International Journal of Engineering Researches and Management Studies

MULTI-LAYER SECURE AUTHENTICATION USING GRAPHICAL PASSWORD FOR ONLINE ENVIRONMENT

Jabez Willmott S.P*, R. Prabhudoss, Boopathy.P

ABSTRACT

Today many institutions have deployed CAPTCHAs to protect their services from automated attacks. In addition to CAPTCHAs for login, CAPTCHAs are also used to prevent malicious manipulation transactions by automated Man-in-the-Middle (MIM) attackers and automated test that humans can pass, but current computer programs can't pass: any program that has high success over a captcha can be used to solve an unsolved Artificial Intelligence (AI) problem.

In this project, we introduce a graphical password system with a supportive sound signature to increase the remembrance of the password is discussed. In this proposed work, a click-based graphical password scheme called Cued Click Points (CCP) is presented. In this system a password consists of sequence of some images in which user can select one click-point per image. In addition user is asked to select a sound signature corresponding to each click point this sound signature will be used to help the user in recalling the click point on an image.

INTRODUCTION

Captchas In General

A CAPTCHA [1] scheme is a challenge-response authentication protocol based on a hard AI problem, which can be easily solved by humans but not by machines. Here, the goal differs from traditional user authentication protocols: to accept humans but reject automated programs. CAPTCHAs can be designed in many forms. The most well-known and widely-deployed form is distorted texts shown as CAPTCHA images. The distortions are chosen in a way that automated programs cannot achieve a comparable recognition rate to what humans can.

Another form of CAPTCHA is based on hard AI problems on image understanding. A typical CAPTCHA of this kind is Asirra [2], which challenges the prover to select all cat pictures from 12 pictures of cats and dogs. The idea of image-based CAPTCHAs has also been generalized to be based on animation, video, and 3-D models. Readers are referred to [3] for a recent survey on CAPTCHAs. The idea of breaking CAPTCHAs has been around for a while. The first public report we know appeared in 2003 [4], in which Mori and Malik proposed recognition based attacks on Gimpy and EZ-Gimpy, two early CAPTCHA schemes based on distorted texts. Later, several other attacks were reported, showing insecurity of more CAPTCHA schemes based on distorted texts. Moreover, hocevor demonstrated results of his attacks on quite a number of CAPTCHA schemes on his web site, which reveals some common pitfalls of weak CAPTCHAs and reported an interesting finding: once a CAPTCHA image based on distorted texts is well segmented, automated programs can recognize those single characters even better than humans. This implies that making segmentation harder is the main way to enhance security of CAPTCHAs based on distorted texts. In [5], Yan and Ahmad showed that a simple pixel-count based attack can break a number of CAPTCHAs offered at Captchaservice.org and deployed by some other web sites. In [6], Yan and Ahmad proposed a new attack on some distorted texts based CAPTCHAs, which can be used to segment CAPTCHA images into single characters with high accuracy.

CAPTCHA APPLICATIONS

A captcha is a program that can generate and grade tests that: (A) most humans can pass, but (B) current computer programs can't pass. Such a program can be used to differentiate humans from computers and has many applications for practical security, including:

Online Polls. Slashdot.com released an online poll asking which the best graduate school in computer science was. As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times.

Free Email Services. Several companies (Yahoo!, Microsoft, etc.) offer free email services, most of which suffer from a specific type of attack: \bots" that sign up for thousands of email accounts every minute. This situation can be improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a captcha of our design to prevent bots from registering for accounts. Their captcha asks users to read a distorted word such as the one shown below (current computer programs are not as good as humans at reading distorted text).

Search Engine Bots. Some web sites don't want to be indexed by search engines. There is an html tag to prevent search engine bots from reading web pages, but the tag doesn't guarantee that bots won't read the pages; it only serves to say \no bots, please". Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly guarantee that bots won't enter a web site, captchas are needed.

Worms and Spam. Captchas also offer a plausible solution against email worms and spam: only accept an email if you know there is a human be- hind the other computer. A few companies, such as www.spamarrest.com are already marketing this idea.

Preventing Dictionary Attacks. Pinkas and Sander [7] have suggested using captchas to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

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 password systems integrating Captcha technology, which we call CaRP (Captcha as g
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.

A NEW SECURITY PRIMITIVE

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 log in. Typical application scenarios for CaRP include:

1) 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.

2) CaRP increases spammer’s operating cost and thus helps reduce spam emails. For an email service provider that deploys CaRP, a spam 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.

RELATED WORK

The Drawing CAPTCHA

The majority of CAPTCHA techniques are designed for use on home computers or laptops. However, increasingly people are accessing network resources using smaller mobile devices as well as regular computers. The development of a CAPTCHA user- verification mechanism suitable for mobile devices is therefore an issue which merits research attention. In particular, the capabilities of touch-screen user- interfaces in mobile devices have proven very convenient for users. People can use their fingers directly on the screen surface to handle all device operations, from dialing voice calls and browsing web- sites to playing games and manipulating graphics. In 2006, a CAPTCHA mechanism for mobile de- vices was proposed in (Shirali-Shahreza et al. 2006) which was appropriate for use on a touch-screen de- vice, named the Drawing CAPTCHA. A Drawing CAPTCHA is an image-based CAPTCHA whereby users draw appropriate lines on a screen in order to pass the CAPTCHA challenge.

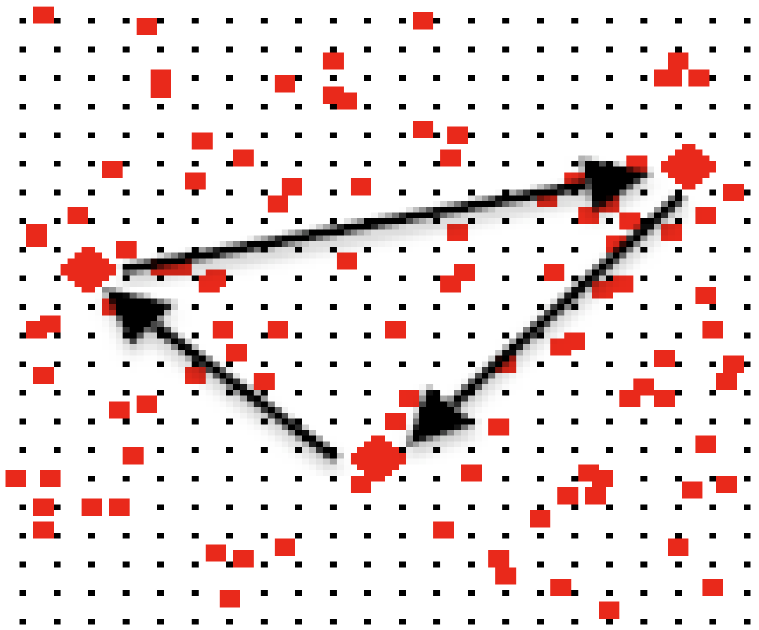

Fig 1: Example of the Drawing CAPTCHA for mobile devices. The user must connect the three red diamonds with lines, to form a triangle. A correct solution is indicated with arrowed black lines.

Fig. 1 shows an example of a Drawing CAPTCHA that is intended for use on a mobile phone. This system generates a gray-colored dotted background to begin with as an obfuscation measure against image-processing attacks. A significant number of large square dots and a small number of even larger diamond-shaped dots are then randomly drawn on the screen. The user is asked to connect the diamond- shaped dots. Therefore, the user must first find three diamond-shaped dots; and secondly, connect them to
each other by drawing lines. There is no need to connect the dots in a sequence. If a user can respond appropriately, they are considered to be human, but if not, they are considered to be potentially an abu-
sive automated program, and denied access.

CLICKABLE CAPTCHA

Another approach to clickable CAPTCHAs was recently proposed by Chow et al. [4]. The approach consists of combining several textual CAPTCHAs into a grid of clickable CAPTCHAs (e.g. a 3-by-4 grid). The solution to the grid is the determination (e.g. by clicking) of the grid elements which satisfy some given requirement. For example, the user may be asked to identify in the grid the subset of CAPTCHAs which embed English words (assuming some, but not all, do). One advantage of this approach is that it relies on existing security assumptions about text-based CAPTCHAs that have been in use for a long time and have been the object of intense scrutiny.

CAPTCHA AS GRAPHICAL PASSWORDS

CaRP: An Overview

CaRP schemes are clicked-based graphical passwords. According to the memory tasks in memorizing and entering 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 recognized objects as cues to enter a password.

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 authentication 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 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 object s, *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. 2. Advanced authentication with CaRP, for example,

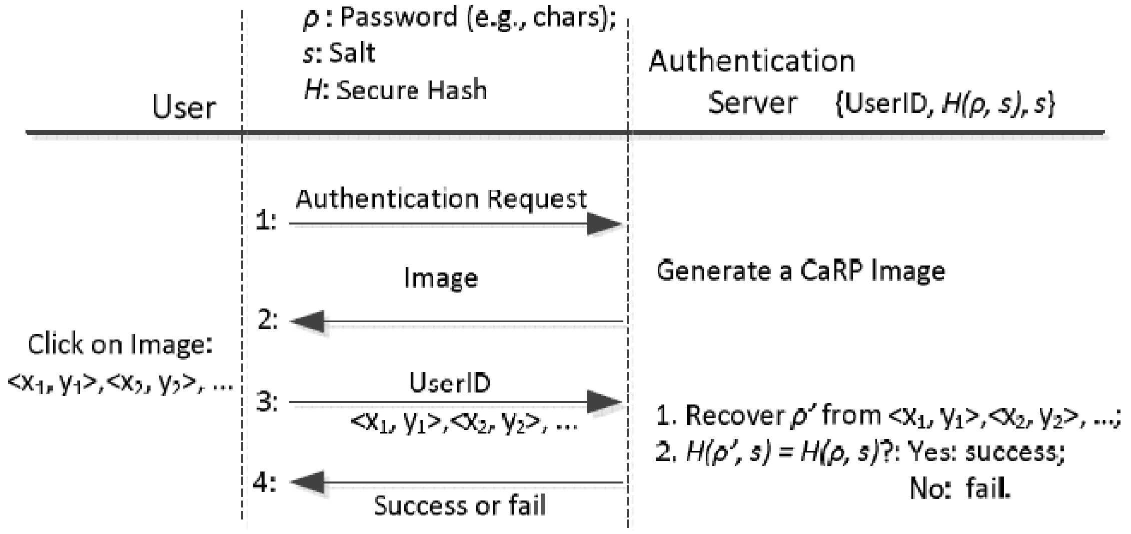
ρ : Password (e.g., chars);
s: Salt
H: Secure Hash
User
Authentication Server {UserID, H(ρ, s), s}
1: Authentication Request
Image
Generate a CaRP Image
2:
Click on Image: <x1, y1>,<x2, y2>, ...
3: UserID <x1, y1>,<x2, y2>, ...
1. Recover ρ′ from <x1, y1>,<x2, y2>, ...;
2. H(ρ′, s) = H(ρ, s)?: Yes: success; No: fail.
4: Success or fail

Fig 2: Flowchart of basic CaRP authentication.

RECOGNITION-BASED CaRP

The 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.

CLICK TEXT

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 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".

CLICK ANIMAL

ClickAnimal is a recognition-based CaRP scheme built on top of Captcha Zoo [9], 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 ClickAnimal images: 3D models are used to generate 2D animals by applying different views, textures, colors, 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. 3 shows a ClickAnimal image with an alphabet of 10 animals.

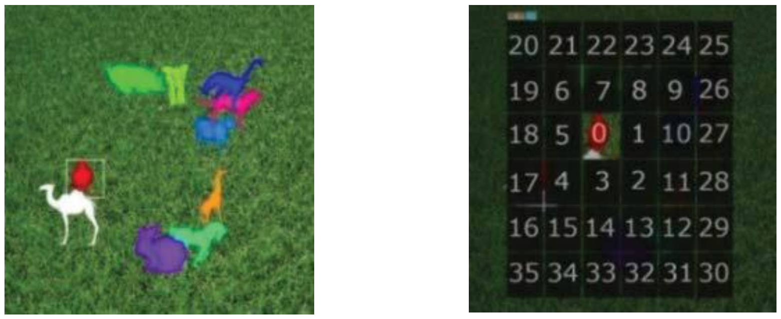

Fig 3: A Click Animal image (left) and 6 × 6 grid (right) determined by red turkey's bounding rectangle.

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.

ANIMAL GRID

AnimalGrid is a combination of ClickAnimal 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 $n \times n$ 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. 3 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.

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.

CONCLUSION

We believe that the fields of cryptography and artificial intelligence have much to contribute to one another. captchas represent a small example of this possible symbiosis. Reductions, as they are used in cryptography, can be extremely useful for the progress of algorithmic development. We encourage security researchers to create captchas based on different AI problems.
We 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 introduces a new family of graphical passwords, which adopts a new approach to counter online guessing attacks. 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.

REFERENCE

1. L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In EUROCRYPT'2003, pages 294{311.
2. J. Elson, J. R. Douceur, J. Howell, and J. Saul. Asirra:A CAPTCHA that exploits interest-aligned manual image categorization. In CCS'2007, pages 366{374.
3. A. Basso and F. Bergadano. Anti-bot strategies based on human interactive proofs. In Handbook of Information and Communication Security, chapter 15, pages 273{291. Springer, 2010.
4. G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In CVPR'2003, pages 134{141.

5. J. Yan and A. S. El Ahmad. Breaking visual CAPTCHAs with na • _ve pattern recognition algorithms. In ACSAC'2007, pages 279{291.
6. J. Yan and A. S. El Ahmad. A low-cost attack on a Microsoft CAPTCHA. In CCS'2008, pages 543{554.
7. Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. In Proceedings of the ACM Computer and Security Conference (CCS' 02), pages 161-170. ACM Press, November 2002.
8. R. Chow, P. Golle, M. Jakobsson, X. Wang and L. Wang. Making CAPTCHAs Clickable. In Proc. Of HotMobile 2008.
9. R. Lin, S.-Y. Huang, G. B. Bell, and Y.-K. Lee, “A new CAPTCHA interface design for mobile devices,” in *Proc. 12th Austral. User Inter. Conf.*, 2011, pp. 3–8.