IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727
PP 21-28
www.iosrjournals.org

Providing Security With Captcha As Graphical Passwords

Neelashree Valanju[1], Kirti Rasal[2], Ambika Nair[3], Anuja Raut[4], S. R. Kakade[5], S. S. Mirajkar[6]

[1](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)
[2](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)
[3](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)
[4](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)
[5](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)
[6](Department of Computer Engineering, Savitribai Phule Pune University, Pune, India)

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

Keywords: *CaRP, Captcha, dictionary attack, Graphical password, hotspots, password, password guessing attack, security primitive.*

I. Introduction

1.1. Graphical passwords: A graphical password in an authentication system that works by having the user select from images, in a specific order, presented in a graphical user interface (GUI). For this reason, the graphical-password approach is sometimes called graphical user authentication (GUA). They are easy to remember and hard to guess thus increasing the security. They can be classified into three categories according 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 [2] wherein a user selects a portfolio of faces from a database in creating a password.

A recall-based scheme requires a user to regenerate the same interaction result without cueing. Draw-A-Secret (DAS) [3] was the first recall-based scheme proposed.

In a cued-recall scheme, an external cue is provided to help memorize and enter a password. PassPoints [5] is a widely studied click-based cued-recall scheme wherein a user clicks a sequence of points anywhere on an image in creating a password, and re-clicks the same sequence during authentication.

1.2. Captcha: CAPTCHA (An acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge response test used in computing to determine whether or not the user is human. It is a program that protects websites against bots by generating and grading tests that humans can pass but computer programs cannot. 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.

II. System Architecture

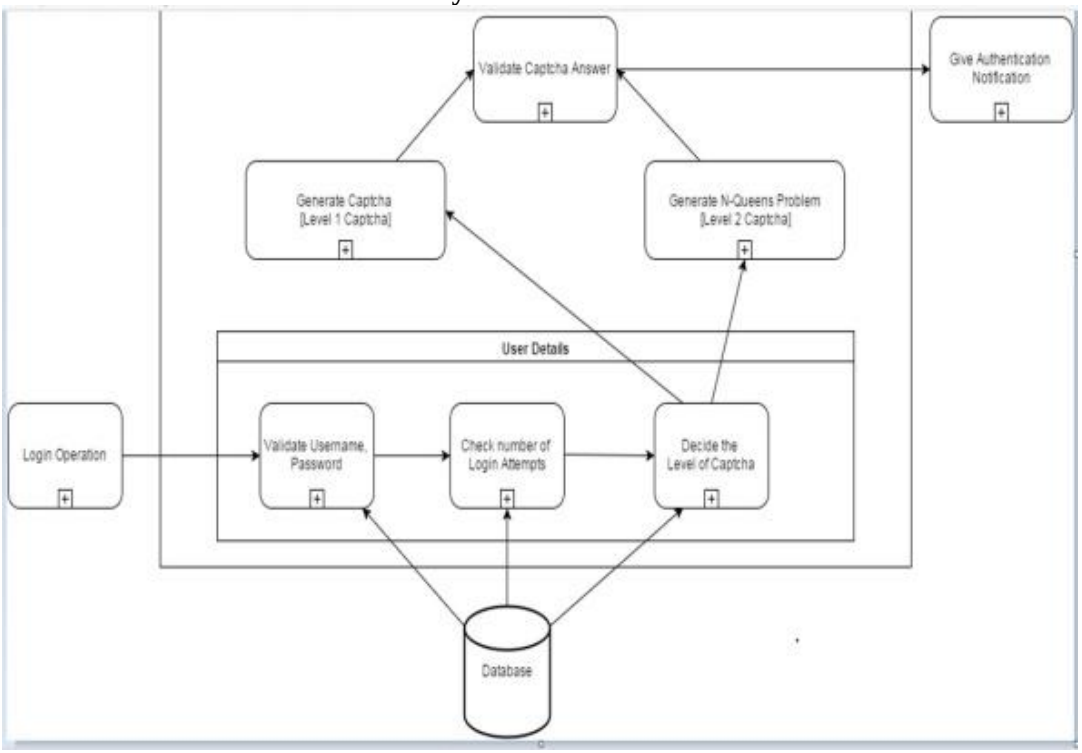

Fig 1. System Architecture

1. In this architecture the main components are User, Trusted Manager and Server.
2. User should have project Software application installed in his/her machine.
3. User sends request to the server to Register and login.
4. User uploads his/her data.
5. Server verifies the Username and stores it in the database.
6. Server sends an authentication to the user.
7. Server sends a CaRP image for verification to a Trusted Manager and the verified image is sent back to the server.

Modules:
Module 1: Register New User: A new user needs to enter all the required details to get registered to our application. Registration includes all personal details of user.
Module 2: Login: User need to enter username and password, to verify whether a user is not a robot, system will show a captcha to the user. If the captcha is correctly solved the user will get authentication. Further, on the basis of number of attempts, the difficulty level of captcha will be increased.
Module 3: Authentication:

- Display Captcha
- Validate Captcha
- Check Number of Attempts
- Increase difficulty level of Captcha.

III. Proposed Scheme

Hard AI problems: Hard AI problems are the AI problems which cannot be solved by computers but can be easily solved by humans. The problems do not have any particular algorithm or solution to be solved. Thus no logic can be developed for any hard AI problems to be solved by the systems.

We define hard in terms of the consensus of a community: an AI problem is said to be hard if the people working on it agree that it's hard. This notion should not be surprising to cryptographers: the security of

most modern cryptosystems is based on assumptions agreed upon by the community (e.g., we assume that 1024-bit integers can't be factored). The concept of a hard AI problem as a foundational assumption, of course, is more questionable than P not equal to NP, since many people in the AI community agree that all hard AI problems are eventually going to be solved. However, hard AI problems may be a more reasonable assumption than the hardness of factoring, given the possibility of constructing a quantum computer. Moreover, even if factoring is shown to be hard in an asymptotic sense, picking a concrete value for the security parameter usually means making an assumption about current factoring algorithms: we only assume that current factoring algorithms that run in current computers can't factor 1024-bit integers. In the same way that AI researchers believe that all AI problems will be solved eventually, we believe that at some point we will have the computational power and algorithmic ability to factor 1024-bit integers. (Shamir and Tromer [13], for instance, have proposed a machine that could factor 1024-bit integers; the machine would cost about ten million dollars in materials.)

We introduce captcha as an automated test that humans can pass, but current computer programs can't pass: any program that has elevated success over captcha can be used to solve an unsolved Artificial Intelligence (AI) problem. We provide numerous novel constructions of captchas. Captcha's have many applications in practical security; our approach introduces a new class of hard problems that can be exploited for security purposes. . Much like research in cryptography has made a positive impact on algorithms for factoring and discrete log, we hope that the use of hard AI problems for security purposes allows us to advance Artificial Intelligence. Two families of AI problems are introduced. They can be used to construct captchas and we show that solutions to such problems can be used for steganographic communication.

3.1. Grid-based Questionnaire:

1. The grid will be of size 4*4 or 5*5.
2. Random images will be added to the grid based on random image generation algorithm.
3. After all the images are added the grid will be displayed to the user.
4. Then a question whose answer lies in the grid itself will be given to the user.
5. The user has to answer the question correctly.
6. If the answer is correct and the ID and password of the user is correct then the user will be given access to his account.
7. Else another captcha with increased difficulty level will appear and the user has to solve it to gain access.

Example is given below with the grid containing images of animals and the user has to answer a particular question asked based on the grid.

Fig 2. Grid-Based Questionnaire

3.2. Random image generation algorithm:

You can use the following algorithm for generating random images.

1. Create an array with each item in the array being the name of an image.
2. Declare a variable which will take the value of a randomly generated number from 0 (because arrays start at 0) to the length of the array of the images - 1. So in an array with 5 images, the variable will take the value of a randomly generated number between 0 and 4.
3. Display the image from the images array at the index that matches the random number. So if the random number generated is 2, we will display the image at index 2. If the random number generated is 4, we will display the image at index 4, and so on.
4. This simple JavaScript game allows you to evolve a small image simply by accepting or rejecting random images, allowing them to "reproduce" into a new generation, and then winnowing down that generation to the "fittest" (ie, closest to your desired outcome) individuals. In other words, without drawing, you can--simply by accepting and rejecting images--create an image that you imagine.

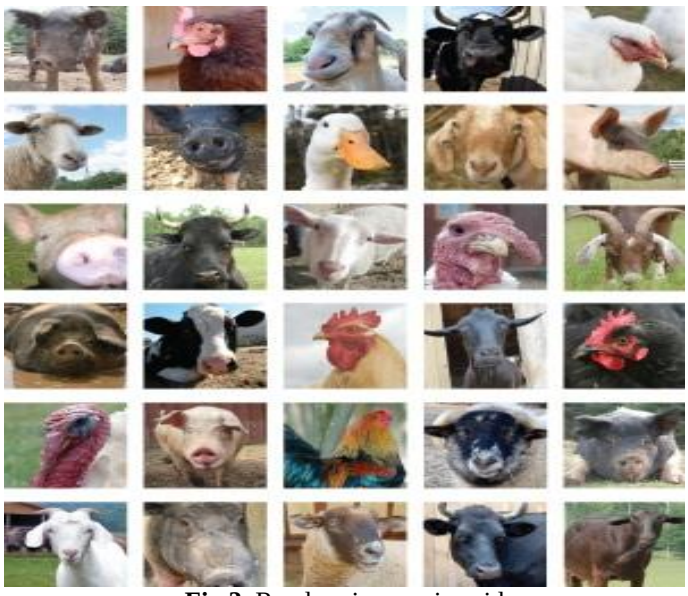

Fig 3. Random images in grid

3.3. N Queen's Technique:

1. Initially few queens will be placed on the chessboard.
2. The user has to place the remaining queens such that no queens attack each other.
3. Start with one queen at the first column first row.
4. Continue with second queen from the second column first row.
5. Go up until find a permissible situation.
6. Continue with next queen.
7. When all the queens are placed correctly and the ID and password of the user is correct then the user will be given access to his account.
8. Else another captcha with increased difficulty level will appear and the user has to solve it to gain access.

Example of a 15 queen's problem with solution is given below.

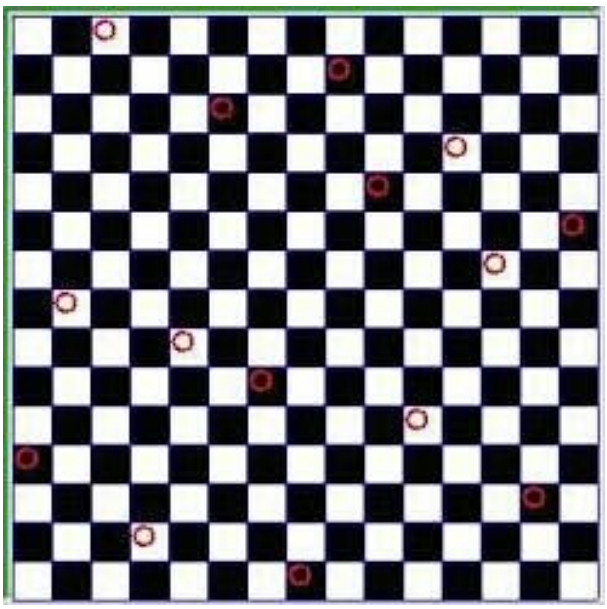

Fig 4. N Queens Problem

3.4. Image Orientation:

1. Set the default angle of image in the database to 0 degree.
2. While displaying the image as captcha, the image will be randomly inclined (by the system) to any angle, for example 30, 60, 90.
3. Two buttons are provided to rotate the image, so that the user can set the image to 0 degree.
4. For example, an image is randomly selected by the system from the database.
5. This image is displayed on the screen inclined to a particular angle.
6. Now the user has to adjust the image back to 0 degree so as to match the image in the database.
7. He can do this adjustment by clicking on the two buttons provided. These buttons are given an angle for orientation (from + to -).
8. Now suppose the image is inclined to an angle 30 degrees and the user brings it back to 0 degree by clicking the buttons provided which has been given some angle for orientation (for e.g. -5 to +5 degree).
9. If the user clicks the +5 button '6' times then the image will be set to 0 degree. That is: - Orientation=30 degree Clicks= 6 of '+5' degree.
10. Formula: Orientation - (Clicks * degree value for each click).
11. Once the image matches the default image of the database, the captcha is solved.

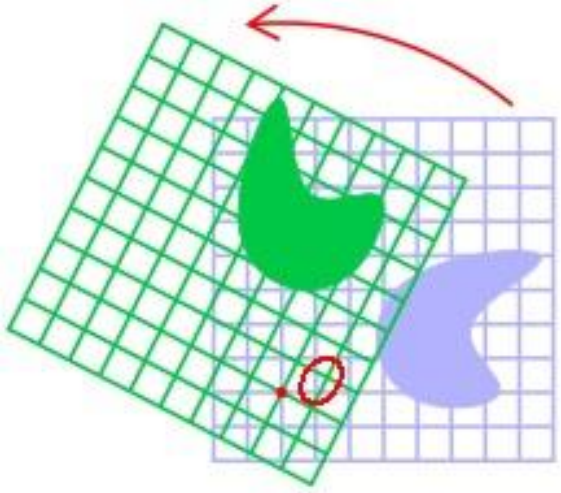

Fig 5. Image orientation

3.5. Sudoku:

1. A Sudoku puzzle is defined as logic based, number placement puzzle. The objective is to fill a 9*9 grid with digits in such a way that each column, each row, and each of the nine 3*3 grids that make up the larger 9*9 grid contains all of the digits from 1 to 9.
2. Each Sudoku puzzle begins with some cells filled in.
3. The user uses these seed numbers as a launching point towards finding the unique solution.
4. It is important to stress the fact that no number from 1 to 9 can be repeated in any row or column.
5. Each row, column, and nonet can contain each number (typically 1-9) exactly once.
6. The sum of all numbers in any nonet, row, or column must match the small number printed in its corner.
7. For traditional Sudoku puzzles featuring the numbers 1 to 9, this sum is equal to 45. This is an important point to review as it isn't uncommon for inexperienced players to get frustrated and leave the game

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Fig 6. Sudoku

IV. Related Work

Security primitives are based on hard mathematical problems. Using hard AI problems for security is emerging as an exciting new paradigm, but has been underexplored. A Fundamental task in security is to create cryptographic primitives based on hard mathematical problems that are computationally intractable. This paradigm has achieved just a limited success as compared with the cryptographic primitives based on hard math problems and their wide applications.

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. As a framework of graphical passwords, CaRP does not rely on any specific Captcha scheme. If one Captcha scheme gets broken, a new robust Captcha scheme can be used to construct a new CaRP scheme. CaRP increases spammer's operating system cost and thus helps reduce spam emails.

Within the scope of this paper, we are going to implement 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. It offers reasonable security and usability and appears to fit well with some practical applications for improving online security. We will also introduce OTP and G-mail confirmation, which is a unique combination and has never been used together before. This will enhance the level of security in our system.

V. Mathematical Model

S={*s, e, X, Y, f, DD, NDD, success, failure*}.
Where,
s: Initial/Start State
e: End state/Final State

X: Set of input
Y: Set of output
f: functions
DD: Deterministic Data
NDD: Non-Deterministic Data
Success: Desired output is generated
Failure: Desired output is not generated

- Start states: The Start State of the System where registration of the user is done.

Users: {UR: Set of registered users, UN: Set of unregistered users}

- Input Set Details:

1. Phase1: During Registration

Input= {User1 Registration Details, User2 Registration Details…..Usern Registration Details}
User Registration Details D= {Firstname, Lastname, Username, Password, Email, Contact}

2. Phase2: During Login

Input = {Login Credentials}
Login Credentials L= {Username, Password}
Username U = {U1, U2….Un}
Password P = {P1, P2….Pn}

- Output Set Details:

1. Phase 1 : During Registration

Output = {Account Creation Confirmation}

2. Phase 2 : During Login Output = {Authentication Confirmation}

- Function Set Details :

1. One system will give authentication to many users. Hence one to many relationship is observed here.
2. One system will have many captchas stored in the database. Hence one to many relationship is observed here.
3. One captcha will have one correct answer. Hence one to one relationship will be observed here.

- Success: If user is able to solve the captcha correctly he will be given authentication.
- Failure: If the solved captcha is incorrect the user will be denied access to the account and after 6 attempts the system will be locked.

VI. Conclusion

We have proposed CaRP, a new security primitive relying on unsolved hard AI problems. CaRP is both a Captcha and a graphical password scheme. 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 shoulder-surfing attacks. CaRP can also help reduce spam emails sent from a Web email service.

In this paper we have two captcha schemes namely image orientation and grid-based questionnaire. These schemes help us to distinguish between a system and a human. Further more if the system detects that the person using the captcha is human and if he is an unregistered user the system will restrict the access by giving a n queen's problem or a Sudoku problem to solve. If the wrong ID and passwords continue the complexity of the problems will be increased and after six attempts the system will be locked. Thus we propose a graphical password authentication mechanism to prevent bots from gaining access to the system. The n queen's and Sudoku problems are used to increase the irritation level of the user who are unauthorized. Therefore, the system can be protected by using captcha as graphical passwords.

References

[1]. [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]. [2] (2012, Feb.). The Science Behind Passfaces [Online].Available: http://www.realuser.com/published/ScienceBehindPassfaces.pdf
[3]. [3] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin, "The design and analysis of graphical passwords," in Proc. 8th USENIX Security Symp., 1999, pp. 1–15.
[4]. [4] H. Tao and C. Adams, "Pass-Go: A proposal to improve the usability of graphical passwords," Int. J. Netw. Security, vol. 7, no. 2, pp. 273–292, 2008.
[5]. [5] S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "PassPoints: Design and longitudinal evaluation of a graphical password system," Int. J. HCI, vol. 63, pp. 102–127, Jul. 2005.
[6]. [6] P. C. van Oorschot and J. Thorpe, "On predictive models and userdrawn graphical passwords," ACM Trans. Inf. Syst. Security, vol. 10, no. 4, pp. 1–33, 2008.
[7]. [7] K. Golofit, "Click passwords under investigation," in Proc. ESORICS, 2007, pp. 343–358.
[8]. [8] A. E. Dirik, N. Memon, and J.-C. Birget, "Modeling user choice in the passpoints graphical password scheme," in Proc. Symp. Usable Privacy Security, 2007, pp. 20–28.
[9]. [9] Adi Shamir and Eran Tromer. Factoring Large Numbers with the TWIRL Device. Unpublished Manuscript, 2003. Available electronically: http://www.cryptome.org/twirl.ps.gz.
[10]. P. C. van Oorschot, A. Salehi-Abari, and J. Thorpe, "Purely automated attacks on passpoints-style graphical passwords," IEEE Trans. Inf. Forensics Security, vol. 5, no. 3, pp. 393–405, Sep. 2010.