Emerging Image Game CAPTCHAs for Resisting Automated and Human-Solver Relay Attacks

Song Gao
Google
gaos@uab.edu

Manar Mohamed
University of Alabama at Birmingham
manar@uab.edu

Nitesh Saxena
University of Alabama at Birmingham
saxena@uab.edu

Chengcui Zhang
University of Alabama at Birmingham
zhang@uab.edu

ABSTRACT

CAPTCHAs represent an important pillar in the web security domain. Yet, current CAPTCHAs do not fully meet the web security requirements. Many existing CAPTCHAs can be broken using *automated attacks* based on image processing and machine learning techniques. Moreover, most existing CAPTCHAs are completely vulnerable to human-solver *relay attacks*, whereby CAPTCHA challenges are simply outsourced to a remote human solver.

In this paper, we introduce a new class of CAPTCHAs that can *not only resist automated attacks but can also make relay attacks hard and detectable*. These CAPTCHAs are carefully built on the notions of dynamic cognitive games (DCG) and emerging images (EI), present in the literature. While existing CAPTCHAs based on the DCG notion alone (e.g., an object matching game embedded in a clear background) are prone to automated attacks and those based on the EI notion alone (e.g., moving text embedded in emerging images) are prone to relay attacks, we show that a careful *amalgamation* of the two notions can resist both forms of attacks. Specifically, we formalize, design and implement a concrete instantiation of EI-DCG CAPTCHAs, and demonstrate its security with respect to image processing and object tracking techniques as well as their resistance to and detectability of relay attacks.

1. INTRODUCTION

Almost every online service relies upon CAPTCHAs [4, 16] to thwart various forms of online attacks and resource abuse. Unfortunately, many existing CAPTCHAs can be broken using automated attacks based on image processing and machine learning techniques (see, e.g., [7, 9, 10, 13, 19, 20]). Moreover, most are vulnerable to human-solver relay attacks, whereby CAPTCHA challenges are simply outsourced to a remote human solver. These attacks do not merely exist in theory – a myriad of CAPTCHA solving services have already emerged in practice [14].

Our research is driven by the question: *can a CAPTCHA scheme be designed that resists both automated attacks and relay attacks?* To answer this question, we turn to two categories of CAPTCHAs from the current literature – *Emerging Image* (EI) CAPTCHAs [17, 18] (known to be resistant to automated attacks) and *Dynamic Cognitive Game* (DCG) CAPTCHAs [12,13] (known to be resistant to relay attacks). However, *no* current scheme is known to be simultaneously resistant to <u>*both*</u> attacks. We aim to achieve this property via a careful combination of the two categories of CAPTCHAs.

In the work of [11], the authors proposed emerging images of *3D objects* and explained the emergence as "the phenomenon by which we perceive objects in an image not by recognizing the object parts, but as a whole, all at once". The authors indicate [11]: "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 [15], leading us to hypothesize that emergence images are hard for automatic algorithms to segment, identify, and recognize". They further go on to argue that: "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". A concrete instantiation of an EI CAPTCHA developed in [17, 18] inherits the above-mentioned characteristics, and is demonstrated to be secure against automated attacks. However, such video-based EI CAPTCHAs are completely vulnerable to relay attacks whereby the static video challenge can be simply forwarded to the remote human-solver.

DCG CAPTCHA [12] is a CAPTCHA that challenges the user to perform a game-like cognitive task interacting with a series of dynamic objects in a static scene. A simple form of DCG CAPTCHAs requires the user to identify the answer object(s) from a set of moving objects, and drag-drop them to the corresponding target object(s). Recently, a startup company, named "are you a human", released a series of such DCG CAPTCHAs [1]. A DCG CAPTCHA exhibits certain interesting properties. First, it is based on a cognitive puzzle, which is easy for humans to understand, but may be difficult for a bot without enough clues. Second, the game-based nature enhances the usability of CAPTCHA solving. Third, due to the dynamic and interactive nature of the underlying game, relaying the game to a remote solver might be challenging. As shown in [12], however, DCG CAPTCHAs based on static background are vulnerable to automated attacks. On the positive side, they were shown to offer resistance to relay attacks [12, 13].

ACSAC '15, December 07-11, 2015, Los Angeles, CA, USA

DOI: http://dx.doi.org/10.1145/2818000.2818006

Our Contributions: We introduce a new class of CAPTCHAs (called EI-DCG), carefully combining the EI and DCG notions, that can not only resist automated attacks but also make relay attacks hard and detectable. Our specific contributions are three-fold:

1. *Design of an EI-DCG CAPTCHA* (Section 3): We design and implement a concrete instantiation of EI-DCG CAPTCHA combining an EI CAPTCHA [17, 18] and a DCG CAPTCHA [12]. Our design further incorporates various countermeasures, including pseudo 3D object rotation, incomplete object contour and tiling background, to reduce the information exposure through the superimposition of consecutive frames, and a fast frame rate to resist the relay attack based on streaming.
2. *Resistance to Automated Attacks* (Section 4): To evaluate the robustness of EI-DCG against automated attacks, we design an attack framework that guesses the centroids of moving objects based on local density variance, for example, by selecting density peaks and valleys as the object centroids, in the binary mask of a frame, in the accumulation of three consecutive frames, and in the frequency map of the accumulation. We demonstrate that EI-DCG can effectively resist this sophisticated attack framework.
3. *Usability and Resistance to Relay Attacks* (Section 5): We conduct a usability study with Amazon Mechanical Turk workers to evaluate the performance of legitimate users in solving EI-DCG challenges. The results show that legitimate users were able to solve the EI-DCG challenges with relatively high accuracies suggesting an acceptable level of usability. Then, we study EI-DCG streaming-based relay attack [13] with human-solvers located in a foreign country (India) connected via a low-speed high-latency channel, and with human-solvers located in close proximity of the attacker (US) connected via a high-speed low-latency channel. The study shows that the solvers in India could only solve less than 1% of EI-DCG challenges successfully, while the solvers in the US could solve only around 13% of EI-DCG challenges.

   Going further, we design a *streaming-based relay attack detection* mechanism, which utilizes the game-solving statistics and machine learning classifiers in order to differentiate legitimate users from human-solvers. The results show that it is possible to detect the streaming-based relay attack with around 65% accuracy, making the overall detection rate to be around 95%.

2. BACKGROUND

A fundamental design objective of EI-DCG CAPTCHA is the same as that of any CAPTCHA: a bot (automated computer program) must only be able to solve CAPTCHA challenges with no better than a negligible probability, and users should be able to solve the challenges with high accuracy. Further, we add that EI-DCG CAPTCHA should offer resilience to relay attacks. In order to achieve these design objectives, we carefully integrate the DCG CAPTCHA proposed in [12,13], which offers resilience to relay attacks, with the EI CAPTCHA proposed in [17,18] (which we refer to as EI-Nu, since it is a variant of the NuCaptcha [2] based on the EI notion), which offers security against automated attacks. The threat model for EI-DCG CAPTCHA therefore also naturally combines the threat models of EI CAPTCHAs and DCG CAPTCHAs.

DCG CAPTCHA is a simple interactive CAPTCHA that consists of several objects that moves randomly within a frame and some static target objects (Figure 1b). DCG CAPTCHA challenges the user to drag a subset of the moving objects (answer objects) to their corresponding target objects. The authors in [12, 13] showed that the dynamic nature of DCG as well as the requirement for multiple interactions between the user and the CAPTCHA facilitate resilience to relay attacks. To support their claims, the authors formalized two types of relay attacks:

(a) (b)

Figure 1: A snapshot of EI-Nu CAPTCHA and DCG CAPTCHA utilized in our work: (a) EI-Nu challenge codeword "7FX". (b) DCG Shape game – blue region contains the stationary target objects, the white region contains the moving possible answer objects. Our resulting EI-DCG construction is a character matching game rendered on top of emerging images (Figure 2(h) depicts a sample EI-DCG frame)

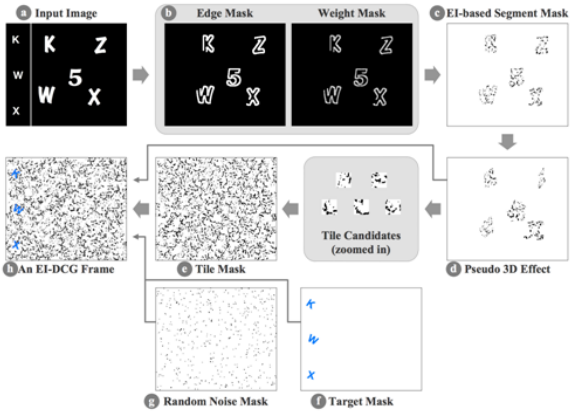

Figure 2: Generating an EI-DCG CAPTCHA frame (this and the other figures are best viewed in color)

1. *Static Relay Attack*: The attacker asynchronously sends static snapshots of the game to the human solver. The human solver has to identify the target objects and then recognize their corresponding answer objects in the subsequent frames as quickly as possible given that the answer objects are moving. The bot then clicks on the position specified by the human-solver as an answer object, drags it and drops it to the pre-specified target object.
2. *Stream Relay Attack*: The attacker utilizes a streaming software, such as VNC, to stream the game frames from the attacker to the human solver, and the game interactions from the human-solver to the attacker.

The authors of [12, 13] argued that the user performance in legitimate setting and in relay attack settings differs. While a legitimate user plays a flash-based game rendered locally by the client machine, a remote solver has to play a streamed game which may be of degraded quality due to the latency of the communication channel between the attacker and the solver. This difference in the quality of the game, resulting in different game playing patterns, was then utilized to detect the relay attack with high accuracy based on machine learning techniques [13] .

EI-Nu CAPTCHA proposed in [17, 18] (Figure 1a) is based on the emerging images notion presented in [11]. EI-Nu is a video based CAPTCHA that contains several characters, collectively called a codeword. It is designed such that a single snapshot of the challenges does not provide enough clues about the codeword in the challenge. However, the user can recognize the codeword by watching multiple frames. The authors in [17, 18] argued that EI-Nu is secure against automated attacks based on existing computer vision and object tracking techniques.

3. DESIGN & IMPLEMENTATION

3.1 Design Overview

Our EI-DCG CAPTCHA has following unique features that differentiate it from EI-based Nu CAPTCHA (EI-Nu) [18] and the original EI-based videos [11].

1. Instead of using 2D objects as in EI-Nu CAPTCHA or real 3D objects with shading as in EI videos, EI-DCG CAPTCHA uses a pseudo 3D object (i.e., projecting a 2D object into 3D space, applying necessary transformations, and projecting it back to 2D) to provide a simulated 3D view, and more importantly to lower the possibility of recovering object contours through accumulation of information from consecutive frames, i.e., to protect against auto-decoding.
2. Hiding information in a single frame is more challenging in EI-DCG CAPTCHA since there are usually more foreground objects (e.g., ≥ 5) than there are in EI-Nu or EI videos. Most EI-based videos in [11] contain one single object, making it relatively easy for a human viewer to focus on the object. Meanwhile, having very few moving objects also leaves adequate room for making each object sufficiently large for easy recognition by human eyes. However, it will be too risky for a DCG CAPTCHA to use less than 5 objects (keeping in mind the random guessing attacks). According to [13], if the locations of moving objects are exposed (e.g., through multi-frame accumulation), a random guess can achieve $\sim$13% success rate given 5 moving objects, 3 target objects, allowing $\leq$2 drag-and-drop attempts per object. In an EI-DCG CAPTCHA frame, we camouflage moving objects in both a single frame and in the accumulation of consecutive frames by tiling the background with deformed and incomplete edge segments from foreground objects in a way similar to that of the EI videos [11].
3. Both EI-Nu and EI videos play a fixed video clip repeatedly, leading to a constant-time requirement for rendering. However, an EI-DCG CAPTCHA challenge demands real-time user interaction with foreground objects, requiring each frame to be generated on the fly and incurring higher computation. We apply a divide-and-conquer strategy to prepare the information needed for generating a frame in advance, and repeatedly use it to efficiently create more new frames.

Creating an EI-DCG frame requires creating both the foreground object mask as well as the background mask. For an input image with both foreground objects and the target objects (Figure 2(a)), the edge mask and the weight mask are computed (Figure 2(b)), which are used to generate the EI visual effect (Figure 2(c)). Large segments that may leave clue for reconstructing object contour will be further split. The pseudo 3D effect for each object is applied based on current rotation and scaling parameters (Figure 2(d)). The remaining area in the frame (i.e., the background) is tiled with segments from foreground objects (Figure 2(e)). Finally, an EI-based frame (Figure 2(h)) is the Gaussian blur of the combination of the foreground mask, tiled background, target object mask (Figure 2(f)), and random noise mask (Figure 2(g)). The details will be provided in the following subsections. An EI-DCG CAPTCHA challenge is configured as follows:

- Dimension: 340(height)$\times$400(width).
- 3 target objects and 5 foreground objects, which are all alphanumeric characters.
- Object moving speed: 3 pixels per frame (ppf).
- Frame rate: 40 frames per second (fps).
- N=40 pairs of foreground and background that record the pixel value and location of foreground and background objects in each frame are rendered.

The purpose of using a higher frame rate (e.g., 40 fps) than usual (e.g., 30 fps) is to increase the robustness against the relay attack. The higher the frame rate, the less information about foreground objects is revealed in one single frame, thus requiring more frames to be read at a time in order to recognize the object. In case of a relay attack, the communication delay between the bot and the human solver's machine could cause loss of synchronization and thus lead to the failure to keep up with the required frame rate, resulting in jittery motion in video play. Since human eyes rely on continuous motion to recognize EI objects, this design choice will make it even harder for remote human-solver to identify the object and play the game effectively.

3.2 Creation of EI-based Foreground Mask

The use of hollow objects may be more secure than solid objects in dynamic CAPTCHAs (e.g., EI-Nu and EI-DCG) because less shape information of a hollow object could be revealed by superimposition of consecutive frames than that for a solid object. Given an input image with both solid moving objects and target objects, we first generate the foreground edge mask, and the normalized foreground weight mask by computing the norm of derivatives. Both masks are dilated with a flat, disk-shaped structuring element in order to enhance the low geometric details that could be later converted into an emerging image. The edge mask (I_{fg_e}) indicates the location of edge pixels, while the weight mask (I_{fg_w}) indicates the relative importance of each edge pixel. We largely followed the instructions of EI-Nu CAPTCHA [18] to create an EI-based foreground mask, with additional countermeasures implemented to deter potential computer-vision based attacks, as discussed below:

1. We generate a noise image I_{bg} with each pixel following a Gaussian distribution. The median filter is used to blur the noise image such that each pixel value is similar to its nearby pixels (Figure 3(a)).
2. We manipulate the pixel values in both the foreground edge mask I_{fg_e} and the noise image I_{bg} to preserve the temporal continuity between two consecutive frames, without revealing too much information about objects.

   In the edge mask, a small portion (e.g., 10%) of the edge of each object is intentionally hidden in the current frame (Figure 3(e)). A different hidden edge segment is selected in the next frame. In this way, in one frame, no complete object contour will be exposed, further lowering the possibility of reconstructing the contour through superimposition of consecutive frames. Meanwhile, according to the *law of closure* in Gestalt theory [8], humans can perceive objects as being whole even if they are not complete. Specifically, when parts of a whole picture are missing, human perception fills in the visual gap. The weight values in the weight mask I_{fg_w} corresponding to the hidden edge segment will also be hidden (Figure 3(b)).

   We denote *common pixels* as the edge pixels shared between the edge masks of two consecutive frames (previous and current), and *discrepant pixels* as the edge pixels that only exist in the current frame. The temporal continuity is preserved through allowing a certain percentage of common pixels to be displayed again in the current frame. Common pixels with non-negative I_{bg} values and discrepant pixels with negative I_{bg} values exchange their values. Discrepant pixels are exchanged in the ascending order of their I_{bg} values until the largest negative pixel is reached. A percentage parameter ρ (e.g., 0.7) 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}.

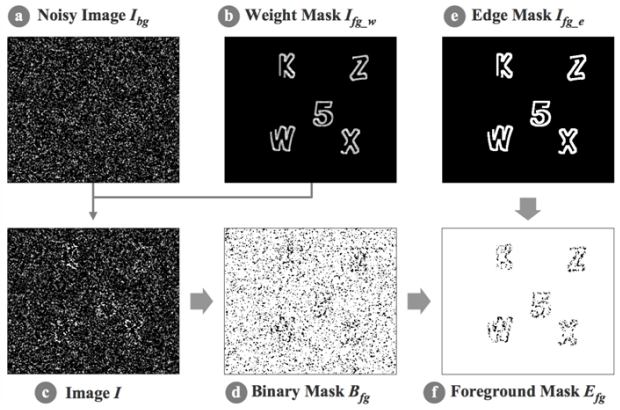

Figure 3: Generating an EI-DCG CAPTCHA frame.

Figure 4: (a) EI-based foreground mask of character "5". (b) Split and erosion on large or long segments, (c) followed by a small rotation and translation. To highlight the changes, we show the original segments in gray in (b) and (c).

3. We create an image I (Figure 3(c)) by combining both the foreground weight mask I_{fg_w} and the noise image I_{bg} using the equation $I(x, y) = I_{bg}(x, y) \times exp(I_{fg\_w}/const)$, where $exp(x)$ is the exponential function. In our design, $const$ is set as 0.6. According to this, pixels with smaller I_{bg} values are more likely to appear as black pixels. Therefore, the swap operation in Step 2 makes common pixels more likely to be displayed as black again in the current frame.

4. We generate the binary mask B_{fg} for foreground objects through binarizing I by setting all the pixels whose values are greater than a user-defined threshold $t < 0$ (e.g., -0.5) to 'white', and the remaining pixels to 'black' (Figure 3(d)).

5. We generate the EI-based foreground mask E_{fg} by removing all the black pixels in B_{fg} that are not on the foreground edge mask I_{fg_e} (Figure 3(e)) such that only those black segments on the object edges remain in the binary mask (Figure 3(f)).

Large segments in the foreground mask may raise a potential risk of exposing object contour. Therefore, we apply a post-processing including split, erosion, and random rotation and translation on segments that have area larger than a threshold t_{area} or their major axis is longer than t_{maxis} (Figure 4). The values of t_{area} and t_{maxis} are determined by the current object size. For a large or long segment, the split point is selected as a random point between 1/3 and 2/3 along the major axis that crosses the center of the segment. Further erosion is applied on the split segments, followed by a small random rotation and translation to further disrupt the continuity between nearby edge segments.

3.3 Pseudo 3D Visual Effect

Taking a closer look at EI-Nu CAPTCHA challenges, we learned that accumulating information from consecutive frames could possibly recover object contours, especially when the primary moving direction of foreground objects is known. In an EI-Nu CAPTCHA challenge, a codeword moves horizontally from the right side of background to the left at a constant speed. Each character in the codeword also moves up and down harmonically, rotates slightly such that neighbor characters overlap each other to some extent.

Thus, we argue that 2D movement as described above may not be sufficient to prevent the reconstruction of the object contour through accumulating information from consecutive frames. *First*, we demonstrate how much information is exposed in consecutive frames. Given three consecutive frames (Figure 5(a)), we converted them into color code representation [13], which replaces a pixel value with a 6-bit color value ([0,63]) consisting of the highest two bits from each 8-bit RGB channel. The color code can help group pixels with similar colors into the same code, facilitating the binarization process. We converted all pixel values in a color code representation that are less than or equal to an empirical threshold (e.g., 21) to white (Figure 5(b)), and black otherwise. Since the primary moving direction is from right to left, during superimposition, the (i+2)-th frame is shifted right 1 pixel, while the i-th frame is shifted left 1 pixel, before they are superimposed on the (i+1)-th frame. The superimposition of three binary masks as shown in Figure 5(c) exposes a large portion of codeword contour. Moreover, removing the background scene mask from the superimposed mask is technically possible and could further increase the accuracy of codeword contour detection.

Second, the orientation difference of a character in 2D space between two consecutive frames is small. If such rotation degree per frame is large, say 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 a human can barely recognize the character. Therefore, the 2D rotation of a character between consecutive frames cannot be too large, thereby making it possible to accumulate contour information through superimposition of consecutive frames

In our EI-DCG CAPTCHA design, therefore, we propose to implement a pseudo 3D effect that includes 3D rotation and scaling for each moving object in order to hinder information accumulation from superimposition. First, the 2D mask of an object is extracted from the EI-based foreground mask (Figure 3(f)). Second, the 2D matrix (the mask) is projected into 3D, in which the X-axis and Y-axis are consistent with the previous axes in 2D, while Z-axis represents the viewer's direction. Third, a degree of rotation is applied to each of the X-, Y-, and Z-axis, respectively. In the end, the 3D geometry is projected back into 2D. Meanwhile, the focal distance is tuned in the projection matrix to scale up/down the object in the current frame.

In our design, the rotation range in degree around X-, Y-, and Z-axis are [-40,40], [-60,60], and [-40,40], respectively. The rotation speed ($degrees\ per frame$) is determined by the equation $r = (2 \times rotation\_range)/N$, where N is the number of pairs of foreground and background as mentioned in Section 2. Therefore, the object rotates around a specific axis from one side to the other, and then back within N rounds. The object also scales up and down between 100% and 184% with a changing speed calculated in the same way as the rotation speed. On top of all that, there is no primary moving direction in our design - each object moves in their own random direction.

To examine the resilience of the above mechanisms to superimposition-based information recovery, we screen-captured three consecutive frames of an EI-DCG CAPTCHA challenge without background or target objects, converted them into binary masks, and showed the captured object contour through superimposition of two and three masks, respectively. As shown in Figure 6, unlike the result in Figure 5(c), none of the superimposition masks expose long segments that are part of the object contours. We can expect that with the addition of background, the superimposition mask will only get noisier.

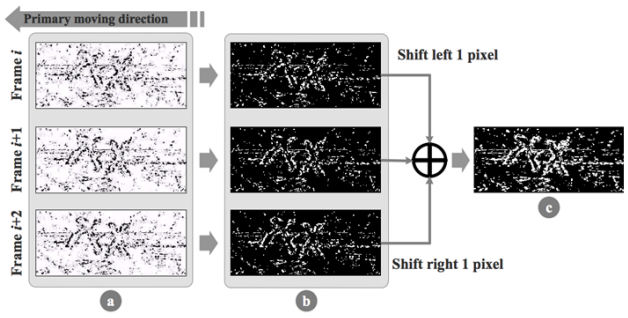

Figure 5: Reconstructing the codeword "7FX". (a) 3 consecutive frames of an EI-Nu CAPTCHA challenge. (b) Binary mask of each frame. (c) Superimposition of three consecutive frames.

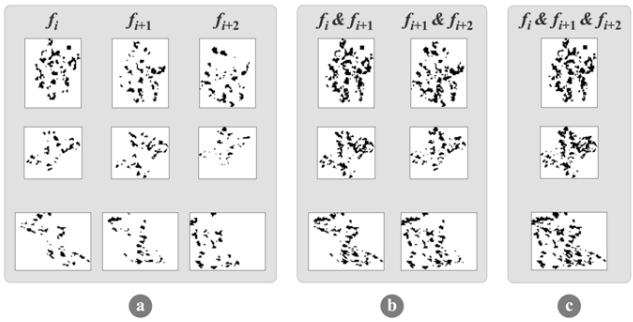

Figure 6: Effect of 3D rotation and scaling on superimposed masks. Objects from left to right: "5", "X", and "Z". (a) Binary masks. (b-c) Superimposition of 2 and 3 consecutive masks.

3.4 Background Tiling

We further identified several key requirements for effectively camouflaging foreground objects in EI-based background.

1. Any subarea of a background should look visually similar to the foreground objects. If long or large segments exist in an EI-based foreground mask, similar segments will also be replicated multiple times in the background. Thus even an exhaustive search in a screen-captured frame based on known object templates will returns multiple candidates, while only one of them may be the real object.
2. Ideally, any subarea in the background should also have similar density as the foreground objects, so that the location of moving objects can be better camouflaged since there is no density anomaly across the whole frame.
3. The background should be tiled, i.e. filled, in a way that can hide objects in the superimposed mask of consecutive frames. One reason that EI-Nu CAPTCHA is vulnerable to superimposition-based attack (Figure 5(c)) is because its background is sparsely and unevenly tiled. To introduce the maximum noise into the superimposed mask, we tile the entire background (except the areas occupied by foreground objects) with segments that exhibit similar density as foreground objects, and thus no prominent subarea that may correspond to foreground objects can be easily identified.

To meet the above requirements, we tile the background using the same idea of the EI video [11] but in a different way of selecting the tile segments and tiling.

Determining the Tile Size: Similar to the original EI, we use the subparts of foreground objects to tile the background. If the tile size is too big, more complete object contour information may be exposed at multiple locations in a frame. And, the larger the tile, the sparser the scene becomes, leaving fewer object candidates in a single frame, leading to less guess work for automated attack. For the same reason, there is also an increased risk associated with information recovery from superimposed masks, if a too large tile size is used. On the other hand, a too small tile size will make the background too crowded, making the relative sparseness in the areas of foreground objects more distinguishable, again easing the auto-attack. Given all the minimum bounding boxes (MBRs) of foreground objects in a frame, we define a tile as a square with size $l_t = min$ (dimensions of all MBRs). Assume the size of a CAPTCHA window is $h \times w$. If there is no overlap in tiling, there will be totally $N_t = floor(h \times w/(l_t \times l_t))$ tiles.

Determining the Tile Segment Candidates: A tile mask should have a similar density as that of the foreground MBR mask. First, from the mask of each of the K foreground objects in a frame, the subarea of dimension $l_t \times l_t$ that has the maximum pixel count for that mask is identified. The average density d_t of these subareas is used as a reference to extract tile segment candidates in K foreground object masks. Second, tile segment candidates are selected by searching each subarea of dimension $l_t \times l_t$ in each foreground object mask that has a density in the range $[d_t - \delta, d_t + \delta]$ where δ is a random value between 0 and $0.1 \times d_t$. Each such subarea is a tile segment candidate, and there are total N_t candidates in a frame.

Determine the Tile Locations: Instead of paving tiles side by side in a regular grid, we vary the locations of tile centers both horizontally and vertically, such that tiles may overlap or separate from each other by a small random amount, thereby further randomizing the segment distribution. Each tile segment candidate obtained in the previous step is further rotated by a random angle and pasted in the background anchored to the center of a randomly selected unpaved tile space.

3.5 EI-DCG Configuration Levels

EI-DCG CAPTCHA is configured at three levels of potential difficulty for the human user (easy, medium, and difficult), using the parameter settings in Table 1. The more "difficult" the CAPTCHA is, the less susceptible it is supposed to be against computer vision-based auto attack. The pixel density decreases when the difficulty level increases, i.e., decreasing amount of foreground information embedded in one single frame.

Table 1: Parameters settings used to generate EI-DCG CAPTCHA challenges at easy, medium, difficult levels.

Threshold	Easy	Medium	Difficult
Foreground threshold t	-0.1	-0.5	-0.7
Percentage of hidden edge γ	0.05	0.1	0.1
Gaussian blur (kernel size, std dev)	(3, 1)	(2, 1)	(2, 1)

4. AUTOMATED ATTACK RESISTANCE

Emerging images [11] are known to be robust to automatic object detection using existing image processing and machine learning techniques. Compared with EI-Nu CAPTCHA (Figure 5), there is no primary moving direction in our proposed EI-DCG CAPTCHA challenge, and thus it is difficult to estimate the object moving speed to compute the shift offset before superimposing consecutive masks. Therefore, the contour information is well camouflaged in both a single binary mask and a superimposed mask.

One challenge in EI-DCG design is the local density difference caused by the presence of hollow objects and the white dilation surrounding the moving objects (used to make objects more visually distinguishable), which may facilitate automated attack. On

one hand, the hollow area and the white dilation area, which accompany the moving object in consecutive frames, may remain white (and thus appear "sparser") in the superimposed mask, indicating possible presence of foreground objects. Also, due to the randomness in the layout of background tiles, the local density of foreground objects in a single frame may occasionally become relatively higher/lower than surrounding areas. In this case, the automated attack could first detect subareas that exhibit such local density anomalies, randomly pick one of them, and drag it to the target. Other automated attacks using object detection in computer vision would be extremely hard, both theoretically and computationally, if not entirely impossible in this case, due to a lack of presence of distinct object visual features such as color, gradient, corner points, edge, or shape, and a lack of prior knowledge of object features. Therefore, neither feature-based nor apearance-based object detection would work well, leaving the best hope with an anomaly-based detection method such as the automated density-based attack that is also computationally efficient.

To evaluate the robustness of EI-DCG CAPTCHA against such density-based automated attack, we applied the automated attack to 3 different masks, i.e., single binary mask (***single***), superimposition of 3 consecutive masks (***3x***), and the frequency map of 3 consecutive masks (***freq***). Our automated attack assumes that the locations of the target objects are already known (e.g., known via a simple image-based relay attack). First, a screen-captured frame is converted into a binary mask with the target area removed. Second, the remaining area (i.e., activity area of moving objects) is divided into $m \times m$ equal-sized subareas (e.g., 40×40 pixels, Figure 7). An $m \times m$ density matrix is created in which each element indicates the black pixel count of the corresponding subarea. The centroids of subareas that correspond to local peak or valley elements in the 2D density matrix are treated as the location candidates of moving objects (Figure 7(a)). Third, the automated attack randomly selects a location candidate and performs a drag-and-drop operation from the cell centroid to each of the target objects.

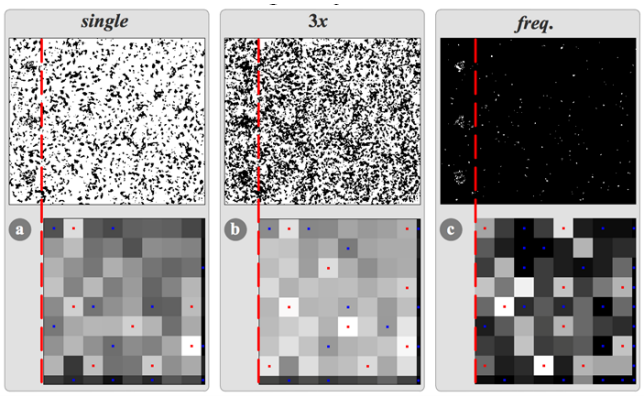

Figure 7: (a-c) Top: single binary mask, superimposition of 3 consecutive binary masks, and binary mask of frequency map with pixels having the highest possible frequency (i.e., 3) shown as white. Bottom: density matrix with grid interval as 40 pixels. The red and blue dots indicate local density peaks and valleys, respectively.

A superimposed mask (***3x***) is generated by superimposing the current binary mask with its previous two consecutive masks (Figure 7(b)). The value of a pixel in the frequency map is the number of times a black pixel appears at that pixel location across the 3 consecutive masks. The frequency map is further binarized so that only those pixels whose value is 3 (the highest possible frequency count of a black pixel) will be shown as white in the final binary mask (Figure 7(c)). The density matrix of such a binary mask records the white pixel count in each subarea (2nd row, Figure 7(c)).

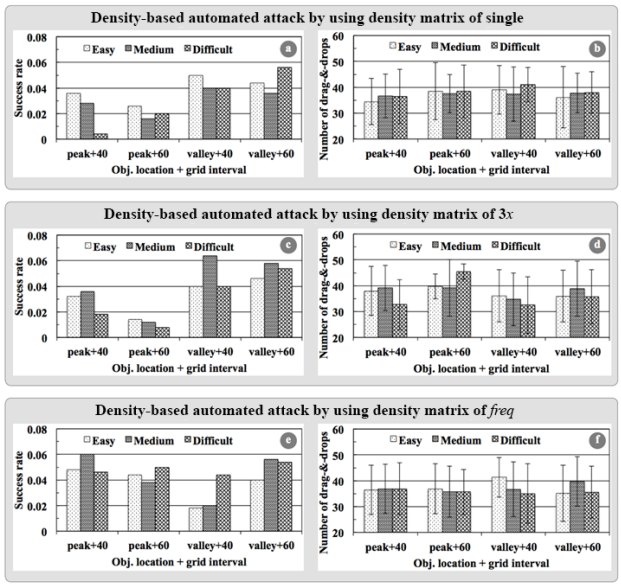

Figure 8: Success rate and number of drag-and-drops in density-based automated attack with ≤50 drag-and-drops for each attack. (a)(c)(e) Success rates by using density matrices of single, $3x$, and freq. (b)(d)(f) The mean and standard deviation of the number of drag-and-drops to complete an EI-DCG CAPTCHA challenge.

We randomly created 100 challenges of EI-DCG CAPTCHAs corresponding to each difficulty level. There are 12 groups of attacks with various parameter settings (Table 2) for each difficulty level. Each group contains 500 attacks on randomly selected challenges from the corresponding 100-challenge set. Each attack will perform drag-and-drop at most 50 times ("time out" otherwise). In our first experiment, we set the maximum number of drag-and-drop attempts allowed to be 21, i.e., each target object will receive ≤7 drops on average, given 3 target objects. This is the threshold parameter that our EI-DCG CAPTCHA implementation will use in practice (later in, Section 5, we will demonstrate that such a thresholdization does not have much impact on the usability of solving EI-DCG challenges by legitimate users).

Table 2: Parameter settings for the density-based automated attack

Parameters	Values
Difficulty level	Easy, Medium, Difficult
Density matrix (DM)	Single mask, 3x superimposition, Freq. map
Obj. locations (OL)	Local density peak, Local density valley
Grid interval (GI)	40 pixels, 60 pixels

Our result indicates that the success rate for density-based automated attack is lower than 0.8%, given ≤7 drag-and-drops per target object on average. Since this attack is based on local density anomaly, that occurs with randomness, the difficulty level is not necessarily reflected in the success rate. This success rate is well-aligned with the acceptable security level for CAPTCHAs (e.g., as specified by Zhu et al. [21]).

Next, we further experiment with ≤50 drag-and-drops in order to gain more insights. As shown in Figure 8 (a)(c), localizing foreground objects by using local valleys in the density matrix of *single* and $3x$ provides a higher success rate than that by using local peaks. In a single binary mask (*single*) or the superimposition of 3 consecutive masks ($3x$), the white dilation area surrounding the moving object (for highlighting the object) may form relatively sparse subareas (valleys), thereby making the valley-based attacks more effective than peak-based attacks. On the other hand, since there

are not many pixels in the binary mask of frequency map ($freq$), the local peaks more likely correspond to the moving traces left by moving objects. Therefore, the success rate of peak-based attack by using density matrix of $freq$ is higher than the other two. For the same reason, a small grid interval, such as 40 pixels, may result in many meaningless valleys in the density matrix of $freq$ (e.g., Figure 7(c)) and thus likely lead to a lower success rate. This explains why in the experiment for $freq$, 'valley+40' has a lower success rate than the other three configurations i.e., 'valley+60', 'peak+40', and 'peak+60' (Figure 8(e)).

Experiment on the number of drag-and-drops (Figure 8(b)(d)(f)) indicates that a successful automated attack usually requires an average of 30∼40 drag-and-drops to complete the CAPTCHA. which is much higher than our threshold of 21.

5. USABILITY AND RELAY ATTACK RESISTANCE

In this section, we present the design and the results of our study to evaluate the usability of our proposed EI-DCGs with the three difficulty levels (Easy, Medium, Difficult). Then, we demonstrate the resilience of EI-DCGs against relay attacks. To this end, we present two relay attack studies we conducted to measure the performance of the Stream Relay attack (Section 2) against EI-DCGs. Finally, we develop a Stream Relay attack detection mechanism based on the differences between the users' solving performance in the attack studies and the usability study. We note that the Static Relay attack against EI-DCGs will naturally fail given that a single or multiple static snapshots of the underlying emerging image does not reveal much meaningful information to a human solver (thus, we do not evaluate this attack).

5.1 Usability Study

In our usability study of EI-DCGs, we use EI-Nu as a baseline – our goal is to compare the usability of EI-DCGs with that of EI-Nu. Basically, we wanted to determine how much usability degradation occurs in EI-DCG over EI-Nu, as a trade-off to enhancing the security against relay attacks. We utilized the Amazon Mechanical Turk (MTurk) service to recruit participants for the study. The study was approved by our University's Institutional Review Board.

5.1.1 Study Design

Each EI-Nu CAPTCHA challenge is of size 285×125 and is displayed as a 6-second video that loops continuously. We asked the participants to type the three characters of the challenge in a textbox and press the "submit" button when they are done. Each EI-DCG challenge is of size 360×400. The user task is to drag-drop the answer objects to their corresponding target objects within 60 seconds (time-out). If the user cannot complete the task within the 60 seconds, the challenge is considered unsuccessful. We generated 100 challenges for each category of tested CAPTCHAs: EI-Nu and EI-DCG (Easy, Medium and Difficult). We employ a within-subjects experimental design, where we ask each participant to solve 5 challenges each for all the four categories. The order of presenting the four categories (EI-Nu, EI-DCG_Easy, EI-DCG_Medium, and EI-DCG_Difficult) followed the standard 4×4 Latin square to reduce the effect of learning biases, while the challenges within each category followed a random order. We recruited 120 MTurk workers, and the experiment took 27 minutes on an average to complete per worker.

We subjected the MTurk workers to a consent form. Then, we asked them to fill-out a demographics form, solve five challenges of one of the tested CAPTCHA categories, and fill-out a survey form about their experience. The survey contains the 10 System Usable Scale (SUS) [6] 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"). We used a similar design to test the rest of the categories. The demographics of the study participants are shown in the second column of Table 3.

Table 3: Demographics of participants in the usability and relay attack studies

	Usability	**Stream Relay Attack**	
Participants Study Type		LSHL	HSLL
Participants Size (N=195)	N = 120	N = 27	N = 48
Age (%)			
<18	0	0	2.1
18 - 24	26.6	16.7	8.3
25 - 34	44.2	70.8	58.3
35 - 50	19.2	8.3	27.1
>50	10	4.2	4.2
Gender (%)			
Female	39.2	25	27.1
Male	60.8	75	72.9
Education (%)			
High School	30.8	0	22.9
Bachelor	43.3	62.5	72.9
Master	24.2	33.3	4.2
PhD	1.7	4.2	0
Field of Study/Profession (%)			
Computer Science	31.7	25	22.9
Engineering	10.8	29.2	10.4
Medicine	0.8	4.2	2.1
Journalism	2.5	4.2	4.2
Finance	5.8	20.8	25
Business	12.5	8.3	12.5
Social sciences	0	0	8.3
other	35.9	8.3	14.6

5.1.2 Results

We evaluated the usability of the tested CAPTCHA categories with respect to the measures of: (1) solving time, (2) error rate, and (3) user experiences, as described below. The results are summarized in Table 4.

Solving time: We calculated the solving time as the time taken by the participants to solve each challenge. In case of EI-Nu, we start measuring the timing from the time the challenge is displayed till the submit button is pressed. Whereas, in case of EI-DCG, we record the timing till the participants drag-drop all the answer objects to their corresponding target objects. We considered in our calculation the time taken only corresponding to the challenges solved successfully. The average solving time is shown in the third column of Table 4.

The time taken to solve EI-DCG challenges is about double the time taken to solve EI-Nu challenges. However, it is still less than 25 seconds on average. Moreover, the time for solving EI-DCG increases with the difficulty level of EI-DCG. A Friedman's test

Table 4: The solving time, error rate, number of drags, number of attempts and SUS scores for the usability study

Challenge Type	**SUS**	**Time (sec)**	**#Drags**	**#Attempts**	**Error Rate**
	mean (std. dev.)				
EI-Nu	71.75 (18.39)	10.34 (6.21)	N/A	N/A	0.16
EI-DCG					
Easy	55.94 (19.75)	19.82 (10.20)	3.82 (1.79)	1.17 (1.86)	0.13
Medium	57.15 (18.09)	21.55 (10.55)	3.82 (1.58)	1.51 (2.42)	0.10
Difficult	56.00 (20.08)	23.34 (11.60)	3.84 (1.85)	1.44 (2.03)	0.13

showed a statistically significant difference between the solving time of the four tested categories ($\chi^2(3)$ = 500.06, p $<$ 0.001). Further analyzing using pairwise Wilcoxon Signed-Rank test with Bonferroni correction, we found a statistically significant difference between all of the tested pairs (p $<$ 0.001).

Error Rate: The error rate represents the percentage of the challenges that were not solved successfully by the participants. The last column of Table 4 shows the error rate for solving each of the tested CAPTCHA categories. All of the categories had low error rate with the minimum error rate for EI-DCG medium as 0.10 and the maximum for EI-Nu as 0.16. The lower error rate in EI-DCGs compared to EI-Nu may be attributed to to the momentary feedback that EI-DCG provides. Whenever the participant drag-drops a correct object to its corresponding target, the object disappears, which informs the user he performed a correct drag-drop. However, EI-Nu does not check the users response until after he submits the whole answer of the challenge.

Further, we analyzed the number of drag-drops performed by the participants, which represents the error rate per drag. We noticed that a minimum of three drag-drops is required to complete any challenge and on an average the users performed less than four drag-drops in all the EI-DCG categories. Finally, we analyzed the number of attempts (clicks that do not correspond to object drag) performed by the users and we found the users performed less than 2 attempts on average for all of the EI-DCG categories. Upon further analysis of the collected logs, we found that some of the participants performed many drags and attempts (up to 37) while solving challenges. However, the fraction of such actions is extremely low, which confirms that we can limit the number of allowed drags and drops to 21 to limit the ability of density-based automated attack (as analyzed in Section 4) successfully without impacting the usability much. The overall error rate, after limiting to 21 drags/attempts, becomes 0.14, 0.11, and 0.14 for Easy, Medium and Difficult EI-DCG, respectively. These errors rates are still similar to that of EI-Nu.

User Experience (SUS Scores): The first column of the Table 4 shows the SUS scores corresponding to the tested CAPTCHA categories. In our SUS calculations, we ignored the responses of 17 participants as they seem to answer the questionnaire randomly, either by giving the same rating to all the questions or at least answer two contradicting questions with the same answer (i.e., we removed the responses from participants who answer both of "I found the system unnecessarily complex" and "I thought the system was easy to use" with "strongly agree").

We find degradation in the user experience in EI-DCG compared to EI-Nu. However, the SUS scores for EI-DCG are still above 50.9, which means EI-DCGs have fair usability [5]. Comparing the SUS scores using Friedman's test shows statistical significant difference among the tested CAPTCHA categories ($\chi^2(3)$ = 87.63, p $<$ 0.0001). Further, we used pairwise Wilcoxon Signed-Rank test with Bonferroni correction to assess the difference between each of the pair of the four categories. Significant differences are found (p $<$ 0.01) between EI-Nu and the three categories of EI-DCG. However, no significant difference is found in the SUS score between any pair of the EI-DCG categories.

Summary: The results of the usability study show some degradation of the user experience represented in lower SUS score and higher time taken to solve EI-DCG challenges compared to EI-Nu challenges. However, the average SUS scores for EI-DCG show that they still have fair usability. Moreover, the error rate decreased slightly compared to EI-Nu. Given that EI-DCG offers higher security than EI-Nu, especially against relay attacks, we believe that this degradation in usability may be acceptable.

5.2 Streaming-based Relay Attack Study

We conducted two studies to investigate the ability of EI-DCG to resist Stream Relay attack introduced in [12]. The studies differs in only the location of the participants. In the first study, tagged Low-Speed High-Latency (LSHL), we recruited participants from a developing country (India), where we expect the users to have a slow Internet connection and they reside on far proximity of the attacker (residing in the USA). In the second study, High-Speed Low-Latency (HSLL), we recruited participants from a developed country (USA), where we expect the users to have a fast Internet connection and they reside in near proximity of the attacker (USA). These two attack models emulate realistic relay attack scenarios [12]. In both models, the human-solvers attempt to solve EI-DCG challenges that are streamed to them in-real time from the attacker's machine using the VNC streaming software.

5.2.1 Study Design

Following the study design in the usability study, we employed a within-subjects experimental design, where we ask each participant to solve 5 challenges for all of the three EI-DCG categories. The order of presenting the three categories followed standard 3×3 Latin square, and the challenges within each category followed a random order. We asked the MTurk workers to connect to a computer which resides in our university (streaming server) via the RealVNC Java applet (streaming client). Then, we subject them to consent an agreement. Next, we ask them to fill-out a demographics form, and solve five challenges of one of the categories. We followed a similar design to test the other categories. The study was approved by our University's Institutional Review Board.

We conducted two separate streaming-based relay attack studies. The two studies differ only in the location of the participants. In the first study (LSHL), we recruited 27 participants in India which simulate the real scenario settings in which the attacker is in USA and hires human-solvers in far and developing countries. In the second study (HSLL), we recruited 48 participants from USA. The second study is to assess how much the performance of the attack will increase when the attacker recruits solvers in near proximity and from developed countries. The demographics of our participants are shown in columns 3-4 of Table 3. The participant characteristics in our relay attack studies are in line with that in our usability study, allowing us to fairly compare the two settings in a between-subjects design.

5.2.2 Results

We evaluated the performance of the participants in solving EI-DCG over the streaming channel with respect to the measures of solving time and error rate. The results are summarized in Table 5.

In the LSHL relay attack setting, only two participants could complete one of the EI-DCG Easy variant, and only one participant could complete one of the EI-DCG medium variant. The overall error rate is therefore 0.99 on an average for the three categories of EI-DCG. The average time taken by the participants to complete the challenges was 42.68 seconds. Moreover, the participants performed much higher number of drag-drops and attempts compared to the participants in the usability study. Moreover, if we limit the number of allowed drags/attempts to 21, the overall error rate for the Easy EI-DCG becomes 1.00.

In the HSLL relay attack setting, the error rate decreased from around 0.99 to 0.87 on average when compared to the LSHL relay attack study. However, the error rate is still considerably high. The completion time and number of drags and attempts for the partici-

Table 5: The solving time, error rate, number of drags, and number of attempts for the India-based and USA-based streaming-based relay attack studies on EI-DCG

EI-DCG Challenge Type	LSHL (human-solver in India)				HSLL (human-solver in USA)			
	Time (sec)	# Drags	# Attempts	Error Rate	Time (sec)	# Drags	# Attempts	Error Rate
	mean (std. dev.)				*mean (std. dev.)*			
Easy	39.05 (5.47)	8.00 (4.24)	24.00 (1.41)	0.98	18.73 (10.62)	4.35 (3.05)	4.71 (15.83)	0.87
Medium	49.92	4.00	15.00	0.99	22.13 (11.69)	4.17 (2.91)	1.63 (1.88)	0.88
Difficult	-	-	-	1.00	24.05 (11.71)	4.53 (2.37)	1.66 (2.89)	0.86

pants who successfully solved the challenges are comparable to the ones in the usability study, except for the number of attempts in the Easy variant. Limiting the number of allowed drags/attempts to our threshold of 21 did not effect the over all error rate. Comparing the successful completion time between each variant of EI-DCG in usability and its corresponding streaming-based relay attack study, using Mann-Whitney test, did not reveal statistically significant differences, however. Further investigation of the collected data shows that only 9 out of the 48 participants (18.75%) were able to complete any EI-DCG challenges. On an average each of them solved around 10.33 out of the 15 challenges.

The above analysis suggests that EI-DCG is very hard to solve via streaming-based relaying, especially when the human-solvers are located far away and have slow Internet connections. The chance of solving EI-DCG challenges seem to increase, but still only marginally, when recruiting the solvers residing in near proximity and having high-speed Internet connections. Even when the solvers can solve the CAPTCHA challenges, we will next show how they can be detected based on different game play patterns compared to legitimate human users.

5.2.3 Attack Detection Technique

We design and implement a machine-learning based EI-DCG Stream Relay attack detection mechanism based on the differences between the participants' performances in completing the challenges in the usability study and the human-solvers' performance in the HSLL relay attack study. We logged all the participants' interactions with the challenges, while they were solving the challenges. We logged the interactions with the challenges regularly and whenever a mouse event is performed. Specifically, at each timestamp, we log the mouse position, mouse status (up or down), and for each object, we log its position and weather it is being dragged or not. From each of the collected challenge log files, we extract a total of sixteen different features, as described below:

- **Number of Drags (ND)**: Total number of drags.
- **Number of Attempts (NA)**: The number of mouse clicks outside of the moving objects.
- **Distance-based Features**: (1) *Distance Drag (DD)*: The total distance of the mouse movement while it is dragging an object; (2) *Distance Attempts (DA)*: Distance of the mouse movement while the mouse status is down but not dragging an object; and (3) *Distance mouse Up (DU)*: Distance of the mouse movement while the mouse status is up.
- **Time-based Features**: (1) *Completion Time (T):* The total time taken by the participant to complete the challenge; (2) *Time drags (TD):* The total time in which an object is being dragged; (3) *Time Attempts (TA):* Time in which the mouse is down and no object is being dragged, and (4) *Time mouse Up (TU):* Time in which the mouse status is up.
- **Speed-based Features** (Distance/Time): Speed of Drags (SD), Speed of Attempts (SA) and Speed of Mouse Up (SU).
- **Acceleration-based features** (Speed/Time): Acceleration of Drags (AD), Acceleration of Attempts (AA) and Acceleration of Mouse Up (AU).
- **Max Attempt (MA)**: The maximum time taken in a single attempt.

For each of the EI-DCG variants, we picked randomly a number of instances of the successfully completed challenges in the usability study equal to the number of successfully completed challenges in the HSLL relay attack study. Then, we implemented a Java program to check which subset of features along with which classifier provides the best results for each of the challenges categories (Easy, Medium, Difficult). We experimented with different classifiers: SVM (C-SVC and nu-SVN, linear, polynomial and radial kernels), Multilayer Perceptron, Naive Bayes, Random Forest, Random Tree, Simple Logistic and Logistic.

As performance measures for our classification task, we used Precision, Recall and F-measure (F1 score), as shown in Equation 1 (TP: True Positive; FP: False Positive; FN: False Negative). Precision measures the security of the proposed system, i.e., the robustness of the system in detecting the relay attack. Recall measures the system usability as low recall leads to high rejection rate of legitimate users' actions. F-measure is the weighted average of both of the precision and recall. We select the classifier with maximum recall, to reduce the rejection of honest users, and have acceptable precision rate.

$$precision = \frac{TP}{TP+FP}; recall = \frac{TP}{TP+FN}; \\ F\text{-}measure = 2 * \frac{precision * recall}{precision + recall} \quad (1)$$

Table 6: Results of using the optimal feature subset for each EI-DCG game in the classification of legitimate user and HSLL streaming-based relay attacker

Challenge Type	Classifier	Features	Precision	Recall	F-measure
Easy	Multilayer Perceptron	ND, T, TD, TA, SD, SU	0.78	0.90	0.84
Medium	Random Tree	ND, T, DD, DA, DU, TU, MA, AD, AU	0.72	0.87	0.79
Difficult	Multilayer Perceptron	T, TU, AD, AA, AU	0.66	0.91	0.76

The results of the best classifier along the best features subset for each of the EI-DCG difficulty level are shown in Table 6. The results show that a classifier can be effectively built such that it rejects around 11% of legitimate users and be able to detect around 65% for HSLL relay attackers. As shown in Table 5, on average only 13% of HSLL relay attackers could solve the EI-DCG challenges, utilizing our proposed detection method 65% of them could be detected. Therefore, the detection rate for the HSLL relay attack is about 95%. The LSHL relay attack is already prevented with at least 98% probability (Table 5). This suggests that the streaming-based relay attack in general has a very low chance of succeeding against EI-DCG CAPTCHAs.

5.3 Discussion

The previous subsection shows that EI-DCGs is resilient to streaming-based relay attacks. Inherently, this ability is due to the fact that the user needs to see the challenge at about 40 fps to be

able to recognize the objects and solve the challenge successfully. That requires high data connection speed with low latency between the attacker and the human-solver.

As the game size is 340×400 and the required frame rate is 40 fps, using 2 bit per pixel will require the connection speed of 10.38 Mbit/s between the attacker and the solver, if no compression is performed. However, RealVNC performs some compression on the data sent between the client and the server to enhance the performance. First, RealVNC sends only the modified pixels between the current frame and the previous frame rather than sending the whole frame. Second, the RealVNC client allows the user to adjust the quality of the images, varying from low quality (8 colors) to best quality (all available colors). In order to evaluate the minimum bandwidth required between the attacker and the human-solver to transfer the game with 40 fps, we performed an experiment in which we connected two laptops over a LAN, installed RealVNC server on one of them and RealVNC client on the other. We varied the RealVNC compression levels starting from the best compression that achieves the best transmission quality. We used Wireshark to capture the packets sent between the two laptops in all of the cases, and then analyzed the average data rate used. The average Mbit/s for the getting all colors with maximum compression is 4.70 Mbit/s, 3.15 Mbit/s for the default settings (256 colors) and 2.22 Mbit/s for the best compression (8 colors). This shows the minimum connection speed required between the attacker and human-solver should be 2.22 Mbit/s to be able to solve the EI-DCG at the human-solver's end. Any slower connection would result in a jittery rendering of the game, and therefore the human-solver will not be able to solve the EI-DCG successfully. This justifies the results of the user studies presented in the previous subsection.

According to [3], the average connection speed is 2.0 Mbit/s in India. Also, since the long distance between the attacker (USA) and the human solver slows down the connection speed, almost all the participants in India could not remain connected to our server with high speed and they failed to successfully solve the challenges. On the other hand, the average connection speed in USA is 11.1 Mbit/s. Also, the distance is much less between the attackers and the human-solvers in the HSLL streaming-based relay attack study. This explains why some of the human-solver were able to successfully solve some of the challenges.

We tested multiple screen-sharing applications, such as TeamViewer and anydesk. All of them require almost similar data rates between the server and the client to transfer the challenges. For example, TeamViewer requires 2.73 Mbit/s for gray-scale, and anydesk requires 2.63 Mbit/sec. Therefore, we conclude that the results of our relay attack studies are not limited to RealVNC.

A third relay attack scenario was suggested in [12], namely, Small Game Relay, whereby the attacker reduces the game size before sending it to the human-solver to lower the data rate required between the attacker and the solvers. However, this scenario cannot be applied to EI-DCG as reducing the game size will make it almost impossible for the human-solver to recognize the moving objects (because of the emerging effect).

6. CONCLUSIONS

We proposed a new class of CAPTCHAs, EI-DCG, based on the notion of emerging images and Dynamic Cognitive Games. EI-DCG applies a series of countermeasures, such as pseudo 3D rotation, hidden edge segments, segment split-erosion-rotation-translation and tiled background, to resist automated object recognition based on both single frames and frame superimposition. Since a human can perceive objects in EI-based frames from motion, playing an EI-based video in a relatively slower frame rate decreases the human recognition rate. Based on this property, a faster frame rate (i.e., 40 fps) was applied for the normal playing of an EI-DCG challenge. The higher the network delay is, the lower the frame rate of EI-DCG becomes on the human-solver's side, providing less opportunity for the solver to complete the challenge successfully. The experiments against both automated and relay attacks indicated the robustness of EI-DCG.

Acknowledgments

The work has been partially supported by a grant from the National Science Foundation (CNS-1255919) and an award from COMCAST.

7. REFERENCES

[1] Are you a human. http://areyouahuman.com/.
[2] Nucaptcha. http://www.nucaptcha.com/.
[3] Akamai. Akamai's state of the internet: Q4 2014 report. http://www.stateoftheinternet.com/resources-connectivity-2014-q4-state-of-the-internet-report.html.
[4] H. S. Baird, A. L. Coates, and R. J. Fateman. Pessimalprint: a reverse turing test. *International Journal on Document Analysis and Recognition*, 2003.
[5] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 2009.
[6] J. Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 1996.
[7] J. M. G. Hidalgo and G. Alvarez. Captchas: An artificial intelligence application to web security. *Advances in Computers*, 2011.
[8] G. Kanizsa. *Organization in vision: Essays on Gestalt perception*. Praeger New York, 1979.
[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 *Symposium on Usable Privacy and Security*, 2009.
[11] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or. Emerging images. In *Transactions on Graphics*, 2009.
[12] M. Mohamed, S. Gao, N. Saxena, and C. Zhang. Dynamic cognitive game captcha usability and detection of streaming-based farming. In *the Workshop on Usable Security (USEC)*, 2014.
[13] 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 *ACM symposium on Information, computer and communications security*, 2014.
[14] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: Captchas-understanding captcha-solving services in an economic context. In *USENIX Security Symposium*, 2010.
[15] J. K. Tsotsos. On the relative complexity of active vs. passive visual search. *International journal of computer vision*, 1992.
[16] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *EUROCRYPT 2003*.
[17] 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. *Transactions On Dependable And Secure Computing*, 2013.
[18] 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*, 2012.
[19] J. Yan and A. S. El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Computer Security Applications Conference*, 2007.
[20] J. Yan and A. S. El Ahmad. A low-cost attack on a microsoft captcha. In *Conference on Computer and communications security*, 2008.
[21] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and design of image recognition captchas. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.