Attacks and Design of Image Recognition CAPTCHAs

Bin B. Zhu*[1], Jeff Yan[2], Qiujie Li[3], Chao Yang[4], Jia Liu[5], Ning Xu[1], Meng Yi[6], Kaiwei Cai[7]

[1] Microsoft Research Asia, Beijing, China
[2] Newcastle University, United Kingdom
[3] Nanjing University of Science and Technology, China
[4] University of Science and Technology of China, Hefei, China
[5] iCare Vision Tech.CO., LTD, Beijing R&D Center
[6] Computer and Information Science, Temple University, USA
[7] Beijing University, Beijing, China

{binzhu, ningx}@microsoft.com, jeff.yan@ncl.ac.uk, jliu@icarevision.cn, mengyi@temple.edu

ABSTRACT

We systematically study the design of image recognition CAPTCHAs (IRCs) in this paper. We first review and examine all IRCs schemes known to us and evaluate each scheme against the practical requirements in CAPTCHA applications, particularly in large-scale real-life applications such as Gmail and Hotmail. Then we present a security analysis of the representative schemes we have identified. For the schemes that remain unbroken, we present our novel attacks. For the schemes for which known attacks are available, we propose a theoretical explanation why those schemes have failed. Next, we provide a simple but novel framework for guiding the design of robust IRCs. Then we propose an innovative IRC called Cortcha that is scalable to meet the requirements of large-scale applications. Cortcha relies on recognizing an object by exploiting its surrounding context, a task that humans can perform well but computers cannot. An infinite number of types of objects can be used to generate challenges, which can effectively disable the learning process in machine learning attacks. Cortcha does not require the images in its image database to be labeled. Image collection and CAPTCHA generation can be fully automated. Our usability studies indicate that, compared with Google's text CAPTCHA, Cortcha yields a slightly higher human accuracy rate but on average takes more time to solve a challenge.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *authentication, unauthorized access*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis – *object recognition.*

General Terms

Security, Human Factors.

Keywords

CAPTCHA, Human Interactive Proof, HIP, security, robustness, Cortcha, image recognition CAPTCHA, IRC, object recognition.

1. INTRODUCTION

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [1][2][3], also known as Human Interactive Proof (HIP), is an automated Turing test in which both generation of challenges and grading of responses are performed by computer programs. CAPTCHAs are based on Artificial Intelligence (AI) problems that cannot be solved by current computer programs or bots, but are easily solvable by humans. A client who provides a correct response to a challenge is presumed to be a human; otherwise a bot. CAPTCHAs have been widely used as a security measure to restrict access from bots.

Text CAPTCHAs are almost exclusively used in real applications. In a text CAPTCHA, characters are deliberately distorted and connected to prevent recognition by bots. Most of the proposed or deployed text CAPTCHAs have been broken [4][5][6][7][8][9]. It is possible to enhance the security of an existing text CAPTCHA by systematically adding noise and distortion, and arranging characters more tightly. These measures, however, would also make the characters harder for humans to recognize, resulting in a higher error rate and an increased level of frustration. There is a limit to the distortion and noise that humans can tolerate in a challenge of a text CAPTCHA. Usability is always an important issue in designing a CAPTCHA [10]. With advances of segmentation and Optical Character Recognition (OCR) technologies, the capability gap between humans and bots in recognizing distorted and connected characters becomes increasingly smaller. This trend would likely render text CAPTCHAs eventually ineffective. Finding alternative approaches in designing CAPTCHAs to replace text CAPTCHAs has become increasingly important. A major effort has been directed to developing CAPTCHAs based on image or object recognition [11][12][13][15][16][17]. Images are rich in information, intuitive to humans, and of a large variation. More importantly, there are still many unsolved AI problems in image perception and interpretation. Images seem to be a better medium than characters for designing CAPTCHAs.

The research of text CAPTCHAs has roughly proceeded in the following way. The earliest inspiration was a clever but rough idea: although recognizing printed fonts was a solved problem, it is hard for OCR to recognize distorted fonts. Therefore early

schemes were designed to make them hard for OCR to recognize. Attacks on the early designs were studied, and the principle of segmentation resistance emerged: computers turn out to perform better than humans in recognizing individual characters, even under severe distortion [18]. However, segmentation, which is to locate individual characters in the right order, is a computationally expensive and combinatorially hard problem. Thus text CAPTCHAs should be designed to be segmentation-resistant [19]. The attack by Yan and El Ahmad [9] further enhanced our understanding of various segmentation resistance mechanisms. Such an iterative process of designs and attacks has led to a better CAPTCHA design. In the meanwhile, failure modes of and some design principles for CAPTCHAs have also started to emerge.

We believe that such an evolutionary process for studying text CAPTCHAs is applicable to the domain of image recognition CAPTCHAs (IRCs). In this paper, we systematically study the design of IRCs. Similar to text CAPTCHAs, the earliest inspiration for the design of IRCs was also a rough, high-level idea: it's difficult for current computers to recognize certain content of an image. However, compared to the extensive security study of text CAPTCHAs, the current collective understanding of failure modes for IRCs is limited. There are few established insights or principles on how to make use of the difficulty of image recognition in a sound way to design secure IRCs. For example, it is well known that a strong IRC should rely on image semantics. However, there is no deep understanding of how to properly make use of image semantics. It is unclear what use of image semantics would fail a CAPTCHA or lead to a more secure design. We will show that by attacking representative IRC schemes and by providing a theoretical explanation to existing attacks, we can derive useful lessons and fundamental principles for guiding the design of IRCs. Our work advances the current collective understanding of the design of such CAPTCHAs.

This paper is organized as follows. We provide in Section 2 a brief yet rather comprehensive review of existing IRCs, and evaluate each scheme against the practical requirements in CAPTCHA applications, particularly in large-scale real-life applications such as Gmail and Hotmail. In Sections 3 and 4 we present attacks on the representative schemes identified in the previous section. For the representative schemes that remain unbroken, we present our novel attacks in Section 3. For the schemes that known attacks are available, we describe the attacks briefly in Section 4. In Section 5, we propose a theoretical explanation why all these schemes have failed. We also define a framework to summarize the lessons we have learned, as well as to provide guidelines for designing robust IRCs. Section 6 presents the design of a novel IRC called Cortcha, along with a study of its security, usability, and potential issues. Section 7 concludes the paper and discusses some future work.

2. EXISTING IRC SCHEMES: HOW GOOD ARE THEY?

2.1 Desired Properties of CAPTCHAs

Early research has summarized some requirements and desirable properties for a CAPTCHA [1][3][12]. We, however, add one additional desirable property that comes from large-scale real-life applications such as Gmail and Hotmail:

- **Scalability**. Scalability measures the range (number) of challenges a CAPTCHA scheme can generate without sacrificing the scheme's the robustness and usability. A scheme is scalable if it can scale up its output (the number of

generated challenges) with acceptable robustness and usability. A scalable scheme can meet the demand of large-scale applications such as Gmail and Hotmail, but an un-scalable one cannot. While it is easy for text CAPTCHAs to achieve scalability, many existing IRCs cannot generate a large number of challenges without sacrificing robustness or usability.

It is worth mentioning a fundamental requirement of CAPTCHAs under the context of IRCs: both challenge generation and the response grading should be automated without human involvement. This requirement ensures that the whole system operates in an automatic manner. This requirement is feasible since it is possible that a task and its reversal have asymmetric complexity. Such a difference in complexity is the base for modern public key cryptosystems. This requirement is easily met by text CAPTCHAs, but has proven to be difficult for many IRCs for the following simple reason. Typically, an IRC builds its security on the difficulty for computers to understand the semantic content of images or visual objects. That is, computers used for generating challenges do not really understand the images or visual objects. As such, many IRCs require human involvement, in which images are manually labeled or selected.

2.2 Metrics for Attack Effectiveness

The first metric in evaluating the effectiveness of an attack is the success rate of the attack. The tolerable success rate of an attack on a CAPTCHA depends on the cost of the attack. A rule of thumb is given in [20]: bots should not have a success rate higher than 0.01%. This is a very challenging number in designing a CAPTCHA. By using IP monitoring such as the token bucket scheme proposed in [15] together with a CAPTCHA, the tolerable success rate of attacks can be relaxed to 0.6% (assuming that *TB-Refill* is 3 for the token bucket scheme). We adopt this threshold in this paper: bots should not have a success rate higher than 0.6%.

The average time needed for an attack to produce a response to a challenge, referred to as the attack response time in this paper, is another metric to evaluate the effectiveness of the attack. An attack should produce a response within the time frame that humans respond to a challenge. Otherwise it is easy to tell if a response is from a bot or human. According to [12], a CAPTCHA should be designed such that humans can respond within 30s. As a result, an effective attack should also respond within 30s.

The following criterion is adopted in this paper: if on average an attack produces a response within 30s with a success rate of 0.6% or higher, the attack is claimed to be effective; otherwise ineffective.

2.3 Existing IRC Schemes

Existing IRCs are all based on the assumption that computers cannot perform well a certain type of task on images. As we shall see later in this paper, many of these assumptions were actually incorrect, and thus the IRC schemes can be successfully attacked. Early IRCs include Bongo [21] in which two groups of visual blocks (e.g., lines, circles, and squares) that humans can find some characteristics to separate them are displayed. A user is asked to classify a visual block into the right group. A random guess results in a success rate of 50%. Pix [21] is another early IRC that uses a large database of labeled images which are pictures of concrete objects (horses, tables, houses, flowers, etc.). It first picks an object label at random and finds six images of that object from the database, randomly distorts them, and then presents to a user to label the object. Labeling an object may be ambiguous. Different

users may label the same object differently. It is also difficult to evaluate an answer automatically. In addition, Pix depends on the language that users use. These problems are addressed in Animal Pix [21] which differs from Pix in the following ways: 1) it uses 12 animals instead of generic objects as the labeled objects; and 2) it asks a user to select from the set of predefined 12 animals instead of entering the object label. The cost is reduced security: a random guess of Animal Pix results in a success rate of 8.3%.

Chew and Tygar [11] proposed three CAPTCHA algorithms based on a database of labeled images generated by collecting the first 20 hits from Google's image search on inputting each word from a list of easily-illustrated words. The first CAPTCHA algorithm (CT_L) presents six images of the same subject, and asks a user to correctly describe the common term associated with the six images to pass the test. The second CAPTCHA algorithm (CT_S) presents two sets of images, with each set containing three images of the same subject, and asks a user to determine if the two sets have the same subject or not. The third CAPTCHA algorithm (CT_A) presents six images, five of the same subject and one of a different subject, and asks a user to identify the image of the different subject. Like Pix and Animal Pix, it is difficult to grade responses automatically for the first CAPTCHA algorithm, and a random guess would result in a sufficiently high success rate, 50% and 16.67%, respectively, for the second and third algorithms. In addition, Google's image search may return inaccurate or irrelevant images. Manual selection may be required to remove bad images. The image database would be too small to meet the scalability requirement.

Asirra [15] relies on the capability gap between humans and bots in distinguishing cats and dogs. It asks a user to identify cats out of a set of 12 photos of both cats and dogs. A large database of labeled images of cats and dogs is needed to generate Asirra challenges. Photos of cats and dogs from Petfinder.com are used in generating challenges. Asirra is not scalable. Petfinder.com has only a limited number of photos of cats and dogs. New photos are added slowly. With a high volume application such as Hotmail, the database is quickly exhausted and photos would have to be repeated, allowing adversaries to use previously used photos of cats and dogs to solve a new challenge.

Website HotCaptcha.com applies a CAPTCHA based on a large database of labeled photos from HotOrNot.com, a popular Website that invites users to post their photos and rate others' photos as "hot" or "not hot". The CAPTCHA asks a user to pick three hot people from nine photos of people presented to the user. Whether a person is hot or not is subjective and culture-dependent. Different people may give different answers. In addition, the success rate by a random guess, which is at least 1 in $C_9^3 = 84$ or 1.19%, may be sufficiently high that renders the CAPTCHA not suitable for many applications such as anti-spam for a free Web email service.

A recent proposal [16] (Orientation) is to exploit the capability gap between bots and humans to identify the orientation of an image. A user is asked to adjust a randomly rotated image to its upright orientation. A large database of candidate images is needed in this CAPTCHA. To generate such a database, images returned from a Web search are first obtained; a suite of automated orientation detectors is then applied to remove those images that can be set upright by a computer; and finally a social feedback mechanism is employed to remove those images hard for humans to set orientation. The quality of this CAPTCHA depends critically on the quality of the image labeling result from the social feedback

mechanism. It is unclear whether there exists an efficient social feedback system that can label a large number of images to meet the demand of a large scale application such as Gmail or Hotmail. In addition, a random guess may result in a sufficiently high success rate. In a challenge, a user is asked to move a scroll bar to adjust the orientation of an image, and the position of the scroll bar is returned for evaluation. The success rate of a random guess depends on the tolerance of variations in setting the upright orientation by different people. The data reported in [16] indicate that the success rate of a random guess when one image is used in a challenge is 4.48% ($= \sqrt[3]{0.009\%}$), which is high enough for many applications that several images are needed in a challenge.

A new CAPTCHA based on 3D object models is recently employed by Yuniti.com [22]. This CAPTCHA presents in a challenge three objects generated from a set of 3D object models, and asks a user to select the matching object from a list of objects for each of the three displayed objects. A major problem for this CAPTCHA is that it is costly to generate a large number of 3D objects for a large scale application. It is also possible for adversaries to reversely build the 3D models from the objects in used challenges, and then to use these models to find the matching objects in the list for the three objects in a new challenge.

A video CAPTCHA using labeled video clips from YouTube is proposed in [17]. A user is asked to label the content of a video clip in a challenge. However, labeling content is subjective; different users may label the same content differently.

In the following subsections, we'll discuss Asirra, ARTiFACIAL, and IMAGINATION. These are the IRCs that we shall examine in more details to learn the lesson why they are successfully attacked.
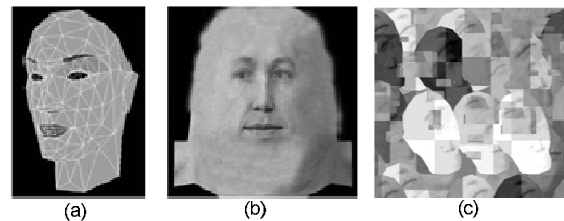

Figure 1. (a) 3D wire model. (b) Cylindrical head texture. (c) Challenge image.

ARTiFACIAL [12] relies on the capability gap between humans and machines in recognizing a human face. Humans can easily recognize a human face even if the face is distorted, partially occluded, or under poorly illumination. A face detector, however, still suffers from head orientation, face asymmetry, lighting and shading, and cluttered background [12]. In ARTiFACIAL, a 3D head model (Figure 1(a)) and a 512×512 pixel cylindrical texture map of an arbitrary person (Figure 1(b)) are used to generate a unique human face with random global head rotation, scaling, translation, and local facial feature deformations to take advantage of the head orientation and face symmetry limitations. The intensity of the face region is perturbed to break the face symmetry and to simulate illumination variances. Finally, a cluttered background is generated by randomly putting confusion heads and facial features on the image. A challenge image is shown in Figure 1(c). A user is required to identify the single human face in a challenge and click the six facial corners (four eye corners and two mouth corners) on the face to pass a test. It is claimed that the success rate for a bot to pass an ARTiFACIAL test is at most 0.0006% [12]. A worth-mentioning feature of

ARTiFACIAL is that in theory an infinite number of challenges can be generated [12].

Figure 2. Challenge images in IMAGINATION: (a) click test, (b) annotate test.

IMAGINATION [13] actually consists of two separate tests: a click test and an annotation test. The two tests are shown in Figure 2. In the click test, a distorted composite image tiled with 8 images is presented. A user has to click a position close enough to the geometric center of any one of the 8 constituent images to pass the test. In the annotate test, a distorted image containing a meaningful object and cluttering curves is presented. To pass a test, a user has to choose the correct label for the image from a list of 15 candidate words. Candidate labels used in this test are generated by adopting a WordNet-based method to avoid ambiguity and to thwart odd-one-out attacks when the correct choice is semantically different from all the others. A random guess of the annotation test results in a success rate of 6.67%.

In generating a challenge image of the click test, the region of the challenge image is randomly partitioned into 8 non-overlapping rectangles. Each rectangle is filled with an image randomly selected from a database, scaled if necessary. The following dithering step is then applied twice: the composite image is randomly divided into another 8 rectangular regions and the Floyd-Steinberg error-diffusion algorithm is applied to each region with independent dithering parameters including base colors (18, randomly chosen in RGB space). To further enhance the security, a factor α chosen randomly in the range of [0.5, 1.5] is used to multiply the spreading quantization error during the dithering. The intuition behind this step is to introduce false image boundaries in the composite image and to blur the true boundaries in hopes of making the image region detection intractable by machines. The resulting composite image is used in the click test. It is claimed that IMAGINATION is resistant to attacks and friendly to humans [13].

The IRCs discussed above, together with Cortcha (our novel IRC to be presented in Section 6) are compared in Table 1 against the CAPTCHA requirements listed in [12] as well as scalability and if manual work is needed for the images to be used to generate challenges. The accuracy rate and solving time in the column "Easy to human" of the table, if presented, are from the original paper that proposes the CAPTCHA. The success rate of no-effort attacks (random guess), if presented, is either from the original paper or calculated previously in this section.

Based on Table 1, we determine that IMAGINATION, ARTiFACIAL and Asirra are representative IRCs that worth a close examination.

Table 1. Evaluation of existing IRCs and Cortcha

IRC	Manual work	Easy to grade	Easy to human	Hard to machine	Univer-sality	Resistance to no-effort attacks	Scalability	Secret database
Bongo	*No*	*Yes*	*Yes*	*No*	*Yes*	*No (50%)*	*No*	*No*
Pix	*Labeling*	*No*	*Subjective*	*Yes*	*No*	*Yes*	*No*	*Yes*
Animal Pix	*Labeling*	*Yes*	*Yes*	*No*	*No[2]*	*No (8.3%)*	*No*	*Yes*
CT_L	*No*	*No*	*Accuracy:76.5% Time: 24s*	*Yes*	*No*	*Yes*	*No*	*Yes*
CT_S	*No*	*Yes*	*Yes*	*No*	*Yes*	*No (50%)*	*No*	*Yes*
CT_A	*No*	*Yes*	*Accuracy: 91% Time: 51s*	*No*	*Yes*	*No (16.6%)*	*No*	*Yes*
ARTiFACIAL	*No*	*Yes*	*Accuracy:99.7%Time: 14s*	*Yes*	*Yes*	*Yes (3.5E-17)*	*Yes*	*No*
IMAGINA-TION	*Click: No Annotation: Labeling*	*Yes*	*Accuracy: 85%*	*Yes*	*Yes*	*Yes (6.2E-7)*	*Click: Yes Annotation: No*	*Yes*
HotCaptcha	*Labeling*	*No*	*Very subjective*	*No*	*No*	*No (1.19%)*	*No*	*Yes*
Asirra	*Labeling*	*Yes*	*Accuracy:83.4%Time: 15s*	*Broken*	*Yes*	*Yes (0.024%)*	*No*	*Yes*
Orientation (with 3 images)	*Removing bad images*	*Yes*	*Accuracy: 84%*	*Yes*	*Yes*	*Yes (0.009%)*	*No*	*Yes*
Video	*Labeling*	*No*	*Accuracy: 90% Time: 22s*	*No*	*No*	*No (>2%)*	*No*	*Yes*
Cortcha	*No*	*Yes*	*Accuracy:86.2% Time: 18.3s*	*Yes*	*Yes*	*Yes (≤0.125%)*	*Yes*	*Yes*

---

[2] Some animals are popular only in a few countries [12].

3. OUR ATTACK ON IMAGINATION

3.1 Basic Ideas on Our Attack

The dithering process during the generation of a click challenge in IMAGINATION produces many false boundaries. To be a good CAPTCHA, some true boundaries should be still readily visible so that humans can easily determine at least one constituent image's location. Let's look how humans would deduce such a location. A candidate region is first located. Then the two sides along the boundary of the candidate are compared. If both sides are very similar, the boundary is likely false, and another candidate should be examined. This process is iteratively applied until a confident image location is found. This image location should agree with the likely locations of the neighboring constituent images. This process is also applied in our attack on the click test of IMAGINATION.

Our attack consists of the following three steps:

- Detect all possible rectangular regions. Each rectangular region represents a candidate image location. These rectangular regions can be ranked according to the likelihood of being a rectangular region.

- Compare objects and textures on both sides along the boundary of each candidate rectangle. An object that crosses a boundary is called a traversing object of the boundary. A boundary with traversing objects is likely a false image boundary. A boundary with very different textures on both sides is likely to be a true image boundary. Any rectangular region with a false boundary is removed from the set of candidates. The likelihood to be a true image location is then adjusted for each survived rectangle.

- Check each candidate's consistency with its neighboring rectangles. The rectangle with the highest likelihood is selected and its geometric center is sent back as the response to the test.

These steps will be described in detail in the following subsections.

3.2 Details of Our Attack

3.2.1 Detection of Candidate Rectangles

To detect all the possible rectangular regions in a composite image, color edge detection is first applied, and vertical and horizontal line segments are then detected. By enumerating possible combinations of these line segments, candidate rectangular regions are generated.

3.2.1.1 Color Edge and Line Segment Detection

Region-based color edge detection is used to detect significant vertical and horizontal color edges in a composite image. This is because a dithered composite image is quite noisy that a local gradient based method would lead to a lot of false edge responses. Before the edge detection, an input composite image is smoothed by a 5×5 Gaussian filter in order to reduce noise. For each location in the image, we draw a circle of radius R and divide it along the diameter at four different directions: $0°$, $45°$, $90°$, and $135°$. The radius R should be selected carefully. A value that is too large would result in imprecise edge localization. A value that is too small would generate many noisy fragments. The color model in each semi-circle is represented as the histogram in a jointly partitioned region by the color components in the Lab color space. The color edge intensity in each different direction is estimated by

calculating the χ^2 distance between the models of the two resulting disc halves:

$$\chi^2(h_1, h_2) = \frac{1}{2} \sum_{n=1}^{\#bins} \frac{(h_{1n} - h_{2n})^2}{h_{1n} + h_{2n}}, \quad (1)$$

where h_1 and h_2 are the color histograms of the two disc halves. The direction with the maximum color edge intensity is considered as the edge direction, and the maximum value as the edge intensity at the current location. The resulting edge candidate map I_c is shown in Figure 3(b) along with the challenge image shown in Figure 3(a). Non-maxima suppression is then applied to I_c to generate a total edge map I_{edge}, shown in Figure 3(c). A binary edge image I_{bin-vh} is obtained by removing all the non-vertical/horizontal edge points after applying a threshold.
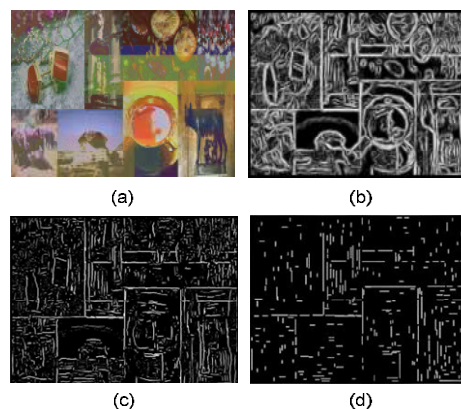

(a)   (b)

(c)   (d)

Figure 3. (a) Original challenge image. (b) Edge candidate map I_c. (c) Total edge map I_{edge}. (d) Horizontal and vertical line segments after the line segment detection is applied.

The horizontal and vertical edge points found above may be slightly off the actual positions by up to a couple of pixels, which makes a boundary not a straight line. We can refine the accuracy of these edge points by applying local gradient based edge detection around the found edge points and adjust the edge point positions if necessary. The two-step procedure described in Appendix 9.1 is applied to the binary image I_{bin-vh} to detect all the potential vertical and horizontal line segments. The resulting vertical and horizontals line segments are shown in Figure 3(d).

3.2.1.2 Generating Candidate Rectangles

Candidate rectangles are generated by enumerate all the possible rectangles from the horizontal and vertical line segments obtained from the last step. A priori knowledge is then applied to remove unlikely image rectangles: a rectangle that is too small or too large is removed. A very small rectangle is unlikely used to fill with an image since it is too hard for humans to recognize. A very large rectangle makes other images too small. The rectangles that are too close to the boundary of the composite image are also removed for the same reason.

In the next step, the candidate rectangles are processed and ranked according to the edge intensity, traversing objects, and edge density variation cues. The detail is described in Appendix 9.2.

3.2.2 Consistency Inference

The a priori knowledge that constituent images cover the whole composite image and that there is no overlapping between any two constituent images is used to check consistency of the survived

rectangles in order to select one as the response to the click test. Two rectangles are said to be neighbors if one contains at least one pixel in the neighborhood of some pixel(s) in the other rectangle. Two neighboring rectangles are said to agree with each other if they share at least one boundary or one boundary of a rectangle is on the extension of a boundary of the other rectangle. Two distinct rectangles are said to be inconsistent with each other if they overlap each other or they are close enough to each other such that the gap between them is too small to hold a constituent image.

The following steps are applied to determine a rectangle with its geometric center as the response to the click test:

1) All the rectangles with a confidence value of 1, if any, are selected. Each selected rectangle is then checked against all the other rectangles in the set of candidates. If any inconsistency is detected, the rectangle is dropped from the selected rectangles. If there is any selected rectangle that survives the inconsistency checking, the one with the largest number of agreed neighboring rectangles is located and its geometric center is returned. Then the attack ends.

2) Each rectangle in the set of candidates is checked against the other rectangles in the set of candidates. If no inconsistency is detected, the rectangle is selected. At the end of this process, if there is any rectangle selected, the one with the largest number of agreed neighboring rectangles and, if there are still multiple choices, with the highest confidence value is located and its geometric center is returned. Then the attack ends.

3) If all the candidate rectangles are inconsistent with some candidate rectangle, the rectangle with the highest confidence is located and its geometric center is returned.
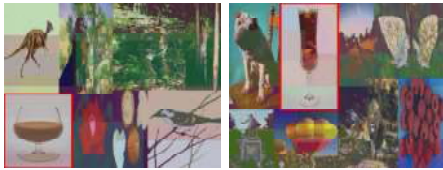

Figure 4. Two challenge images and the image regions (enclosed by red lines) returned by our attack.

3.3 Attacking Results

We have used both our own implementation of IMAGINATION's click test and IMAGINATION's online service [14] to evaluate our attack algorithm. Using our own implementation, the evaluation process was automated and fast, but our implementation might be different from the actual IMAGINATION. Therefore we collected 109 click test images from IMAGINATION's online service, and attacked them to compare with the result from our own implementation. Unlike our own implementation, the evaluation using IMAGINATION's online service could not be automated since the service applied an annotation test after a click test, denied access if failing either test more than a small threshold number. Each collected image was first manually labeled to locate perceivable image boundaries. The output of our attack algorithm was then compared with the labeled result to determine if the attack was successful or not. For the 109 collected click test images, our attack solved 81 test images correctly, resulting in a success rate of 74.31%. Figure 4 shows the image regions (enclosed by the red lines) returned by our attack algorithm for two collected challenge images. This success rate agrees with the result evaluated with our own implementation of IMAGINATION's click test where 2000 click test images were used. Our average attack speed was 0.962s

per image when running on a PC with 3.2GHz Intel P4 and 2GB memory.

If we use a random guess for the annotate test of IMAGINATION, the success rate will be 1 in 15, or 6.67%. By combining the attack results of IMAGINATION's two tests, our attack algorithm can achieve an overall success rate of 74.31% × 6.67%, or 4.95%. This number can be increased if a technique better than a random guess is used for the annotate test. In fact, this annotate test has several shortcomings, e.g., difficult to build an image database large enough to meet the demand of a large scale application, language-dependent, and poor rejection of a random guess.

4. OTHER ATTACKS
4.1 Low-level Features and Semantics
A typical image contains rich information which can be classified roughly into two types: low-level features and high-level semantics. Low-level features are the information that can be extracted from an image with little or nothing to do with perception or understanding of the image. Many low-level features have been developed for various tasks. Commonly used low-level features include color, shape, texture, color layout, among others. *Color* is a widely used feature. A color feature can be represented by the color histogram which is a distribution of the colors in an image. *Texture* refers to repeated patterns with varying intensities or colors such as grassland. Contrast is a simple representation of texture. *Shape* represents a visual object, represented by the outer boundary of the shape or the entire region of the shape. *Color layout* includes both the color feature and the spatial relations. More low-level features can be found in [23]. Computers are typically good at extracting low-level features from an image.

High-level semantics, on the other hand, is associated with perception or interpretation of an image such as identifying semantically meaningful objects contained in an image, and relationships of these objects. Low-level features are typically deterministic, i.e., the same or similar result is produced when a low-level feature is extracted from the same image by any computer or most humans at any time. High-level semantics, on the other hand, may be subjective and user-dependent, especially when interpretation is applied during extraction. Different semantic meanings may be generated when the same image is perceived by different people or by the same person at different times. There is still a large gap between low-level features and high-level semantics. Image understanding or general object recognition aims to reduce such a gap but still remains a hard AI problem in computer vision.

4.2 Attack on Asirra
Golle [24] designed a machine learning attack on Asirra. In this attack, an image is partitioned and divided into uniform blocks. The discriminative features used in the attack are the block's color patterns and 5x5 texture tiles. Machine learning on the labeled training data produces a classifier that has achieved a success rate of 82.7% in distinguishing a cat from a dog used by Asirra, much higher than a random guess does. For a 12-image Asirra, the success rate is 10.3%. However, no insightful explanation was given on why a seemingly hard object recognition problem can be readily solved by a machine learning attack.

4.3 Attack on ARTiFACIAL
We have developed a machine learning attack on ARTiFACIAL [25]. There are two stages in the attack: detect the face in a

challenge and then locate the six facial corner points on the face. Based on the observation that the intensity perturbation introduced by ARTiFACIAL could be largely removed in the gradient domain which represents spatial variations of the image's intensities, we have designed a gradient-domain based face detector that learns the structural shapes of facial components to detect the face in a challenge image. Figure 5(a) shows the gradient domain representation of the challenge image shown in Figure 1(c). Then the intensity perturbation which manifests as horizontal and vertical lines in the gradient domain is neutralized (Figure 5(b)), thanks to their very different patterns from the gradient of a human face.
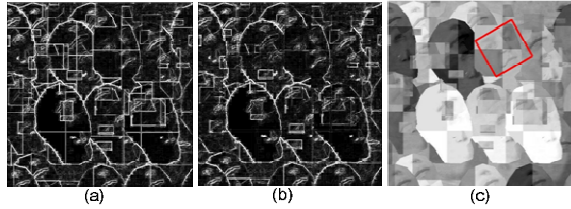

(a)   (b)   (c)

Figure 5. (a) Gradient image of the challenge image shown in Figure 1(c). (b) After line filtering. (c) Face detection result.

The discriminative facial structural features obtained by machine learning are applied to detect location and orientation of the face. Figure 6 shows the top 5 features produced by the machine learning process. They represent structure features on the eyes and nose of a face. When tested on 800 challenge images, the face detection rate was 42.0%. The red tilted rectangle in Figure 5(c) shows the detected face for the challenge shown in Figure 1(c). After the face detection, a facial component-based discriminative algorithm and a refinement algorithm is then applied to the detected face to locate the six corner points. The success rate to correctly identify the six corner points on a face detected at the first stage was 42.9%. The overall success rate to pass an ARTiFACIAL test is therefore $42.0\% \times 42.9\%$, or 18.0%. The average time to produce a response was 1.47s when tested with the 800 challenges on a PC with 3.2GHz Intel P4 and 2GB memory.
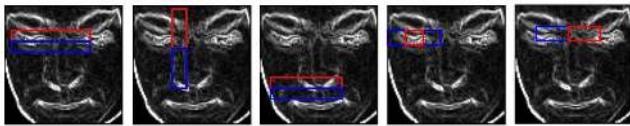

Figure 6. First 5 features produced by the learning procedure.

5. A SIMPLE FRAMEWORK

In this section, we propose a simple framework for understanding the design of a good IRC. We first examine the design flaws in the three reprehensive IRCs that led to successful attacks. We then propose three guidelines in designing robust IRCs.

5.1 Lessons from Successful Attacks

The task in the click test of IMAGINATION [13] is to distinguish authentic image boundaries from false image boundaries such that at least the boundary of one constituent image is identified. Humans decide that a boundary is likely false if the two sides of the boundary are correlated since two randomly selected images are unlikely correlated. This eliminating process is iteratively applied until an image's boundary is confidently identified. This iterative process can be readily performed by machines through computing low-level features of the image. That is, a cognitive decision about whether both sides of a boundary are correlated or not can be approximated by detecting similarity of textures and continuity of

traversing objects. No image recognition or semantics is necessarily used. This explains why the click test of IMAGINATION fails.

- **Lesson #1**. *An IRC that does not rely on image semantics is doomed to be vulnerable to automated machine attacks.* For such an IRC, the human's natural cognitive "algorithm" for passing the CAPTCHA test can be imitated or approximated by machines automatically computing certain low-level discriminative features – such a task can be readily performed by a computer, and sometimes done even more accurately by a computer than by humans. Instead, image recognition task must be introduced in an IRC.

The task to solve an Asirra challenge is a binary classification problem, as the image is of either a dog or a cat. It is still an open problem how human beings exactly carry out such classification tasks, but it is believed that cognitive capability of image recognition is needed to perform the tasks. Although computers do not have such a capability, Asirra was broken by computers for a simple reason: often it is sufficient to compute low-level features to achieve binary classification.

A common method for binary classification is to identify a set of discriminative low-level features and use machine learning on empirical training data to automatically learn both the common patterns among the individual objects of each type, and the most effective discriminative patterns to distinguish objects of one type from objects of the other type. The training data contain positive samples which are objects of the type that the classifier wants to identify (cats in Asirra's case) and also negative samples which are objects of the other type. Golle's attack [24] used two simple yet highly discriminative low-level features: texture and color patterns. Texture patterns are highly discriminative in this case. For example, cats usually have patterns of furs and whiskers, which are very different from those of dogs. Although not used in Golle's attack, shapes would be another highly discriminative low-level feature since cats and dogs typically have very distinctive ear shapes. In fact, a recent study [26] on cat detection using machine learning on texture and shape patterns claimed a success rate of above 90% for cat detection. Note that cat detection is harder than identifying cats from a collection of cats and dogs since detection faces much more variations in the background or cluttering objects.

Although it is unclear how much semantic discrimination humans use in solving Asirra's binary classification problem, we have experienced that in many cases, substantial low-level features in the images aided humans to distinguish cat images from dog images. On the other hand, rather than an IQ test, a CAPTCHA is designed to let most humans easily pass in a short time. To be user friendly, an IRC based on a binary classification problem needs to use images of objects that can be easily and unambiguously identified by most humans. This implies that the objects of each type share easily perceived common features, some of which are highly discriminative that humans can readily apply to distinguish between the two types. Some of these perceivable discriminative features should be associated with or reflect in low-level features; otherwise it is unlikely for most humans to produce the same result for a challenge, as required by the usability of an IRC, since high-level semantics tends to be subjective and use-dependent. Low-level features, on the other hand, are typically deterministic, and can lead to a consistent and unambiguous result by most humans. By identifying and selecting a set of highly discriminative low-level features, adversaries can apply machine learning on empirical data to find effective decision criteria to distinguish between the two

types, resulting in an automated effective attack. Therefore, we have the following lesson.

- **Lesson #2**. *A user-friendly IRC based on a binary classification problem to classify an object into one of two fixed types a priori of concrete objects[3] is highly likely vulnerable to machine learning attacks.*

ARTiFACIAL [12] relies on face detection. In general, an object detection problem is harder than a binary classification problem since the negative samples for the latter case have much less variations, making the classification problem easier. In our attack on ARTiFACIAL, the spatial patterns of facial features are learned by machines as the discriminative features to identify the face in a challenge image with cluttering background. The a priori information that there exists one and only one face in a challenge image is also exploited to improve the detection rate. As such, an IRC based on object detection does not appear to be able to provide a significantly more secure design than a binary classification based IRC.

- **Lesson #3**. *A user-friendly IRC based on detecting a concrete object of a fixed type a priori is very likely to be vulnerable to machine learning attacks.*

5.2 Guidelines for Designing Robust IRCs

From the lessons we learned in Section 5.1, we would summarize three guidelines to shed light on the design of a robust IRC.

- **Guideline #1**. *Rely on unambiguous high-level semantics.*

This guideline is directly derived from the first lesson. One problem associated with using semantics is that the retrieved semantic information from an image tends to be subjective and user-dependent. A good example is image labeling, where different people may give the same image different labels. This intrinsic ambiguity in semantics makes it difficult to generate CAPTCHA challenges using image semantics.

However, there are still some ways to use semantics without ambiguity in the answer and thus they can likely be used in IRC design. An example is spatial relationships of objects. Humans can understand relative sizes, shape changes and occlusion relationships of different objects, and can deduce spatial relationships of different objects. This cognitive process involves recognition of objects, and thus is still hard for computers to perform. Another example is logical relationships of objects. Like spatial relationships of objects, deducing logical relationships of objects also involves recognition of objects, a hard AI problem.

- **Guideline #2**. *Boost robustness with more variations such as relying on recognition of multi-type objects for either detection or classification.*

Based on the second and third lessons, the task of either binary classification of objects from two types a priori or detecting an object of one type a priori may leave sufficient low-level features that can be efficiently exploited by machine learning methods, resulting in a seemingly image recognition problem solved by computers. We can make an IRC more secure by making machine

learning attacks much harder than before. This can be achieved, for example, by increasing the number of concrete object types used in an IRC. Multi-label classification is much harder for computers than binary classification.

It appears that Guideline #2 is a general principle which is applicable to all CAPTCHAs including text schemes. For example, by increasing variations of segmentation-resistant mechanisms, text distortion methods and fonts, and then randomly selecting one or more fonts, a segmentation-resistance mechanism and a text distortion method in generating the current challenge, we can have a text CAPTCHA that is more robust than the state-of-the-art, making an attacker's life much harder.

- **Guideline #3**. *Disable machine learning by eliminating the possibility of using empirical data or a priori knowledge such as the types of objects. This means that a current challenge is independent of the past challenges in terms of computable features such as low-level image features.*

An intrinsic feature for all machine learning attacks is that they typically rely on empirical data to learn effective discriminative features and decision criteria before becoming effective. The most fundamental solution to deal with these attacks is therefore to disable machine learning by making the past challenges uncorrelated with the current or future challenges. In this way, the discriminative features or decision criteria learned from the past challenges would be ineffective to solve the current or a future challenge. This can be achieved by randomly selecting a type and an object of the type to generate a challenge, with both the number of types and the number of individual objects of each type being sufficiently large, infinite ideally, so that it is intractable for the current computing capability. This is the ultimate goal, although hard to achieve.

In principle, Guideline #3 is applicable to all CAPTCHAs. For text CAPTCHAs, however, it is still an open problem whether we can create an unlimited number of segmentation-resistant mechanisms, or an unlimited number of combinations of segmentation-resistant mechanisms and text-distortion methods.

6. A NOVEL IMAGE BASED CAPTCHA

A novel IRC, Cortcha (Context-based Object Recognition to Tell Computers and Humans Apart), is presented in this section.

Intuitive ideas. Based on the third guideline, an IRC should use an unlimited number of different types of objects. This can be easily achieved by crawling images in the Internet. But how do we generate challenges? In general, computers do not understand the semantics of an image, but we want to avoid labeling images in generating challenges. Computers do a poor job in segmenting an image into semantically meaningful objects. Objects that are automatically segmented by computers were considered not suitable for an IRC. Instead, semantically meaningful objects were considered necessary in generating IRC challenges in order for human users to produce consistent answers. This explains why many existing IRCs require manual labor such as labeling or selecting images. It is a dilemma seemingly hard to solve.

Our insight is that although an object that is segmented by a computer might be poor cognitively, but if the object is surrounded by its original context in the image, then the object is readily recognizable by humans. By exploiting the context, objects segmented by computer can be used in an IRC. The magic of

---

[3] A concrete object means an object that is unambiguously and easily defined and identified. IRCs usually rely on concrete objects, since for these objects human users can generate consistent answers that can be easily verified by a machine.

context solves the dilemma, and an IRC can be designed without labeling any image.

To use context for humans to recognize a computer-segmented object, we can crop the object and detach it from its original image, and then ask a user to use the image as a context cue to identify the detached object from a set of decoy objects. The hole left by cropping in the original image must be filled. Otherwise the detached object can be easily deduced by comparing the contour of a candidate object with the shape of the hole. The filling should not allow bots to locate the cropped region but should leave some semantic hints such as unnaturalness to enable humans to quickly locate the region. Image inpainting [27], which is to incrementally fill a hole with the best matching blocks, can be modified to achieve the goal.

These intuitive ideas led to the development of Cortcha. In Cortcha, a user is asked to identify, among a set of candidate objects, an object detached from an image, and then place it back to its original position in the image.

Advantages. The major innovation in Cortcha is to exploit the surrounding context to recognize an inaccurately segmented and thus often semantically meaningless object. Compared with existing IRCs, Cortcha has the following advantages:

- *No need to manually label any image.* Context-based object recognition makes it possible to use semantically meaningless objects in our design. Therefore object segmentation can be done by computers and the whole challenge generation process can be fully automated.
- *An unlimited number of types of objects can be used in Cortcha.* This can effectively disable the learning process in machine learning attacks.
- *Cortcha is scalable.* In Cortcha, the tasks of source image collection and challenge generation can both be automated. By crawling the Internet, a large number of images can be quickly and continuously added to Cortcha's image database. Cortcha can therefore meet the demanding requirement of a large scale application such as Hotmail.

6.1 Detailed Description

Cortcha consists of the following stages: collecting images into the database, generating challenges, displaying challenges, and grading responses.

6.1.1 Image Database

Cortcha relies on a secret database of images. The huge number of images in the Internet combined with the fact that the content-based image retrieval at the Internet scale is still in its infancy makes it feasible to convert these public images into Cortcha's secret database. It is highly unlikely for adversaries to find out the original Internet image used in a Cortcha challenge before the current CAPTCHA session expires.

Not all images are suitable for Cortcha. We discard small-sized images as well as noisy ones which have a large ratio of high frequency energy to low frequency energy. We also discard monotonic images, whose absolute gradient values are on average small or whose histogram entropy is small. However, instead of discarding large images, we crop them into suitable sizes.

6.1.2 Image Segmentation and Object Selection

An image is first processed to identify its salient objects. We apply the JSEG [28] method to segment the image into objects. The boundary of each object is refined to align with the gradient edges. Small-sized objects are merged with their best matched neighbors. We then assign each object a perceptual significance value, which is calculated with the saliency detection scheme proposed in [29].

Not all the resulting objects are suitable as the object to be cropped from the image for generating a challenge. We have observed improved usability when both the object to be cropped and its surrounding context in the image are semantically meaningful. Humans can correlate the semantics of an object with that of the surrounding context in solving a challenge. Such semantic correlation cannot be exploited by computers. This observation leads us to discard objects with a small significance value and those that are thin, monotonic, or overly large-sized. An object's thickness is measured by the maximum value of its distance transform. A thin or monotonic object tends to carry little semantic information. A large object tends to make its surrounding context of little semantic meaning.

We also discard objects which share local or global similarity with the remaining image. Such similarity may be exploited by attackers to deduce a correct answer to a challenge. Local similarity is calculated by comparing the local color histogram and texture on both sides of the object's boundary. Global similarity is calculated by using SIFT [30] to extract scale-invariant local features. The features from the objects are compared with those from the remaining image to detect any similar object in the remaining image.

If there exists any survived object at the end of the above process, the image, along with the survived objects, is inserted into Cortcha's database for future use. With all the measures adopted above, it is still possible that a survived object is segmented incorrectly in terms of cognition, carrying little semantic information. Cortcha allows such a case since the context in the inpainted image can be leveraged by humans in recognizing the object. This is a key advantage over other IRCs such as Pix [21], Chew and Tygar [11], Asirra [15], and the orientation CPATCHA [16], which all require semantically meaningful objects, and thus have to involve human labors to label or select suitable images. On the contrary, the image segmentation and object selection process in Cortcha can be fully automated.

6.1.3 Image Inpainting

To generate a challenge, an image is randomly selected from the database. Then, an object is randomly selected from the objects stored along with the image. The image and its objects are then all deleted from the database. A buffer region of n-pixels surrounding the object is created in the image. The object and the buffer region are then cropped from the image. The buffer region is used to remove possible traces of local similarity between the object and the remaining image. The cropped region is then filled with an inpainting algorithm modified from [27], as described next.

We first locate a region surrounding the cropped region and calculate its color histogram. The database is then searched to find an image that matches the color histogram best. The found image is used as the primary source while the remaining image as the secondary source in inpainting. The hole to be filled is divided into blocks. These blocks are filled sequentially. In filling a block, a matching block from the source is needed. The primary source is searched first for a matching block. If no matching block is found, the secondary source is searched to find a set of matching blocks. We then randomly select a block from the set as the source block. It

is possible to use several images from the database as the primary source for the inpainting process.

Two scheduling schemes are used to determine which block to be filled next. The first scheme places a block with a large curvature and a shorter distance to the cropped boundary at a higher priority. The second scheme treats a block with structured neighboring blocks at higher priority. The first schedule is applied to boundary blocks of the cropped region while the second schedule is applied to internal blocks. For the blocks lying between the two types of blocks, either schedule can be applied, depending on a random selection process. The first schedule ensures smooth transition at the boundary while the second schedule maintains structured filling. Sharp changes at the boundary might be correlated with the contour of the detached object. Lack of structures may indicate a filling region. They both can help an attacker solve a challenge.

If any part of the detached object's boundary lies in the boundary of the image, that part may indicate the block's location in the image. This leaking information exploitable by attackers can be effectively removed by applying outpainting, an inpainting process for the reversal direction, to grow the object beyond its boundary at the part that lies in the image's boundary. As described later in Section 6.2, the extra pixels "filled" by the outpainting process are invisible when the detached object is aligned correctly with the inpainted image in solving a Cortcha challenge.

6.1.4 *Generating a Challenge*
The detached object, outpainted if necessary, is used to search the database to find the best matched *L-1* objects in the image database. These objects are from distinct images. The search is based on the color histogram and the complexity, measured as the averaged absolute values of the object's gradient. More advanced technologies such as SIFT [30] can also be applied. These *L-1* objects are optionally further processed by warping, rotating, or, if lacking of structures, randomly embedding with a similarly colored visual object. The resulting objects are used as decoy objects. These optional distortions make a decoy object look odd to humans so that the authentic object can be easily identified by human eyes. If the detached object can be detected by a computer, e.g., a human or a cat face, similar computer-detectable objects are retrieved from the database as decoy objects. None of the optional distortions is applied in this case. The reason to treat generic objects and computer-detectable objects differently is to prevent attacks from "recognizing" an object or detecting the distortion applied to decoy objects to solve a challenge.

The detached object and its *L-1* decoys form *L* candidate objects. They, as well as the inpainted image, are scaled in size by a factor that is empirically chosen. We use the bicubic interpolation for image scaling. A random noise is then added to the scaled image and candidate objects. The purpose of both image scaling and random noise is to remove any quantization or other patterns in an image that can be exploited to deduce the inpainted region or the detached object.

6.2 Solving Cortcha Challenges
A Cortcha challenge displays an inpainted image along with *L* candidate objects. Figure 7 shows an actual challenge from our current implementation, in which eight candidate objects were used. A user selects a candidate object, and drags it to move around or drop to a position of the inpainted image. Wherever the object is on top of the inpainted image, surrounding the object, a buffer region (of the same width as we use in the challenge generation process) is

created. The buffer region is cropped and then filled by an image smoothing method. Effectively, a composite image is created by combining the inpainted image and the candidate object on the spot. The resulting composite image is presented to the user, but only the pixels within the inpainted image's boundary are visible. As a result, when the detached object is corrected aligned with the inpainted image, the outpainted portion, if exists, is invisible. At each trial location, if the composite image looks natural and semantically meaningful, the detached object and its due position are found. The challenge is thus solved. Figure 8 shows the result when the detached object is correctly placed back.
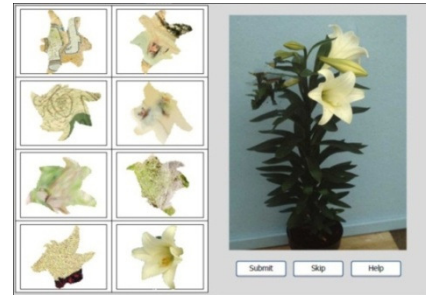

Figure 7: A Cortcha challenge with 8 candidate objects on the left and the inpainted image on the right.

Figure 8. A successfully solved Cortcha challenge: the right panel shows the composite image when the detached object was placed at the correct position.

6.3 Usability

6.3.1 *Experimental Settings*
Our database included 30,000 images that we crawled from the Internet. The longest side of each image was 500 pixels. 350 Cortcha challenges were then generated from this database. All the thresholds needed in generating these challenges were empirically determined from a small set of representative samples, and then applied to all the images. The average time for generating a challenge (following the whole process as described in Section 6.1) was 122s on a PC with 3.2GHz Intel P4 and 2GB memory. Note that many steps in this process can be performed offline. Therefore, offline preprocessing will significantly reduce the average time required for online generation of challenges.

A website was used in our usability study. A participant browsed the website to start a test. Each test consisted of 20 randomly selected Cortcha challenges presented sequentially. The responses and solving times were recorded by our web server. After the test, each participant was asked to answer a questionnaire as an exit survey. We invited a group of interns who had never exposed to Cortcha to participate in our study. Most of them were graduate students in their twenties. 84 volunteers participated and completed the study. In our experiment, the position tolerance for a response

was set to be 10.0% of the inpainted image's height and width. The buffer region was set to have a width of *n=3* pixels.

6.3.2 Results

We examine Cortcha based on the following three components, partially quoted from [31]:

1. **Learnability**. Figure 9 shows the average solving time for each of the 20 sequentially presented challenges. Note that different participants received different challenges even for the same index of challenge since each challenge in the usability experiments was randomly selected from the 350 generated challenges. We can see from the figure that the participants improved the solving time significantly as their experiences built up. They spent 25.6s on average to solve the first challenge, and the solving time dropped quickly to 20s or below after the first two challenges. This indicates that Cortcha is fairly easy and quick to learn.
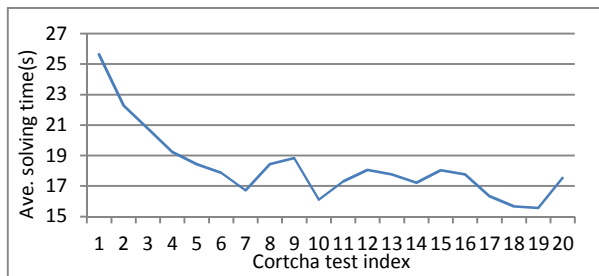

Figure 9. Average solving time vs. the challenge index.

2. **Errors and efficiency**. The overall accuracy rate was 86.2%, and the average solving time among all the 84 participants and 20 challenge indexes was 18.3s. As compared in Table 1, Cortcha has a human accuracy rate that is slightly higher than Asirra, Orientation and IMAGINATION, significantly higher than CT_L, but lower than ARTiFACIAL, Video and CT_A. On average, Cortcha takes slightly more time to solve than Asirra and ARTiFACIAL, but less than other IRCs with a solving time reported.

We have also conducted a usability study of Google's text CAPTCHA with the same group of people. The text challenges from Google's Gmail site were used in the experiments. Each participant was asked to solve 20 text challenges sequentially. 105 volunteers participated and completed the test. The average accuracy rate was 82.8% and the average solving time was 7.9s. Compared with Google's text CAPTCHA, Cortcha has a slightly higher human accuracy rate but takes more than double the time to solve a challenge.

It is worth mentioning that in the Cortcha's usability study, more than half (43 out of 84) of the participants had an accuracy rate higher than 90%, and the participants with a higher accuracy rate (>80%) spent a significantly more time (3s more on average) than those who performed poorly (with an accuracy rate lower than 80%). Since the challenges were randomly selected in the experiment, this may indicate that the participants with a lower accuracy rate might not have paid sufficient attention.

We observed that 51% of the tested challenges were solved in less than 15 seconds, but we also got a long tail for the distribution of the solving time, up to 40 seconds or even more. In Cortcha, some challenges are easier to solve than others. For example, if a detached object is semantically meaningful and

the unnaturalness of the inpainted region is apparent to human eyes, this challenge will be easy to solve. Otherwise it can take a longer time to solve, or lead to more erroneous responses.

3. **Satisfaction**. We collected 72 valid responses in the exit survey. About 78% ($\approx$56/72) of the responses indicated a difficulty level of either medium/acceptable or easy for them to learn how to solve Cortcha. About 1/3 of the responses preferred Cortcha to traditional text-based CAPTCHAs. Many of them liked Cortcha because it was interesting, secure, and keyboard-free. Complaints were mainly on the difficulty of some Cortcha challenges. Specifically, for some challenges, it was hard to either determine where the inpainted region was or differentiate between candidate objects with similar color patterns.

6.3.3 Discussions

After examining each challenge and its responses, we concluded that most of the "difficult" images had the following common characteristics: the detached object lay inside a repeated texture. An almost seamless inpainting result was produced: although different from the original image, the inpainted image looked natural. There was no unnaturalness in the inpainted region that the participants could leverage to locate the original position of the detached object. The surrounding context was mainly textures, almost irrelevant semantically to the detached object. This inspires us to add a posteriori check: if the inpainted region has a similar pattern as its neighborhood, the challenge might be hard to solve and should not be used. We will implement this posteriori check to further improve Cortcha's usability.

Another inspiring finding is: easy challenges were dominant in our randomly generated challenges. 294 out of the 350 challenges have been tested more than three times. 173 out of the 294 challenges (58.8%) were found to be very easy to solve, and the participants did not make any mistake on them. Less than 10% (24) of the 294 challenges were found difficult. Their accuracy rates were below 50%. This observation inspires us to explore a heuristic approach towards image filtering to remove difficult images. This approach can improve the human accuracy rate and reduce the solving time.

6.4 Robustness

6.4.1 Random Guess Attacks

When 8 candidate objects are used in a Cortcha challenge, a random selection has a chance of 1/8 or 12.5% to be correct. A random search for the correct alignment position results in a probability of 10%*10% or 1.0% to be correct. Therefore a random guess attack could have a success rate of 0.125%, which is however much smaller than the design criterion (0.6% success for bots, see Section 2.2).

A random guess attack can be made even harder by using a larger number of candidate objects. For example when *L*=12, the success rate of a random guess attack will be 0.083%. This rate can be further reduced. For example, when a larger inpainted image is used, the tolerance of alignment position variations can be tightened from the current 10% to, say 5%, reducing the success of a random guess attack to 0.021%.

6.4.2 Other Possible Attacks

Using machine learning to classify or recognize objects was an effective attack on Asirra and ARTiFACIAL, but it will not work on Cortcha since the objects used in a current challenge are uncorrelated with those used in other challenges. However, object segmentation and image inpainting used in generating Cortcha

challenges may leave traces or characteristics, which can be exploited to tell the detached object or its position in the image. Possible attacks that we can think of are as follows.

An image may show a specific noise or quantization pattern such as an imaging sensor noise pattern or a JPEG quantization pattern. These patterns have been used in image forensics to detect image forgery and identify image source [32]. A correlation of such a pattern in the detached object and in the remaining image may help identify the detached object. The inpainted region may show a distinct pattern that helps locate the inpainted region. However, these patterns can be destroyed by scaling the image's size with a nonlinear bicubic interpolation and by adding noise to the image. In Cortcha, both inpainted images and candidate objects are scaled in size using a bicubic interpolation, and then random noise is added, as described in Section 6.1.4. Therefore noise or quantization patterns cannot be exploited to attack Cortcha. More sophisticated measures such as those in [33] can also be adopted by Cortcha to thwart attacks based on other image forensic technologies.

There might exist some local similarity between an object and its surrounding region due to light reflection, shadowing, inaccurate segmentation, or other issues. Such local similarity, calculated with computable features such as color or texture patterns, can be exploited to solve a challenge. This issue is addressed in Cortcha by aligning an object's boundary with its gradient edges, cropping an object which contains little or no local similarities (currently based on color and texture) with its surrounding region, and using a buffer region to separate a detached object and the remaining image. From our observation on the 350 Cortcha challenges used in our study, a buffer region with a width of $n=3$ pixels can effectively remove color-based local similarity between a detached object and the remaining image. Such a buffer region does not have a significant perceptual impact on the composite image when a human user is solving a challenge.

Low-level features might be used to detect possible correlations between a detached object and the remaining image. Such a correlation would increase the chance to identify the detached object. To address this threat, we make sure that all the candidates have a similar correlation with the inpainted image. For example, readily computable low-level features such as color histogram and complexity are used to find decoy objects. When the image database is sufficiently large, decoy objects are all similar to the detached object in terms of the applied low-level features. In addition, SIFT [30] is used in Cortcha to prevent the existence of a similar object in the remaining image. It is possible that attackers apply other low-level features or better algorithms to calculate correlations or detect similar objects. However, these low-level features or algorithms can also be adopted by Cortcha to thwart this type of attack.

A possible brute force attack is to use the same object segmentation method as used in Cortcha to solve a challenge. In such an attack, each candidate object is tested at all the possible positions. At each trial, the same object segmentation procedure is applied to the composite image. If the segmentation result agrees with the object under test and the trial location, the detached object and its location are deduced. However, this brute force attack will not be effective. At a trial position, the object under test is unlikely to have a smooth transition with the image at its boundary, thus leading to an agreeing segmentation result. Therefore this attack will produce many false alarms, as confirmed by our preliminary experimental results.

Artifacts produced by the image inpainting might be exploited to deduce an inpainted region. Wu et al. proposed an attack to identify an inpainted region [34]. Their attack compares different parts of an image to detect abnormal similarity between destination blocks and their source blocks in the inpainting process, and then deduces the inpainted region. In Cortcha's inpainting, the source blocks are mainly from the primary source, a secret image in the database. When the database is large and contains a large variation of different images, the cropped image is unlikely to contribute many blocks during inpainting. In addition, when filling a block, our method applies a smoothing operation to smooth the pixels from the source block and those in the destination blocks. Such a smoothing operation removes most of the similarity between a destination block and its source block. Therefore the attack proposed in [34] is ineffective on Cortcha.

A lousy inpainting method may leave some artifacts to indicate where inpainting starts. This location information can be correlated with the contour of the detached object, leading to a possible attack. It is also possible that an inpainting method produces a smoothed region that lacks of structures as compared to the remaining image. Therefore, detecting structureless regions may help locate the inpainted region. Our inpainting method applies different schemes to decide which block is filled next. It ensures that an inpainted region is still rich in structures. It also prevents inpainting artifacts from forming a long edge in parallel with the detached object's contour. We have conducted an experiment to find all the edges in an inpainted image that are in parallel with some portions of the detached object's contour. The preliminary result showed that most of the found edges were irrelevant to the detached object's contour. Therefore correlating inpainting artifacts with a candidate object's contour does not lead to locating the inpainted region.

Another possible brute force attack is the following. Each candidate object is tested at all the possible locations. At each trial, the region covered by the candidate is taken as an object and cropped from the image. The same inpainting procedure as that used in generating Cortcha challenges is then applied to fill the cropped region. The inpainted result is then compared with the region covered by the candidate. If they are similar, the candidate is presumed to be cropped from that position. The original inpainting algorithm proposed in [27] would suffer from this attack, since the inpainting procedure is deterministic and all the source blocks used in inpainting are publically available. Our inpainting algorithm selects source blocks from a secret image whenever possible, and introduces randomness in selecting blocks from the public remaining image. These measures prevent the above attack from producing an inpainting result similar to ours. On the other hand, this brutal force attack in general would hardly be practical, since it requires applying the inpainting process, which is slow, many times.

6.5 Legal Issues for Deployment

Copyrights of images used by Cortcha might cause legal issues that prevent Cortcha from being deployed in the real world. This is a common issue for all the IRCs that use crawled images, such as the CAPTCHAs proposed by Chew and Tygar [11] and the orientation CAPTCHA [16] introduced by Google. Cortcha likely faces even more copyright issues since it modifies images to generate challenges, and some of the resulting images may appear unpleasantly. We leave these issues for lawyers to deal with in near term, and for future technology innovations in the longer term.

7. CONCLUSIONS

We have attempted a systematic study of image recognition CAPTCHAs. We provided a thorough review of the state-of-the-art, presented a novel attack on a representative scheme, and analyzed successful attacks on the other representative schemes. Learned from these attacks, we defined for the first time a simple but novel framework for guiding the design of robust image recognition CAPTCHAs. The framework led to our design of Cortcha, a novel CAPTCHA that exploits semantic contexts for image object recognition. Our usability study showed that Cortcha yielded a slightly better human accuracy rate than Google's text CAPTCHA. Cortcha offers the following novel features. Image labeling is entirely avoided. The source image collection and challenge generation are fully automated. An infinite number of object types are used to generate Cortcha challenges. Objects used in the current challenge are independent of the objects used in previous challenges. This independence makes powerful machine learning attacks useless in attacking Cortcha. Being scalable, Cortcha is a stride forward for image recognition CAPTCHAs towards practical applications. Our future work includes improving Cortcha's speed, a large-scale usability study, and a thorough evaluation of Cortcha's robustness.

8. REFERENCES

[1] Ahn, L. von, Blum, M., and Langford, J. 2003. Telling humans and computers apart automatically. Comm. of the ACM. 46 (Aug. 2003), 57-60.

[2] Ahn, L. von, Blum, M., Hopper, N. J., and Langford, J. 2003. CAPTCHA: Using hard AI problems for security. Eurocrypt'2003.

[3] Baird, H. S. and Popat, K. 2002. Human interactive proofs and document image analysis. In Proc. of Document Analysis Systems 2002. 507–518.

[4] Hocevar, S. PWNtcha - Captcha Decoder web site. http://sam.zoy.org/pwntcha/.

[5] Mori, G. and Malik, J. 2003. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In Proc. IEEE Conf. on Computer Vision & Pattern Recognition, 2003.

[6] Moy, G., Jones, N., Harkless, C., and Potter, R. 2004. Distortion estimation techniques in solving visual CAPTCHAs. In IEEE Conf. on Computer Vision & Pattern Recognition, 2004.

[7] Chellapilla, K. and Simard, P. 2004. Using machine learning to break visual human interaction proofs. Neural Information Processing Systems (NIPS'04), MIT Press.

[8] Yan, J. and El Ahmad, A. S. 2007. Breaking Visual CAPTCHAs with naive pattern recognition algorithms. In Proc. Ann. Comp. Security Applications Conf. 2007, 279-291.

[9] Yan, J. and El Ahmad, A. S. 2008. A low-cost attack on a Microsoft CAPTCHA. In ACM CCS'2008, 543-554.

[10] Yan, J. and El Ahmad, A. S. 2008. Usability of CAPTCHAs or usability issue in CAPTCHA design. In Proc. 4th Symposium on Usable Privacy and Security (2008), 44-52.

[11] Chew, M. and Tygar, J. D. 2004. Image Recognition CAPTCHAs. In Proc. 7th Info. Security. LNCS 3225, 268-279.

[12] Rui, Y. and Liu, Z. 2004. ARTiFACIAL: Automated reverse Turing test using FACIAL features. Multimedia Systems 9 (2004), 493–502.

[13] Datta, R., Li, J., and Wang, J. Z. 2005. IMAGINATION: A robust image-based CAPTCHA Generation System. In ACM Multimedia 2005, 331-334.

[14] IMAGINATION demo system. http://goldbach.cse.psu.edu/s/captcha/

[15] Elson, J., Douceur, J. R., Howell, J., and Saul, J. 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In ACM CCS'2007, 366-374.

[16] Gossweiler, R., Kamvar, M., and Baluja, S. 2009. What's up CAPTCHA? a CAPTCHA based on image orientation. In WWW'2009, 841-850.

[17] Kluever, K. A. and Zanibbi, R. 2009. Balancing usability and security in a video CAPTCHA. In Proc. Symp. Usable Privacy and Security, (2009).

[18] Chellapilla, K., Larson, K., Simard, P., and Czerwinski, M. 2005. Computers beat humans at single character recognition in reading-based Human Interaction Proofs. In 2nd Conference on Email and Anti-Spam (CEAS'05), 2005.

[19] Chellapilla, K., Larson, K., Simard, P., and Czerwinski, M. 2005. Building Segmentation Based Human-friendly Human Interaction Proofs. In 2nd Int'l Workshop on Human Interaction Proofs, Springer-Verlag, LNCS 3517, 2005.

[20] Chellapilla, K., Larson, K., Simard, P., and Czerwinski, M. 2005. Designing Human Friendly Human Interaction Proofs (HIPs). In Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI'05). 711-720.

[21] Ahn, L. von. 2005. Human Computation. Ph. D. dissertation, Carnegie Mellon University, CMU-CS-05-193.

[22] http://www.yuniti.com/register.php.

[23] Rui, Y., Huang, T. S., and Chang, S.-F. 1999. Image retrieval: current techniques, promising directions, and open issues. J. of Visual Comm. & Image Representation, 10, (1999), 39-62.

[24] Golle, P. 2008. Machine learning attacks against the Asirra CAPTCHA. In ACM CCS'2008, 535-542.

[25] Zhu, B. B., LI, Q., Liu, J, and Xu, N. Machine learning attacks on ARTiFACIAL. 2010. Submitted for publication.

[26] Zhang, W., Sun, J., and Tang, X. 2008. Cat head detection - how to effectively exploit shape and texture features. In Proc. ECCV 2008, Part IV, LNCS 5305 (2008), 802–816.

[27] Sun, J., Yuan, L., Jia, J., and Shum, H.-Y. 2005. Image completion with structure propagation. In Int. Conf. Computer Graphics and Interactive Techniques, (2005). 861-868.

[28] Deng, Y. and Manjunath, B. S. 2001. Unsupervised segmentation of color-texture regions in images and video. IEEE Trans Pattern Analysis and Machine Intelligence, 23(8) (2001), 800-810.

[29] Liu, T., Sun, J., Zheng, N. N., Tang, X., and Shum, H-Y. 2007. Learning to detect a salient object. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, (2007) 1-8.

[30] Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. Int. J. of Computer Vision. 60 (2), 91-110.

[31] Nielsen, J. 2003. Usability 101: Introduction to Usability. http://www.useit.com/alertbox/20030825.html.

[32] Ng, T.-T., Chang, S.-F., Lin, C.-Y., and Sun, Q. 2006. Passive-blind image forensics. Chapter 15, Multimedia Security

Technologies for Digital Rights Management, edited by Zeng, W., Yu, H., and Lin, C.-Y., Academic Press, (2006), 383-412.

[33] Gloe, T., Kirchner, M., Winkler, A., and Böhme, R. 2007. Can we trust digital image forensics? In ACM Multimedia 2007, 78-86.

[34] Wu, Q., Sun, S. J., Zhu, W., and Li, G. H. 2009. Identification of inpainted images and natural images for digital forensics. Journal of Electronics (China), 26 (3) (2009), 341–345.

[35] Canny, J. 1986. A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8 (1986):679-714.

9. APPENDIX

9.1 Line Segment Detection

The following two-step procedure is applied to the binary edge image I_{bin-vh} obtained in Section 3.2.1.1 to find all the potential vertical and horizontal line segments.

- **Tracing step.** For the horizontal direction, we scan each row from left to right to search connected edge points. A line segment tracing ends when a gap occurs or the edge point drifts away from the current line segment's row index estimated by averaging the traced points. A similar operation is applied for the vertical direction. The result of this step is a collection of short horizontal and vertical line segments.
- **Clustering step.** In this step, the line segments from the previous step are divided into different subsets with an online clustering algorithm according to adjacency. The online clustering algorithm for horizontal line segments works as follows.

Online Segments Clustering Algorithm

Input: A set of short horizontal line segments
1. For each new arrived line segment, find its nearest cluster in the current cluster set $C = \{C_1, C_2, \ldots, C_N\}$. If C is empty or the nearest distance exceeds a threshold σ, a new cluster C_{N+1} is generated. Otherwise the line segment is added to the nearest cluster C_n.
2. Update the row index for each C_n. The row index of C_n is calculated by averaging the row index weighted by the line length on all the members in C_n.
3. For each cluster in C, sort its line segments by their left end points. Merge two line segments if their horizontal gap is within a tolerance threshold θ_g.
4. Recalculate the row index for each line segment by minimizing the average distance to adjacent binary horizontal edge points.

Output: Final horizontal line segments

9.2 Cleansing and Ranking Rectangles

Candidate rectangles are processed and ranked according to the following visual cues executed in the order of their presentation in this section.

9.2.1 Edge Intensity Cue

The first cue is the edge intensity on each side of a rectangle, which indicates how visible the rectangle is. This cue is useful to find a true image rectangle since at least one constituent image should be easily perceived by humans. It is worth noting that false boundaries generated by the dithering process tend to be weak enough to avoid confusing humans.

Given a boundary L of N points, the edge intensity cue along L is defined as follows:

$$F_{intensity}(L) = \sqrt{\frac{1}{N}\sum_{p=1}^{N} I(p_{max})w(p)}; \quad p \in L, \quad (2)$$

where $I(p_{max})$ is the edge intensity of the nearest local maxima p_{max} to location p along the perpendicular direction. This intensity is capped by a maximum value I_{th}. The weight function $w(p)$ takes into account the distance d between p and p_{max}:

$$w(p) = \begin{cases} 1 - \left[\frac{d(p, p_{max})}{d_{max}}\right]^2; & d(p, p_{max}) < d_{max} \\ 0, & otherwise \end{cases}, \quad (3)$$

Where d_{max} is the maximum distance to search for the nearest local maxima, and $d(x, y)$ is the distance between two points x and y. Rectangles with at least one side whose $F_{intensity}$ is smaller than a threshold are removed from the list of candidate rectangles.

9.2.2 Traversing Object Cue

A true image region boundary does not have any traversing object while a false one may have since dithering may not make a traversing object disappear. The following steps are applied to find traversing objects.

The canny edge detection [35] with an adaptive threshold is applied to each color channel, and the results from the color channels are properly aligned and merged. Vertical and horizontal line segments are not considered as part of an object to avoid interference from the true or false image boundaries. Then the edge density on each side of a rectangle boundary is calculated in the same way as that described next in Appendix 9.2.3. No traversing object cue is applied to the boundary if the edge density on either side is too large to avoid mismatched objects across the boundary, or if there is no contour long enough on either side. Otherwise contour segments along the boundary are searched and matched in the following way: for each contour segment long enough along the boundary, contour segments on the other side of the boundary are searched, and a matched contour is found if 1) both contour segments have adjacent ending points; 2) the contour segment to be matched is also long enough; 3) the two contour segments are smooth enough across the boundary. If two matched contour segments across the boundary are found, they form a traversing object.

If a traversing object is detected along a boundary, the boundary is presumed to be false, and all the associated rectangles are removed from the set of candidates.

9.2.3 Edge Density Variation Cue

If the textures on both sides of a boundary are very different and the change of texture aligns with the boundary, the boundary is likely to be a true image region boundary. We use edge density as a measure of texture. Edge density variation across the boundary is our third cue. The edge density in a rectangle area R is defined as follows:

$$D(R; \theta) = \frac{\sum_{x_{i1} \leq x < x_{i2}, y_{i1} \leq y < y_{i2}} I_{bin-all}(x, y; \theta)}{(x_{i2} - x_{i1})(y_{i2} - y_{i1})}, \quad (4)$$

where $I_{bin-all}$ is the binary edge map after applying a threshold θ on the total edge image I_{edge}. Dithering may cause edges on one side less visible than the other side. A uniform threshold may result in significantly fewer edges on one side. This problem can be addressed by applying a different threshold on each side of the boundary, θ_1 on one side and θ_2 on the other. The edge density variation is calculated as follows:

$$F_{variation}(L) = \frac{|D(R_1; \theta_1) - (R_2; \theta_2)|}{D(R_1; \theta_1) + D(R_2; \theta_2)} \quad (5)$$

Thresholds θ_1 and θ_2 are chosen to minimize Eq. (5) under the following conditions:

1) Both θ_1 and θ_2 are larger than a minimal threshold value θ_{min} and their difference is within a range: $|\theta_1 - \theta_2| \leq \Delta$;
2) At least one side has its edge density larger than a minimum edge density unless θ_{min} is reached.

If the minimum edge density variation across a boundary F_{min} is large than a threshold F_{th}, the boundary is presumed to be a true image region boundary and is assigned a confidence value of 1. A boundary that is a part of the boundary of the composite image is also assigned a confidence value of 1. Other boundaries are each assigned a confidence value of $\frac{F_{min}}{F_{th}} * 0.9$, where $F_{min} \leq F_{th}$. The confidence of a rectangle is the average of the confidences of its four side boundaries.