CAPTCHA Design

Color, Usability, and Security

Most user interfaces use color, which can greatly enhance their design. Because the use of color is typically a usability issue, it rarely causes security failures. However, using color when designing CAPTCHAs, a standard security technology that many commercial websites apply widely, can have an impact on usability and interesting but critical implications for security. Here, the authors examine some CAPTCHAs to determine whether their use of color negatively affects their usability, security, or both.

Ahmad El Ahmad and Jeff Yan
Newcastle University, UK

Wai-Yin Ng
Chinese University of Hong Kong

Color plays a major role in increasing usability in systems ranging from TV remote controls (whose buttons are highlighted in different colors to make them easy to spot) to complicated GUIs (where users' navigation from one area to another is effectively guided by different colors). When used properly, color can greatly enhance user interface designs.[1] Color in interfaces is thus typically a usability issue and has rarely led to any security failures.

Here, however, we examine the use of color in CAPTCHAs (Completely Automated Public Turing Test to Tell Computers and Humans Apart),[2] now a standard security mechanism that many commercial websites deploy to address spam and other online abuses. We focus on text-based CAPTCHAs, which are the most common and which rely on sophisticated distortion of text images, rendering them unrecognizable to current pattern-recognition algorithms but recognizable to human eyes. Some of our discussion is also relevant to other types of CAPTCHAs (such as image-based schemes that typically require users to perform an image-recognition task), which we discussed in detail in our previous technical report.[3]

A good CAPTCHA must be human-friendly and robust enough to resist computer programs that attackers write to automatically pass CAPTCHA tests. To strike the right balance between usability and security, designers must address many issues other than color, but these are discussed elsewhere.[4] Solving the accessibility issues that CAPTCHAs cause — for instance, by exploring alternatives[5] — is also important and of practical relevance, but is beyond this article's scope (although our discussion can help improve accessibility by identifying inappropriate use of color in deployed CAPTCHAs).

CAPTCHAs have commonly used color for the following reasons:

- Color is a strong attention-getting mechanism.

Published by the IEEE Computer Society
1089-7801/12/$31.00 © 2012 IEEE

- It's appealing and can make CAPTCHA challenges interesting.
- It can facilitate recognition, comprehension, and positive affect.
- Color can make CAPTCHA images compatible with webpage color so they look less intrusive.[6]
- Occasionally, color can provide variation to fit different user preferences — for example, a CAPTCHA service might let users choose their favorite color configurations.[7]
- Some CAPTCHAs also use colors to defend against automated attacks.

Here, we present some case studies showing that using color can actually reduce CAPTCHA usability, has negatively affected security, or is problematic in terms of both usability and security. These case studies cover both colorful and monochromatic CAPTCHAs. We look at CAPTCHAs from the earliest days to the latest designs and examine both simple and more complex attacks. Overall, these case studies constitute a cohesive and fairly comprehensive investigation of color use, leading to valuable lessons for designing robust and usable CAPTCHAs.

Color CAPTCHAs

Let's first examine several colorful CAPTCHAs.

Gimpy-r, designed at Carnegie Mellon University, was a well-known early CAPTCHA that used colorful challenges;[2] Figure 1a shows four examples. Yahoo deployed this scheme to protect its online services. The dominant color of distorted texts in each challenge image always had the lowest intensity among all colors used, and this color (often black) never appeared in the background. This made it easy for a computer program to extract the challenge text. Figure 1b shows the text extracted by our automatic program, which looks only for black pixels.

The images in Figures 1a and 1b show what the challenges look like for humans and computers, respectively; they provide just about the same level of security. The colorful background is useless for security — in fact, it can confuse people and thus decrease the scheme's usability.

In general, breaking a CAPTCHA (in the sense of writing computer programs that automatically solve the test) involves *segmentation* — locating individual characters in the right order — and *recognition* — recognizing which character is which.

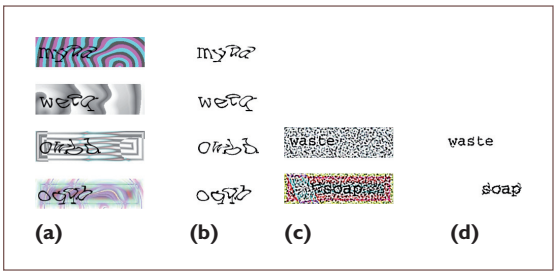

Figure 1. Two early CAPTCHAs (deployed around 2001). We took (a) the original challenge for Gimpy-r and (b) extracted the text using our automatic program. We did the same for (c) the original challenge for EZ-gimpy, (d) extracting only the black pixels. Both the original and text-extracted images provide the same level of security.

We had 100 percent success for segmentation with the Gimpy-r images using *color filling segmentation* (CFS).[8] CFS aims to detect every connected large component, which often corresponds to each individual character (or stroke). Our algorithm first detects a black pixel in the image in Figure 1b and then traces all its black neighbors until all the connecting black pixels are *traversed* — that is, a component is identified and segmented. Next, the algorithm locates a black pixel outside the area of the already identified components and starts another traversal to identify the next component. This continues until the algorithm has identified all black components. This method is effectively like using a distinct color to flood each connected component. In the end, each black component is highlighted with a distinct color, and the number of colors used is the number of black components in the image.

After segmentation, applying standard techniques to recognize each individual character at a high speed is trivial. For example, a neural network achieved a success rate of roughly 95 percent for recognizing individual characters heavily distorted in different ways.[9] So, we were able to break Gimpy-r with an overall success rate of approximately 81 percent ($\approx$100% * 0.95^4). A common design goal for CAPTCHA security is to prevent a bot from achieving a success rate of more than 0.01 percent.[6] Clearly, if we can reduce breaking a CAPTCHA to the problem of recognizing individual characters in its challenge, then the CAPTCHA is effectively broken.

We observed the Gimpy-r design flaw in EZ-gimpy (see Figures 1c and 1d), another

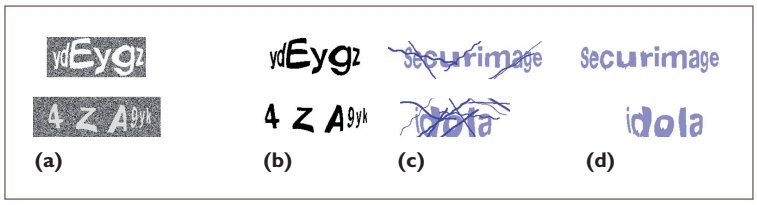

Figure 2. Two Securimage CAPTCHAs. We can see (a) the first scheme, which uses a gray-scale image and (b) the text our automatic program extracted. (c) The second scheme uses random arcs to intersect and connect its challenge characters, but (d) we can still extract the text. Both the original images and those extracted with our program provide the same level of security. (These schemes were created in 2008; we broke them in 2009.)

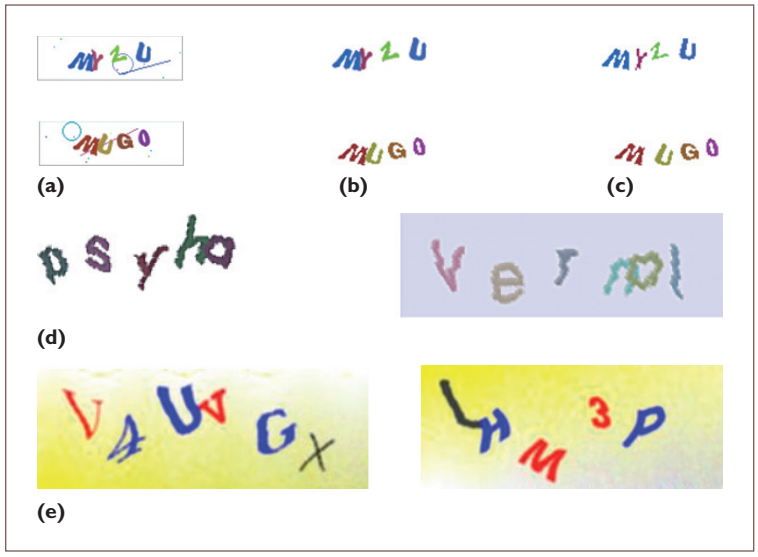

Figure 3. Color misuse in various CAPTCHAs. We can see (a) the original Cryptographp challenges, (b) the challenges after background noise removal, and (c) the final segmented results. (d) Two FreeCap examples and (e) two LinkedIn examples have similar security problems. (Cryptographp was created in 2006 and FreeCap in 2003; we broke both in 2007, and LinkedIn in 2009.)

well-known early CAPTCHA designed by the same CMU team and also deployed by Yahoo in the past.

Gabriel Moy and his colleagues developed distortion-estimation techniques to break Gimpy-r (78 percent success) and EZ-Gimpy (99 percent success).[10] Greg Mori and Jitendra Malik have broken EZ-Gimpy (92 percent success) using sophisticated object-recognition algorithms.[11] However, both works focused more on advancing computer vision algorithms in recognizing objects than on identifying CAPTCHA design flaws. The methodology we demonstrated — extracting challenge text, segmenting the text, and then recognizing individual characters — is generic to CAPTCHA security analysis but can also pinpoint design flaws. So, we use this methodology throughout the article.

Flaws similar to those found in Gimpy-r have occurred in CAPTCHAs that use gray-scale images, which have pixels of white, black, and many shades of gray. An example is the Securimage CAPTCHA (see Figure 2a) available from the open source CAPTCHA service at www.phpcaptcha.org. By detecting all pixels of the foreground color (white), we effectively extracted all challenge characters (see Figure 2b). We then automatically detected with 100 percent success each individual character using the CFS method. As such, we effectively broke this scheme. On the other hand, in this case, gray-scale images don't appear to degrade usability.

Another Securimage scheme (see Figure 2c) uses random arcs to intersect and connect its challenge characters to defend against segmentation attacks. This scheme uses three colors: white for the image background, lavender for the challenge text, and blue for the arcs. Because these colors are distinct, identifying and removing the arc is trivial. We first identify (by distinct color) arc portions that are outside the challenge text (that is, they don't intersect with the text) and remove them by converting their color to the background white. We then convert the remaining arc portions to text color (see Figure 2d). Afterward, the CFS method achieved 100 percent segmentation success; existing character-recognition technology can recognize the characters with a 95 percent success rate.

To make challenges look interesting, some CAPTCHAs generate images in which adjacent characters have distinct colors. The Cryptographp CAPTCHA (available as a WordPress plug-in and at www.captcha.fr) is such a scheme (see Figure 3a). However, this design feature turns out to be a fatal security mistake.

The original challenges used random shapes such as circles and intersecting lines as noise. However, removing such noise is trivial because these shapes' and lines' thickness differs significantly from that of characters in the images. Figure 3b shows the images after noise removal. Typically, segmenting overlapping characters is difficult. The state of the art[9] suggests that a text CAPTCHA should rely on such segmentation-resistant mechanisms to provide security.

However, because each character has a different (dominant) color in this scheme, by picking all pixels with the same color, we effectively segmented overlapping characters, as Figure 3c shows. We tested this method on 50 random challenges generated by this Cryptographp scheme and achieved 100 percent success for segmentation. The average segmentation speed was roughly 60 ms per challenge on a standard desktop computer. Any competent attacker could thus bypass this scheme instantly.

We observed a similar mistake in FreeCap (see Figure 3d), another popular CAPTCHA (available at www.puremango.co.uk/tag/captcha). In this scheme, adjacent letters have different colors, helping us segment touching and overlapping characters, which would otherwise be much harder. Note that FreeCap's designer is security savvy: his previous work identified a well-known attack in which simply re-using a known challenge image's session ID could bypass some early CAPTCHAs.[12]

Finally, we observed the same mistake in the latest CAPTCHA from LinkedIn, the popular social networking website (www.linkedin.com). For the LinkedIn examples in Figure 3e, we first applied *posterization*, which reduces the number of colors in the images. We then located the region containing overlapping or touching characters. Next, we easily separated "U" from "V" and "L" from "H" by picking up all pixels of the same color in the concerned region to form a character.

BotBlock, available at http://chimetv.com/tools/botblock, demonstrates that misusing complex color combinations in a CAPTCHA can cause both usability and security problems. This scheme, as shown in Figure 4a, uses random letters that appear in different places in a given challenge. The scheme introduces a sophisticated color-management method: backgrounds comprise multiple color blocks of random shapes, and foreground colors also occur in the background.

However, this color scheme often made it hard for people with normal vision (including us) to recognize challenge texts, and even harder for vision-impaired people, such as those with color blindness, to do so. (Color blindness affects roughly 8 percent of adult males and 1 percent of adult females in North America and Europe, and the most common form is difficulty discriminating between red and green.[1])

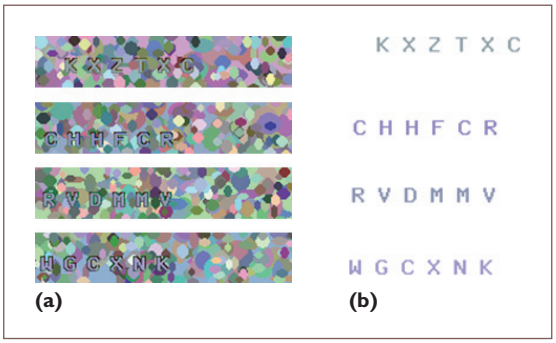

Figure 4. BotBlock CAPTCHA. We can see (a) the original sample challenges and (b) challenge text our automatic program extracted. Images in (a) and (b) provide the same level of security. (Its first deployment date is unknown, but we broke this scheme in 2007.)

Moreover, this scheme relied too much on color for security. We tested 100 samples and all were indeed resistant to the best optical character-recognition program on the market. Unfortunately, a design error made removing all the complex background trivial: the scheme has an exploitable color pattern for foreground texts — that is, the same color occurs repetitively. By looking for this pattern, we successfully extracted the challenge text in all samples we tested. The scheme's robustness is equivalent only to that for the challenges Figure 4b shows, which are trivial to decode.

For example, the letters in Figure 4b were vulnerable to a "pixel count" attack we discovered.[13] That is, by counting the number of foreground pixels in each character, we could recognize most of the characters. For a few with identical pixel counts, simply analyzing their geometrical shapes allowed our algorithm to tell them apart. Such simple attacks successfully broke all 100 random samples we tested.

Black and White CAPTCHAs

Let's now look at two black and white text CAPTCHAs: the first was until recently deployed by Megaupload.com, one of the largest file-sharing websites in the world; the other is from BotDetect, a commercial CAPTCHA system that claims a wide deployment by major websites (see http://captcha.biz). We show that such a simple combination of colors in CAPTCHAs can still go wrong.

Figure 5a shows the Megaupload CAPTCHA, whose key innovation was to combine two mechanisms from early research[8,14] that were

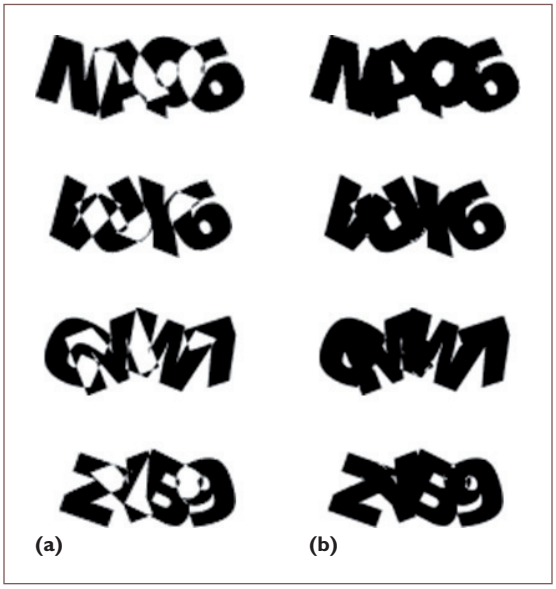

Figure 5. The Megaupload CAPTCHA. We can see (a) four example challenges, with the correct answers being NAQ6, VUX6, GMW7, and ZYB9, respectively. In (b), we see the same challenges but with the Gestalt feature turned off. (This scheme's first deployment date is unknown, but we broke it in 2009.)

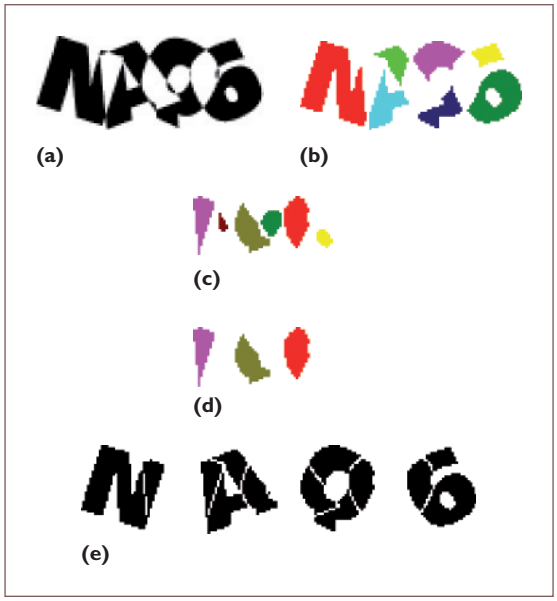

Figure 6. A segmentation attack on the Megaupload CAPTCHA. We took (a) an original image and (b) extracted all the black components (removing a small amount of black noise). We then (c) extracted all the white components except the main image background and (d) extracted the shared white components by removing loops. Finally, we (e) merged black and shared white components to form individual characters.

already proven effective at increasing security. The first mechanism makes characters overlap and connect with each other; in general, current computers aren't good at segmenting connected characters. The second mechanism uses Gestalt psychology: although some portions of each character are removed, humans can infer the whole picture from only partial information, whereas machines can't. Combining these mechanisms was clever and easy to implement: the Megaupload CAPTCHA connected each character (in black) with its neighbors and changed the connecting areas to the background color (white). (The latter is effectively an XOR operation.)

It appears that using Gestalt psychology also contributes to this CAPTCHA's usability: once you get used to the scheme, it's reasonably easy to read which character is which with accuracy. The usability improvement this feature introduced is more apparent when we compare images in Figure 5a with those in Figure 5b — the latter are the same images but with the Gestalt feature turned off.

The segmentation attack we used on this CAPTCHA works as follows. We first used the CFS method to extract black components, which define most of each character's actual content and are never shared with adjacent characters. Figure 6b shows the result of extracting all black components, each highlighted with a different color for illustrative purposes.

The second step of our attack was to identify and extract shared white components, which are the connecting areas between adjacent characters. We first applied the CFS method to detect all white components. Then, we excluded the main image background (that is, the outside of the image text), which is the largest white component in terms of pixel count; Figure 6c shows the remaining white components. With some heuristics, we also excluded, with a high success rate, loops that occur normally as part of a character's shape, such as those found in "A," "Q," and "6," and loops that occur when connecting characters together. For example, the connection between "V" and "U" in the second example in Figure 5a created a connection loop. Figure 6d shows that we successfully identified all the shared white components.

The final step was to put the shared white components in the right location to merge with corresponding black components to form each

complete character. We know that n number of characters when connected horizontally should typically produce $n - 1$ connection areas between them, and that shared white components that are juxtaposed vertically must belong to the same connection area. So, in an example that has only four characters, as in Figure 6,

- all shared white components inside the first connection area and all black components to its left will be merged to form the first character;
- all shared white components inside the first and second connection areas and all black components between them will be merged to form the second character;
- all shared white components inside the second and third connection areas and all black components between them will be merged to form the third character; and
- all shared white components inside the third connection area and all black components to the area's right will be merged to form the fourth character.

Finally, we converted the color of merged components to black, and kept individual characters away from each other horizontally. Figure 6e shows the final segmented result.

Our attack achieved a segmentation success rate of more than 78 percent; it took roughly 120 ms on average to segment each challenge on a standard desktop computer. This implies that we could break the Megaupload CAPTCHA (using segmentation and then recognition) with an overall success rate of 63.7 percent ($\approx$78.25% * 0.95^4). Details of our attack are available elsewhere.[15]

The BotDetect CAPTCHA that we examine is a "chess scheme" according to its designers. As Figure 7a shows, this scheme is effectively like embedding characters within a chess board. As each character is divided into numerous components, either black or white, and all the characters are mixed and connected with the chess board, the designers expect that automated programs will fail in extracting and recognizing the embedded characters. However, a simple attack works as follows.

We first detect each chess box. If the majority of pixels in a box are black, we reverse all the pixels that are originally black to white, and the pixels that are originally white, if any, to black. If the majority of pixels in the box are white, we do nothing. Figure 7b shows that our automated program successfully extracted all the characters. Locating and recognizing each character is trivial, so the scheme is broken.

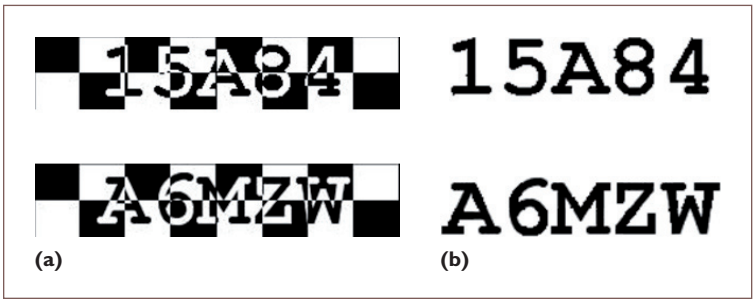

Figure 7. A BotDetect CAPTCHA. We took (a) sample challenges and (b) extracted the challenge text using our automated program. (This scheme was created in 2006; we broke it in 2009.)

Lessons Learned

Using color in CAPTCHAs can be tricky. Many CAPTCHAs, ranging from the earliest schemes to the latest design, and including the widely used LinkedIn, Megaupload, and BotDetect schemes as well as some relatively less-known designs, have had fatal security vulnerabilities due to their imprudent use of color. Using complex or fancy color schemes has raised usability concerns as well. It is far more difficult than it appears, for instance, to tell what kind of color images would cause problems for color-blind people, given the various types of color blindness.

We can summarize our common method for attacking CAPTCHAs into the following procedure: separating foreground from background; identifying connected components in the foreground and, when necessary, separating the connected components into individual characters; and subsequently recognizing individual characters. We can thus derive the following guidelines for using color:

- Uniformity in the foreground or background reduces resistance to segmentation attacks in general. A good security design principle is minimal uniformity in the foreground and background, with a controlled cost for readability.
- Contrast between foreground and background, or contrast among foreground characters, also reduces resistance to segmentation attacks.

For this reason, coloring that increases such contrasts might be undesirable.
- Perceptually connected but physically disconnected components are another good security design principle; however, coloring might enhance or destroy perceptual grouping with security consequences. So, a careful evaluation of the end design's security is essential.

However, engineering a CAPTCHA that entirely relies on color arrangement to provide reasonable security and usability simultaneously doesn't appear to be easy. Instead, we consider this an open problem.

To avoid the potentially complicated consequences of usability and security, a text CAPTCHA should use a simple color scheme; using color should aim at increasing a CAPTCHA's usability rather than its security. In particular, a CAPTCHA should rely on better-understood segmentation-resistant mechanisms,[8] rather than complex color management, to provide security.

Recent versions of several major CAPTCHAs have adopted these lessons and don't use complex color schemes:

- Microsoft uses a simple color scheme in which the foreground (that is, the challenge text) is dark blue and the background light gray.
- Google uses a single color (green, red, or blue) for all text and a white background.
- Yahoo uses black and white only.
- reCAPTCHA (http://recaptcha.net) also uses black and white only (reCAPTCHA is the latest design from the same team that invented Gimpy-r and EZ-gimpy).

The well-known "Las Vegas effect" on using color in interface design suggests that using fewer colors can be better than using too many. It appears that this principle also applies to text CAPTCHAs, but in this context, it isn't only a usability principle but also a security lesson.

Acknowledgments

Part of this work was conducted while Jeff Yan was a visiting faculty member in the Department of Information Engineering, Chinese University of Hong Kong.

References

1. L.W. MacDonald, "Using Color Effectively in Computer Graphics," *IEEE Computer Graphics & Applications*, vol. 19, no. 4, 1999, pp. 20–35.
2. L. von Ahn, M. Blum, and J. Langford, "Telling Humans and Computer Apart Automatically," *Comm. ACM*, vol. 47, no. 2, 2004, pp. 56–60.
3. A.E. Ahmad and J. Yan, *Colour, Usability and Security: A Case Study*, tech. report CS-TR 1203, School of Computing Science, Newcastle Univ., May 2010; www.cs.ncl.ac.uk/publications/trs/papers/1203.pdf.
4. J. Yan and A.E. Ahmad, "Usability of CAPTCHAs or Usability Issues in CAPTCHA Design," *Proc. 4th Symp. Usable Privacy and Security* (SOUPS 08), ACM Press, 2008, pp. 44–52.
5. "Inaccessibility of CAPTCHA — Alternatives to Visual Turing Tests on the Web," W3C Working Group note, 23 Nov. 2005; www.w3.org/TR/turingtest.
6. K. Chellapilla et al., "Building Segmentation-Based Human-Friendly Human Interaction Proofs," *Proc. 2nd Int'l Workshop Human Interaction Proofs*, LNCS 3517, Springer, 2005, pp. 1–26.
7. T. Converse, "CAPTCHA Generation as a Web Service," *Proc. 2nd Int'l Workshop Human Interactive Proofs* (HIP 05), LNCS 3517, Springer, 2005, pp. 82–96.
8. J. Yan and A.E. Ahmad, "A Low-Cost Attack on a Microsoft CAPTCHA," *Proc. 15th ACM Conf. Computer and Communications Security* (CCS 08), ACM Press, 2008, pp. 543–554.
9. K. Chellapilla et al., "Computers Beat Humans at Single Character Recognition in Reading-Based Human Interaction Proofs," *Proc. 2nd Conf. Email and Anti-Spam* (CEAS 05), 2005; www.ceas.cc/2005/papers/160.pdf.
10. G. Moy et al., "Distortion Estimation Techniques in Solving Visual CAPTCHAs," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition* (CVPR 04), IEEE CS Press, 2004, pp. 23–28.
11. G. Mori and J. Malik, "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition* (CVPR 03), IEEE CS Press, 2003, pp. 134–141.
12. H. Yeend, "Breaking CAPTCHAs without Using OCR," blog, 30 Nov. 2005, www.puremango.co.uk/cm_breaking_captcha_115.php.
13. J. Yan and A.E. Ahmad, "Breaking Visual CAPTCHAs with Naïve Pattern Recognition Algorithms," *Proc. Ann. Computer Security Applications Conf.* (ACSAC 07), IEEE CS Press, 2007, pp. 279–291.
14. H.S. Baird, M.A. Moll, and S.Y. Wang, "A Highly Legible CAPTCHA that Resists Segmentation Attacks," *Proc. 2nd Int'l Workshop Human Interaction Proofs*, LNCS 3517, Springer, 2005, pp. 27–41.

15. A.E. Ahmad, J. Yan, and L. Marshall, "The Robustness of a New CAPTCHA," *Proc. 2010 European Workshop System Security* (EuroSec 10), ACM Press, 2010, pp. 36–41.

Ahmad El Ahmad is a PhD student in the School of Computing Science at Newcastle University, UK. His research interests include computer security, particularly the design of secure and usable CAPTCHA systems. El Ahmad has an MSc in computing science from Newcastle University. He's a student member of IEEE. Contact him at ahmad.salah-el-ahmad@ncl.ac.uk.

Jeff Yan is a lecturer in the School of Computing Science at Newcastle University, UK, where he's a founding research director of the Center for Cybercrime and Computer Security. His recent research includes systems security and human aspects of security. Yan has a PhD in computer security from Cambridge University. He held a visiting faculty post with the Department of Information Engineering, Chinese University of Hong Kong. He is the corresponding author for this article. Contact him at jeff.yan@ncl.ac.uk.

Wai-Yin Ng is an associate professor in information engineering at the Chinese University of Hong Kong. His current research focus is in complex networks, a young, vibrant science concerned with connectivity, complexity, and emergent phenomena in both natural and artificial systems. Ng has a PhD in control engineering from the University of Cambridge. Contact him at w.ng@cantab.net.