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. ACM.

[9] H. Gao, H. Liu, D. Yao, X. Liu, and U. Aickelin. An Audio CAPTCHA to Distinguish Humans from Computers. pages 265 –269, jul. 2010.

[10] A. Hindle, M. W. Godfrey, and R. C. Holt. Reverse Engineering CAPTCHAs, 2008.

[11] S. Hocevar. PWNtcha - Caca Labs. http://caca.zoy.org/wiki/PWNtcha. Online; accessed 8-October-2010.

[12] Microsoft. Sign up - Windows Live. https://signup.live.com/signup.aspx?lic=1. Online; accessed 11-October-2010.

[13] Microsoft. Microsoft Human Interaction Proof (HIP). http://download.microsoft.com/download/3/2/0/320e814a-d969-4c6c-a26e-2f3115032d4c/Human_Interaction_Proof_Technical_Overview.doc, 2006. Online; accessed 26-September-2010.

[14] Pavel Simakov. Cracking WaterCap CAPTCHA In 24 Hours. http://www.softwaresecretweapons.com/jspwiki/cracking_watercap_captcha_in_24_hours, May 2007. Online; accessed 21-Septempber-2010.

[15] Pavel Simakov. WaterCap Strong PHP CAPTCHA With Negative Spaces And Shadows. http://www.softwaresecretweapons.com/jspwiki/watercap_strong_php_captcha_with_negative_spaces_and_shadows, May 2007. Online; accessed 21-Septempber-2010.

[16] S. A. Ross, J. A. Halderman, and A. Finkelstein. Sketcha: a captcha based on line drawings of 3D models. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 821–830, New York, NY, USA, 2010. ACM.

[17] M. Shirali-Shahreza and S. Shirali-Shahreza. CAPTCHA for Blind People. pages 995 –998, dec. 2007.

[18] Statistics Finland. Online Statistics Course – Demography and population statistics – History of Finland's population statistics – What does your personal identity number tell? http://www.stat.fi/tup/verkkokoulu/data/vt/04/12/index_en.html. Online; accessed 16-November-2010.

[19] L. von Ahn, M. Blum, N. Hopper, and J. Langford. ESP-PIX. http://server251.theory.cs.cmu.edu/cgi-bin/esp-pix/esp-pix. Main site: http://www.captcha.net. Online; accessed 11-October-2010.

[20] L. von Ahn, M. Blum, N. Hopper, and J. Langford. SQUIGL-PIX. http://server251.theory.cs.cmu.edu/cgi-bin/sq-pix. Main site: http://www.captcha.net. Online; accessed 11-October-2010.

[21] Wikipedia. Kanizsa triangle — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Kanizsa_triangle&oldid=370923946, 2010. Online; accessed 22-September-2010.

[22] Wikipedia user Fibonacci. Kanizsa_triangle.svg — Wikimedia Commons. http://commons.wikimedia.org/w/index.php?title=File:Kanizsa_triangle.svg&oldid=37290457, 2007. License: Creative Commons Attribution-Share Alike 3.0 Unported. Online; accessed 22-September-2010.

[23] J. Yan and A. S. El Ahmad. A Low-cost Attack on a Microsoft CAPTCHA. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, pages 543–554, New York, NY, USA, 2008. ACM.

[24] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and Design of Image Recognition CAPTCHAs. CCS'10, 2010.