# Captcha as Graphical Passwords

Kalyani S Kumar

*Assistant Professor,Dept of Information Science & Engineering,*
GSSSIETW, Mysore

*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. A new security primitive based on hard AI problems, namely,a novel family of graphical password systems built on top of Captcha technology, which is called 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-surfi 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 Pass Points, 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**

*Keywords*-**Graphical password, password, hotspots, CaRP, Captcha, dictionary attack, password guessing attack, securityprimitive.**

## I. INTRODUCTION TO CAPTCHA

The early days of Internet and web usage, people, business and companies are facing a disastrous threat that comes out of leakage of sensitive data and information. Also, blocking of the network is one of the possible ways to harm a server, storage or service. The progress of Internet, Web Security has become an important issue. To deal with such problems, John Langford, Nicholas J. Hooper and Luis Von Ahn proposed techniques known as CAPTCHA. CAPTCHA is an acronym for "Completely Automated Public Turning Test to tell Computers and Humans Apart". Typically, a Turing test is performed by a human on a machine. Whilst in CAPTCHA schemes, the test is performed by computers to identify the users of the system, particularly, to test whether the user is a Human or not.

The test is for the humans, that is, it is solvable only by humans and not by machine, system or program. The user is required to provide a correct response to the test and then the user is permitted to access the work. When a correct response is received; it is presumed that the response arrived because of a Human user. The three main aspects or features of a good CAPTCHA technique are: humans to solve, Easy to generate automatically, Almost impossible or difficult for any other computer program, to solve. A good CAPTCHA also has two important aspects: Security and Usability. The security dimension determines its strength for preventing the variant attacks, while usability the necessity of "user friendly" when CAPTCHA is deployed

CAPTCHAs are by definition fully automated, requiring little human maintenance or intervention to administer. This has obvious benefits in cost and reliability.By definition, the algorithm used to create the CAPTCHA must be made public, though it may be covered by a patent. This is done to demonstrate that breaking it requires the solution to a difficult problem in the field of artificial intelligence (AI) rather than just the discovery of the (secret) algorithm, which could be obtained through reverse engineering or other means.

Modern text-based CAPTCHAS are designed such that they require the simultaneous use of three separate abilities invariant recognition, segmentation, and parsing to correctly complete the task with any consistency.

Invariant recognition refers to the ability to recognize the large amount of variation in the shapes of letters. There are nearly an infinite number of versions for each character that a human brain can successfully identify. The same is not true for a computer, and teaching it to recognize all those differing formations is an extremely challenging task.

Segmentation, or the ability to separate one letter from another, is also made difficult in CAPTCHAs, as characters are crowded together with no white space in between.

Context is also critical. The CAPTCHA must be understood holistically to correctly identify each character. For example, in one segment of a CAPTCHA, a letter might look like an "m." Only when the whole word is taken into context does it become clear that it is a "u" and an "n."

Each of these problems pose a significant challenge for a computer, even in isolation. The presence of all three at the same time is what makes CAPTCHAs difficult to solve.

Unlike computers, humans excel at this type of task. While segmentation and recognition are two separate processes necessary for understanding an image for a computer, they are part of the same process for a person. For example, when an individual understands that the first letter of a CAPTCHA is an "a", that individual also understands where the contours of that "a" are, and also where it melds with the contours of the next letter. Additionally, the human brain is capable of dynamic thinking based upon context. It is able to keep multiple explanations alive and then pick the one that is the best explanation for the whole input based upon contextual clues. This also means it will not be fooled by variations in letters.

## TYPES OF CAPTCHA

CAPTHCAs means presenting a challenge response test to the users or humans. They are classified based on what is distorted that is whether characters, digits, or images. Some types of CAPTCHAs are given below

## 1. Text CAPTCHA

CAPTCHA based on text is easy to implement and requires a large question bank. Also contain number of classes of characters and digits so that problem occurs for user to identify the correct character and digit. User identifies the correct character and provides the text in the space provided o the form. There are various forms of text based CAPTCHA.

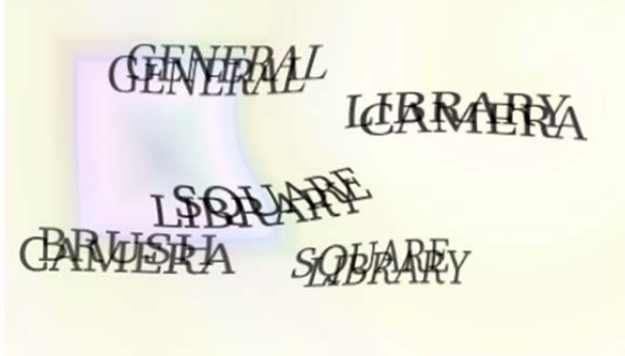**Gimpy**: It is reliable text-based CAPTCHA, developed by Yahoo and Carnegie, Mellon University.



Fig.1 Example of Gimpy-Text

**Ez-Gimpy**:-Simplified version of Gimpy that is developed by Henry Baired, which is used by Yahoo messenger in case of their signup page.



Fig.2 Example of Ez- Gimpy-Text

**Baffle-Text**:-It is developed by Henry Baired at California University at Berkeley. It is modified version of Gimpy.



Fig.3 Example of Baffle-Text

**MSN-CAPTCHA**:-This type of CAPTCHA contains 8 characters (Uppercase) and digits are used and background color is gray, foreground is dark blue.



Fig. 4 Example of MSN-CAPTCHA

## 2. Image CAPTCHA

Image or graphical CAPTCHA is a technique which offers challenge text in which the human user has ability to identify the hidden meaning in the image which is displayed. Various implementation of graphical based Captcha are as follows;

**Bongo**:-It is developed by Mikhail M. Bongard who is pattern recognition expert. Visual pattern recognition problem is solved by human. In Bongo, two series of block i.e left and right, and both have different characteristics from each other.
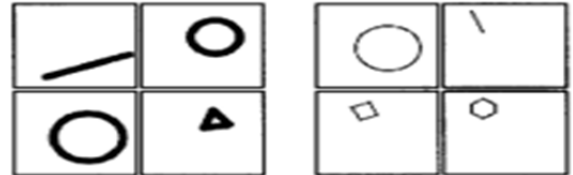


Fig.5 Example of bongo

**PIX**: It include large database of photographic and animated images, which are related to daily needs. There are the images given and user has to recognized the image which is similar to original one. For example - pick the common features among the following 3 pictures =" AEROPLANE"



Fig.6 Example of Pix

## 3. Audio CAPTCHA

This type of CAPTCHA based on sound-based system. Instead of showing text or image, a sound is played which user must recognized and type the word. It contains downloadable audio-clips. ECO is the first sound based system implemented by Nancy Chan of the City University in Hong Kong.
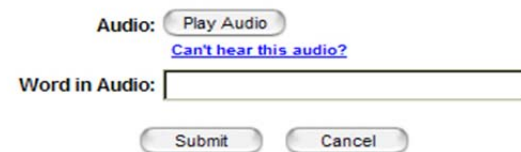


Fig.7 Example of Audio CAPTCHA

## 4. Video CAPTCHA

This type of CAPTCHA is newer technique and more secure than exiting CAPTCHA. A video taken from database that has words that describe video and user has to match word with video and submit even before completing the video.
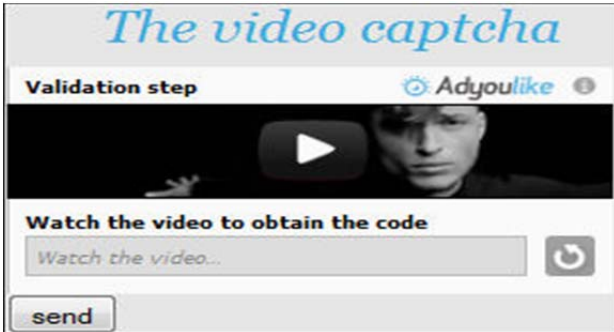


Fig.8 Example of video CAPTCHA

## 5. Puzzle CAPTCHA

In puzzle based CAPTCHA a picture or image is divided into segment. And user has to arrange this segment to form complete image in correct format.
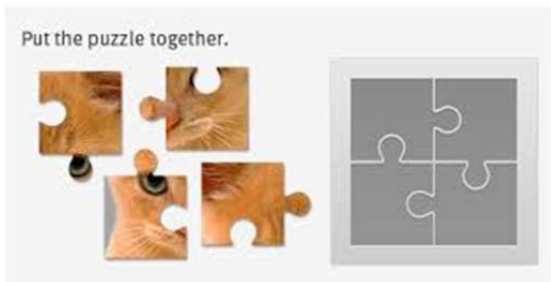


Fig.9 Example of  Puzzle CAPTCHA

This document is a template.  An electronic copy can be downloaded from the conference website.  For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website.

## II. CARP

A fundamental 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, is an exciting new paradigm. 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 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. An 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. Defense 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 ) and incurs expensive helpdesk costs for account reactivation.

It is vulnerable to global password attacks 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 increas-ing threat to bypass Captchas protection, wherein Captcha challenges are relayed to humans to solve. Koobface  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 technologies.

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:**

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 . 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 trafficThe entire document should be in Times New Roman or Times font. Type 3 fonts must not be used. Other font types may be used if needed for special purposes

### III. AN OVERVIEW OF GRAPHICAL PASSWORDS
**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.

A recognition-based scheme requires identifying among decoys the visual objects belonging to a password portfolio. A typical scheme is Passfaces where in 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 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 is also similar but uses a large set of computer-generated "random-art" images. Cognitive Authentication 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 end

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) 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 improves DAS's usability by encoding the grid intersection points rather than the grid cells. BDAS 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 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) is similar to PassPoints but uses one image per click, with the next image selected by a deterministic function. Persuasive Cued Click Points (PCCP) 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. 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. A study reported that a significant portion of passwords of DAS and Pass-Go 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 i.e., spots likely selected in creating passwords. Hotspots were exploited to mount successful guessing attacks on PassPoints : a significant portion of passwords were broken with dictionaries of 226 to 235 entries, as compared to the full space of 243 passwords.

**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 The following principle has been established: text Captcha should rely on the difficulty of character segmenta-tion, which is computationally expensive and combinatorially hard

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 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. IRCs based on binary object classification or identification of one concrete type of objects are likely insecure. 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.

**Captcha in Authentication**

It was introduced 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 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 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 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 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.

### Other Related Work

Captcha is used to protect sensitive user inputs on an untrusted client.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.
.

### IV. CAPTCHA AS GRAPHICAL PASSWORDS
### 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 findingthe password. Mathematically, let S be the set of password guesses before any trial, $\rho$ be the password to find, T denote a trial whereas Tn denote the n-th trial, and $p(T = \rho)$ be the probability that $\rho$ 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\En-1,i.e., the relative complement of En-1 in S. If $\rho \in S$, then we

have
$$p (T = \rho | T1 \_= \rho, \ldots, Tn-1 \_= \rho) > p(T = \rho),$$
$$(1)$$
and
$$En \to S$$
$$p(T = \rho | T1 \_= \rho, \ldots, Tn-1 \_= \rho) \to 1$$
$$\_$$
with $n \to |S|,$
$$(2)$$

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, the distinction 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:
$$p(T = \rho | T1, \ldots, Tn-1) = p(T = \rho), \forall n$$
$$(3)$$

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 . By examining the ecosystem of user authentication, 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 identify,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 leading to creation of CaRP, a new family of graphical passwords robust to online guessing attacks.

### 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.

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. 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.

**Converting Captcha to CaRP**

In principle, any visual Captcha scheme relying on recognizing 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.

Some IRCs rely on identifying objects whose types are not predefined. A typical example is Cortcha 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.

**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 $H(\rho, s)$ for each user ID, where $\rho$ 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 objects, $\rho\_$, that the user clicked on the image. Then AS retrieves salt s of the account, calculates the hash value of $\rho\_$ 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.
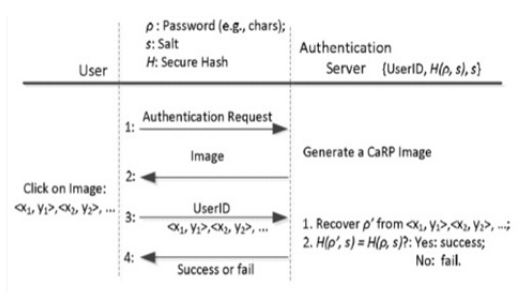


Fig 10: Flowchart of basic CaRP authentication

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.

## V. 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.

**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., $\rho$ ="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. 11 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 $\rho$ = "AB#9CD87".



Fig 11:Click-Text image with 33 characters

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. 11 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 $\rho$ = "AB#9CD87".

Fig 12:Captcha Zoo with horses circled red.



Fig 13: A ClickAnimal image      Fig 14:6 × 6 grid

### ClickAnimal

Captcha Zoo  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. 12 shows a sample challenge wherein all the horses are circled red. ClickAnimal is a recognition-based CaRP scheme built on top of Captcha Zoo, 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. 13 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

### AnimalGrid

The number of similar animals is much less than the number of available characters. ClickAnimal has a smaller alphabet,and thus a smaller password space, than Clickext. CaRP should have a sufficiently-large effective password space to resist human guessing attacks. AnimalGrid'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 is a candidate but requires drawing on the grid.

To be consistent with ClickAnimal, we change from drawing to clicking: Click-A-Secret (CAS) wherein a user clicks the grid cells in her password. 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 × 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. 14 shows a 6 × 6 grid when the red turkey in the left image of Fig. 14 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. 14. 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 grid cells following the first animals in her password, and then gets back to the ClickAnimal 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 pointse.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

## VI. 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 clickvariation of 3 pixels or less. TextPoint, a recognition recall CaRP scheme with an alphabet of characters, is presented next, followed by a variation for challenge response authentication.
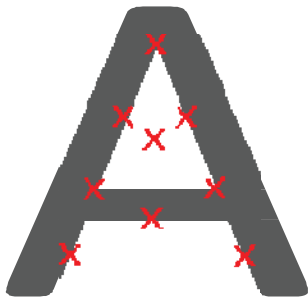
### TextPoints



Fig 15: A invariant points in A

Characters contain invariant points. Fig. 15 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,the selection 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 although 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. 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:

1. 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.

2. 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 TextPoints.

**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. TextPoints can be modified to fit challenge-response authentication. This variation is called TextPoints for Challenge-Response or TextPoints4CR. Unlike TextPoints 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 TextPoints 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 $\tau$ , $\mu \geqslant 4\tau$ .

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 clickable point at least $\tau$ 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 distancealong 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 , widely used in generating text Captcha challenges, is then used to warp the image. The result is a TextPoint4CR image wherein every clickable point would tolerate at least $\tau$ of click errors. Selection of target points should try to reduce warping distortion caused by mapping Authentication. In entering a password, a user-clicked point is replaced by the grid-cell it lies in. If click errors are within $\tau$ , 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, TextPoints4CR requires the authentication server to store passwords 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.

## VII. SECURITY ANALYSIS
### 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 which modern text Captcha schemes rely on. According to, the complexity 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 = αM P(N), where α > 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 α30P(N)/(α10P(N)) = α20 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. 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 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 O again.

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.

**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:

1. 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.

2. 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 internal object-point $\alpha$ on one image A that is non-negligibly dependent of an internal object-point $\beta$ 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 $\alpha$. In the testing phase, point $\beta$ on image B is used to query the oracle. Since point $\alpha$ is non-negligibly dependent of point $\beta$, 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 dictionary would always succeed in attacking existing graphical passwords.

**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 $33^8 \approx 2^{40}$ for ClickText with an alphabet of 33 characters, $10^8 \approx 2^{26}$ for ClickAnimal with an alphabet of 10 animals, and $10 \times 46^7 \approx 2^{42}$ for AnimalGrid with the setting as ClickAnimal plus $6 \times 6$ 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 $0.5 \cdot 33^8 \cdot 30/ (3600 \cdot 8 \cdot 1000 \cdot 365) \approx 2007$ years to break a ClickText password, $0.5 \cdot 10^8 \cdot 30/(3600 \cdot 8 \cdot 1000) \approx 52$ days to break a ClickAnimal password, or $0.5 \cdot 10 \cdot 46^7 \cdot 30/(3600 \cdot 8 \cdot 1000 \cdot 365) \approx 6219$ years to break an AnimalGrid password.

Human guessing attacks on TextPoints require a much longer time than those on ClickText since TextPoints 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 id for text passwords. A recent study on text passwords 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.

**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 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, it can make a rough estimation of the cost. Its assumeed that the cost to click one password on a CaRP image is the same as solving a Captcha challenge. Using the lowest retail price, $1, reported [34] to solve 1000 Captcha challenges, the average cost to break a 26-bit password is $0.5 \cdot 2^{26} \cdot 1/1000$, or about 33.6 thousand US dollars.

**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

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 shouldersurfer, 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 shouldersurfing attacks.

### 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.

### VIII. CONCLUSION & FUTURE ENHANCEMENT

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: 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.

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, CaRP to inspire new inventions of such AI based security primitives.

#### REFERENCES

[1] R. Biddle, S. Chiasson, and P. C. van Oorschot, "Graphical passwords: Learning from the first twelve years," ACM Comput. Surveys, vol. 44, no. 4, 2012.

[2] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: CAPTCHAs—Understanding CAPTCHA-Solving Services in an Economic Context," in Proc. USENIX Security, 2010, pp. 435–452

[3] H. Gao, X. Liu, S.Wang, and R. Dai, "A new graphical password scheme against spyware by using CAPTCHA," in Proc. Symp. Usable Privacy Security, 2009, pp. 760–767.

[4] 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.