Towards New Security Primitives Based on Hard AI Problems

Bin B. Zhu[1] and Jeff Yan[2]

[1] Microsoft Research Asia, Beijing, China
binzhu@microsoft.com
[2] School of Computing Science, Newcastle University, UK
Jeff.Yan@ncl.ac.uk

Abstract. Many security primitives are based on hard mathematical problems. Using hard AI problems for security has emerged as an exciting new paradigm (with Captcha being the most successful example). However, this paradigm has achieved just a limited success, and has been under-explored. In this paper, we motivate and sketch a new security primitive based on hard AI problems.

Keywords: Captcha as gRaphical Passwords (CaRP), passwords, cross-device authentication.

1 Thwart Password Guessing: A New Method

PassPoints [1] is a well-studied graphical password scheme, where a user clicks on an image and the ordered sequence of her click-points is used to derive a password.

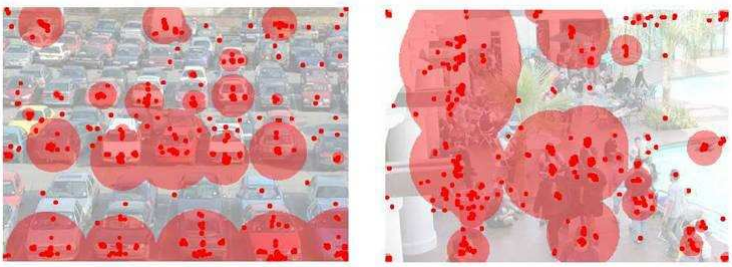

Fig. 1. Hotspots in PassPoints (taken from [2])

PassPoints has an inherent security weakness: it is easy for an attacker to automatically identify all salient points in an image using standard image processing methods. Then running through random combinations of salient points will lead to a brute force attack on passwords. To make things worse, when given

B. Christianson et al. (Eds.): Security Protocols 2013, LNCS 8263, pp. 3–10, 2013.

an image, people tend to prefer some image points (e.g. eye-catching ones) over others when creating passwords. These more popular points are 'hot spots' of the image, as shown in Figure 1, and an attacker can exploit them for an effective dictionary attack, significantly reducing the security of PassPoints [2,3].

To address the above problems, we have pondered a new approach to mitigating password guessing attacks.

Password guessing, on text or graphical passwords, online or offline, is typically a deterministic elimination process. Each guess reduces the remaining search space, and a next guess has a higher chance for success. While more and more password candidates get eliminated, the probability of a current guess being correct increases, and this probability finally approaches 1. Naturally, a classic defense is to increase the password space.

But, how about thwarting the deterministic elimination process? What if previous guesses do not contribute to reducing the password space, and thus a next guess is just like starting from scratch? Is this possible?

Salient points in PassPoints harm security but help memorability, as these points are often structural and they facilitate users to remember their click-points. It is impractical to force users to choose non-salient points, as these will be hard to remember. If we do not want to increase the image size to boost the password space, the only option remaining seems to make it hard for computers to exploit salient points. If the points a user clicks to login in a session cannot be correlated to the points she clicks in other sessions, then it is likely that a previous guess is not correlated with the next. One way of achieving this is the following: a different image is used for each session, and in each of the images, a user's password points appear in different forms, different locations, etc. This way, each automated guess will not reduce the password search space any more.

On the other hand, there must exist some invariant components in all the images used in different login attempts, otherwise users cannot use anything as passwords. We also need a password to remain the same for a user so that the authentication server can use it to verify her.

The above two requirements are similar to that of an ideal Captcha. In particular, as an established principle in Captcha design: to defeat machine learning attacks, each Captcha challenge should be computationally independent of the other [4]. If a new image is used in every login attempt and there is no computationally detectable correlation among these images, then the salient points or hotspots collected from previously used images will not help to locate the target points in the next image. As such, an adversary cannot build a dictionary with entries consistent for different login attempts to mount a dictionary attack.

The above thoughts have led to the concept of CaRP (Captcha as gRaphical Passwords), a new family of graphical passwords robust to online guessing attacks. Their relationship with Captcha also indicates how to construct CaRP schemes from various Captchas.

2 CaRP: Captcha as gRaphical Passwords

CaRP is a family of graphical password systems created with Captcha technology. Just like PassPoints, a user clicks on a CaRP image and the sequence of her clicks creates a password. However, each CaRP image is automatically generated by a Captcha generator, and thus is also a Captcha challenge. Just like a session key, a CaRP image is never reused across different sessions. Even for the same user, a new CaRP image is needed for every login attempt. To the contrary, in PassPoints a user always uses the same image to click her password, and many users use the same image for their password input, which leads to successful attacks exploiting hotspots.

Pinkas and Sander [5] introduced a protocol to protect passwords from online dictionary attack with Captchas[1]. Captcha and password are separate entities in this protocol, but are intrinsically combined in CaRP, which is *both a Captcha and a graphical password (scheme)*.

The notion of CaRP is simple but generic, and it can have multiple instantiations. Many Captcha schemes, regardless of whether they are text based or image recognition based, can be converted to a CaRP scheme. We provide a number of examples as follows.

2.1 ClickText

ClickText is a CaRP scheme built on top of text Captcha. Unlike normal text Captchas, a CaRP image should contain all the alphabet to allow a user to form any allowed password. Figure 2 shows a ClickText image with an alphabet of 33 characters. In ClickText images, characters can be arranged randomly on 2D space. This is another major difference from traditional text Captchas in which characters are typically ordered from left to right. Using ordinary text Captcha is not suitable in this context, as it is hard to arrange all the characters one dimensionally in a reasonably small space. Also, there is no order among characters in a CaRP image whereas the order is needed for characters in a normal Captcha image so that users can type them in. Therefore, we propose a new problem, 2D text segmentation, as the underlying hard AI problem for ClickText.

A ClickText password is a sequence of characters in the alphabet, e.g. $\rho =$ '$AB\#9CD87$', which is similar to a text password. To enter a password, the user clicks on the image the characters in her password in the order, 'A', 'B', '#', '9', 'C', 'D', '8', and then '7'.

When a CaRP image is generated, each character's location is tracked to produce a ground truth. The authentication server relies on the ground truth to

[1] In the PS protocol, a user is required to solve a Captcha challenge after entering her valid user name and password, unless a valid browser cookie from a previous successful login is available. If the user name and password pair is invalid, with a probability determined by a deterministic function, the user will receive a Captcha challenge to solve before being denied access to her account.

Fig. 2. A ClickText image with 33 characters

identify the characters corresponding to user-clicked points. The server does not store passwords in the clear, but their cryptographic hashes.

ClickText does not use visually-confusing characters. For example, letter 'O' and digit '0' may cause confusion in a CaRP image, and thus one of the characters should be excluded from the alphabet.

2.2 ClickAnimal

Captcha Zoo [6] is an image recognition scheme whose security relies on both object segmentation and binary object classification. It uses 3D models of two similar animals, e.g. dog and horse, to generate 2D animals with different textures, colors, lightings and poses, and then places them on a cluttered background. A user clicks all the horses in a challenge image to pass the test. Figure 3 shows a sample challenge where all the horses are circled red.

Fig. 3. A challenge in Captcha Zoo with horses circled red (taken from [6])

We can turn Captcha Zoo into a CaRP scheme, by introducing additional similar animals such as dog, horse and pig into the alphabet. In this new CaRP which we call ClickAnimal, a password is a sequence of animal names such as $\rho = `Turkey, Cat, Horse, Dog, ...'$. For each animal, one or more 3D models are built. The Captcha generation process is applied to generate ClickAnimal images,

wherein 3D models are used to generate 2D animals by applying different views, textures, colors, lightning effects, and, optionally, distortions. Different views applied in this step generate many different 2D shapes for the same animal, which, together with other anti-recognition mechanisms applied in this step, makes it hard for automatic recognition to identify the generated 2D animals. 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 part should not be occluded in order for humans to identify. Figure 4 shows a ClickAnimal image with an alphabet of 10 animals.

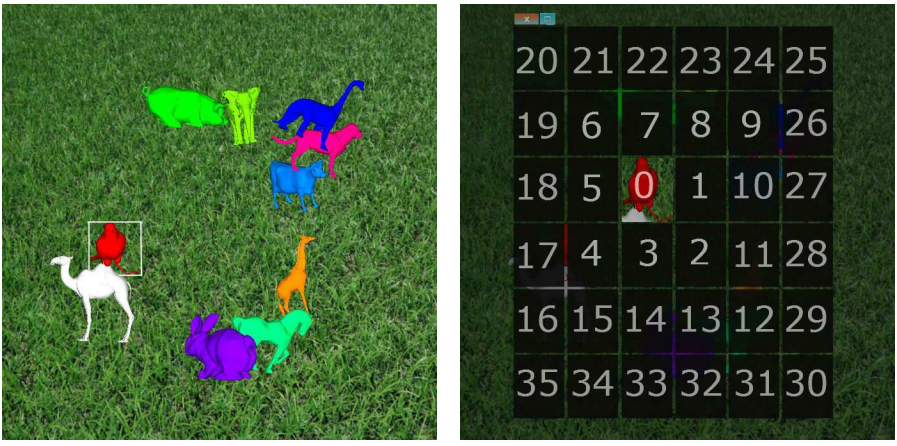

Fig. 4. A ClickAnimal image (left) and a 6×6 grid (right) determined by the red turkey's bounding rectangle

2.3 AnimalGrid

The number of similar animals is much less than the number of available text characters. ClickAnimal has a smaller alphabet, and thus it implies a smaller password space than ClickText does. CaRP should have a sufficiently-large effective password space to resist human guessing attacks. ClickAnimal's password space can be increased by combining a grid scheme as follows, leading to a new CaRP which we call AnimalGrid.

To enter a password, a ClickAnimal image is displayed first. After an animal is selected, an $n \times n$ grid appears, with the grid-cell size equaling the bounding rectangle of the selected animal. All grid cells are labeled to help a user identify them. Figure 4 shows a 6×6 grid when the red turkey in the left image was selected. A user can select zero to multiple grid-cells to form her password. Therefore a password is a sequence of animals interleaving with grid-cells, e.g. $\rho =$ '$Dog, Grid(2), Grid(1); Cat, Horse, Grid(3)$', where $Grid(1)$ means the grid-cell indexed as 1, and grid-cells following an animal means that the grid is determined by the bounding rectangle of the animal. A password must begin with an animal.

3 Application Scenarios

CaRP's typical applications include the following.

E-banking. Many e-banking systems have deployed Captchas to protect customers from automated online password attacks. For example, ICBC (`http://www.icbc.com.cn/`), the largest bank in the world, requires solving a Captcha for every login attempt. We envisage that it is faster and more convenient for people to use CaRP than the combined effort of entering a password and then solving a Captcha.

Cross-device authentication. Typing passwords is cumbersome on touch devices such as smartphones and tablets, where click/touch-based input is convenient. CaRP can offer the same password entry experience across different types of devices, including desktops, smartphones and tablets. Therefore, it is inherently a cross-device authentication mechanism, and a single implementation can simultaneously serve a wide range of different devices. On the contrary, text passwords are more friendly to desktop users, but less so to smartphone or tablet users.

Spam mitigation. CaRP can be deployed to increase a spammer's operating cost, and thus likely help reduce junk emails. For an email service that deploys CaRP, human involvement is compulsory to access an account; a spam bot cannot log into any account even if it knows the password. If CaRP is used together with a policy of throttling the number of outgoing emails allowed per login session, a spam bot will need regular human assistances, and each time it sends out only a limited number of emails. All these will reduce a spammer's productivity.

4 Security Analysis

The computational intractability of hard AI problems such as object recognition is fundamental to the security of CaRP. Existing analyses on Captcha security were mostly case by case or used an approximation approach. No theoretic security model has been established yet. Segmenting similar objects (e.g. characters) is considered as a computationally-expensive and combinatorially-hard problem [7], which modern text Captcha schemes rely on. According to [7], the complexity of object segmentation is exponentially dependent of the number of objects contained in a challenge, and polynomially dependent of the size of the Captcha alphabet. A Captcha challenge typically contains 6 to 10 characters, whereas a CaRP image typically contains 30 or more characters. Therefore, ClickText is much more secure than normal text Captcha. Furthermore, characters in a CaRP scheme are arranged two-dimensionally, which further increases segmentation difficulty due to an additional dimension to segment. ClickAnimal 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 the security of 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.

CaRP offers protection against online dictionary attacks on passwords, which have been for long time a major security threat for various online services.

Defending against online dictionary attacks is a subtler problem than it might appear. Intuitive countermeasures such as limiting the number of logon attempts do not work, for two reasons:

- They cause denial-of-service attacks (which were exploited to lock highest bidders out in final minutes of eBay auctions [8]) and incurs expensive helpdesk costs for account reactivation.
- They are vulnerable to global password attacks [5], where adversaries intend to break into any account rather than a specific one, and thus they try each password candidate on multiple accounts. This way, the number of guesses on each account is made below the threshold, thus avoiding triggering account lockout.

CaRP makes it much harder for bad guys to perform automated guess attacks. Even when a human is involved, the attack is still expensive and slowed down. CaRP also offers protection against relay attacks, which have been an increasing threat to online applications protected by Captchas. In a relay attack, Captcha challenges are relayed to humans to solve, with their answers returned.

CaRP is robust to shoulder-surfing attacks, if combined with Microsoft's dual-view technologies [9] that show two sets of completely different images simultaneously on the same LCD screen: one for private, and the other for public. When a CaRP image is displayed as private, attackers can capture a user's click-points but not the private image, but these points are useless for a next login session (where a new CaRP image will be used).

CaRP is robust to cross-site scripting attacks targeting at stealing users' graphical passwords, although other click-based graphical passwords such as PassPoints are vulnerable to such attacks.

However, a longitudinal evaluation is needed to establish the effective password space for each CaRP instantiation. CaRP is vulnerable if a client is compromised, and the image and user-clicked points can both be captured.

5 Usability

Initial user studies with several schemes proposed in Section 2 are encouraging. Still, CaRP requires a user to handle a Captcha-like challenge each time to login. This might have a usability impact, but it can be mitigated by serving CaRP images of different difficulty levels, according to an account's login history and whether a known machine is used for login.

The optimal configuration for achieving good security *and* usability remains an open question for CaRP, and further studies are needed to refine each implementation for actual deployments.

6 Summary

It is a fundamental method in computer security to create cryptographic primitives based on hard mathematical problems that are computationally intractable.

Using hard AI problems for security, initially proposed in [10], is an exciting new paradigm. Under this new paradigm, the most notable primitive invented is Captcha. However, the new paradigm has achieved just a limited success, if compared with the number of cryptographic primitives based on hard math problems and the wide applications of such primitives. We have showed that it is indeed possible to construct new security primitives based on hard AI problems.

Like Captcha, CaRP utilizes unsolved AI problems. However, a password is much more valuable for attackers than a free email account that Captcha typically protects. Therefore there are probably more incentives for the attackers to hack CaRP than Captcha. That is, CaRP can attract more efforts than ordinary Captcha does to the following win-win game: if the attackers succeed, they contribute to improving AI by providing solutions to open problems. Otherwise, our system stays secure, contributing to practical security.

Overall, CaRP appears to be a step forward in the paradigm of using hard AI problems for security. What else can be invented this way? We expect CaRP to inspire new inventions of AI based security primitives.

Acknowledgements. We thank Peter Ryan for very helpful discussions, and thank Tim Barclay for proofreading our camera-ready version, which improved the writing quality of this paper.

References

1. Wiedenbeck, S., Waters, J., Birget, J.C., Brodskiy, A., Memon, N.: PassPoints: design and longitudinal evaluation of a graphical password system. Int. J of HCI 63, 102–127 (2005)
2. Thorpe, J., van Oorschot, P.C.: Human-seeded attacks and exploiting hot spots in graphical passwords. USENIX Security (2007)
3. Dirik, A.E., Memon, N., Birget, J.-C.: Modeling user choice in the PassPoints graphical password scheme. ACM SOUPS (2007)
4. Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition CAPTCHAs. ACM CCS, 187–200 (2010)
5. Pinkas, B., Sander, T.: Securing passwords against dictionary attacks. ACM CCS, 161–170 (2002)
6. Lin, R., Huang, S.-Y., Bell, G.B., Lee, Y.-K.: A new Captcha interface design for mobile devices. In: Australasian User Interface Conference (2011)
7. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In: Baird, H.S., Lopresti, D.P. (eds.) HIP 2005. LNCS, vol. 3517, pp. 1–26. Springer, Heidelberg (2005)
8. Wolverton, T.: Hackers attack eBay accounts. ZDNet (March 26, 2002), `http://www.zdnet.co.uk/news/networking/2002/03/26/hackers-attack-ebay-accounts-2107350/`
9. Kim, S., Cao, X., Zhang, H., Tan, D.: Enabling concurrent dual views on common LCD screens. In: Sig. CHI 2012, pp. 2175–2184 (2012)
10. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)