INTERNATIONAL JOURNAL OF SCIENTIFIC PROGRESS AND RESEARCH (IJSPR) ISSN: 2349-4689

Primitive Image Security Based On Capcha

[1]KavyaShree N, [2]Mr.PrasannaKumar M J, [3]Kavya J, [4]Shwetha S N
[1]P G Scholar , [24]Asst. Prof. , [3] U G Scholar
[124] CS&E, BGSIT, Mandya
[3]IS&E, SSIT, Tumkur

Abstract— Many security primitives are based on hard mathematical problems. Using hard AI problems for security is emerging as an exciting new paradigm, but has been under-explored. In this paper, we present a new security primitive based on hard AI problems, namely, a novel family of graphical password systems built on top of Captcha technology, which we call Captcha as graphical passwords (CaRP). CaRP is both a Captcha and a graphical password scheme. CaRP addresses a number of security problems altogether, such as online guessing attacks, relay attacks, and, if combined with dual-view technologies, shoulder-surfing attacks. Notably, a CaRP password can be found only probabilistically by automatic online guessing attacks even if the password is in the search set. CaRP also offers a novel approach to address the well-known image hotspot problem in popular graphical password systems, such as PassPoints, that often leads to weak password choices. CaRP is not a panacea, but it offers reasonable security and usability and appears to fit well with some practical applications for improving online security.

Index Terms— Graphical password, password, hotspots, CaRP, Captcha, dictionary attack, password guessing attack, security primitive.

I. INTRODUCTION

AFUNDAMENTAL task in security is to create crypto - graphic primitives based on hard mathematical problems that are computationally intractable. For example, the problem of integer factorization is fundamental to the RSA public-key cryptosystem and the Rabin encryption. The discrete logarithm problem is fundamental to the ElGamal encryption, the Diffie-Hellman key exchange, the Digital Signature Algorithm, the elliptic curve cryptography and so on.

Using hard AI (Artificial Intelligence) problems for security, initially proposed in [17], is an exciting new par-adigm. Under this paradigm, the most notable primitive invented is Captcha, which distinguishes human users from computers by presenting a challenge, i.e., a puzzle, beyond Manuscript received April 15, 2013; revised July 22, 2013 and October 14, 2013; accepted February 21, 2014. Date of publication March 19, 2014; date of current version April 21, 2014. This work was done when G. Bao and M. Yang worked as interns at Microsoft Research Asia. The associate editor coordinating the review of this manuscript and, U.K. (e-mail: jeff.yan@ncl.ac.uk).

G. approving it for publication was Prof. Carlo Blundo. B. B. Zhu and N. Xu are with Microsoft Research Asia, Beijing 100080, China (e-mail: binzhu@microsoft.com; ningx@microsoft.com).J. Yan is with Newcastle University, Newcastle NE 1 7RUBao is with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: guanbo.bao@gmail.com).M. Yang is with Sichuan University, Chengdu 610207, China (e-mail: djyangmaowei@gmail.com).Color versions of one or more of the figures in this paper are available online at the capability of computers but easy for humans. Captcha is now a standard Internet security technique to protect online email and other services from being abused by bots.

However, this new paradigm has achieved just a limited success as compared with the cryptographic primitives based on hard math problems and their wide applications. Is it possible to create any new security primitive based on hard AI problems? This is a challenging and interesting open problem. In this paper, we introduce a new security primitive based on hard AI problems, namely, a novel family of graphical pass-word systems integrating Captcha technology, which we call CaRP (Captcha as graphical Passwords) . CaRP is click-based graphical passwords, where a sequence of clicks on an image is used to derive a password. Unlike other click-based graphical passwords, images used in CaRP are Captcha challenges, and a new CaRP image is generated for every login attempt.

The notion of CaRP is simple but generic. CaRP can have multiple instantiations. In theory, any Captcha scheme relying on multiple-object classification can be converted to a CaRP scheme. We present exemplary CaRPs built on both text Captcha and image-recognition Captcha. One of them is a text CaRP wherein a password is a sequence of characters like a text password, but entered by clicking the right character sequence on CaRP images. CaRP offers protection against online dictionary attacks on passwords, which have been for long time a major security threat for various online services. This threat is widespread and considered as a top cyber security risk [13]. Defence against online dictionary attacks is a more subtle problem than it might appear. Intuitive countermeasures such as throttling logon attempts do not work well for two reasons: It causes denial-of-service attacks (which were exploited to lock highest bidders out in final minutes of eBay auctions [12]) and incurs expensive helpdesk costs for account

reactivation. It is vulnerable to global password attacks [14] whereby adversaries intend to break into any account rather than a specific one, and thus try each password candidate on multiple accounts and ensure that the number of trials on each account is below the threshold to avoid triggering account lockout. CaRP also offers protection against relay attacks, an increasing threat to bypass Captcha protection, wherein Captcha challenges are relayed to humans to solve. Koobface [33] was a relay attack to bypass Facebook's Captcha in creating new accounts. CaRP is robust to shoulder-surfing attacks if combined with dual-view tech CaRP requires solving a Captcha challenge in every login. This impact on usability can be mitigated by adapting the CaRP image's difficulty level based on the login history of the account and the machine used to login. Typical application scenarios for CaRP include: CaRP can be applied on touch-screen devices whereon typing passwords is cumbersome, esp. for secure Internet applications such as e-banks. Many e-banking systems have applied Captchas in user logins [39]. For example, ICBC (www.icbc.com.cn), the largest bank in the world, requires solving a Captcha challenge for every online login attempt. CaRP increases spammer's operating cost and thus helps reduce spam emails. For an email service provider that deploys CaRP, a spam bot cannot log into an email account even if it knows the password. Instead, human involvement is compulsory to access an account. If CaRP is combined with a policy to throttle the number of emails sent to new recipients per login session, a spam bot can send only a limited number of emails before asking human assistance for login, leading to reduced outbound spam traffic. The remaining paper is organized as follows: Background and related work are presented in Section II. We outline CaRP in Section III, and present a variety of CaRP schemes in Sections IV and V. Security analysis is provided in Section VI. A usability study on two CaRP schemes that we have imple-mented is reported in Section VII. Balance of security and usability is discussed in Section VIII. We conclude the paper with Section IX.

II. BACKGROUND AND RELATED WORK

A. Graphical Passwords

A large number of graphical password schemes have been proposed. They can be classified into three categories accord-ing to the task involved in memorizing and entering passwords: recognition, recall, and cued recall. Each type will be briefly described here. More can be found in a recent review of graphical passwords [1].A recognition- based scheme requires identifying among decoys the visual objects belonging to a password portfolio. A typical scheme is Passfaces [2] wherein a user selects a portfolio of faces from a database in creating a password. During authentication, a panel of candidate faces is presented for the user to select the face belonging to her portfolio. This process is repeated several rounds, each round with a different panel. A successful login requires correct selection in each round. The set of images in a panel remains the same between logins, but their locations are permuted. Story [20] is similar to Passfaces but the images in the portfolio are ordered, and a user must identify her portfolio images in the correct order. Déjà Vu [21] is also similar but uses a large set of computer-generated "random-art" images. Cognitive Authentication [22] requires a user to generate a path through a panel of images as follows: starting from the top-left image, moving down if the image is in her portfolio, or right otherwise. The user identifies among decoys the row or column label that the path ends.

This process is repeated, each time with a different panel. A successful login requires that the cumulative probability that correct answers were not entered by chance exceeds a threshold within a given number of rounds. A recall-based scheme requires a user to regenerate the same interaction result without cueing. Draw-A- Secret (DAS) [3] was the first recall -based scheme proposed. A user draws her password on a 2D grid. The system encodes the sequence of grid cells along the drawing path as a user-drawn password. Pass-Go [4] improves DAS's usability by encoding the grid intersection points rather than the grid cells. BDAS [23] adds background images to DAS to encourage users to create more complex passwords. In a cued-recall scheme, an external cue is provided to help memorize and enter a password. PassPoints [5] is a widely studied click- based cued-recall scheme wherein a user clicks a sequence of points anywhere on an image in creating a password, and re-clicks the same sequence during authentication. Cued Click Points (CCP) [18] is similar to PassPoints but uses one image per click, with the next image selected by a deterministic function. Persuasive Cued Click Points (PCCP) [19] extends CCP by requiring a user to select a point inside a randomly positioned viewport when creating a password, resulting in more randomly distributed click-points in a password.

Among the three types, recognition is considered the easiest for human memory whereas pure recall is the hardest [1]. Recognition is typically the weakest in resisting guessing attacks. Many proposed recognition-based schemes practically have a password space in the range of 213 to 216 passwords [1]. A study [6] reported that a significant portion of passwords of DAS and Pass-Go [4] were successfully broken with guessing attacks using dictionaries of 231 to 241 entries, as compared to

the full password space of 258 entries. Images contain hotspots [7], [8], i.e., spots likely selected in creating passwords. Hotspots were exploited to mount successful guessing attacks on PassPoints [8]–[11]: a significant portion of passwords were broken with dictionaries of 226 to 235 entries, as compared to the full space of 243 passwords.

B. Captcha

Captcha relies on the gap of capabilities between humans and bots in solving certain hard AI problems. There are two types of visual Captcha: text Captcha and Image-Recognition Captcha (IRC). The former relies on character recognition while the latter relies on recognition of non-character objects. Security of text Captchas has been extensively studied [26]– [30]. The following principle has been established: text Captcha should rely on the difficulty of character segmentation, which is computationally expensive and combinatorial hard [30].Machine recognition of non-character objects is far less capable than character recognition. IRCs rely on the difficulty of object identification or classification, possibly combined with the difficulty of object segmentation. Asirra [31] relies on binary object classification: a user is asked to identify all the cats from a panel of 12 images of cats and dogs. Security of IRCs has also been studied. Asirra was found to be susceptible to machine-learning attacks [24]. IRCs based on binary object classification or identification of one concrete type of objects are likely insecure [25]. Multi-label classification problems are considered much harder than binary classification problems. Captcha can be circumvented through relay attacks whereby Captcha challenges are relayed to human solvers, whose answers are fed back to the targeted application.

C. Captcha in Authentication

It was introduced in [14] to use both Captcha and password in a user authentication protocol, which we call Captcha-based Password Authentication (CbPA) protocol, to counter online dictionary attacks. The CbPA-protocol in [14] requires solving a Captcha challenge after inputting a valid pair of user ID and password unless a valid browser cookie is received. For an invalid pair of user ID and password, the user has a certain probability to solve a Captcha challenge before being denied access. An improved CbPA-protocol is proposed in [15] by storing cookies only on user-trusted machines and applying a Captcha challenge only when the number of failed login attempts for the account has exceeded a threshold. It is further improved in [16] by applying a small threshold for failed login attempts from unknown machines but a large threshold for failed attempts from known machines with a previous successful login within a given time frame. Captcha was also used with recognition-based graphical passwords to address spyware [40], [41], wherein a text Captcha is displayed below each image; a user locates her own pass-images from decoy images, and enters the characters at specific locations of the Captcha below each pass-image as her password during authentication. These specific locations were selected for each pass-image during password creation as a part of the password.

In the above schemes, Captcha is an independent entity, used together with a text or graphical password. On the contrary, a CaRP is both a Captcha and a graphical password scheme, which are intrinsically combined into a single entity.

D. Other Related Work

Captcha is used to protect sensitive user inputs on an untrusted client [35]. This scheme protects the communication channel between user and Web server from keyloggers and spyware, while CaRP is a family of graphical password schemes for user authentication. The paper [35] did not introduce the notion of CaRP or explore its rich properties and the design space of a variety of CaRP instantiations.

III. CAPTCHA AS GRAPHICAL PASSWORDS

A. A New Way to Thwart Guessing Attacks

In a guessing attack, a password guess tested in an unsuccessful trial is determined wrong and excluded from subsequent trials. The number of undetermined password guesses decreases with more trials, leading to a better chance of finding the password. Mathematically, let S be the set of password guesses before any trial, ρ be the password to find, T denote a trial whereas Tn denote the n-th trial, and p(T = ρ) be the probability that ρ is tested in trial T . Let En be the set of password guesses tested in trials up to (including) Tn . The password guess to be tested in n-th trial Tn is from set S\ E n −1, i.e., the relative complement of En−1 in S. If ρ S, then we have p (T = ρ|T1 = ρ , . . . , Tn−1 = ρ) > p(T = ρ), (1)

And p(T = ρ|T1 = ρ , . . . , Tn−1 = ρ) → 1 → | |

En → S with n S , (2) where |S | denotes the cardinality of S. From Eq. (2), the password is always found within |S| trials if it is in S; otherwise S is exhausted after |S| trials. Each trial determines if the tested password guess is the actual password or not, and the trial's result is deterministic.

To counter guessing attacks, traditional approaches in designing graphical passwords aim at increasing the effective password space to make passwords harder to guess and thus require more trials. No matter how secure a graphical password scheme is, the password can always be found by a brute force attack. In this paper, we distinguish two types of guessing attacks: automatic guessing attacks apply an automatic trial and error process but S can be manually constructed whereas human guessing attacks apply a manual trial and error process. CaRP adopts a completely different approach to counter automatic guessing attacks. It aims at realizing the following equation:

$$p(T = \rho | T1, \ldots, Tn-1) = p(T = \rho), \quad n \qquad (3)$$

in an automatic guessing attack. Eq. (3) means that each trial is computationally independent of other trials. Specifically, no matter how many trials executed previously, the chance of finding the password in the current trial always remains the same. That is, a password in S can be found only probabilistically by automatic guessing (including brute-force) attacks, in contrast to existing graphical password schemes where a password can be found within a fixed number of trials.

How to achieve the goal? If a new image is used for each trial, and images of different trials are independent of each other, then Eq. (3) holds. Independent images among different login attempts must contain invariant information so that the authentication server can verify claimants. By examining the ecosystem of user authentication, we noticed that human users enter passwords during authentication, whereas the trial and error process in guessing attacks is executed automatically. The capability gap between humans and machines can be exploited to generate images so that they are computationally-independent yet retain invariants that only humans can iden-tify, and thus use as passwords. The invariants among images must be intractable to machines to thwart automatic guessing attacks. This requirement is the same as that of an ideal Captcha [25], leading to creation of CaRP, a new family of graphical passwords robust to online guessing a B. CaRP: An Overview In CaRP, a new image is generated for every login attempt, even for the same user. CaRP uses an alphabet of visual objects (e.g., alphanumerical characters, similar animals) to generate a CaRP image, which is also a Captcha challenge. A major difference between CaRP images and Captcha images is that all the visual objects in the alphabet should appear in a CaRP image to allow a user to input any password but not necessarily in a Captcha image. Many Captcha schemes can be converted to CaRP schemes, as described in the next subsection.

CaRP schemes are clicked-based graphical passwords. According to the memory tasks in memorizing and enter-ing a password, CaRP schemes can be classified into two categories: recognition and a new category, recognition-recall, which requires recognizing an image and using the recog-nized objects as cues to enter a password. Recognition-recall combines the tasks of both recognition and cued-recall, and retains both the recognition-based advantage of being easy for human memory and the cued-recall advantage of a large password space. Exemplary CaRP schemes of each type will be presented later.

C. Converting Captcha to CaRP

In principle, any visual Captcha scheme relying on recogniz-ing two or more predefined types of objects can be converted to a CaRP. All text Captcha schemes and most IRCs meet this requirement. Those IRCs that rely on recognizing a single predefined type of objects can also be converted to CaRPs in general by adding more types of objects. In practice, conversion of a specific Captcha scheme to a CaRP scheme typically requires a case by case study, in order to ensure both security and usability. We will present in Sections IV and V several CaRPs built on top of text and image-recognition Captcha schemes. Some IRCs rely on identifying objects whose types are not predefined. A typical example is Cortcha [25] which relies on context-based object recognition wherein the object to be recognized can be of any type. These IRCs cannot be converted into CaRP since a set of pre-defined object types is essential for constructing a password.

D. User Authentication With CaRP Schemes

Like other graphical passwords, we assume that CaRP schemes are used with additional protection such as secure channels between clients and the authentication server through Transport Layer Security (TLS). A typical way to apply CaRP schemes in user authentication is as follows. The authentica-tion server AS stores a salt s and a hash value H (ρ , s) for each user ID, where ρ is the password of the account and not stored. A CaRP password is a sequence of visual object IDs or clickable- points of visual objects that the user selects. Upon receiving a login request, AS generates a CaRP image, records the locations of the objects in the image, and sends the image to the user to click her password. The coordinates of the clicked points are recorded and sent to AS along

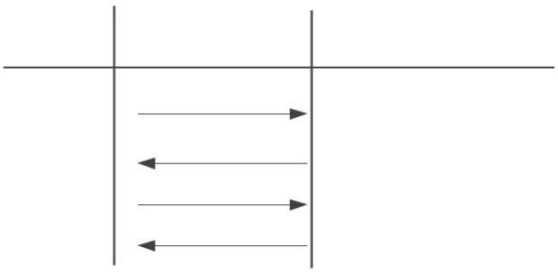

Fig. 1. Flowchart of basic CaRP authentication.

with the user ID. AS maps the received coordinates onto the CaRP image, and recovers a sequence of visual object IDs or clickable points of visual objects, ρ , that the user clicked on the image. Then AS retrieves salt s of the account, calculates the hash value of ρ with the salt, and compares the result with the hash value stored for the account. Authentication succeeds only if the two hash values match. This process is called the basic CaRP authentication and shown in Fig. 1.

Advanced authentication with CaRP, for example, challenge-response, will be presented in Section V-B. We assume in the following that CaRP is used with the basic CaRP authentication unless explicitly stated otherwise.

To recover a password successfully, each user-clicked point must belong to a single object or a clickable-point of an object. Objects in a CaRP image may overlap slightly with neighboring objects to resist segmentation. Users should not click inside an overlapping region to avoid ambiguity in identifying the clicked object. This is not a usability concern in practice since overlapping areas generally take a tiny portion of an object.

IV. RECOGNITION-BASED CARP

For this type of CaRP, a password is a sequence of visual objects in the alphabet. Per view of traditional recognition-based graphical passwords, recognition-based CaRP seems to have access to an infinite number of different visual objects. We present two recognition-based CaRP schemes and a variation next.

A. ClickText

ClickText is a recognition-based CaRP scheme built on top of text Captcha. Its alphabet comprises characters without any visually-confusing characters. For example, Letter “O” and digit “0” may cause confusion in CaRP images, and thus one character should be excluded from the alphabet. A ClickText password is a sequence of characters in the alphabet, e.g., ρ =“AB#9CD87”, which is similar to a text password. A ClickText image is generated by the underlying Captcha engine as if a Captcha image were generated except that all the alphabet characters should appear in the image. During generation, each character’s location is tracked to produce ground truth for the location of the character in the generated image. The authentication server relies on the ground truth to identify the characters corresponding to user clicked points. In ClickText images, characters can be arranged randomly

ZHUetal.: NEW SECURITY PRIMITIVE BASED ON HARD AI PROBLEMS

Fig. 2. A ClickText image with 33 characters.

Fig. 3. Captcha Zoo with horses circled red.

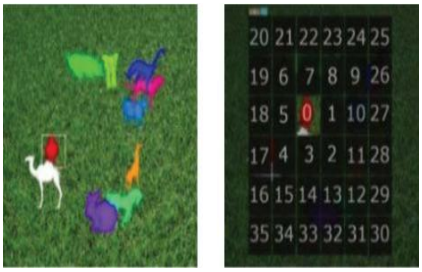

Fig. 4. A ClickAnimal image (left) and 6 × 6 grid (right) determined by red turkey’s bounding rectangle. On 2D space. This is different from text Captcha challenges in which characters are typically ordered from left to right in order for users to type them sequentially. Fig. 2 shows a ClickText image with an alphabet of 33 characters. In entering a password, the user clicks on this image the characters in her password, in the same order, for example “A”, “B”, “#”, “9”, “C”, “D”, “8”, and then “7” for password ρ = “AB#9CD87”.

B. Click Animal

Captcha Zoo [32] is a Captcha scheme which uses 3D models of horse and dog to generate 2D animals with different textures, colors, lightings and poses, and arranges them on a cluttered background. A user clicks

all the horses in a challenge image to pass the test. Fig. 3 shows a sample challenge wherein all the horses are circled red Click Animal is a recognition-based CaRP scheme built on top of Captcha Zoo [32], with an alphabet of similar animals such as dog, horse, pig, etc. Its password is a sequence of animal names such as ρ = "Turkey, Cat, Horse, Dog,…." For each animal, one or more 3D models are built. The Captcha generation process is applied to generate Click Animal images: 3D models are used to generate 2D animals by applying different views, textures, colours, lightning effects, and optionally distortions. The resulting 2D animals are then arranged on a cluttered background such as grassland. Some animals may be occluded by other animals in the image, but their core parts are not occluded in order for humans to identify each of them. Fig. 4 shows a ClickAnimal image with an alphabet of 10 animals. Note that different views applied in mapping 3D models to 2D animals, together with occlusion in the following step, produce many different shapes for the same animal's instantiations in the generated images. Combined with the additional anti-recognition mechanisms applied in the mapping step, these make it hard for computers to recognize animals in the generated image, yet humans can easily identify different instantiations of animals.

C. Animal Grid

The number of similar animals is much less than the number of available characters. Click Animal has a smaller alphabet, and thus a smaller password space, than ClickText. CaRP should have a sufficiently-large effective password space to resist human guessing attacks. Animal Grid's password space can be increased by combining it with a grid-based graphical password, with the grid depending on the size of the selected animal.

DAS [3] is a candidate but requires drawing on the grid. To be consistent with , we change from drawing to clicking: Click-A-Secret (CAS) wherein a user clicks the grid cells in her password. Animal Grid is a combination of Click Animal and CAS. The number of grid-cells in a grid should be much larger than the alphabet size. Unlike DAS, grids in our CAS are object-dependent, as we will see next. It has the advantage that a correct animal should be clicked in order for the clicked grid-cell(s) on the follow-up grid to be correct. If a wrong animal is clicked, the follow- up grid is wrong. A click on the correctly labeled grid-cell of the wrong grid would likely produce a wrong grid- cell at the authentication server side when the correct grid is used.

To enter a password, a ClickAnimal image is displayed first. After an animal is selected, an image of n × n grid appears, with the grid-cell size equaling the bounding rectangle of the selected animal. Each grid-cell is labeled to help users identify. Fig. 4 shows a 6 × 6 grid when the red turkey in the left image of Fig. 4 was selected. A user can select zero to multiple grid-cells matching her password. Therefore a password is a sequence of animals interleaving with grid-cells, e.g., ρ = "Dog, Grid 2 , Grid 1 ; Cat, Horse, Grid 3 ", where Grid 1 means the grid-cell indexed as 1, and grid-cells after an animal means that the grid is determined by the bounding rectangle of the animal. A password must begin with an animal.

When a ClickAnimal image appears, the user clicks the animal on the image that matches the first animal in her password. The coordinates of the clicked point are recorded. The bounding rectangle of the clicked animal is then found interactively as follows: a bounding rectangle is calculated and displayed, e.g., the white rectangle shown in Fig. 4. The user checks the displayed rectangle and corrects inaccurate edges by dragging if needed. This process is repeated until the user is satisfied with the accuracy of the bounding rectangle. In most cases, the calculated bounding rectangle is accurate enough without needing manual correction.

Once the bounding rectangle of the selected animal is identified, an image of n × n grid with the identified bounding rectangle as its grid-cell size is generated and displayed. If the grid image is too large or too small for a user to view, the grid image is scaled to a fitting size. The user then clicks a sequence of zero to multiple grid-cells that match the cells following the first animals in her password, and then gets back to the Click Animal image. For the example password ρ given previously, she clicks a point inside grid-cell 2 , and then a point inside grid-cell 1 to select the two grid-cells. The coordinates of user-clicked points on the grid image (the original one before scaling if the grid image is scaled) are recorded. The above process is repeated until the user has finished entering her password. The resulting sequence of coordinates of user-clicked points, e.g., "AP 150, 50 ,GP 30,66 , GP 89,160 , AP 135,97 ,…" where "AP x,y " denotes the point with coordinates x,y on a ClickAnimal image, and "GP x,y " denotes the point with coordinates x,y on a grid image, is sent to the authentication server.

Using the ground truth, the server recovers the first animal from the received sequence, regenerates the grid image from the animal's bounding rectangle, and recovers the clicked grid-cells. This process is repeated to recover the password the user clicked. Its hash is then calculated and compared with the stored hash.

V. V. RECOGNITION-RECALL CARP

In recognition-recall CaRP, a password is a sequence of some invariant points of objects. An invariant point of an object (e.g. letter "A") is a point that has a fixed relative position in different incarnations (e.g., fonts) of the object, and thus can be uniquely identified by humans no matter how the object appears in CaRP images. To enter a password, a user must identify the objects in a CaRP image, and then use the identified objects as cues to locate and click the invariant points matching her password. Each password point has a tolerance range that a click within the tolerance range is acceptable as the password point. Most people have a click variation of 3 pixels or less [18]. TextPoint, a recognition-recall CaRP scheme with an alphabet of characters, is presented next, followed by a variation for challenge-response authentication.

A. TextPoints

Characters contain invariant points. Fig. 5 shows some invariant points of letter "A", which offers a strong cue to memorize and locate its invariant points. A point is said to be an internal point of an object if its distance to the closest boundary of the object exceeds a threshold. A set of internal invariant points of characters is selected to form a set of clickable points for TextPoints. The internality ensures that a clickable point is unlikely occluded by a neighboring character and that its tolerance region unlikely overlaps with any tolerance region of a neighboring character's clickable points on the image generated by the underlying Captcha engine. In determining clickable points, the distance between any pair of clickable points in a character must exceed a threshold so that they are perceptually distinguishable and their tolerance regions do not overlap on CaRP images. In addition, variation should also be taken into consideration. For example, if the center of a stroke segment in one character is selected, we should avoid selecting the center of a similar stroke segment in another character. Instead, we should select

Fig. 5. Some invariant points (red crosses) of "A"

a different point from the stroke segment, e.g., a point at one-third length of the stroke segment to an end. This variation in selecting clickable points ensures that a clickable point is context-dependent: a similarly structured point may or may not be a clickable point, depending on the character that the point lies in. Character recognition is required in locating clickable points on a TextPoints image lthough the clickable points are known for each character. This is a task beyond a bot's capability.

A password is a sequence of clickable points. A character can typically contribute multiple clickable points. Therefore TextPoints has a much larger password space than ClickText. Image Generation. TextPoints images look identical to ClickText images and are generated in the same way except that the locations of all the clickable points are checked to ensure that none of them is occluded or its tolerance region overlaps another clickable point's. We simply generate another image if the check fails. As such failures occur rarely due to the fact that clickable points are all internal points, the restriction due to the check has a negligible impact on the security of generated images.

Authentication. When creating a password, all clickable points are marked on corresponding characters in a CaRP image for a user to select. During authentication, the user first identifies her chosen characters, and clicks the password points on the right characters. The authentication server maps each user-clicked point on the image to find the closest clickable point. If their distance exceeds a tolerable range, login fails. Otherwise a sequence of clickable points is recovered, and its hash value is computed to compare with the stored value.

It is worth comparing potential password points between TextPoints and traditional click-based graphical passwords such as PassPoints [5]. In PassPoints, salient points should be avoided since they are readily picked up by adversaries to mount dictionary attacks, but avoiding salient points would increase the burden to remember a password. This conflict does not exist in TextPoints. Clickable points in TextPoints are salient points of their characters and thus help remember a password, but cannot be exploited by bots since they are both dynamic (as compared to static points in traditional graphical password schemes) and contextual:

Dynamic: locations of clickable points and their contexts (i.e., characters) vary from one image to another. The clickable points in one image are computationally independent of the clickable points in another image, as we will see in Section VI-B.

Contextual: Whether a similarly structured point is a clickable point or not depends on its context. It is only if within the right context, i.e., at the right location of a right character.

These two features require recognizing the correct contexts, i.e., characters, first. By the very nature of Captcha, recognizing characters in a Captcha image is a

task beyond computer's capability. Therefore, these salient points of characters cannot be exploited to mount dictionary attacks on Text Points.

B. TextPoints4CR

For the CaRP schemes presented up to now, the coordinates of user-clicked points are sent directly to the authentication server during authentication. For more complex protocols, say a challenge-response authentication protocol, a response is sent to the authentication server instead. Text Points can be modified to fit challenge-response authentication. This variation is called Text Points for Challenge-Response or TextPoints4CR.

Unlike Text Points wherein the authentication server stores a salt and a password hash value for each account, the server in TextPoints4CR stores the password for each account. Another difference is that each character appears only once in a TextPoints4CR image but may appear multiple times in a Text Points image. This is because both server and client in TextPoints4CR should generate the same sequence of discretized grid-cells independently. That requires a unique way to generate the sequence from the shared secret, i.e., password. Repeated characters would lead to several possible sequences for the same password. This unique sequence is used as if the shared secret in a conventional challenge-response authentication protocol. In TextPoints4CR, an image is partitioned into a fixed grid with the discretization grid-cell of size μ along both directions. The minimal distance between any pair of clickable points should be larger than μ by a margin exceeding a threshold to prevent two clickable points from falling into a single grid- cell in an image. Suppose that a guaranteed tolerance of click errors along both x-axis and y-axis is τ , we require that $\mu \geq 4\tau$.

Image Generation. To generate a TextPoints4CR image, the same procedure to generate a TextPoints image is applied. Then the following procedure is applied to make every click-able point at least τ distance from the edges of the grid-cell it lies in. All the clickable points, denoted as set , are located on the image. For every point in , we calculate its distance along x-axis or y-axis to the center of the grid-cell it lies in. A point is said to be an internal point if the distance is less than $0.5\mu-\tau$ along both directions; otherwise a boundary point. For each boundary point in , a nearby internal point in the same grid-cell is selected. The selected point is called a target point of the boundary point. After processing all the points in , we obtain a new set comprising internal points; these are either internal clickable points or target points of boundary clickable points. Mesh warping [36], widely used in generating text Captcha challenges, is then used to warp the image so that maps to . The result is a TextPoint4CR image wherein every clickable point would tolerate at least τ of click errors. Selection of target points should try to reduce warping distortion caused by mapping to. Authentication. In entering a password, a user-clicked point is replaced by the grid-cell it lies in. If click errors are within τ, each user-clicked point falls into the same grid-cell as the original password point. Therefore the sequence of grid-cells generated from user-clicked points is identical to the one that the authentication server generates from the stored password of the account. This sequence is used as if the shared secret between the two parties in a challenge-response authentication protocol.

Unlike other CaRP schemes presented in this paper, Text-Points4CR requires the authentication server to store pass-words instead of their hash values. Stored passwords must be protected from insider attacks; for example, they are encrypted with a master key that only the authentication server knows. A password is decrypted only when its associated account attempts to log in.

VI. VI. SECURITY ANALYSIS

A. Security of Underlying Captcha

Computational intractability in recognizing objects in CaRP images is fundamental to CaRP. Existing analyses on Captcha security were mostly case by case or used an approximate process. No theoretic security model has been established yet. Object segmentation is considered as a computationally-expensive, combinatorially- hard problem [30], which modern text Captcha schemes rely on. According to [30], the com-plexity of object segmentation, C, is exponentially dependent of the number M of objects contained in a challenge, and polynomially dependent of the size N of the Captcha alphabet: $C = \alpha^{M P(N)}$, where $\alpha > 1$ is a parameter, and P() is a polynomial function. A Captcha challenge typically contains 6 to 10 characters, whereas a CaRP image typically contains 30 or more characters. The complexity to break a Click-Text image is about $\alpha^{30 P(N)}/(\alpha^{10 P(N)}) = \alpha^{20}$ times the complexity to break a Captcha challenge generated by its underlying Captcha scheme. Therefore ClickText is much harder to break than its underlying Captcha scheme. Furthermore, characters in a CaRP scheme are arranged two-dimensionally, further increasing segmentation difficulty due to one more dimension to segment. As a result, we can reduce distortions in ClickText images for improved usability yet maintain the same security level as the underlying text Captcha. Click Animal relies on both

object segmentation and multiple-label classification. Its security remains an open question.

As a framework of graphical passwords, CaRP does not rely on any specific Captcha scheme. If one Captcha scheme gets broken, a new and more robust Captcha scheme may appear and be used to construct a new CaRP scheme. In the remaining security analysis, we assume that it is intractable for computers to recognize any objects in any challenge image generated by the underlying Captcha of CaRP. More accurately, the Captcha is assumed to be chosen-pixel attack (CPA)-secure defined with the following experiment: an adversary A first learns from an arbitrary number of challenge images by querying a ground-truth oracle O as follows: A selects an arbitrary number of internal object-points and sends to O, which responds with the object that each point lies in. Then A receives a new challenge image and selects an internal object-point to query This time O chooses a random bit b ← {0, 1 } to determine what to return: It returns the true object if b = 1; otherwise a false object selected with a certain strategy. A is asked to determine whether the returned object is the true object that the internal object-point lies in or not. A Captcha scheme is said to be CPA-secure if A cannot succeed with a probability non-negligibly higher than ½, the probability of a random guess.

B. Automatic Online Guessing Attacks

In automatic online guessing attacks, the trial and error process is executed automatically whereas dictionaries can be constructed manually. If we ignore negligible probabilities, CaRP with underlying CPA-secure Captcha has the following properties: Internal object-points on one CaRP image are computationally-independent of internal object-points on another CaRP image. Particularly, clickable points on one image are computationally-independent of clickable points on another image.

Eq. (3) holds, i.e., trials in guessing attacks are mutually independent.

The first property can be proved by contradiction. Assume that the property does not hold, i.e., there exists an inter-nal object-point α on one image A that is non-negligibly dependent of an internal object-point β on another image B. An adversary can exploit this dependency to launch the following chosen-pixel attack. In the learning phase, image A is used to learn the object that contains point α. In the testing phase, point β on image B is used to query the oracle. Since point α is non-negligibly dependent of point β , this CPA-experiment would result in a success probability non-negligibly higher than a random guess, which contradicts the CPA-secure assumption. We conclude that the first property holds. The second property is a consequence of the first property since user-clicked internal object-points in one trial are computationally-independent of user-clicked internal object-points in another trial due to the first property. We have ignored background and boundary object -points since clicking any of them would lead to authentication failure. Eq. (3) indicates that automatic online guessing attacks can find a password only probabilistically no matter how many trials are executed. Even if the password guess to be tested in a trial is the actual password, the trial has a slim chance to succeed since a machine cannot recognize the objects in the CaRP image to input the password correctly. This is a great contrast to automatic online guessing attacks on existing graphical passwords which are deterministic, i.e., that each trial in a guessing attack can always determine if the tested password guess is the actual password or not, and all the password guesses can be determined by a limited number of trials. Particularly, brute-force attacks or dictionary attacks with the targeted password in the dic-tionary would always succeed in attacking existing graphical passwords.

C. Human Guessing Attacks

In human guessing attacks, humans are used to enter passwords in the trial and error process. Humans are much slower than computers in mounting guessing attacks. For 8- character passwords, the theoretical password space is 338 ≈ 240 for ClickText with an alphabet of 33 characters, 108 ≈ 226 for Click Animal with an alphabet of 10 animals, and 10 × 467 ≈ 242 for Animal Grid with the setting as Click Animal plus 6 × 6 grids. If we assume that 1000 people are employed to work 8 hours per day without any stop in a human guessing attack, and that each person takes 30 seconds to finish one trial. It would take them on average 0.5 • 338 • 30/ (3600 • 8 • 1000 • 365) ≈ 2007 years to break a ClickText password, 0.5 • 108 • 30/(3600 • 8 • 1000) ≈ 52 days to break a Click Animal password, or 0.5 • 10 • 467 • 30/(3600 • 8 • 1000 • 365) ≈ 6219 years to break an Animal Grid password. Human guessing attacks on Text Points require a much longer time than those on ClickText since Text Points has a much larger password space.

Just like any password scheme, a longitudinal evaluation is needed to establish the effective password space for each CaRP instantiation. This requires a separate study similar to what Bonneau [42] did for text passwords. A recent study on text passwords [42] indicates that users tend to choose passwords of 6–8 characters and have a strong dislike of using non-alphanumeric characters, and

that an acceptable benchmark of effective password space is the expected number of optimal guesses per account needed to break 50% of accounts, which is equivalent to 21.6 bits for Yahoo! users. If we assume that ClickText has roughly the same effective password space as text passwords, it requires on average 1000 people to work 1.65 days or one person to work 4.54 years to find a ClickText password.

D. Relay Attacks

Relay attacks may be executed in several ways. Captcha challenges can be relayed to a high- volume Website hacked or controlled by adversaries to have human surfers solve the challenges in order to continue surfing the Website, or relayed to sweatshops where humans are hired to solve Captcha challenges for small payments. Is CaRP vulnerable to relay attacks? We make the same assumption as Van Oorschot and Stubblebine [15] in discussing CbPA-protocol's robustness to relay attacks: a person will not deliberately participate in relay attacks unless paid for the task. The task to perform and the image used in CaRP are very different from those used to solve a Captcha challenge. This noticeable difference makes it hard for a person to mistakenly help test a password guess by attempting to solve a Captcha challenge. Therefore it would be unlikely to get a large number of unwitting people to mount human guessing attacks on CaRP. In addition, human input obtained by performing a Captcha task on a CaRP image is useless for testing a password guess.

If sweatshops are hired to mount human guessing attack, we can make a rough estimation of the cost. We assume that the cost to click one password on a CaRP image is the same as solving a Captcha challenge. Using the lowest retail Pr reported [34] to solve 1000 Captcha challenges, the average cost to break a 26-bit password is 0.5 • 226 • 1/1000, or about 33.6 thousand US dollars.

E. Shoulder-Surfing Attacks

Shoulder-surfing attacks are a threat when graphical passwords are entered in a public place such as bank ATM machines. CaRP is not robust to shoulder-surfing attacks by itself. However, combined with the following dual-view technology, CaRP can thwart shoulder-surfing attacks.

By exploiting the technical limitation that commonly-used LCDs show varying brightness and color depending on the viewing angle, the dual-view technology can use software alone to display two images on a LCD screen concurrently, one public image viewable at most view-angles, and the other private image viewable only at a specific view-angle [38]. When a CaRP image is displayed as the "private" image by the dual-view system, a shoulder-surfing attacker can capture user-clicked points on the screen, but cannot capture the "private" CaRP image that only the user can see. However, the obtained user-clicked points are useless for another login attempt, where a new, computationally-independent image will be used and thus the captured points will not represent the correct password on the new image anymore.

To the contrary, common implementations of graphical password schemes such as PassPoints use a static input image in the same location of the Screen for each login attempt. Although this image can be hidden as the private image by the dual-view technology from being captured by a shoulder-surfer, the user-clicked points captured in a successful login are still the valid password for next login attempt. That is, capturing the points alone is sufficient for an effective attack in this case.

In general, the higher the correlation of user-clicked points between different login attempts is, the less effective protection the dual-view technology would provide to thwart shoulder-surfing attacks.

F. Others

CaRP is not bulletproof to all possible attacks. CaRP is vulnerable if a client is compromised such that both the image and user-clicked points can be captured. Like many other graphical passwords such as CCP and PCCP, CaRP schemes using the basic CaRP authentication are vulnerable to phishing since user-clicked points are sent to the authentication server. However, CaRP schemes such as TextPoints4CR used with challenge-response authentication are robust to phishing to a certain level: a phishing adversary has to mount offline guessing attacks to find out the password using the verifiable data obtained through a successful phishing attack.

VII. EMPIRICAL EVALUATIONS

A. Implementations

ClickText and Animal Grid were implemented using ASP.NET. ClickText was implemented by calling a configurable text Captcha engine commercially used by Microsoft. This Captcha engine accepts only capital letters. As a result, we chose the following 33 characters in our usability studies: capital letters except I, J, O, and Z, digits except 0 and 1, and three special characters "#", "@", and "&". The last three special characters were chosen to balance security and users' strong dislike of using non-alphanumeric characters in text passwords [42]. Characters were arranged in 5 rows. Each character was randomly rotated from −30◦ to 30◦ and scaled From

60% to 120%. Neighbouring characters could overlap up to 3 pixels. Warping effect was set to the light level. Each image was set to 400 by 400 pixels. Fig. 2 in Section IV-A shows an image generated with the above setting.

In our implementation of Animal Grid, we used an alphabet of 10 animals: bird, cow, horse, dog, giraffe, pig, rabbit, camel, element, and dinosaur. Each animal had three 3D models. The number of animals in a Click Animal image ranged randomly from 10 to 12, with the extra animals randomly selected from the alphabet. In generating an animal object, one of the three 3D animal models was randomly selected, and posed at a random view in generating a 2D object. Each animal was assigned a colour randomly selected from a set of 12 colours. Generated 2D objects were placed randomly on a grass background, with the main part of each animal not occluded by other animals. Each Click Animal image was also set to 400 by 400 pixels. A 6 × 6 grid was used for CAS. Cells were labelled clockwise starting from cell 0. Fig. 4 in Section IV shows an example of generated Click Animal images and an example of grid images. There was a cross icon on top of a grid image that a user could click to close the grid image.

B. Usability Study

1**) Experimental Settings**

We conducted an in-lab usability study to compare Click-Text, AnimalGrid, PassPoints, text password (Text), and text password combined with text Captcha (P + C). P + C was used to simulate a CbPA-protocol when a Captcha challenge was used in login. In P + C, a user was asked to enter a password and solve a Captcha challenge generated with the same Captcha engine used in ClickText. Each Captcha challenge contained 6 to 8 random characters. Keyboard input was used to create and enter passwords for Text and P + C as well as to enter user IDs for all the schemes. As explained later, Text and P + C were conducted as if they were a single scheme to participants.We recruited 40 (30 males and 10 females) voluntary senior and graduate students majoring in engineering and sciences, with ages ranging from 20 to 28 years (the average age = 23.4 and the standard deviation = 1.74). For pragmatic rea-sons, they were recruited from interns working at Microsoft Research Asia. None of them had studied security or was involved in any security usability study before. They were involved in this work solely as participants in our usability study. All participants were trained to get familiar with each authentication scheme and their experimental tasks before our data collection. During the experiment, one of the authors got each participant when it was time for the participant to take a test, which ensures that we could collect the required data from every participant. Each scheme was tested in the following setting: a participant used a web browser to interact with an authentication server, creating passwords or logging into the server. Once a participant submitted his/her credentials to the server, the browser would show the login result.

The schemes were classified into two categories according to their types of passwords: Animal Grid and PassPoints in the first category, and the remaining schemes in the second category. A password for the schemes in the second category was a string of characters.

Each participant was asked to create a new password never used previously for each scheme, 4 in total, and a user ID for all the schemes. Each created password consisted of 8 characters or click- points. We also made it explicit that participants were not allowed to write down their passwords. Each password must meet the following minimum complexity requirements. A password must contain at least one letter, one digit, and one non-alphanumerical character for both Text and ClickText, and at least three different animals for Animal Grid. No repeating patterns such as “A#A#…” or “Dog, Dog,…” were allowed. For PassPoints, click-points in a password must be distinct (i.e., no click-point was inside another click-point’s tolerance range). Each password was verified immediately after creation.

The study was partitioned into two stages. Two schemes were tested in each stage. In the first stage, two schemes, one from each category, were randomly selected for a participant to test. One scheme had a string of text characters as a password while the other had a string of animals and grid cells or click-points as a password. During the study, each participant was asked to log in with the following intervals between two consecutive login tests: one hour after creation, one day, one week, and three weeks. In each test, a participant was allowed three tries to log in. If he/she failed three attempts, his/her password was considered forgotten, and no more test would be conducted with the participant for that specific scheme. In the second stage, the remaining two schemes were tested in the same way as above. In the end, each participant was required to fill a questionnaire to compare ClickText and Animal Grid with PassPoints and Text, and to compare ClickText with P +C, in terms of ease of use as a password system, taking both memorizing and entering a password into consideration.

A participant’s login time in each trial was recorded by the server. We define the login time as the duration from

the time when the server received a login request to the time when the server gave its response to the login request, which includes the time to enter user ID and password, to generate a CaRP image, and to communicate between the server and a participant's browser. For Text and P + C, a participant was asked to enter a password. If successful, the server recorded the time as the login time for Text, and then generated a Captcha challenge and sent to the user to solve. If the participant failed with the challenge, another challenge was generated and used. This process was repeated until the server received a correct answer to a challenge. Then the server recorded the time as the login time for P + C, which included the time that the participant failed to solve a challenge.

2) Experimental Results

Usability. Among all the recorded login attempts, 24.4% failed. Tests after a larger interval tended to have more failed attempts. Some participants contributed significantly more failed attempts than others. At the end of tests, 40 (100%) participants remembered their PassPoints passwords, 39 (97.5%) remembered their passwords of both ClickText and Animal Grid, and 34 (85%) remembered their Text passwords. One participant forgot the Animal Grid password at the one-hour test, and another one forgot the ClickText password at the one-week test. For Text, two participants forgot their passwords at the one-week test, and four forgot at the three-week test. PassPoints scored the best in memorability whereas Text scored the worst. This may be partially due to the fact that hotspots were allowed for PassPoints passwords, and that Text passwords had a much larger alphabet than both ClickText and Animal Grid.

Table I shows the login time averaged over the 40 participants' successful login attempts and the sample standard deviation as well as the maximum and minimum login times for each scheme. ClickText, AnimalGrid and P + C had similar average login time whereas PassPoints had a little shorter average login time. Text had a much shorter average login time than the other schemes. Each scheme had a large sample standard deviation relative to the average login time, indicating large variations of login time for each scheme, which is confirmed by the great difference between the minimum and maximum login times in each column shown in Table I. This is mainly caused by large individual differences. We did not detect obvious patterns indicating that a test with a longer interval had a larger login time than a test with a shorter interval. We did notice that some participants had a much larger login time when the preceding trial failed, but many other participants didn't follow this observation.

The passwords in our tests were used much less frequently than typical usage of a password in practice since we would like to test password memorability for each scheme. We expect improved results when a password is used more frequently. Table II shows the comparison results of different scheme for ease of use as a password system. We assign a value ranging from 1 to 5 to each category, indicating the spectrum from "much more difficult" to "much easier". ClickText has a mean value of 3.2 and a median value of 3 as compared to PassPoints, and a mean of 2.85 and a median of 2 as compared to Text. AnimalGrid has a mean of 3.325 and a median of 4 as compared to PassPoints, and a mean of 3.5 and a median of 4 as compared to Text. ClickText has a mean of 3.875 and a median of 4 as compared to P + C. Security. We also analysed the security of the passwords we collected for Text and ClickText, with a popular password-cracking tool, John the Ripper version 1.7.9 [43]. Given our sample size of 40 users, the distribution of passwords will not be as complete as a larger study with significantly more u but such an analysis can provide at least an indication of their security.

John the Ripper has three operation modes: "single crack", "wordlist", and "incremental". In the "single crack" mode, login names and other account information as well as a large set of mangling rules are used to generate password guesses. In the "wordlist" mode, a list of words and word mangling rules are used to generate password guesses. In the "incremental" mode, a brute force attack is applied, with guesses being tested at the descending order of their likelihood to be a password. In our study, the default parameters and settings were used for John the Ripper except that length of a password was set to 8 characters, and that the recommended wordlist "all.lst" from [44] was used in the "wordlist" mode. When operat-ing in both "single crack" and "wordlist" modes, John the Ripper didn't find any password for either Text or ClickText. When running in the "incremental" mode for 24 hours on a HP Compaq Elite 8100 PC with Intel Core i7-870 CPU, 8GB RAM and 64-bit Windows 7 OS, John the Ripper found two of 40 (i.e., 5.0%) passwords for Text but didn't find any password for ClickText. The small difference in the results of the "incremental" mode is probably because the unusual alphabet set in ClickText increased a user's chance to choose more random passwords in order to meet the same password complexity requirement applied to both Text and ClickText. A separate longitudinal study is needed to fully understand the distribution and security of ClickText passwords that users would select.

We conclude from the cracking results that the pass-words the participants selected for Text and ClickText were reasonably strong, which matches our expectation of the password complexity requirement described in the previous subsection.

3) Discussions

Wiedenbeck et al. [5] studied memorability of text pass-words and PassPoints with 20 participants for each scheme: their recall rate after one week was 65% and 70%, respectively. Our memorability results are higher than theirs for both text and PassPoints passwords.

The sample sizes in both [5] and our studies were too small to explain conclusively the difference in the results. However, this difference could probably be explained by the difference of the subjects in both studies. First, although randomly chosen, our participants were from the pool of interns working at Microsoft Research Asia, who were selected typically from leading universities. Second, our participants were used to strong text passwords as their Microsoft accounts were strictly enforced with strong password policies (e.g. each account must use a complex password, and a password has to be changed regularly, with the new password having to be significantly different from previously used ones). Third, our participants were much younger, with an average age of 23.4 years as compared to the average of 32.9 years in Wiedenbeck et al.'s experiment. However, we do not think this sample bias introduced an undue impact on the main goal of our experiment, namely, comparing the performance of CaRP with other authentication schemes. Nonetheless, user studies with a diverse sample pool are needed for CaRP, which are our future work.

C. Computation Load on Server

Compared with many graphical password schemes, generation of CaRP images is an extra load on the server side. We tested our implementations of ClickText and Animal Grid on the same HP Compaq Elite 8100 PC we used to run John the Ripper, as described in Section VII-B. For images of 400x400 pixels, the average speed was 10.68 images per second in generating a ClickText image with 33 characters and 0.86 images/s in generating an Animal Grid image with 10 to 12 animals. Our implementations were single-threaded without code optimization, and did not take any advantage of multi-core capability of the test machine. Much faster image generation should be viable by exploiting multi-core architecture of today's servers and by optimizing the code.

VII. BALANCE OF SECURITY AND USABILITY

Some configurations of CaRP offer acceptable usability across common device types, e.g. our usability studies used 400 × 400 images, which fit displays of smart phones, iPads, and PCs. While CaRP may take a similar time to enter a password as other graphical password schemes, it takes a longer time to enter a password than widely used text passwords. We discuss two approaches for balancing Carp's security and usability.

A. Alphabet Size

Increasing alphabet size produces a larger password space, and thus is more secure, but also leads to more complex CaRP images. When the complexity of CaRP images gets beyond a certain point, humans may need a significant amount of time to recognize the characters in a CaRP image and may get frustrated. The optimal alphabet size for a CaRP scheme such as ClickText remains an open question.It is possible to use a fixed subset of the alphabet to generate CaRP images for a user if the server receives her user ID before sending an image. In this case, the authentication server allows a user to create her password from the full alphabet. Once the password is created, the server finds a suitable subset of a reasonable size, which contains all the symbols in the password. The server stores the subset or its index for the account, and retrieves it later when the account attempts to log in to generate a CaRP image. This scheme is suitable when the alphabet must be large while some people would log in on small- screen devices for which an image using the full alphabet would be too complex to quickly identify the objects in the image.

B. Advanced Mechanisms

The CbPA-protocols described in Section II-C require a user to solve a Captcha challenge in addition to inputting a password under certain conditions. For example, the scheme described in [16] applies a Captcha challenge when the number of failed login attempts has reached a threshold for an account. A small threshold is applied for failed login attempts from unknown machines but a large threshold is applied for failed attempts from known machines on which a successful login occurred within a given time frame. This technique can be integrated into CaRP to enhance usability:

A regular CaRP image is applied when an account has reached a threshold of failed login attempts. As in [16], different thresholds are applied for logins from known and unknown machines. Otherwise an "easy" CaRP image is applied. An "easy" CaRP image may take several forms depending on the application

requirements. It can be an image generated by the underlying Captcha generator with less distortion or overlapping, a permuted "keypad" wherein undistorted visual objects (e.g. characters) are permuted, or even a regular "keypad" wherein each visual object (e.g., character) is always located at a fixed position. These different forms of "easy" CaRP images allow a system to adjust the level of difficulty to fit its needs. With such a modified CaRP, a user would always enter a password on an image for both cases listed above. No extra task is required. The only difference between the two cases is that a hard image is used in the first case whereas an easy image is used in the second case.

VIII. CONCLUSION

These have proposed CaRP, a new security primitive relying on unsolved hard AI problems. CaRP is both a Captcha and a graphical password scheme. The notion of CaRP intro-duces a new family of graphical passwords, which adopts a new approach to counter online guessing attacks: a new CaRP image, which is also a Captcha challenge, is used for every login attempt to make trials of an online guessing attack computationally independent of each other. A password of CaRP can be found only probabilistically by automatic online guessing attacks including brute-force attacks, a desired security property that other graphical password schemes lack. Hotspots in CaRP images can no longer be exploited to mount automatic online guessing attacks, an inherent vulnerability in many graphical password systems. CaRP forces adversaries to resort to significantly less efficient and much more costly human-based attacks. In addition to offering protection from online guessing attacks, CaRP is also resistant to Captcha relay attacks, and, if combined with dual-view technologies, shoulder-surfing attacks. CaRP can also help reduce spam emails sent from a Web email service. Our usability study of two CaRP schemes we have implemented is encouraging. For example, more participants considered AnimalGrid and ClickText easier to use than PassPoints and a combination of text password and Captcha. Both AnimalGrid and ClickText had better password memora-bility than the conventional text passwords. On the other hand, the usability of CaRP can be further improved by using images of different levels of difficulty based on the login history of the user and the machine used to log in. The optimal tradeoff between security and usability remains an open question for CaRP, and further studies are needed to refine CaRP for actual deployments.

Like Captcha, CaRP utilizes unsolved AI problems. However, a password is much more valuable to attackers than a free email account that Captcha is typically used to protect. Therefore there are more incentives for attackers to hack CaRP than Captcha. That is, more efforts will be attracted to the following win-win game by CaRP than ordinary Captcha: If attackers succeed, they contribute to improving AI by providing solutions to open problems such as segmenting 2D texts. Otherwise, our system stays secure, contributing to practical security. As a framework, CaRP does not rely on any specific Captcha scheme. When one Captcha scheme is broken, a new and more secure one may appear and be converted to a CaRP scheme. Overall, our work is one step forward in the paradigm of using hard AI problems for security. Of reasonable security and usability and practical applications, CaRP has good potential for refinements, which call for useful future work. More importantly, we expect CaRP to inspire new inventions of such AI based security primitives.

REFERENCES

1. Biddle, S. Chiasson, and P. C. van Oorschot, "Graphical passwords: Learning from the first twelve years," ACM Comput. Surveys, vol. 44, no. 0, 2012. (2012, Feb.). The Science Behind Passfaces [Online]. Available: http://www.realuser.com/published/ScienceBehindPassfaces.pdf

2. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin, "The design and analysis of graphical passwords," in Proc. 8th USENIX Security Symp., 1999, pp. 1–15.

3.H. Tao and C. Adams, "Pass-Go: A proposal to improve the usability of graphical passwords," Int. J. Netw. Security, vol. 7, no. 2, pp. 273–292, 2008.

S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "PassPoints: Design and longitudinal evaluation of a graphical password system," Int. J. HCI, vol. 63, pp. 102–127, Jul. 20 P. C.

4. van Oorschot and J. Thorpe, "On predictive models and user-drawn graphical passwords," ACM Trans. Inf. Syst. Security, vol. 10, no. 4, pp. 1–33, 2008. K. Golofit, "Click passwords under investigation," in Proc. ESORICS, 2007, pp.

5.A. E. Dirik, N. Memon, and J.-C. Birget, "Modeling user choice in the passpoints graphical password scheme," in Proc. Symp. Usable Privacy Security, 2007, pp. 20–28.

6.J. Thorpe and P. C. van Oorschot, "Human-seeded attacks and exploiting hot spots in graphical passwords," in Proc. USENIX Security, 2007, 03–118.

7. P. C. van Oorschot, A. Salehi-Abari, and J. Thorpe, "Purely automated attacks on passpoints-style graphical passwords," IEEE Trans. Inf. Forensics Security, vol. 5, no. 3, pp. 393–405, Sep.