This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

Emerging-Image Motion CAPTCHAs: Vulnerabilities of Existing Designs, and Countermeasures

Song Gao, Manar Mohamed, Nitesh Saxena, and Chengcui Zhang

Abstract—Based on the notion of "emergence", Xu et al. (Usenix Security 2012; TDSC 2013) developed the first concrete instantiation of *emerging-image moving-object* (EIMO) CAPTCHAs using *2D hollow objects* (codewords), shown to be usable and believed to be secure. In this paper, we highlight the hidden security weaknesses of such a 2D EIMO CAPTCHA design. A key vulnerability is that the camera projection on 2D objects is constant (unlike 3D objects), making it possible to reconstruct the underlying codewords by superimposing and aggregating the temporally scattered parts of the object extracted from consecutive frames. We design and implement an automated attack framework to defeat this design using image processing techniques, and show that its accuracy in recognizing moving codewords is up to 89.2%, under different parameterizations. Our framework can be broadly used to undermine the security of different instances of 2D EIMO CAPTCHAs (*not just the current state-of-the-art by Xu et al.*), given the generalized and robust back-end theories in our attack, namely the methods to locate a codeword, reduce noises and accumulate objects' contour information from consecutive frames corresponding to multiple time periods. As a countermeasure, we propose a fundamentally different design of EIMO CAPTCHAs based on *pseudo 3D objects*, and examine its security as well as usability. We argue that this design can resist our attack against 2D EIMO CAPTCHAs, although at the cost of reduced usability compared to the – *now insecure* – 2D EIMO CAPTCHAs.

Index Terms—CAPTCHAs, security, usability, computer vision.

1 Introduction

Almost every online service relies upon CAPTCHAs [5, 19] to thwart various forms of online attacks. The offensive-defensive research in CAPTCHA security (especially focusing on automated or auto- attacks) has continued [8, 9, 10, 14, 22] for more than ten years, advancing the science towards developing secure CAPTCHAs. In this paper, we follow the path of this established line of research explicitly focusing on a recently introduced breed of CAPTCHAs, called the *Moving-object* (MO) CAPTCHAs [20, 21].

MO CAPTCHAs embed a text identification task into dynamic moving objects, in an attempt to enhance both the security with respect to auto-attacks and the usability compared to static text-based CAPTCHAs. One of the representative commercial instantiations of MO CAPTCHAs is *NuCaptcha* [4], which challenges the user with a few characters flowing across a dynamic scene following a certain trajectory, and requires the user to recognize the highlighted characters (referred to as the *codeword*). Such designs add another layer of resistance to existing auto-attack algorithms used to break static text-based CAPTCHAs, because it is significantly harder to extract dynamic texts through tracking and distinguish the codeword from other characters. Moreover, since the codeword is relatively easy for humans to recognize, the usability of NuCaptcha might be better than most static text-based CAPTCHAs [20, 21].

However, Xu et al. [20, 21] recently reported compromising the security of NuCaptcha by using image processing and machine learning techniques with a success rate more than 70%. Similar work was done by Bursztein that uses single frame segmentation and recognition to break NuCaptcha [2]. As a countermeasure to their auto-attack, Xu et al. considered developing MO CAPTCHAs based on the notion of *emerging-images* (EI), introduced by Mitra et al. [13]. Theoretically, an *emerging-image moving-object* (EIMO) CAPTCHA makes it easy for a human user to perceive a shape from motion, while making it difficult, if not impossible, for auto-recognition using existing computer vision techniques. The authors proposed an instantiation of EIMO CAPTCHAs based on *2D objects*, which we call *EI-Nu* in this paper and is the focus of this study. Xu et al. demonstrated that EI-Nu exhibits a usability level similar to that of NuCaptcha, and showed that it is resilient to their attack against NuCaptcha and argued that it poses "significant challenges to existing computer vision methods" [21].

Let us first briefly review the notion of emergent images. In the seminal work that introduced this notion [13], the authors proposed emerging images of (pure) *3D objects*[1], and explained the emergence as "the phenomenon by which we perceive objects in an image not by recognizing the

- *S. Gao is with Google, USA (e-mail: gaoskddm@gmail.com)*
- *M. Mohamed is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: manar.kasem.mohamed@temple.edu)*
- *N. Saxena and C. Zhang are with the Computer and Information Sciences Department, University of Alabama, Birmingham, AL 35294 USA (e-mail: saxena,czhang02@uab.edu)*

1. A 3D EI-based video demo is available at: http://graphics.stanford.edu/~niloy/research/emergence/emergence_image_siga_09.html

object parts, but as a whole, all at once". As the authors indicate [13], "Humans cannot instantaneously detect the object in such images, and can probably recognize it only after several iterations that take into account numerous relationships between hypothetical objects and their context. The computational complexity of this human processing is believed to be extremely high [16], leading us to hypothesize that emerging images are hard for automatic algorithms to segment, identify, and recognize. Taking into account the complexity of the task, and the lack of a clear understanding of how humans solve the problem, it is highly unlikely, if not impossible, that these types of tasks could be carried out by bots in the near future."

The EI-Nu CAPTCHA proposed in [1, 20, 21] does have the above-mentioned characteristics of emerging images. However, in this paper, we identify several critical differences between the 2D object-based EI-Nu CAPTCHA of [20, 21] and the 3D object-based emerging image/video of [13], which seem to highlight the security weaknesses of the former. We explain these differences and weaknesses in Section 3.1. One of the key differences is that the camera projection is variable on 3D objects but constant on 2D objects, making it possible to reconstruct 2D objects by superimposing and connecting the temporally scattered parts of the object extracted from consecutive frames.

In this paper, we focus on investigating the security of 2D emerging CAPTCHA instead of (pure) 3D emerging CAPTCHA for the following reason. 3D emerging images have limited choices on the type of objects they can embed, because the visual details of the object are removed in a 3D emerging image in order to prevent automated attack, and a human's perception of emergence effect depends on their familiarity with the object. Therefore, only those simple objects that are commonly seen, and have *distinct* shape features, can be embedded into a 3D emerging image and must have a size large enough to be perceivable. This limits not only the diversity of such images but also the number of objects that can be embedded, therefore lowering the feasibility of easy design of secure and usable 3D EI CAPTCHAs.

Our Contributions: We show that the weaknesses of 2D emerging CAPTCHAs mentioned above can be exploited to effectively compromise the security of EI-Nu CAPTCHA of [20, 21], the current state-of-the-art in EIMO CAPTCHAs. Specifically, we report on an automated attack framework to defeat EI-Nu CAPTCHA, and propose fundamental countermeasures based on *pseudo (not pure) 3D effects* and *real dynamic background*. As such, we believe that we are advancing the state of science in CAPTCHA security by: (1) demonstrating and quantifying the security limitations of a CAPTCHA type previously believed to be secure, and (2) proposing and evaluating countermeasures. A repository of our CAPTCHA samples, attack demonstrations and defense strategies are available online at: https://sites.google.com/site/breakingeimo1/. Our specific contributions in this paper are as follows:

1) *Design of a Novel Automated Attack Framework:* We designed an attack framework using image processing to recover the shape contour of the codeword in a "bottom-up" manner, i.e., by reconstructing the shape contour first from individual binary masks, then from the combination of a set of consecutive masks, and finally from the combination of multiple sets. An extensive evaluation of thousands of CAPTCHA challenges (including limited samples available from developers' own website [1]) demonstrates that our attack can undermine the security of EI-Nu CAPTCHAs with a non-trivial success rate (i.e., up to 89.2%) under different parameterizations (Sections 3 and 4).
2) *New Countermeasures and Usability Evaluation:* We develop a set of countermeasures to enhance the security of EI-Nu CAPTCHAs (broadly applicable to 2D EIMO CAPTCHAs in general). Specifically, we propose a novel variant based on pseudo 3D effects (i.e., rotation and scaling) and dynamic background with dynamics mainly existing in low geometric details that can be well transmitted into EI-based frames, examine its security, and evaluate its usability. We demonstrate that this variant can resist almost all auto-attacks, in our knowledge, based on accumulated information. This improvement in security, however, comes at the cost of reduced usability compared to the - *now shown insecure* - 2D variant (EI-Nu). (Section 5).

Significance and Impact of Our Work: Emerging images represent a relatively new notion being applied to the CAPTCHA domain, and promise to defeat most existing object detection methods that rely on common visual features, such as edge, color, and corner point. The *significance and impact of our work* therefore lies in systematically exploring and highlighting the security vulnerabilities of a representative instance of such CAPTCHAs, and associated countermeasures. While such CAPTCHAs have not currently been deployed, they may soon be given their attractive properties. Our work represents a preemptive effort to analyze the security of these CAPTCHAs in advance of their deployment.

Generalized and Robust Attack/Countermeasures: Our attack targets EI-Nu CAPTCHAs, "*the first concrete instantiation of the notion of Emerging Images applied to captchas*" [20, 21] with certain parameter setting, such as the minimum bounding rectangle (MBR) of a codeword, the number of characters in a codeword (e.g., 3), and the threshold of the trivial segment size. However, *without loss of generality*, our methods to locate a codeword, reduce noises, and accumulate edges from consecutive masks of multiple time periods, are generally-applicable and robust to defeat all similar 2D EI designs. In the same vein, our countermeasures are generally applicable to other 2D EI designs.

2 Background

The authors in [20, 21] proposed a 4-step process in generating 2D EI-Nu. However, it is difficult to reproduce the exact same effect by following their instructions due to a lack of detailed explanation about implementation choices such as parameter settings. Therefore, we independently reconstructed EI-Nu CAPTCHAs by following the original instructions with certain modification as explained below, followed by the proof of visual equality of the reconstructed version and the original EI-Nu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

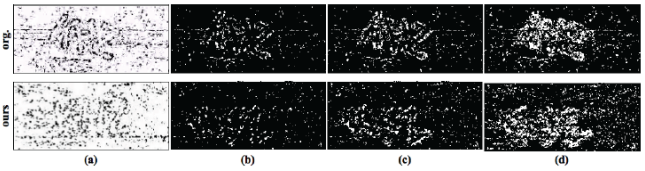


Fig. 1: Comparison between the original EI-Nu CAPTCHA and our design with codeword "KHZ". (a) A single frame image. (b) A single binary mask after removing the background scene. (c) Superimposition of 2 consecutive binary masks. (d) Superimposition of 5 consecutive binary masks.

1) Generate a noisy frame I_{bg} with each pixel following a Gaussian distribution (same as the original instruction). Due to a lack of instructions for mixing the current frame and the previous frame to preserve the temporal continuity, we implemented our own method based on our best understanding of the principal and best effort to observe the mixing pattern. Here *common pixels* refer to those object edge pixels shared between the edge mask of the current frame and that of the previous frame, while *discrepant pixels* denote those edge pixels that only exist in the current frame. The temporal continuity is preserved by allowing a certain percentage of common pixels to be displayed again in the current frame. The formula for generating the combined frame I (in Step 3) indicates that pixels with smaller I_{bg} values have a higher possibility to be displayed as black pixels. Therefore, common pixels with non-negative I_{bg} values and discrepant pixels with negative I_{bg} values exchange their values. Discrepant pixels are exchanged in ascending order of their I_{bg} values until the largest negative pixel is reached. A percentage parameter ρ is used to control the exchange percentage of common pixels so as to control the degree of temporal continuity. The swap operation preserves the current Gaussian distribution in I_{bg}. Finally, a median filter is used to blur the noise frame.
2) Generate the edge mask I_{fg} for both foreground codeword and the background scene (same as the original instruction).
3) A combined frame I is created by combining the foreground edge mask I_{fg} and the noisy frame I_{bg} using: $I(x, y) = I_{bg}(x, y) \times exp(I_{fg}/const)$, where $exp(x)$ is the exponential function. As indicated in [20, 21], in this way, pixels near the edges of characters in I can be made noisier than other pixels (same as the original instruction).
4) In the original Step 4, the final binary frame is generated through binarizing I by setting all the pixels whose values are greater than a user-defined exposure degree threshold t <0 to 'white', and the remaining pixels to 'black'. However, it is difficult to reproduce the exact same effect by following the original instructions due to a lack of explanation of the value selection for t. Since the pixel values in a noisy frame follow the Gaussian distribution, blurring (like in the original Step 1) could shrink the value range, which increases the threshold sensitivity of t in Step 4. Consequently, a small change of t value would often lead to too many/few white pixels being generated, easily causing the resulting frame to be visually dissimilar to that in the original design. To address this issue, two thresholds, i.e., $t_1 < t_2 < 0$, are used to generate the final binary frame. In particular, t_2 is applied only to the pixels on the contour of codewords and its surrounding area, and t_1 is applied to the noise background, enabling a finer granularity of control and thus reducing the sensitivity when only one threshold is used.

To prove that our version produces visually similar effects as the original EI-Nu CAPTCHA, we provide a comparison of the two versions with respect to two primitive visual similarity measures:

- A single screen-captured frame image.
- A single binary mask of a codeword, or superimposition of multiple consecutive binary masks. The method to generate such masks will be introduced in Section 3.

The first standard is used to check whether similar visual effect is presented to the human user, while the second standard relates to whether similar clues are exposed to attacking algorithms. As shown in Figure 1(a), only partial shape contours, represented as edge segments, for both codewords and the background scene are exposed in a single frame image. The codeword shape is blurred in overlap of 5 consecutive binary masks (Figure 1(d)). Moreover, the exposed contour (i.e., edge segments) of the codeword in a single frame, a single binary mask, and superimposition of 2 consecutive masks from the reconstructed version is perceivably less than that of the original EI-Nu CAPTCHA. Therefore, our design is at least as secure as or even more secure than the original design in terms of resilience to vision-based auto-attacks.

3 Attack on EI-Nu CAPTCHA

In this section, we highlight the weaknesses of the EI-Nu CAPTCHA, and describe our automated attack framework that exploits these weaknesses.

3.1 Design Weaknesses

We have identified following weaknesses in the EI-Nu CAPTCHA design, the first three of which are significant.
1) **2D Camera View:** The recognition of 2D shapes is relatively easier than that of 3D shapes since the former have a constant camera projection, which as we mentioned earlier, makes it easier to connect scattered parts from consecutive frames into meaningful objects after superimposition.
2) **Dynamic Background:** Even though the input edge mask I_{fg} consists of both the codewords and the background scene, as will be seen in Section 3.2, the background can be recovered to some extent by collecting clues from consecutive frames. With the background mask removed from each frame, the task of codeword auto-recognition becomes easier.
3) **Accumulated Knowledge:** Although each character rotates within a certain angle range, overlaps with neighbor characters to some extent, and moves up and down in a wavy motion, there is a primary moving direction (e.g., left to right) with a low to moderate moving speed while other movements between two consecutive frames are comparably trivial. Therefore, once the location of the codeword is identified, it is possible to superimpose consecutive frames based on this primary direction so as to collect contour segments of the codeword.
4) **Puzzle Data Size:** Each CAPTCHA challenge loops continuously within a short video (6 seconds) to: (1) limit

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

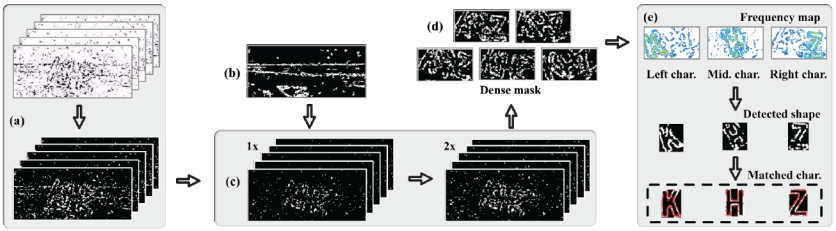


Fig. 2: Auto-attack framework for EI-Nu CAPTCHA. (a) Collect frame images and convert them into binary masks. (b) Detect background scene mask. (c) Remove background scene mask from each binary mask (1x), and generate overlapped masks (2x). (d) Generate dense masks from overlapped masks. (e) Shape separation and character recognition (In the dashed rectangle, red: detected shapes; white: templates). (This and other figures are best viewed in color.)

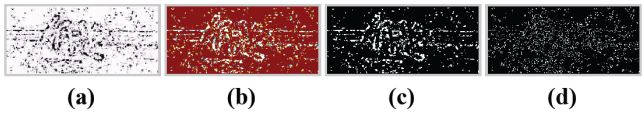


Fig. 3: The capture of frame images. (a) Original frame; (b) Color code image; (c) Binary mask; (d) Removed noise.

the CAPTCHA file size; and (2) prevent the accumulative collection of codeword contours. For example, the longer the video, the more likely it is for the codeword to reappear in the same orientation but with different contour segments (due to the randomness in generating I_{bg}), facilitating the reconstruction of codeword shapes. However, one problem with using a short video is all the information is stored in a relatively small number of frames, making it computationally more feasible for automated algorithms to mine these patterns.

5) **Color:** Since each frame is a binary image, only black pixels (which usually constitute the smaller portion of a frame) that come from the edge mask, and random noises need to be explored by the automated attack program that is computationally feasible.

3.2 Our Attack Framework

Our EI-Nu attack framework consists of five steps (Figure 2).

1) ***Capture of Frame Images:*** A set of consecutive frames is screen-captured from an EI-Nu CAPTCHA challenge, and converted into binary masks (Figure 2(a)).

2) ***Background Scene Detection:*** The background scene is reconstructed by accumulative learning from the binary masks multiple times (Figure 2(b)).

3) ***Preprocessing of Binary Masks:*** The background edge is removed from each binary mask. Multiple consecutive masks (e.g., 2) are superimposed to form a clearer shape of codewords, which is called overlapped mask (Figure 2(c)).

4) ***Generation of Dense Masks:*** A set of dense masks is generated from the overlapped masks (Figure 2(d)).

5) ***Character Separation and Recognition:*** All dense masks are matched with each other to form the frequency map for each character. The detected shape is extracted from the frequency map using a constant frequency threshold. A Generalized Hough Transform (GHT) [6] is performed to match each shape to a character template (Figure 2(e)). See Appendix A for further explanation of the use of GHT.

3.2.1 Step 1: The Capture of Frame Images

As mentioned in Section 3.1, each CAPTCHA challenge loops continuously within a 6-second video clip. We screen-capture consecutive frames from this 6-second clip. The shape recognition only needs to focus on segments consisting of black pixels, which represent codewords, background scene and random noise. However, due to the quality loss in the conversion from frame images to a video clip in generating the CAPTCHA, a screencaptured frame image becomes a color image with a very low saturation (i.e., grayish). To obtain a binary frame mask, we first convert an RGB frame image into an image I with color code representation [24], which replaces a pixel value with a 6-bit color value consisting of the highest two bits from each 8-bit RGB channel. A sample color code image is shown in Figure 3(b), in which different colors represent different color code values. First, such a simplification operation will drastically reduce the number of colors from 256^3 to $2^6 = 64$, ease the selection of the threshold value, and remove noises. Second, color code can help group pixels with similar colors into the same code, facilitating the binarization process. We empirically select the dominant bin value (e.g., 21 in our case) that is the second darkest among all dominant bins from colorcode histogram as the threshold t_b for binarization. All pixel values in I that are lower than or equal to t_b are made white (Figure 3(c)), and black otherwise. A lot of pixels irrelevant to the codeword as well as some edge pixels from the codeword are removed in this binarization procedure (Figure 3(d)).

3.2.2 Step 2: Background Scene Detection

As mentioned in Section 3.1, one of the drawbacks of the original 2D EI-Nu design is the use of a weak dynamic background (Figure 6 in [1]) that is not sufficiently dynamic because most dynamics/movements exist in the high geometric details in the original natural scene videos but get largely lost during conversion into binary EI-based frames, while low geometric details (the relatively static video content) are largely preserved in the resulting binary frames, which is an innate limitation of binary emerging images. Therefore, such a background scene is ineffective in hiding the location of foreground codewords. As can be seen from the EI-Nu samples on our demo website, certain dynamics from the original natural scene video is still preserved, however not sufficient to border against our proposed attack especially the recovery and removal of background which is the key to the successful extraction of the codeword.

For the four natural scene videos used in the experiments of [20, 21], we convert each frame into gray image and recover the background by treating the most frequent gray value among all frames as the true color from background. Our study shows that this operation can recover most con-

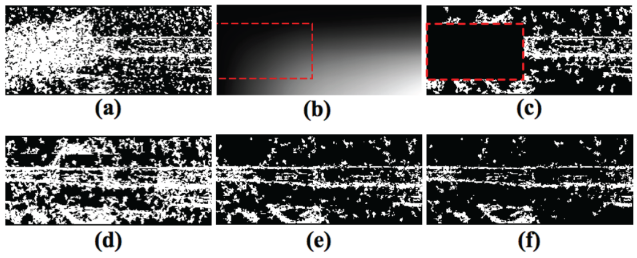


Fig. 4: Background Mask Detection (corresponding to the natural scene in Figure 4(g)). (a) Superimposition of 25 consecutive binary masks. (b) Localization of the codewords in integral image. (c) Incomplete background mask after removing the overlapping codewords area fMBR (dashed rectangle). (d) A single complete background mask Bi. (e) Intersection of 5 complete background mask. (f) Intersection of 20 complete background mask.

tents of all four natural scene videos used in [20, 21]. Next, we will show how a video background like the one used in original EI-Nu CAPTCHA (Figure 1), despite the existence of moderate vibration effect and random noises, can be extracted automatically. Further, based on the observations from this study, we implemented new countermeasures with dynamic background (details in Section 5.2).

This process starts with randomly selecting a starting frame (e.g., ith frame) from the n collected binary masks. Let $b|_i^{(i+k-1)mod\ n}$ denote the incomplete background mask (Figure 4(c)) generated by superimposing k consecutive binary masks (Figure 4(a)) that corresponds to a very short period Δt and removing the subarea f_{MBR} of overlapping codewords. To construct a complete background, multiple incomplete backgrounds are generated while the codeword travels across the background scene. For this purpose, after the **first** starting frame (i^{th} frame) is selected and the first incomplete background mask is constructed, the process moves on to select the second starting frame ((i+k)$^{\text{th}}$ frame) and generate the second incomplete background mask, and continues until all the removed foreground subareas, when superimposed together, cover the full width of the CAPTCHA window, or the maximum number of iterations is reached. One complete sweep of the codewords across the CAPTCHA challenge window is usually sufficient to provide enough incomplete background masks to form a complete background mask B_{i}, but often with a lot of noises (Figure 4(d)). Since noise pixels appear randomly in each frame, one way to reduce noise in the background mask is to generate multiple background masks using different **first** starting frames (e.g., i^{th}, (i+1)$^{\text{th}}$, (i+2)$^{\text{th}}$,.., (i+p)$^{\text{th}}$ where p<k), and group the common white pixels shared by those background masks to form the final background B (Figure 4(e-f)).

Background detection is described in the following equations, in which $olp(i,k)$ is the superimposition of k (e.g., 25) masks starting from the i^{th} mask, r is the number of iterations to generate one complete background mask, and p is the total number of complete background masks used to generate the final background mask B.

$$b|_i^{(i+k-1)mod\ n} = opl(i,k) - f_{MBR} \qquad i \subseteq [1,n] \tag{1}$$

$$B_i = b|_i^{(i+k-1)mod\ n} \bigcup b|_{(i+k)mod\ n}^{(i+2k-1)mod\ n} \bigcup \ \cdots \ \bigcup b|_{(i+(r-1)k)mod\ n}^{(i+rk-1)mod\ n} \tag{2}$$

$$B = B_i \bigcap B_{(i+1)mod\ n} \bigcap \ .. \ \bigcap B_{(i+p-1)mod\ n} \tag{3}$$

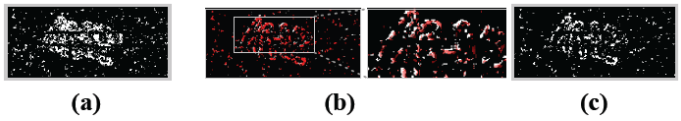


Fig. 5: Effects of superimposing and shifting on binary mask of the original EI-Nu CAPTCHA. The background is removed in the above images. (a) Superimposition of 5 consecutive binary masks. (b) The degree of discrepancy of the codewords between 2 consecutive masks - the global view and the zoomed-in view. Red color represents pixels from (i- 1)$^{\text{th}}$ frame, while white color represents pixels in the i^{th} frame. (c) 2x-overlapped mask based on the two masks in (b).

The remaining task is to locate the overlapping codeword subarea f_{MBR} from a superimposed mask. Based on some a prior knowledge, (e.g., the number of characters nc, the font and font size) in a CAPTCHA challenge, a minimum bounding rectangle (MBR) can be defined, the dimensions of which are determined by the maximum possible MBR for the nc characters arranged side-by-side. As mentioned in Section 2, pixels near the edges of characters are made noisier than other pixels, meaning that there are more noise pixels near the edges than the other areas. Therefore, the task to locate the overlapping codeword subarea is to find the MBR with the maximum pixel count in the superimposed binary mask. The *integral image* [18] is used to quickly calculate the pixel count in an MBR (Figure 4(b)). The size of an MBR is further adjusted, usually made smaller, after the removal of trivial segments (e.g., <10 pixels) from the MBR.

3.2.3 Step 3: Preprocessing of Binary Masks

Both the background scene and the random noise introduce errors in auto-recognition of the codewords. It is necessary to remove these artifacts from the binary mask before the shape detection. However, since the background scene and the random noise share the same color with the codewords, removing them may also remove some contour/edge pixels of the codewords. The generation instruction of EI-Nu CAPTCHA, as mentioned in Section 2, suggests that the edge segments preserve certain level of temporal continuity. Therefore, the higher the temporal continuity, the higher the probability that similarly shaped segment will appear again in the current frame at about the same location, thus revealing less new edge information. Preserving temporal continuity is critical in preventing the fast revelation of the character's shape. Superimposing multiple consecutive binary masks over a relatively long period (e.g., ≥5 frames) will blur the character shape due to self-rotation and movements (i.e., horizontal and/or vertical movement) of the codeword (Figure 5(a)). However, there is an upper bound for both the rotation speed and the moving speed for the sake of usability of solving CAPTCHAs. If the movement is too much and too fast, the user may feel dizzy when watching the CAPTCHA, in turn decreasing the usability.

Thus, superimposing consecutive binary masks over a relatively short period (e.g., < 5 frames) may still be used to recover partial shape contours with an acceptable degree of distortion. Figure 5(b) shows the degree of discrepancy of the codewords between two consecutive binary masks of an original EI-Nu challenge, which indicates that the orientation difference of a character, and the wavy up and down movement between TWO consecutive frames (i.e., 2x-

overlapped mask, Figure 2(c)) are very minute. First, the orientation difference of a character between two consecutive frames is trivial. If such rotation degree per frame is large, e.g., 10 degrees per frame and the frame rate is 30 fps, the character will rotate 150 degree in 0.5 second, in which case even human could barely recognize the character. Second, the height of the CAPTCHA window restricts the up and down moving offset between TWO consecutive frames. The wavy up and down is controlled by a sine function y(t)=$A\times$sin($\omega\ t+\psi$), where y is the vertical position of the letter, t is the frame id, and A, ω, ψ are adjustable parameters. The vibration needs to be restricted under certain level that does not undermine the usability severely. For example, assume the moving offset is 3 pixels per frame and the frame rate is 30 fps. The codeword can vertically travel across 90 pixels in one second, which corresponds to about 72% of the height of the CAPTCHA window (i.e., 125 pixels) and more than half of the sine period. It is difficult and inconvenient for a human to recognize codewords at such fast speed. Therefore, 1 or 2 pixels per frame for up and down movement are more user-friendly, resulting in relatively trivial frame-to-frame difference in at least 2x-overlapped masks compared with the character size and edge thickness.

The preprocessing on the binary masks consists of two steps: 1) subtract the background scene mask from each binary mask, and 2) generate the *kx-overlapped* mask, which accumulatively collects contour information from the combination of k consecutive masks through shifting and superimposing operations. To determine the value of k, factors to consider include the extent of exposure of the shape contour in consecutive masks, the moving speed of the codeword, and the frame rate. If the extent of exposure is low (e.g., high temporal continuity), a small k cannot provide enough contour information. However, if the moving speed and/or the frame rate is high, a large k may over-blur the shape contour. Our design may have a different moving speed and/or frame rate than the original design. We find that the optimal k for attacking the original EI-Nu CAPTCHA is 2, while the optimal k for attacking our design is 3.

Step 2 includes two operations, shifting and superimposing. To generate the i^{th} 2x-overlapped mask, without loss of generality, the previous mask $b_{i\text{-}1}$ is left-shifted by 1 pixel and superimposed on the i^{th} binary mask b_i. Note that the shifting offset value vary depending on the moving speed of codewords. The moving speed can be estimated by measuring the distance between the two MBRs' centroids that come from the beginning and ending of a set of consecutive frames. Also, the frame rate can be calculated by counting the number of unique frames per second collected using high sampling rate (e.g., 1/90 second). In our attack on the original EI-Nu, a constant shifting offset due to the constant moving speed.

As shown in Figure 5(c), the 2x-overlapped mask of an original EI-Nu CAPTCHA exposes more shape information, which can be used in character recognition. To generate the i^{th} 3x-overlaped mask, there are two options in selecting the two neighbor masks ($b_{i\text{-}2}$, $b_{i\text{-}1}$, b_i) or ($b_{i\text{-}1}$, b_i, b_{i+1}). Because of the temporal continuity between consecutive frames, the difference of edge segments between $b_{i\text{-}2}$ and b_i is larger than that between b_{i+1} and b_i, therefore, exposing more shape information when superimposed. In our experiment, we use ($b_{i\text{-}2}$, $b_{i\text{-}1}$, b_i) to generate the i^{th} 3x-overlapped mask. The shift distance of $b_{i\text{-}2}$ is twice as much as that of $b_{i\text{-}1}$.

3.2.4 Step 4: Generation of Dense Masks

Only partial shape exists in a 2x-overlapped mask and is not sufficient for separation of characters from each other or for character recognition. One way to obtain better shape information is to use more consecutive frames in generating kx-overlapped masks. However, using too many consecutive frames (e.g., $k \geq 5$) will introduce too much noise due to character rotation and movement. Another way is to use multiple kx-overlapped masks, where k is a relatively small value, in order to collect more shape contour via cross-matching between two or more such overlapped masks. To match the codeword shape in one kx-overlapped mask with that in a neighbor overlapped mask, a divide-and-conquer strategy is adopted to match each subarea that contains only one character in an overlapped mask with the corresponding subarea in a neighbor mask by testing different orientations and locations of each character subarea during matching.

First, the codeword subarea f_{MBR} in each kx-overlapped mask (e.g., k=2) is extracted by using the same method presented in Section 3.2.2. Since the background scene has been removed from an overlapped mask, the codeword subarea becomes the densest subarea, and thus easy to detect. Not all detected MBRs provide the correct location of the codeword. For example, sometimes the background scene is not thoroughly removed, or the codeword is split into two parts and appears on both sides of the CAPTCHA window. An MBR may contain partial codeword and partial background scene segments or noises. Such MBRs are removed based on the following two criteria:

- The width of an MBR must be larger than a predefined threshold that indicates the minimum width of the side-by-side arrangement of three characters
- The total pixel count in an MBR must be larger than a predefined threshold that indicates the minimum segment pixel count of an incomplete codeword contour.

Instead of exhaustively matching an MBR with all of its neighbor MBRs, only a few MBR candidates are selected based on their entropy values for the sake of efficiency. The MBR entropy is calculated as $H(X) = -\sum p(x_i) log_2 p(x_i)$, where X denotes an MBR, $x_i \in X$ is a 9×9 block of pixels, and $p(x_i)$ denotes the pixel density of the i^{th} block, i.e., the number of white pixels divided by the area of the cell. Usually, the lower the entropy, the more evident and cleaner the shape contours contained in that MBR. Top qualified MBRs are selected one by one in the ascending order of their entropy values, and typically no more than 5 top candidates are selected.

Further, each MBR candidate, together with the MBRs of its previous k neighbors (e.g., k=4), is horizontally and evenly divided into three subareas, each of which roughly contains one character. Subareas in each neighbor MBR are treated as templates while subareas in the current MBR candidate are treated as the target. Generalized Hough Transform (GHT) [6], which is an object detection method for arbitrary shapes, is applied to match the character shape

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

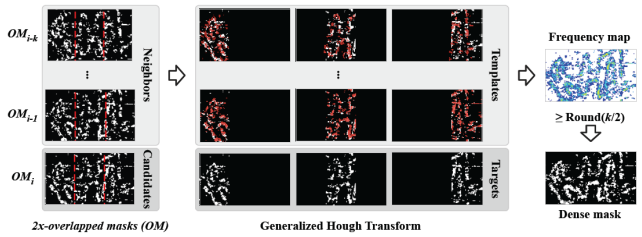


Fig. 6: Flowchart for generating the dense mask.

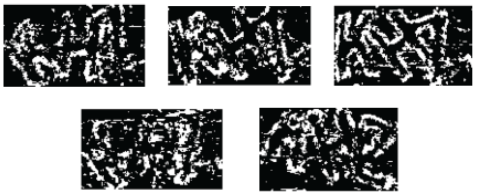


Fig. 7: The 5 dense masks corresponding to the top 5 MBR candidates (codeword: "KHZ").

between the MBR candidate and its neighbors. The spatial correlation among several consecutive masks results in a small rotation range (e.g., $\pm$10 degrees) used in GHT. Moreover, the temporal continuity preserves the overlapping of certain edge segments of the same character from consecutive masks. A frequency map (see Figure 6) is generated to show the matching frequency of each white pixel in the MBR candidate. Then, a *dense mask* for each MBR candidate is created through binarizing the corresponding frequency map by using a fixed threshold (i.e., $Round(k/2)$), in which white pixels represent those that are highly frequently matched to the templates from neighbor MBRs. The above process is illustrated in Figure 6. The dense masks corresponding to the top five MBR candidates are shown in Figure 7.

3.2.5 Step 5: Character Separation and Recognition

The main challenge in breaking static text-based CAPTCHAs, such as Google CAPTCHA and reCAPTCHA, is to separate characters from each other in order to perform optical character recognition (OCR) [7]. A similar challenge exists in breaking EI-Nu CAPTCHAs, due to the overlap between characters. Unlike static text-based CAPTCHAs, or NuCaptcha, that can utilize some robust features such as corner points and closed loops to determine the boundary between characters, the character contour, consisting of edge segments, in EI-Nu CAPTCHA is often unsmooth and incomplete. Therefore, it is challenging to find cutting points on edge segments to reliably separate characters.

To address the above problem, we apply the same cross matching method (Figure 6) to dense masks in order to find the frequently matched pixels in each subarea group (i.e., one group for each set of left, middle, and right subareas). For each subarea group, the target image is chosen as the one with the maximum pixel count, while the other subareas are used as templates. As shown in Figure 8, the first dense mask is selected as the target in shape detection of the left character. Pixels with matching frequency larger than a threshold (e.g., 2 for 5 masks) are included in the final character mask for that subarea group. Since the variation in the orientation of the same character in different dense masks may be significant, a relatively large rotation angle range (e.g., $\pm$60 degrees) is used in GHT. Since the middle character overlaps with both the left and the right characters, its detection is also largely affected by the detection of those two characters. Therefore, the left and the right characters are detected before the middle character. In particular, the middle character is detected based on the frequency map after the pixels of the other two characters are removed.

In recognizing characters, it is unreliable to use the character pixel count [23] in the left, middle, or right subareas due to the irregularity of edge segments. For the sake of efficiency and simplicity, we again rely on GHT to compare the detected shapes (used as object templates) with known character templates (used as target images). Assume the pixel counts of the template and the target image are N_1 and N_2, respectively. The matching similarity scope P is calculated in (5), where comm(N_1, N_2) returns the count of common pixels shared by the template and the target image: $P = p_1 \times p_2 = (comm(N_1, N_2)/N_1) \times (comm(N_1, N_2)/N_2)$.

It is noteworthy that evenly dividing the MBR into three subareas could undermine the shape contour detection due to unequal width of characters (e.g., "W" and "M" are wider than most others while "J" is narrower.) The fewer the characters in a codeword, the less is such an impact because there are fewer division errors. For example, according to our experimental results on 101 4-character codewords (3 attack attempts each), a codeword such as "WWK" is fully recognizable because the division error did not exceed the mitigating capability of GHT, while "MWWK" could not be fully recognized. In this batch of experiments, 40 out of the 101 codewords contain a mix of normal-sized characters and wide/narrow characters such as "M", "W", and "J". The overall average success rate is $\sim$ 67%, while the average success rate on unequal-width codewords is 30%. Some successful attacks on unequal-width codewords include "49GW", "9VWZ", and "368W", etc. As for "MWWK", although our algorithm never recognized the word completely correctly, the best result is "FWWK" which is pretty close and can be acceptable if the CAPTCHA does not demand a 100% correct answer (such as the Multi-digit Number Recognition from Street View[2] and Google reCaptcha[3]).

It is also worth noting that character combinations such as "MW", "VW", "83", "5S", "O0Q", "1lI", "NM" should be avoided in generating codewords, since they are also hard for human solvers to distinguish in EI videos. However, in our experiments on the 101 4-character codewords, we did not exclude those combinations. The actual success rate of our algorithm could be higher in a more practical setting that excludes those combinations.

4 Attack Implementation and Evaluation

Our attack is implemented in MATLAB, except of some functions, e.g., rotating image, calculating integral image and matrix matching, are implemented with C based APIs in OpenCV. Our experiments with the attack framework focus on evaluating the security of the EI-Nu CAPTCHAs under

2. http://arxiv.org/abs/1312.6082
3. https://www.cs.cmu.edu/~biglou/reCAPTCHA_Science.pdf

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

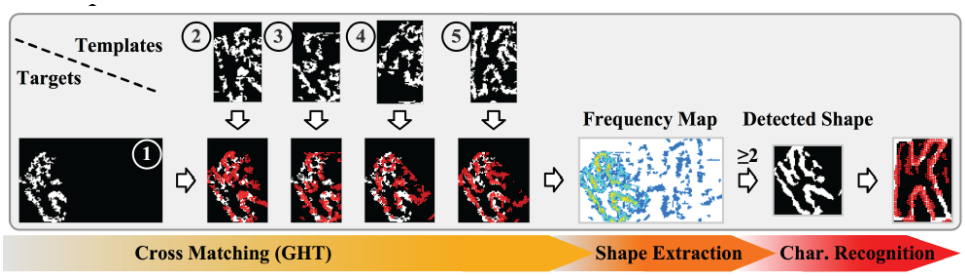


Fig. 8: Separation of left character in a codeword, and character recognition using known character template images. The number labels are the dense mask IDs. Different colors in the frequency map represent different matching frequencies: sky blue (1), cyan (2), green (3), yellow (4), and brown (5).

different difficulty levels. All experiments are performed on a MAC air laptop with hardware configuration: 2 GHz Intel Core i7, 8GB 1600MHz DDR3, and 250G SSD; and software configuration: OS X 10.8.5, MATLAB R2013b.

To evaluate our attack, we first implemented our versions of the EI-Nu CAPTCHAs as described in Section 2.2. In our implementation, each challenge loops on a 6-second video clip that uses a canvas of size 285(W)×125(H). A codeword moves across the background scene from right to left with constant speed (represented as *pixels per frame* (*ppf*)), and moves up and down harmonically. Each character in a codeword also rotates around its centroid. A vibration effect is implemented for the background scene by randomly shaking the scene image up and down. The frame rate is 27 fps when moving speed of codewords is 2 ppf, and 20 fps for 3 ppf for better perception to human users.

As mentioned in Section 2, three parameters were used in our CAPTCHA implementation against which our attack was evaluated: (1) A percentage parameter ρ is used to control the degree of temporal continuity; (2) A parameter *const* is used to calculate the intensity image I; and (3) Different t values ($t < 0$) are used to generate the binary masks for codeword, background scene, and random noises. Table 1 shows the range of the parameters used for controlling the segment density of codeword, background scene, and noises. In all experiments, we only vary the parameter values for codewords since they determine the extent of exposure of shape contour (e.g. a challenge with t_2 = -0.75 have about 6% less codeword segment pixels than a challenge with t_2=-0.70). A set of constant parameter values is applied to generate segments of background scene and noises in each frame, thereby maintaining a similar visual effect as the original design. Two different horizontal moving speeds for a codeword, i.e., *ppf* = {2, 3}, are tested to show its security impact. A color image (Figure 11(a)) is used to generate the background scene edge mask. The templates used in attacking the original EI-Nu CAPTCHA are obtained by copying the character contour from NuCaptcha challenges (available from NuCaptcha website), and scaling it to an appropriate size. In our versions, the font style and size of each template is Arial narrow bold and 60 pt.

Before attacking our versions, we applied the attack on the two original EI-Nu CAPTCHA challenges with codeword "KHZ" and "7FX", 100 times each. Due to the randomness in selecting the first frame for detecting the background in Step 2 (Section 3.2.2), the detected background can be slightly different. Therefore, the best dense masks may not always be selected, leading to errors in character recognition. In order to improve the efficiency of GHM matching, we scaled down each detected shape by 20%. The average success rates are 65% and 52%, respectively, in attacking the two challenges. A lower accuracy in recognizing "7FX" than that of "KHZ" is partially due to the mismatch in detecting the shape of "7" with template "T". The detected dense masks and the shapes of the "7FX" challenge are shown in Figure 9. This demonstrates that our attack can effectively solve the EI-Nu challenges proposed in [20, 21].

Fig. 9: Results of dense mask candidates and character recognition for attacking the original EI-Nu CAPTCHA.

TABLE 1: Range of parameters used in generating CAPTCHAs.

	ρ	$const$	$t < 0$
Codeword	{0.5, 0.6, 0.7, 0.8, 0.9}	0.6	{-0.70, -0.75} (t_2)
Bg. Scene	0.5	0.3	-0.40
Noises	0.5	0.6	-0.85 (t_1)

The first experiment with our EI-Nu CAPTCHAs is to explore the security impact of the moving speed and the threshold t_2 that controls the segment density of codeword. Two different degrees of temporal continuity {0.5 and 0.7} are tested . Together with two moving speeds {2 and 3} ppf and two t_2 values {-0.70 and -0.75}. A total of 8 EI-Nu groups have been tested, each group contains 500 CAPTCHA challenges with randomly generated unique 3 characters codeword. In order to adapt the attack to our version of CAPTCHAs, two parameter settings are tuned in Step 3 (Section 3.2.3). First, 3x-overlapped mask that combines 3 consecutive binary masks, are generated in Step 3 instead of 2x-overlapped mask. Moreover, when the moving speed is 2 ppf, the 1st neighbor mask (i.e., b_{i-1}) is left-shifted by 1 pixel, and the 2nd neighbor (i.e., b_{i-2}) is leftshifted by 2 pixels. When the moving speed is 3 ppf, the shifting offset is adjusted to 2 and 4 for the 1st and 2nd neighbor, respectively.

TABLE 2: Success Rate (%) of the Proposed Attack under Different Moving Speed and Threshold t_2.

TC	Speed(*ppf*) \ t_2	-0.70	-0.75
0.5	2	**89.2**	80.4
	3	76.8	**66.2**
0.7	2	**57.8**	51.6
	3	48.8	**40.6**

Table 2 indicates that the success rate decreases with the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

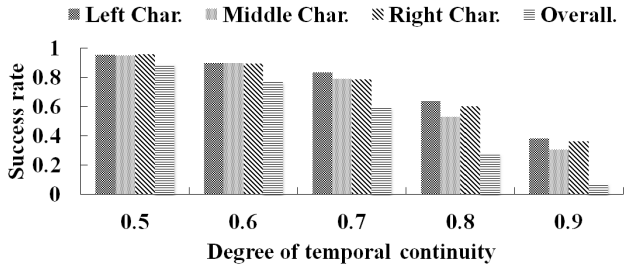


Fig. 10: Success rate of each character location in a codeword and overall success rate under different degrees of temporal continuity.

increase of the codeword moving speed and/or the decrease of the threshold t_2. Each character in a codeword rotates, and moves up and down, so there is a certain degree of distortion on the codeword shape in the 3x-overlapped mask. A higher moving speed will further distort the object's shape in the overlapped mask. Decreasing the threshold t_2 allows less amount of contour pixels to be displayed in the final binary frame. Thereby, the completeness of the detected shape from speed 3 CAPTCHAs with a small threshold t_2 is worse than that of speed 2 with a relatively larger t_2.

In the second test, we focus on assessing the security impact of the temporal continuity. As mentioned in Section 2, to preserve the temporal continuity between two consecutive binary frames, a selected common pixel always swaps its I_{bg} values with the current largest discrepant pixel that has a negative I_{bg} value. The parameter ρ, which relates to the degree of temporal continuity, is varied from 0.5 to 0.9 in our experiment at a fixed interval 0.1, while t_2 = -0.7 and moving speed = 2 ppf.

As shown in Figure 10, the success rate decreases when the degree of temporal continuity increases. A higher degree of temporal continuity implies that common pixels have a higher probability to be displayed in both the current frame and the previous frame while the display of those discrepant pixels with the largest negative I_{bg} values are more likely to be suppressed, leading to less revelation of new information in the current frame. Therefore, under a higher temporal continuity, in order to collect the complete shape contour information, more consecutive masks need to be analyzed, making the challenge more difficult to break. However, while consecutive masks within a short period ($<$5 consecutive masks) may not be able to provide enough contour pixels to recover an object shape, consecutive masks in a long period may over-blur the shape contour after superimposition. Therefore, such a dilemma leads to a very low success rate of the proposed attack when the temporal continuity is very high, such as 0.9. We did not experiment with any ρ value lower than 0.5 because the success rate is already very high at this degree of continuity.

In each group of the above test, the success rate in recognizing characters in each location of a codeword is summarized in Figure 10. Take the left character as an example, the *success rate* is calculated as the ratio of the number of recognized left characters to the total number of challenges in a test group (i.e., 500). Since the middle character overlaps characters from both left and right sides, it is more difficult to detect.

Finally, the evaluation of the processing time of the proposed attack is analyzed from two aspects, namely, the total completion time of an attack and the proportion of time of each step. In all our previous groups of test, the average completion time of an attack is relatively stable (i.e., 96.15 seconds) given a specific degree of temporal continuity (i.e., ρ=0.5). As the degree of temporal continuity increases from 0.6 to 0.9 , the average completion time decreases from 92.74 to 76.87 seconds due to the decrease in the number of recovered contour pixels. Fewer contour pixels result in faster matching between the detected shape and the templates. However, the attack success rate decreases sharply (Figure 10).

All modules in each step are executed linearly. The bottleneck is in Steps 4 (19% of computing time) and 5 (55% of computing time), which use GHT matching to generate dense masks and object shapes, and perform character recognition. During character recognition, one detected character contour needs to be matched to each of the 36 templates (26 letters and 10 numbers). As shown in Figure 12 (Appendix B), more than half of the processing time is spent on the last step of the attack.

There are several ways to improve the efficiency of our attack: (1) Upgrading the current hardware; (2) Writing each module in a more efficient, lower level language, such as C/C++; and (3) Using parallelized algorithms. For example, the last step could be applied on the left and the right characters simultaneously. Moreover, a detected shape can be matched to all templates in parallel.

5 Countermeasures and Usability

In this section, we introduce a new design of EIMO CAPTCHAs that can defeat almost all auto-attacks, to our knowledge, based on accumulated information, and pursue a study to evaluate its usability. Before introducing our new design, we consider and rule out several straight-forward countermeasures.

5.1 Straight-forward Countermeasures

We explore several possible countermeasures to improve the security of EI-Nu CAPTCHAs against our auto-attacks, while keeping in mind the usability aspects. The first natural counter-attack is to increase the number of characters displayed in the challenge window, referred to as *Extended* in [20, 21]. However, simply increasing the number of characters is not necessarily an effective defense. First, Extended will only increase the completion time of our attack linearly, but still affordable with proper parallelism as suggested in Section 4. Second, to examine the impact of longer codewords, we performed experiments on 101 4-character codewords (three attack attempts each), and the average success rate is $\sim$67%, which is a noticeably drop compared to the average success rate of attacking 3-character codewords (89%). However, this success rate is still considered sufficiently high. We did not perform experiments on 5- or more character codewords, because: (1) The current window size can hold up to 5 characters but leave very little space for codeword movement, thus significantly increasing the difficulty for the human solver because there would be much less time ($<$ 2 seconds with a moving

speed of 2ppf) that all the characters of the codeword appear simultaneously in the window. (2) Given the fact that emerging image contains very limited visual cues, the font size cannot be too small, posing a limit on the number of characters (e.g., $\leq$5) in the challenge window. According to the study in [20, 21], the variant of NuCaptcha (which did not even have any emergent effect) considerably increased the difficulty of character recognition for human users due to confusing pairing of characters, missing letters, and extra ones. Their usability study also indicates that solving a long codeword is the least preferred by the user, and thus the most disliked. (3) It is not practical to further enlarge the window in order to accommodate long codewords because a too-large CAPTCHA window takes up too much space on a web page, which may degrade the UI experience of web browsing.

Another natural way that may counter our attack is to reduce the exposure of visual clues in each frame. This is similar to the idea of *Transparent* countermeasure proposed in [20, 21]. We did test our attack with different t_2 values, which is used to control the extent of exposure of shape contour. In our experiments, we decrease the t_2 value (i.e., causing less exposure) gradually to a level that would still yield human recognizable characters (though increasingly harder). The results in Table 2 indicate that the success rate drops with the decrease of t_2 value but the lowest success rate of our attack is still above 40%.

Increasing the overlap between adjacent characters, referred to as *Overlapping* in [20, 21], could be another viable mechanism of mitigating our attack. However, too much overlapping is likely to be challenging to both the attack and human users (usability), especially when combined with emerging effect and hollow characters that leave very little visual cues. We already demonstrated that our attack is effective in defeating the EI-Nu CAPTCHAs that do contain small-medium amount of overlapping. We followed the original design in [20, 21] to set up the character distance.

As shown in [20, 21], none of the above obvious countermeasures is considered effective in defeating auto-attacks against NuCaptcha without significantly compromising usability. They also seem unworkable against our auto-attack against EI-Nu.

Our attack effectively leverages the temporal information in the moving objects not only by exploring the accumulated shape information from a series of consecutive frames, but also by exploring higher-level accumulated information from lower-level accumulated information in a bottom-up fashion, and doing so at different moments of time to compensate for the incomplete and inconsistent visual cues exposed at different points of time. In the case of (2D-based) EI-Nu, although there are not enough visual cues in one frame that help distinguish characters from the background, and the codeword has no seemingly temporally consistent appearance (e.g., due to rotation, emerging effect, and wavy movement), the 2D character has a relatively consistent shape that does not change over time. What keeps changing from one frame to another is the specific set of edges segments (containing the shape information) exposed in each frame and the local transformations that make them look different if the same edge segment is displayed again. Also, although the background already incorporates emerging effects and appears to be "dynamic", the amount of dynamics embedded in largely "fixed" low geometric details is not sufficient to prevent foreground/background separation.

In summary, none of the above-mentioned obvious countermeasures fundamentally improve the design paradigm of 2D EIMO CAPTCHAs, and therefore still susceptible to the proposed attack and falling into a subclass of computer vision problems that have a computationally feasible solution and are characterized by relatively consistent object shapes with rigid 2D transformations, partial (and locally transformed) shape information exposed in a frame, and an emerging background that has largely "fixed" low geometric details.

5.2 A Fundamentally Different Design

In this section, we introduce a new design of EIMO CAPTCHA that addresses the fundamental limitations of the current 2D EIMO CAPCHAs. Our design consists of the following countermeasures.

The **first countermeasure** aims at drastically (rather than gradually) increasing the amount of dynamics embedded in the background, preventing the background detection. Instead of randomly moving the background up and down to implement the vibration effect, the CAPTCHA challenge window will start from an initial location in a larger background scene canvas, and randomly move up, down, left, or right with a constant speed (Figure 11(b)) that is fast enough in order to produce fast changing background, equivalent to drastically changing the low geometric details in the background in each frame. The background scene edge mask within the current CAPTCHA window is used in generating the current EI-based frame mask. Due to the fast dynamic change of the scene content, a "static" background scene can no longer be constructed, not even in a remotely reliable way, by superimposing multiple consecutive frame masks because such methods rely on reasonably consistent low geometric details or gradually varying appearance over time. In this case, pixel segments from the background scene will present them in the overlapped mask as noises, preventing effective detection of the codeword. In contrast, the original EI-Nu design [20, 21] converts each frame from a natural scene video (with slow varying appearance) into an emerging image as the background, resulting largely "fixed" low geometric details in the background, which is not sufficient to prevent foreground/background separation.

The **second countermeasure** is to add a pseudo 3D effect in the 2D object movement, which increases the problem space by an order of magnitude (vs. gradual increment) and effectively invalidates the assumption that the object shape remains constant over time. The implementation of pseudo 3D is simpler than that of the real 3D effect in the original EI videos [13], which depends on 3D objects and their shadow. Each character in a codeword rotates around the vertical axis, and also scales up and down randomly, producing a visual effect as if the character is moving toward/away from the human viewer. Figure 11(c) shows the pseudo 3D effect (codeword: "M3E") and dynamic background content in two edge masks. Such enhancement can largely prevent character separation through cross matching among multiple dense masks that are collected from different periods

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

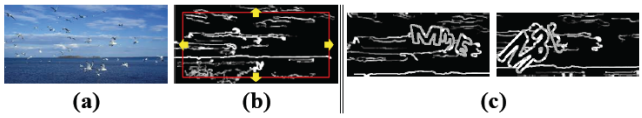


Fig. 11: Countermeasures against our proposed autoattack. (a) The natural scene image. (b) CAPTCHA window randomly roams in the canvas. (c) Pseudo 3D effect and dynamic background content. (The contrast of the edge mask is enhanced for better visualization.)

of the CAPTCHA video. However, 3D effect by itself is not guaranteed to be a sufficient defense, because occasionally we found that some superimposition masks (constructed from the version with 3D effect only) can still expose long object contours that can be visually recognized by a human solver, although not fully recognizable by our attack approach.

Our resulting EIMO CAPTCHA variant, termed 3D-EI, incorporates both of the above countermeasures. We can expect that, with two levels of security (3D effect foreground and dynamic background), the superimposition mask will only get much noisier, thus drastically reducing the success of attacks. In fact, 3D-EI not only defeats our attack against EI-Nu, but is also resistant to almost all commonly used computer vision based auto-attacks, to our knowledge, based on accumulated information. For example, it defeats the character pixel count approach [7], due to the irregularity of edge segments, i.e., the edge is no longer smooth, equal-width, or sufficiently continuous, not to mention the significant amount of noise existent in the subarea. The fact that it defeats our attack shows that 3D-EI is also resistant to shape matching attacks based on accumulated information. The color filling segmentation [7] relies on the availability of complete boundary of each character. The authors of emerging images have already proved the robustness of emerging images against conventional edge detectors (e.g., Canny edge detector). Therefore, no complete boundary can be detected in either a single frame image or a dense mask due to absence of information in the local neighborhood area, therefore cannot be used to attack 3D-EI. Examples of 3D-EI can be found at https://sites.google.com/site/breakingeimo1/proposed. The difference in the amount of background dynamics between the original EI-Nu and 3D-EI is obvious, so is the increased difficulty in recognizing 3D characters.

5.3 Usability Study: 3D-EI/EI-Nu/Nu

We now present a study to assess the usability of the proposed pseudo 3D effect EI CAPTCHA (3D-EI) compared with the EI-Nu CAPTCHA and the commercial NuCaptcha (Nu). In particular, we wanted to determine how much usability degradation occurs in 3D-EI over EI-Nu by adding security to our attack against EI-Nu. NuCaptcha was used as a baseline for our comparative usability study. We utilized the Amazon Mechanical Turk (MTurk) to recruit participants for the study. We chose to utilize MTurk as it would allow us to collect large amount of data from participants from various age groups, and backgrounds. Experimenting with the proposed CAPTCHA in a lab study would give us only insight of the usability of the proposed CAPTCHA among limited pool of participants (usually University samples) and thereby we will not be able to understand how a diverse set of people would perceive the CAPTCHA challenges. However, prior to conducting the MTurk study, small pilot lab studies were conducted in various stages of the CAPTCHA development process. Our university's Institutional Review Board approved the project.

5.3.1 Study Design

Each CAPTCHA challenge was of size 285(W)×125(H) and displayed as a 6-second video that loops continuously. We generated 100 challenges for each of 3D-EI and EI-Nu, and downloaded 100 challenges for Nu from its website [3]. A within-subjects experimental design was employed, where each participant was asked to solve exactly 10 challenges of the three categories (either correctly or incorrectly), the total number of solved challenges was 1200 challenges for each of the tested category. All participants completed all the tests. To reduce the effect of learning biases, the order of presenting the three categories followed a standard 3×3 Latin Square, and the challenges within each category followed a random order (a similar design was used in [21]). A total of 120 MTurk workers were recruited for the study and paid $1.0 each for their efforts that took on average 15 minutes. The participants were divided equally across the Latin square (40 participants per Latin square).

The MTurk workers were subjected to a consent agreement, and then asked to fill-out a demographics form, solve ten challenges of one of the categories (selected randomly from the hundred pre-generated/downloaded challenges), and fill-out a survey form about user experience. The survey contained 10 System Usable Scale (SUS) standard questions, each with 5 possible responses (5-point Likert scale, where strong disagreement is represented by "1" and strong agreement is represented by "5"). The same design was used to test the three categories of CAPTCHAs. Given the online nature of the study, in order to make sure that the participants pay attention while answering the survey, we inserted a dummy question at random location in each survey which asks the user to select a specific rating.

The participants in our study were from various age groups, education levels and backgrounds. Age group: 1.67% were $<$ 18 years, 21.67% 18-24 years, 50% 25-34 years, 15.83% 35-50 yearsand 10.83% $>$ 50 years[4]. Gender: 59.17% male and 40.83% female. Education: 36.67% high school graduate, 45% hold bachelor degree, 16.67% hold master degree and 1.67% hold a PhD degree. The participants were from various backgrounds such as Computer Science, Engineering, Medicine, Social Science, Finance, Business, Education, Art, Sales, Chemistry, Architecture, Construction, etc.

5.3.2 Results

The three CAPTCHA categories were evaluated in terms of (1) solving time, (2) number of successes, and (3) user experiences, as described below. The overall results are summarized in Table 3.

We have performed several statistical tests to evaluate how the three CAPTCHA categories differ from each other.

4. The age group of the participants in our study is similar to the age distribution of MTurk workers http://www.behind-the-enemy-lines.com/2015/04/demographics-of-mechanical-turk-now.html.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2719031, IEEE Transactions on Dependable and Secure Computing

Our selection of the statistical tests follows the analysis represented in [21]. We used one-way repeated-measures ANOVA test to analyze the differences between the solving time and the number of success mean. Post-hoc Tukey HSD test is used to determine between which pairs the difference occurred, whenever ANOVA revealed a significant difference. For SUS, we used the non-parametric Friedman's test. Whenever overall significant difference was found, we used post-hoc pairwise Wilcoxon Signed-Rank tests with Bonferroni correction to see in which of the pairs the difference occurred.

TABLE 3: Overall usability results for 3D-EI, EI-Nu and Nu

	Solving Time (s) *mean (sdev)*	# of Success *mean (sdev)*	SUS *mean (sdev)*
3D-EI	9.99 (5.46)	8.35 (1.72)	45.11 (21.16)
EI-Nu	6.68 (3.32)	9.23 (1.51)	58.13 (21.17)
Nu	6.19 (3.59)	9.62 (1.61)	79.62 (15.25)

Solving Time: The solving time was calculated as the time taken by the participants to solve each challenge. We considered in our calculation both the time taken that results in correct as well as the incorrect responses. The average solving time across all participants is shown in Table 3 (column 1). The results show that the time taken to solve EI-Nu is 1.08 times the time taken to solve Nu, and the time taken to solve 3D-EI is 1.50 times the time taken to solve EI-Nu on average. A one-way repeated measure ANOVA test showed overall statistically significant difference between solving time of the three variants ($F(2, 1200) = 287.73$, $p < 0.0001$). Upon further inspection, Tukey HSD tests showed significant differences between all of the pairs of the variants: $p < 0.01$ for the 3D-EI and EI-Nu pair, $p < 0.01$ for the 3D-EI and Nu pair, and $p < 0.05$ for the EI-Nu and Nu pair.

Number of successes: Table 3 (column 2) shows the mean and standard deviation of the number of successes for solving each of the tested CAPTCHA variants. Each participant was asked to solve 10 challenges of the three categories. Mean number of success is calculated as the average number of challenges that is solved correctly by the participants. The results show an average decrease of 9.53% in the solving accuracy of 3D-EI compared to EI-Nu, and an average decrease of 4.05% in the solving accuracy of EI-Nu compared to Nu. Comparing the differences in mean accuracies between the three categories using one-way repeated measure ANOVA, we found statistically significant difference ($F(2, 120) = 19.93$, $p < 0.0001$). Further, the Tukey HSD test showed significant differences between 3D-EI and EI-Nu ($p < 0.01$) and between 3D-EI and Nu ($p < 0.01$); no significant differences were found between EI-Nu and Nu, however.

Analyzing the individual participant responses, we found out the most amount of errors (65.22%) in NuCaptcha were contributed by the three participants who likely did not understand the task fully and were inputting the first moving word rather than the red codeword (the actual challenge). Further, we found that some of the participants could not distinguish between visually similar characters such as {7, T, 1}, {O, Q, 0, C}, and {S, 5, 8}, which reduced their performance in solving the three tested CAPTCHA categories (62.12% of the committed error was because of the confusable characters.). We also analyzed the position of errors in the challenge/codeword of the three tested CAPTCHAs (shown in Table 4). As we can see, most errors in 3D-EI occur in the second letter which is due to the overlapping with the left and the right characters in the codeword.

TABLE 4: Location of errors within the challenges/codewords

	1st Char Only	2nd Char Only	3rd Char Only	More Than 1 Char
3D-EI	31	72	54	37
EI-Nu	28	20	20	24
Nu	1	3	4	37

User Experience (SUS Scores): Three participants answered one of the dummy questions incorrectly and we removed their responses in that SUS survey from our analysis. Table 3 (column 3) shows the SUS scores (out of 100) for the three CAPTCHA categories. Nu's usability was very high as expected. However, both 3D-EI and EI-Nu variants seem to have usability on the lower side, given usable industrial software systems generally exhibit SUS scores of 70 or more [12], highlighting the usability challenges underling emerging-image CAPTCHAs in general. Clearly, 3D-EI scores are lower compared to those for EI-Nu. Comparing the SUS scores Friedman's test showed overall significant difference ($p < 0.0001$). Further, pairwise Wilcoxon Signed-Rank test with Bonferroni correction was used to to assess the difference between each of the three pairs. Significant differences were found ($p < 0.01$) between all the pairs of the three categories.

Familiarity: To evaluate whether the performance of CAPTCHA solving improves with practice, for each of the variants, we compared the average solving time and solving accuracy between the first and last attempt of the participants. The results are shown in Table 5. For both 3D-EI and EI-Nu, we found improvement in the participants' performance reflected by the decrease in solving time and increase in solving accuracy. Comparing the solving time using paired t-test, we found a statistical difference between the first and last attempts of both EI-Nu and EI-3D. No such statistical differences were found for NuCaptcha, however. Also, we found that only 52% of the participants could solve the 3D-EI variant the first time they saw it, while the percentage increased to 92% on their 10th attempt. This analyses suggests that participants performed better and their tolerance to the 3D effect and dynamic background increased with more exposure to these emerging CAPTCHA variants.

TABLE 5: Comparing the first and last attempts for the three CAPTCHA variants

	First Attempt		Last Attempt	
	Solving Time *mean (sdev)*	Solving Accuracy(%)	Solving Time *mean (sdev)*	Solving Accuracy(%)
3D-EI	13.83 (8.21)	76.67	9.24 (4.28)	82.50
EI-Nu	8.94 (3.45)	90.00	6.03 (2.41)	96.67
Nu	7.05 (3.93)	95.00	7.88 (5.62)	95.83

Summary: Our results show some degradation in the usability of 3D-EI CAPTCHAs when compared to EI-Nu

CAPTCHAs (comparable to the degradation in the usability of EI-Nu CAPTCHAs over NuCaptchas). On the positive side, users could get become better at solving 3D-EI CAPTCHAs as they become more familiar with them over time. Thus, we believe that a significant improvement in the security provided by our 3D variant over EI-Nu would make it a valuable CAPTCHA that may still be used in web applications that require high security against automated attacks.

6 Conclusions

Emerging images provide humans with an easy way to perceive an object, meanwhile, posing a significant challenge for computers to decode the underlying content. As the first representative and user-friendly instantiation of emerging image CAPTCHA, EI-Nu CAPTCHA was proposed believed to be secure against auto-attacks. However, we have identified several security vulnerabilities in this design and translated them into a full attack against this scheme. The key weakness is to use constant camera projection on 2D objects, which provides limited, but temporally continuous contour information in consecutive frames.

Possibly all auto-attacks employing accumulated information would fail against our new construction due to two reasons. First, the dynamic background content eliminates the possibility for generating static background mask in the attack. Second, the pseudo 3D objects have different camera projection in different period of a CAPTCHA video, thereby reducing the chance to recover the object shape through cross matching between frames in different periods. This improvement in security, however, comes at the cost of reduced usability compared to the (now shown insecure) 2D variant (EI-Nu), which may still be acceptable in high security web applications. Broadly, our work highlighted the security and usability challenges associated with emerging-image CAPTCHAs not known before.

References

[1] Emerging captcha. http://www.cs.unc.edu/~yix/VideoCaptcha/.

[2] How we broke the nucaptcha video scheme and what we propose to fix it. https://www.elie.net/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it.

[3] Nucaptcha. http://docs.nucaptcha.com/embed/basic2/outdoor.

[4] Nucaptcha. whitepaper: Nucaptcha & traditional captcha. http://www.nucaptcha.com.

[5] H. S. Baird, A. L. Coates, and R. J. Fateman. Pessimalprint: a reverse turing test. *International Journal on Document Analysis and Recognition*, 5(2-3):158–163, 2003.

[6] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[7] A. S. El Ahmad, J. Yan, and M. Tayara. *The robustness of Google CAPTCHA's*. Computing Science, Newcastle University, 2011.

[8] J. M. G. Hidalgo and G. Alvarez. Captchas: An artificial intelligence application to web security. *Advances in Computers*, 83:109–181, 2011.

[9] G. Keizer. Spammers' bot cracks microsoft's captcha. Computer World, Available at: http://www.computerworld.com/article/2536901/security0/spammers--bot-cracks-microsoft-s-captcha.html, 2008.

[10] K. A. Kluever and R. Zanibbi. Balancing usability and security in a video captcha. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 14. ACM, 2009.

[11] B. B. Le Cun, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

[12] J. R. Lewis and J. Sauro. The factor structure of the system usability scale. In *Human Centered Design*, pages 94–103. Springer, 2009.

[13] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or. Emerging images. In *ACM Transactions on Graphics (TOG)*, volume 28, page 163. ACM, 2009.

[14] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W.-B. Chen. A three-way investigation of a game-captcha: automated attacks, relay attacks and usability. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 195–206. ACM, 2014.

[15] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, pages 958–958. IEEE Computer Society, 2003.

[16] J. K. Tsotsos. On the relative complexity of active vs. passive visual search. *International journal of computer vision*, 7(2):127–141, 1992.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[18] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[19] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Advances in Cryptology-EUROCRYPT 2003*, pages 294–311. Springer, 2003.

[20] Y. Xu, G. Reynaga, S. Chiasson, J. Frahm, F. Monrose, and P. Van Oorschot. Security analysis and related usability of motion-based captchas: Decoding codewords in motion. *IEEE Transactions On Dependable And Secure Computing*, 11(5), 2013.

[21] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. C. van Oorschot. Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *USENIX Security Symposium*, pages 49–64, 2012.

[22] J. Yan and A. S. El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008.

[23] J. Yan and A. S. El Ahmad. Captcha robustness: A

security engineering perspective. *Computer*, 44(2):0054–60, 2011.

[24] C. Zhang, W.-B. Chen, X. Chen, R. Tiwari, L. Yang, and G. Warner. A multimodal data mining framework for revealing common sources of spam images. *Journal of Multimedia*, 2009.

APPENDIX

A: The Choice of GHT

There are several representative classification methods, such as ANN [11, 15] and AdaBoost [17], using features of objects (e.g., edges and corners) for character/object recognition. However, different from the relatively clean training data used in [20], our detected shape contours consist of sporadic small segments and do not have smooth edges or clear corners (Figure6). Therefore, there are no robust features that can be reliably used to train a classifier. Instead, we use GHT-based direct matching to accommodate for noisy dense masks and rotations.

B: Processing Time of Each Step in our Attack

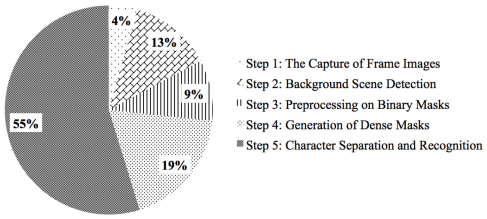
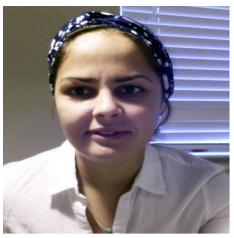


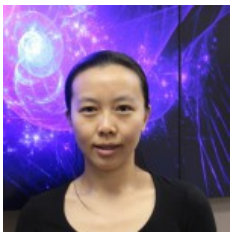
Fig. 12: Proportion of the processing time of each step in the proposed attack.

Song Gao is a software engineer, Google (April 2015-present). Song Gao received his PhD in computer and information sciences from the University of Alabama at Birmingham, USA in December 2014. His research interests include multimedia data mining, machine learning, image processing, and computer and network security. The projects he participated include service abuse resilience design of playful gamebased CAPTCHAs, a photographic method for human body composition assessment, identification of image spam authorship, machine classification of Melanoma and Nevi from skin lesions, texture-based authorship classification of web images, etc. Gao has an MS in computer science from Chongqing University of Posts and Telecommunications, China.



Manar Mohamed is an Assistant Professor of Computer and Information Sciences at Temple University. Her main research interests are usable security, mobile security, applied machine learning, and data sciences. She has published several journal, conference and workshop papers, many at top-tier venues in Computer Science, including: IEEE Transactions, ACM WiSec, and IEEE Percom. She received her PhD degree from the University of Alabama at Birmingham, and her BSc and MSc degrees from Arab Academy for Science, Technology and Maritime Transport, Egypt. Mohamed's work has received extensive media coverage, for example, ACM TechNews, Slashdot and Computer World.

Nitesh Saxena is an Associate Professor of Computer and Information Sciences at the University of Alabama at Birmingham (UAB), and the founding director of the Security and Privacy in Emerging Systems (SPIES) group/lab. He works in the broad areas of computer and network security, and applied cryptography, with a keen interest in wireless and mobile device security, and the emerging field of usable security. Saxena's current research has been externally supported by multiple grants from NSF and NIJ, and by gifts/awards/donations from the industry, including Google (2 Google Faculty Research awards), Cisco, Comcast, Intel, Nokia and Research in Motion. He has published over 110 journal, conference and workshop papers, many at top-tier venues in Computer Science, including: IEEE Transactions, ISOC NDSS, ACM CCS, ACM WWW, ACM WiSec, ACM ACSAC, ACM CHI, ACM Ubicomp, IEEE Percom, IEEE ICME and IEEE S&P. On the educational/service front, Saxena currently serves as the director and principal investigator for the UAB's Scholarship for Service (SFS) program and a co-director for UAB's MS program in Computer Forensics and Security Management. He serves as an Associate Editor for flagship security journals, IEEE Transactions on Information Forensics and Security (TIFS), and Springer's International Journal of Information Security (IJIS). Saxena's work has received extensive media coverage, for example, at NBC, MSN, Fox, Discovery, ABC, Bloomberg, MIT Tech Review, ZDNet, ACM TechNews, Yahoo! Finance, Communications of ACM, Yahoo News, CNBC, Slashdot, Computer World, Science Daily and Motherboard.



Chengcui Zhang is a Professor of Computer and Information Sciences at the University of Alabama at Birmingham (UAB). She works in the broad areas of multimedia databases and information retrieval, multimedia data mining, multimedia security and forensics, Geoinformatics, and applied Bioinformatics. She has published over 150 refereed articles, many at the top tier venues in computer sciences including IEEE Transactions, IEEE Multimedia, ACM Multimedia (MM), IEEE International Conference on Data Mining (ICDM), ACM Conference on Communication and Computer Security (CCS), and IEEE International Conference on Multimedia and Expo (ICME). Dr. Zhang's research has been externally supported by NSF, NIH, and by awards/gifts from the industry, including IBM, eBay, and Comcast. Dr. Zhang was the Chair of IEEE Technical Committee on Semantic Computing from 2014-2016 and has been serving as the secretary and newsletter editor for IEEE Technical Committee on Multimedia Computing. She has also served in leading roles for many IEEE and ACM conferences, including a program chair for IEEE ICME'14, two times as a program chair for IEEE International Conf. on Information Reuse and Integration (IRI) in 2012 and 2013, a program chair for the 6th FTRA International Conf. on Information Technology Convergence and Services (ITCS 2014), a program vice-chair for the 2007 International Conference on Intelligent Pervasive Computing (IPC-07), etc. She is an Associate Editor of IEEE Transactions on Multimedia and on the Editorial Board for 6 other journals.