An Attempt Based Textual Graphical Password Approach For User Authentication To Protect Against Shoulder Surfing Attacks

By

SOHAIB KHAN

NUST201463273MRCMS64214F

Masters of Science in Systems Engineering

Supervised by

Dr. Adnan Maqsood

RESEARCH CENTER FOR MODELING AND SIMULATION (RCMS)

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST), ISLAMABAD, PAKISTAN

May, 2018

An Attempt Based Textual Graphical Password Approach For User Authentication To Protect Against Shoulder Surfing Attacks

Supervised by

Dr. Adnan Maqsood

Research Center for Modeling and Simulation (RCMS)

A thesis submitted to the National University of Sciences and Technology in partial fulfillment of the requirement for the degree of Masters of Science

May, 2018

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by Mr. **Sohaib Khan**, Registration No. **NUST201463273MRCMS64214F** of **RCMS** has been vetted by undersigned, found complete in all aspects as per NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature with stamp:_____

Name of Supervisor: Dr. Adnan Maqsood

Date: _____

Signature of HoD with stamp: _____

Date: _____

Countersign by

Signature (Dean/Principal): _____

Date: _____

APPROVAL

It is certified that contents of the thesis entitled "**An Attempt Based Textual Graphical Password Approach for User Authentication to Protect Against Shoulder Surfing Attacks** " submitted by Mr. **Sohaib Khan**, Registration No. **NUST201463273MRCMS64214F** of **RCMS** have been found satisfactory as partial fulfillment for award of MS/MPhil degree.

Name of Supervisor: **Dr. Adnan Maqsood**

Signature:_____

Date: _____

Name of GEC member 1: **Muhammad Tariq Saeed**

Signature: _____

Date: _____

Name of GEC member 2: **Fawad Khan**

Signature: _____

Date: _____

Name of GEC member 3: **Ammar Mushtaq**

Signature: _____

Date: _____

Dedication

Dedicated to my beloved parents whose prayers and sacrifices made me what I am today. This achievement is a part of their dream to give me the best education as they could.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to nay other University or Institution.

_____ _____

Date Sohaib Khan

Acknowledgments

I would like to thank Almighty ALLAH who is the greatest of all, who has provided me with strength to achieve this milestone. I would like to show my deepest gratitude to my supervisor Dr. Adnan Maqsood for his support, guidance and mentor-ship throughout this project. I sincerely appreciate the efforts of my GEC members Engr. Fawad Khan and Dr. Ammar Mushtaq for giving their insightful input at every step. Specially Mr. Muhammad Tariq Saeed without his assistance and vision it was impossible for me to complete this project, the way he motivated me at every step and keep pushing me to do more and better is admirable. I am highly obliged to my all family members, without their unending support, tolerance and prayers it was impossible to complete this work.

Contents

List of Abbreviations

AES	Advanced Encryption Standard
2FA	Two Factor Authentication
DES	Data Encryption Standard
S3PAS	Scalable Shoulder Surfing Resistant Textual-Graphical Password Authentication Scheme
CIA	Confidentiality, integrity, availability
GPU	Graphical Processing Unit

List of Tables

List of Figures

Abstract

Most common schemes typically used for user authentication are based on textual passwords. Such schemes are highly vulnerable to keyloggers, shoulder surfing and brute force attacks. The increase in security breaches during the last decade have forced security analysts to use graphical pass word schemes that lead to increased complexity and decreased user friendliness. In this study, we present a comparison of existing techniques with Scalable Shoulder Surfing Resistant Textual-Graphical Password Authentication Scheme (S3PAS) on the basis of password strength, algorithmic complexity, user friendliness and randomness. After comparison we have developed an open-source algorithm for S3PAS to provide protection against keyloggers and shoulder surfing attacks. The proposed architecture of scheme requires original password only for registration process and session password for login process. Session password will be different for each login attempt, so if attackers somehow steal session password they will not be able to use that session password for next attempt to login. Consequently there is an improved trade-off between user friendliness and protection against keyloggers and shoulder surfing attacks. Also time complexity of proposed algorithm for the technique can be written in the form of linear function ax + b which is better than existing techniques..

Chapter 1

Introduction

1.1 Information Security

Information Security is the practice of protection of data from those with malicious intentions to preserve its Confidentiality, Integrity and Availability (CIA). CIA is the basics of information security as shown in 1.1

The prioritization out of three is done on the type of data or the process at hand. To help aid in security and protection of data, information security is further divided into five aspects [1] [2]. The five aspects of information security includes

1. Accountability: Trace-ability of actions performed by user on a system and how these actions will be held against user responsible for breaches.
2. Assurance: Making sure that security measures must be designed and tested to ensure assurance.

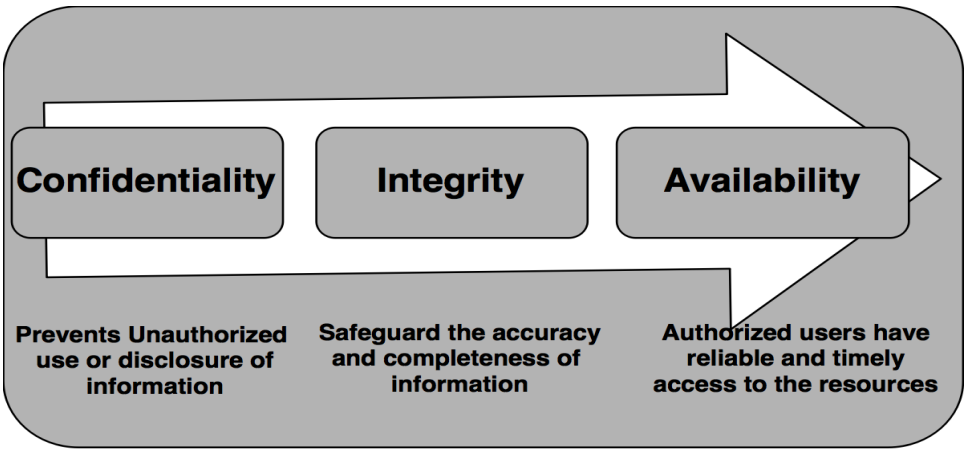

Figure 1.1: CIA Traid of Information Security. Confidentiality is set of rules to limit access to data, integrity is Maintaining the originality of data and availability is reliable access to data for authorized users.

3. Authentication: Positive identification of users i.e. ensuring that users logging into the system are who they say they are.

4. Authorization: Determining the individuals are allowed to access certain data.

5. Accounting: Record of which data has being accessed and by whom.

Security of the system can not be ensured by looking solely at the components of the system. Interaction of these components is more that individual components themselves [3]. Authentication is known as the corner stone of most netwroking models. Attacks against authentication such as incidents of data breaches and identify theft in the recent past has really made it important to implement secure authentication techniques. These threats can be external or from inside the system by untrustworthy users [4]. According to Bilge in "All your contacts are belong to us: automated identity theft attacks on social networks" [5] most of the social security credentials are vulnerable to automated attack. Out of these, authentication is most important in network security models because if attacker is able to exploit

authentication scheme, they can gain access to the system and data as a legitimate user.

1.2 Authentication

Authentication is the positive identification of a system or person who wants to gain access to the system. In authentication process, user identification is done through textual passwords, smart-cards, graphical passwords, bio-metrics, two factor authentication and so on. Authentication systems are used to provide security to user's data to preserve its confidentiality and integrity. Usability of any authentication system is an important factor because users can misuse the system which can lead to security failures [6].

Existing authentication systems suffer from various vulnerabilities. Various encryption protocols such as AES (Advance Encryption Protocol) are used these days for protection of data. On the other hand advancement in technology has also increased the level of threat. All efforts encryption of data is useless when attackers have access to the system by obtaining the credentials of users by exploiting authentication systems/techniques [7].

1.3 Attacks Against Authentication Systems

Authentication systems are vulnerable to multiple types of attacks which can be categorized into two classes; capture attacks and guessing attacks [8].

1. Guessing attacks take place at the time of login when an attacker tries to access

the account by guessing the password. These attacks include brute force and dictionary attacks. These attacks can be online where attacker tries to login by guessing password or offline on stolen password file.

2. Capture attacks include tricking users to disclose their passwords or recording it with the help of malwares which record passwords by installing malicious software without the knowledge of user.

In existing authentication systems users are required to enter their credentials every time they need access to the system. These actions of entering usernames and passwords exposes users to keyloggers and shoulder surfing attacks.

1.3.1 Keyloggers and Shoulder Surfing Attacks

Keyloggers are software that record every keystroke made by user usually covertly. So how strong a graphical or a textual password is; does not matter when every keystroke is being logged. It doesn't matter whether the password is human memorable or not, when it can be recorded in the form of keystroke. Every key pressed by a user is logged by these keyloggers and user credentials can easily be extracted from these logs. In result attacker can use these credentials to log-in to the system as legitimate user and can gain access to the data [9].

Same is the case with shoulder surfing attacks. In this type of attack person sitting beside or standing behind a user can see and memorize user credentials. Similarly camera installed at a certain angle can record every keystroke without even the knowledge of user, which can later be used to see the entered credentials.

All security measures no matter how strong cannot protect the system, if the at-

tacker is logged in as legitimate user and can do whatever they want to do without raising any alarm [10].

1.4 Problem Statement

Authentication plays a key role in protection of data to preserve its confidentiality and integrity. Passwords are used in most of the authentication schemes and often present multiple difficulties to users. In security systems, textual passwords are often considered as weak link. In order to overcome the weaknesses of textual passwords, many graphical password schemes are introduced which provide better security than simple textual passwords. These graphical passwords are still vulnerable to multiple attacks mainly the capture attacks. Passwords can be recorded either with the help of key-loggers, through shoulder surfing attacks or phishing attacks. This information can be used to get access of a system.

Main focus of this study is to mitigate key-logger and shoulder surfing attacks by introducing a textual-graphical password authentication scheme. This scheme will allow the users to access their data anywhere without exposing their password to these attacks.

1.5 Research Objectives

Objectives of this research endeavour are to:

- Propose and implement Textual-Graphical password scheme to provide protection against keyloggers and shoulder surfing attacks.

- Compare and evaluate implemented technique with existing techniques on the basis of complexity, randomness and user friendliness.

1.6 Contributions

The contributions of this thesis investigates the vulnerabilities of existing authentication schemes to keyloggers and shoulder surfing attacks. We provide a textual-graphical password authentication scheme to mitigate these attacks. This scheme is based on S3PAS (Scalable Shoulder Surfing Resistant Password Authentication Scheme) and is implemented using Firebase for data storage. This thesis provides an open sources authentication technique which can be used in different authentication applications.

This thesis also compares proposed authentication scheme with existing authentication schemes on basis of certain factors which includes user comfort-ability, randomness, complexity and key space.

1.7 Structure of Thesis

The organization of rest of the thesis is as follows

- Chapter 1 includes introduction of thesis, problem statement, motivation and objectives of research.
- Chapter 2 presents literature review of related work including the topics of textual passwords, graphical password, existing authentication schemes, key-

logger and shoulder surfing attacks.

- Chapter 3 introduces the textual-graphical password authentication scheme. We introduced this scheme to mitigate keylogger and shoulder surfing attacks.

- Chapter 4 presents results of proposed scheme and its effectiveness against shoulder surfing and keylogger attacks. It also provides a comparison of proposed scheme with existing authentication schemes on the basis of multiple factors.

- Chapter 5 includes conclusion and future work.

Chapter 2

Literature Review

2.1 User Authentication

2.1.1 Origin and History

Back in 1960 passwords were first used in Massachusetts Institute of Technology (MIT) on its time-sharing computer called Compatible Time-Sharing System (CTSS) for accomodation of multiple users on various terminals. This was used to identify individuals and allowing them access to the system [11]. Security aspect was not under consideration at that time and passwords were stored in simple plain text files. First documented breach was done by Allan Scherr in 1962 at MIT who wanted to increase his allotted time to complete his performance simulation.
In 1966 a bug displayed the password file upon logging into the system this happens because passwords were stored in plain text.
Earliest documented attempt to encrypt password was carried out by US air force

in early 1970s over the Multics User Control Subsystem Mainframe (MULTICS). It converts passwords through one way transformation, 90% of the passwords were cracked on Multics in order to determine weak password by Paul Karger and his team. During 1972-1974 security assessment of Multics was done and breached successfully. This success motivated developers to develop stronger methods to protect passwords. This was the turning point that leads to the development of the complex algorithms and methods.

Sixth edition of UNIX (Uniplexed Information and Computing System) operating system implements a password cipher that simulates M-209 cipher machine used in World War II and it uses password as an encryption key. This was done to counteract the weaknesses of the cipher [12]. In 1978 study was performed by Robert Morris on possible attacks against the M-209 cipher key and calculated that average password length used at that time can be cracked using brute force dictionary attack against 62 alphanumeric character in approximately 318 hours.

Later version of UNIX start using DES (Data Encryption Standard) for faster encryption [11]. Passwords were used as keys for encrypting the a constant which was initially zero, 25 iterations were performed and the resulting bits were repacked to a string of 11 printable characters[13]. In 1977 DES was announced as national standard for encryption [14].Initially DES was vulnerable and the systems using it were at risk because password length was limited to eight characters.

2.1.2 Encrypted Passwords

Up till early 80s, user names and passwords were stored in the same file and readable to all users. In 1988, the concept of password shadowing was introduced by Sun

Micro systems which separates user names from passwords and restricts user access to password hashes and make it readable to only rootuser [12]. The main advantage of hashing was that it was a one way function and was extremely difficult to reverse and its slowness adds to the time required [11].

Password auditing and validation tools like Npasswd [15] for checking password for easy guess ability, Crack [16] developed in 1996, a very powerful tool at the time cracks password file by performing auditing sweeps. LANMAN hashing was introduced by Microsoft in 1990 in which password length was limited to fourteen characters which was later compromised. It was followed by NTLM (NT LAN Manager) for windows NT. LANMAN uses DES encryption, it weakens the password as it reduces the length of password and was not case sensitive. Unlike LANMAN NTLM uses MD4 hashes. MD4 was introduced by Ronald Rivest in 1990 [17]. It does not use salting for strong encryption, later on its weaknesses were compromised in 1995-96 [17].

BSD/OS (Berkeley Software Design) extended DES-based crypt to support long passwords as up till this time long passwords were not supported which was a major vulnerability it uses 24bit salt. In 1994 MD5 was introduced with up to 1000 iterations and 48 bit of salting. Linux distributions adopted using MD5 in late 1990s. Rainbow tables attack tools like Qcrack and Bitslice were introduced in 1977, they uses rainbow tables and have large databases of pre calculated hashes of dictionary words [18].

In late 1990s password attacks were extended to networks. Attackers targeted passwords in transit by sniffing compromised servers.Introduction protocols like kerberos and SSH (in 1995) were introduced to provide network authentication without exposing keys. Web developers build application password security around PHP MD5 module but it still lacks salting and hashing in early 2000. NTLM hashing was

introduced in windows NT4 2000 and in newer systems.

In 2000 DES encryption was cracked by Electronic Frontier Foundation in 23 hours. Microsoft worked on NTLM and improved its security in Windows NT and newer systems by using long passwords and distinction between upper and lower case letters. Microsoft introduces SAM file to store password hashes. To add another layer of security these SAM files were also encrypted by Microsoft using SYSKey encryption. SAMDUMP/SAMDUMP2 can recover SYSKey that was used from system hive by using BKhive to dump the SAM file. This provides hacker with a copy of user's hashed password. Also these SYSKeys do not protect data in the memory [19].

MD5 digest and AES (Advance Encryption Standard) key were introduced in windows vista, windows 7, 8 2008, 2008R2. AES was developed by NIST in 2001 which is also known as Rijndael. In networked domain environment passwords are stored in NTDS. DIT database and are centrally managed through active directory. Later in 2002 government replaced DES by AES as standard for encryption.

With increasing security demanding more and more computational power; attackers began using distributed computing to increase available computing power. SANS password calculator estimates that ten character password using both uppercase and lowercase letters can be cracked in one day time using a cluster GPU.

A 25 GPU password cracking unit was introduced in conference in Norway in 2012 and have the capability of

- 14 character Windows XP password in six minutes.
- Eight character NTLM password in 5.5 hours
- 180 billion MD5 hashes per second.

- 63 billion SHA1 per second
- 71000 bcrypt hashes per second
- 363000 SHA512 crypt hashes per second

They were working on to extend the hashcat implementation using virtual open clusters to support up to 128 GPUs. And they achieved 90% to 95% success rate in brute forcing against LinkedIn password hash that was leaked online.
With this much computational power available something more was required, so the concept of two factor authentication was introduced which can be in the form of a card and pin or password and sms code or one time password tokens.
In 2003 2FA by RSA security take over 72% of the market. Paypal introduces 2FA in 2007. Blizz authenticator in 2008. Initially in 2010, phone calls were first method of authetication supported by Duo Security. Later Duo introduced SMS delivered on time pass codes in early 2011. Duo push was invented in 2010 and released in 2011, it offers asymmetric cryptography and mutually authenticated secure communication over data network [20]. One time password tokens were used by RSA security till 2011, at the time they had to replace 40 million of its secure ID tokens because hackers attains the algorithm used to generate one time passwords. There is also enforcement of security policies and practices related to passwords where user is forced to use minimum of eight character password. Also the password must contain uppercase, lowercase, numeric and symbols [21]. Users are warned not to store passwords in simple plain text files and no storage of multiple passwords even in the encrypted form. Password strength is measured before it is allowed to be selected, this work is done by special strength checking meters [15]. Even the concept of 3FA is under consideration which will includes physical token , password and biometric

data. Biometric data can be in the form of a finger print scan or a voice print.
The work is still in progress in this field as fool proof security can never be provided and hackers are coming up with more advanced approaches to compromise a system by getting in to the system legally by getting password/credentials required illegally.

2.2 Textual Passwords

Most of the techniques used for authentication involve the use of traditional textual passwords as first step of authentication. These textual passwords are vulnerable to brute force, dictionary, keyloggers, shoulder surfing and other kind of attacks. As users tend to select password which they can easily remember so it is far from fulfilling the requirements of a strong password. It only results in making these passwords vulnerable to threats.
Password recovery tools like John the Ripper provide dictionaries which can be used for recovery as well as cracking passwords and have higher success rate against these weaker passwords [22].
Even with the adaptation of restrictive password policies like the use of special characters, alphanumeric and combination of upper case and lower case letters in passwords, it still does not prevent a user from selecting a weak password. Mostly passwords follow patterns like use of capitalized letter in the start of a password and numeric in last making these passwords vulnerable to smart attacks but these passwords are far better than the passwords selected without the enforcement of any restrictive policy. However, users entering same password repeatedly on multiple devices are still susceptible to different kind of threats and it requires further actions

to be taken in order to minimize such threats.

2.2.1 Two Factor Authentication (2FA)

As per the name suggests it needs more than a user name and a password to log-in into the system. An extra pass-code or token is always required for second step of authentication. This token is usually a code sent to some other device owned by user. It can be sent through a message or as an email from the service provider. As effective as it sounds against different kind of attacks, it also has its drawbacks and limitations [23]. One of the major issue in 2FA is its adaptability and acceptance to users. Very small chunk (6.5%) of users are actually using 2FA provided by Google and other service providers. 2FA needs more time than usual one factor authentication systems and like in case of Google 2FA [24].

2.3 Graphical Passwords

With the limitations of textual passwords and their vulnerability to attacks, new authentication schemes/methods are introduced including the use of pictures as passwords. Various schemes have been introduced over the past years which can be classified into two main categories including recall based and recognition based techniques [25].

2.3.1 Recall Based Techniques

In this type of techniques user is asked to reproduce a drawing or to repeat a selection made earlier at the time of registration. Few techniques are discussed as

Blonder Technique

This scheme was presented by G.E blonder. In this scheme an image with tap regions is presented to user and user is asked to click in those tap regions and in a specific sequence as shown in 2.1.
This scheme has certain vulnerabilities in which memorable space is a major one. As user can not click with full independence because of predetermined tap regions.

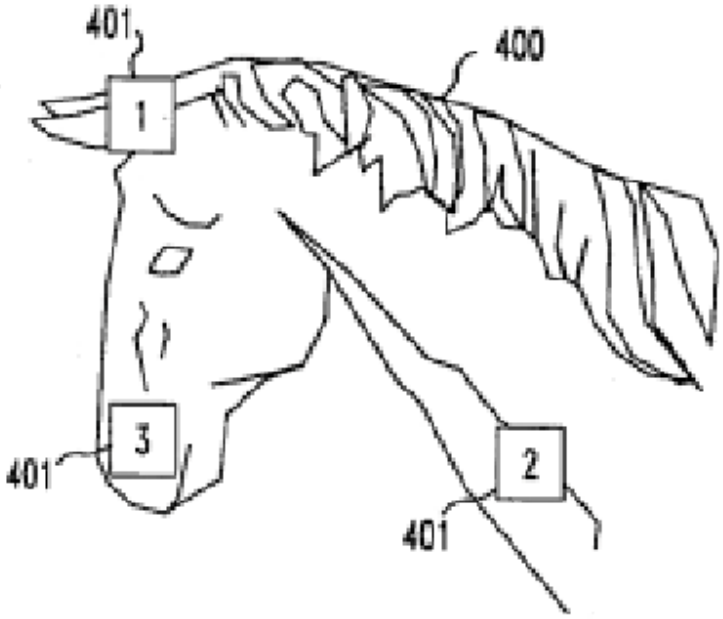

Figure 2.1: Blonder Technique: User is presented with an image with tap regions and user is asked to tap those regions in a specific sequence which user needs to remember at the time of login [26].

Pass-Face Scheme

During the registration process, the user is asked to select four images. These images then need to be identified correctly for two consecutive times. When its time to log-in; the user is presented with the grid of nine faces and has to choose the previously selected images [27] as shown in 2.2.

Figure 2.2: Pass-Face Scheme: In this technique user needs to recognize and select set of images selected at the time of registration. (Source: Suo [28])

However it should be kept in mind that a single password image will be displayed in each grid. This method is predictable and is affected by certain factors including attractiveness, gender and race. Pass-faces are very memorable over long intervals [29, 30].

Pass-Point Technique

At the time of registration user is asked to click on random points in a picture and user has to click on the same points at the time of log in. If clicked points are within the range of defined tolerance only then user will be logged in to the system [31]. These pass-points are difficult to learn, remember and take comparatively greater time than textual passwords. These points are also vulnerable to shoulder surfing attacks with no randomness.

2.3.2 Recognition Based Techniques

In this type of techniques, users are presented with a set of picture and they select few of them. During authentication process user is again presented with a set images and they identify and select images they have chosen earlier. Some of the techniques along with their vulnerabilities are discussed as

Shoulder Surfing Resistant Graphical Password Authentication

A two step authentication technique asking user to select minimum of six images from a grid of 25 for step I authentication, then to select three questions and to answer these questions user has to click on images for step II authentication. For log in phase user has to select correct graphical password in the same sequence as user will be presented with a grid of random images including first pair of selected images. By selecting the image displayed at the intersection of row and column of password pair displayed user will complete first step of authentication. Then the previously selected images and three questions numbers (instead of whole questions)

will appear and user has to click in region of answers as selected at the time of registration. This technique provides some randomness but asks users to remember too much including images, order of images and order of questions selected. It becomes harder and harder to memorize all these things as length of the password increases [32].

Click Based Graphical Password

This scheme provides two step log in process. The first step involves traditional textual password whereas second step requires clicks within the defined range on previously selected image at the time of registration [33]. Similar to pass-point technique, using same image over and over again provides no protection against shoulder surfing and keyloggers.

Captcha as Graphical Password (CaRP)

Captcha is another technique which if combined with graphical passwords, has proven to be very effective against security attack. However the generation of CaRP images provide an extra load on server side as these captcha images have to be generated at every log in [34]. Other than that these images are very hard to read specially for the physically challenged user which results in making it useless for a certain audience, such an example is shown below

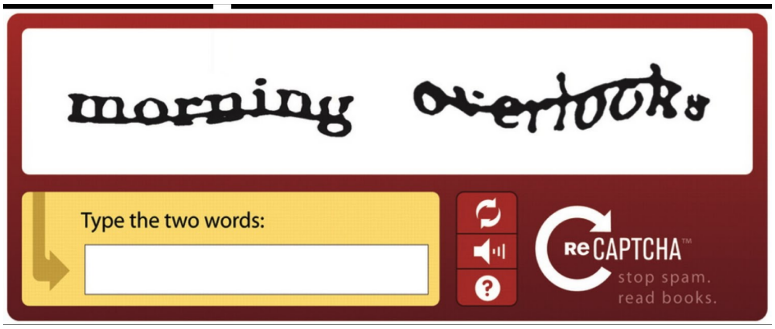

Figure 2.3: Captcha as Graphical Password, in this technique users need to enter the distorted text from the image displayed displayed[34]

Sobrado and Birget Method

This technique prompts user to select images from different objects displayed at the time registration process then the user has to click inside the area formed by the selected images to get access [35] as shown in 2.4.

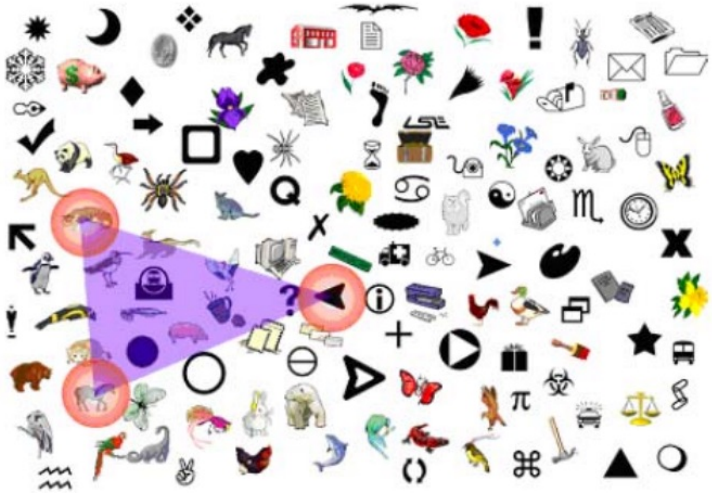

Figure 2.4: Grid of objects for Sobrado and Birget scheme where user needs to click inside the triangle formed by objects selected at the time of registration. [35]

Display becomes crowded and it becomes harder to remember/recognize selected objects.

S3PAS (Scalable Shoulder Surfing Resistant Textual-Graphical Password Authentication Scheme)

A Scalable Shoulder Surfing Resistant Textual-Graphical Password Authentication Scheme provides resistance against shoulder surfing attacks and keyloggers. In this scheme user is asked to select a user-name and textual password at the time of registration.

For log-in phase, an image is generated consisting of random characters along with characters of password selected by user during registration phase. User has to click within the areas formed by three characters of password scattered randomly in generated image. Slider window of one character is used to form three paired characters [36]. For example, if user has four character password, say "1Cdf" then the following pairs will be formed according to logic used 1Cd, Cdf, df1 and f1C. Now user has to click in the area of triangles formed by pairs in the specific sequence to complete the process of authentication.

2.3.3 Keyloggers and Shoulder Surfing Attacks

How strong a graphical or a textual password is does not matter when every keystroke is being logged or someone is watching over the shoulder. It doesn't matter whether the password is human memorable or not, when it can be recorded in the form of keystroke or video. Every key pressed by a user is logged by these keyloggers and user credentials can easily be extracted from these logs. In result attacker can use these credentials to log-in to the system as legitimate user and can gain access to system [9].

Same is the case with shoulder surfing, person sitting beside or standing behind a

user can see and memorize user credentials. Similarly camera installed at an angle to record every keystroke without even the knowledge of user. All security measures no matter how strong cannot do anything if attackers are logged in as legitimate users and can do whatever they want to do without leaving any trace [10].

2.4 Missing Links in Literature

Many authentication schemes have been proposed over the last decade to protect user credentials against shoulder surfing and key-logger attacks. There is no open sources textual-graphical password authentication scheme which protects password against shoulder surfing attacks and is also user friendly. Adaptability rate for such schemes is very low.

Work on protection of data against shoulder surfing and keyloggers is more focused on hiding the credentials in plain sight but still users have to enter password at the time of login. However this scheme will reduce the gap and will help the researchers up to some extent in finding the missing links between data protection against shoulder surfing attacks and user friendliness. Also this scheme will mitigate shoulder surfing as user will not need to add their original password at the time of login.

Chapter 3

Methods

3.1 Implementation of Textual-Graphical Password Authentication scheme

In order to make S3PAS (Scalable Shoulder Surfing Resistant Textual-Graphical Password Authentication Scheme) technique more effective, easier to learn and user friendly we have made certain modifications/improvements in our proposed technique.

3.1.1 Registrations Phase

In our proposed scheme, users are asked to enter their full name, a unique user-name and a password with the minimum length of 12 characters at the time of registration. Interface for registration is shown in 3.1 (a). User credentials are stored in Firebase.

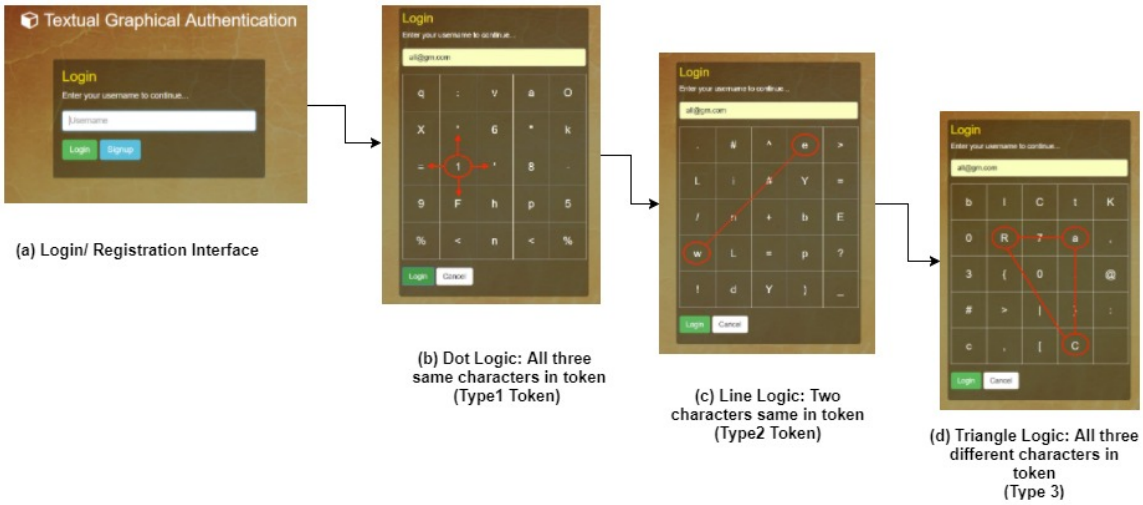

Figure 3.1: Randomly Generated Grid, Circles show password characters in a token and lines represent the area where user needs to click. (a)Shows the login interface where user provides username, and password will be fetched from Firebase after clicking login button. (b) Represents Dot Logic where user needs to click any character pointed by arrow. (c) Shows Line logic where user needs to click on characters lying on line between token characters. (d) Represents Triangle Logic where user needs to click in the area enclosed by triangle formed by token characters and in the end user needs to click login.

3.1.2 Login Phase

At the time of log-in, user is asked to enter previously selected user-name in step I of authentication as shown in fig 3.1(b). At the back end, password against the entered user name is requested and is split into group of three characters each; starting from the first character. These groups are named as tokens. These tokens have three types depending upon the similarity of three characters in a token which we will explain later with example. The type of token will determine what type of logic will be used in authentication process.

Lets take an example in order to explain how the proposed technique works. Credentials required at the time of log-in are username e.g. "all@gm.com" and password e.g. "111weeCaR". In first step of authentication user will enter "all@gm.com"

against which password will be requested from database (Firebase). Password selected for this example will cover all three types of token and authentication logic implemented according to type of token. Password will be converted in to tokens as follows

- Token1: 111 (Type1: Consists of three same characters)
- Token2: wee (Type2: Consists of two same characters))
- Token3: CaR (Type3: Consists of three different characters))

A random grid of 5x5 will be displayed consisting of random characters along with first token characters scattered randomly, as type1 token consists of all three same character so it will be displayed only one time at a random position in grid as shown in figure 3.1 (c).
For the type1 token we named the logic used for authentication as "Dot Logic". In this logic user has to click on a button in grid above, below, before or after the token character displayed in grid as indicated in figure 3.1 (d) by arrows. Here four characters ("=F) are the right choice. If the user clicks on right button, authflag will be set as true otherwise it will be set as False. After making first click on grid, another grid with random characters along with second token characters scattered randomly will be displayed. As in type2 token two of the three characters are different so those two different characters will be displayed in grid at random positions as shown in figure

For type2 token logic used for authentication is named as "Line Logic", as the name clearly indicates that user has to click on any button on the line between the two token characters. Here # and n are the right options. Here authflag will be

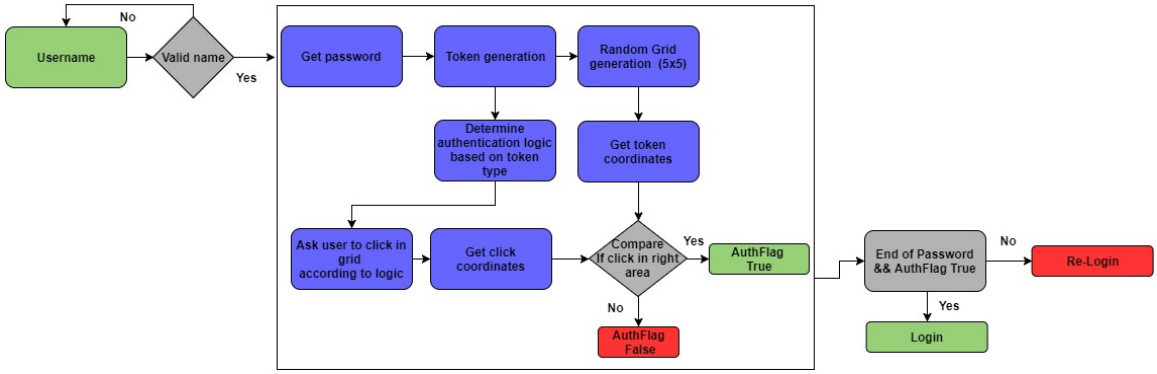

Figure 3.2: Implementation Sequence of Proposed Technique: Scheme starts with username as input, if username is valid password is fetched from Firebase and tokens are generated. Each token is placed randomly in grid of 5x5 and user is asked to click in grid according to logic. Click coordinates when compared with implemented logic decides the status of "Auth Flag". At the end of password if "Auth Flag" status is True user will get access otherwise process stats again.

set as True only when user has selected the right option and also the previous state of the authflag is True, if any condition fails authflag will be set to False.

After making second click, again user will be presented with a grid of random characters including all three different characters from type3 token scattered randomly as shown in figure 3.1 in encircled form for clarity.

For type3 token logic used for authentication is named as "Triangle Logic" because here user needs to click on any button within the area of triangle. Here the best and the most clear right option is '0', but in order to make this scheme more user friendly some tolerance introduced and '—' is also a right option as this button also lies within the area of the triangle. For authflag, same conditions will be applied as in previous step.

3.2 Algorithm For Proposed Approach

Here algorithm of proposed technique along with flow diagram is given for more clarity and better understanding of how technique works.

```
1  Function Procedure (username)
2      get password from Firebase (username);
3      token= preprocessing(password);
4      while End of password do
5          determine logic (token);
6          if token type 1 then
7              Dot logic(token);
8          else if token type 2 then
9              Line Logic(token);
10         else
11             triangle logic (token);
12         end
13         grid[][]= generate random grid(token);
14         get token coordinated(token, grid);
15         Function on click
16             get click coordinates(click);
17     end
18     contd...
```

```
18 Function Authentication(click coordinates, token coordinates)
19     case Token Type 1 do
20         if click coordinate is above, below, right or left to token coordinates
            then
21             Auth Flag= True;
22         else
23             Auth Flag= False;
24         end
25     end
26     case Token Type 2 do
27         if click coordinate on line and between token coordinates then
28             Auth Flag= True;
29         else
30             Auth Flag= False;
31         end
32     end
33     case Token Type 3 do
34         if click coordinate in area triangle formed by token coordinates then
35             Auth Flag= True;
36         else
37             Auth Flag= False;
38         end
39     end
40 contd...
```

```
40 Function Login(Auth Flag, End of Password)
41     if end of Password && Auth Flag then
42         Authentication = Success;
43     else
44         Authentication = Fail;
45     end
```

3.3 Description of Algorithm

This technique starts with username as input from user. This username is also the input of a function which gets password from Firebase stored against a particular username, after confirming the authenticity of username. Then a while loop on line 4 will start, which will execute till the length of password. In this while loop function "determine logic" function on line 5 will decide what logic will be used based on the type of token generated.

The generated token will be placed randomly in a grid of 5x5 along with other randomly generated characters. "get token coordinates" function on line 14 will get the grid coordinates of password characters which will be compared with coordinates of click received using "get click coordinates" function on line 16. On line 17 "Authentication" function will determine depending on token type (Type 1 on line 18, type 2 on line 25 and type 3 on line 32) whether click made was in right area or not. At the end of password "Login" function on line 40 will determine that authentication was successful or not based on the status of Auth Flag.

3.4 Time Complexity of Algorithm

Time taken for execution of our algorithm depends on the size of input so running time of algorithm is defined as the size of input "n" so max number of tokens will be n/3. Also execution time for each line is different. Assuming cost for ith line is Ci. Algorithm with cost of lines alongwith with number of time a line will execute is presented below

```
1  Function Procedure (username)
2      get password from Firebase (username);      ..... 0 * 2
3      token= preprocessing(password);      ..... 0 * 2
4      while End of password      .... c4 * n/3
5       do
6          determine logic (token);      .... c6 * n/3
7          if token type 1      .... 0 * n/3
8           then
9              Dot logic(token); ..... c9 * n/3
10         else if token type 2      .... 0 * n/3
11          then
12             Line Logic(token);      .... c12 * n/3
13         else
14             triangle logic (token);      .... c14 * n/3
15         end
16     end
```

```
contd....
grid[][]= generate random grid(token)      .... c17 * n/3;
get token coordinated(token, grid);      .... c18 * n/3
Function on click
    get click coordinates(click);      .... c20 * n/3
Function Authentication(click coordinates, token coordinates)
    CaseToken Type 1      .... c22 * n/3
    Comparing click and token coordinates;
    Return True or False on basis of comparison;
    CaseToken Type 2      .... c25 * n/3
    Comparing click and token coordinates;
    Return True or False on basis of comparison;
    CaseToken Type 3      .... c28 * n/3
    Comparing click and token coordinates;
    Return True or False on basis of comparison;
Function Login(Auth Flag, End of Password)
    if end of Password && Auth Flag      .... 0 * n/3
     then
        Authentication = Success;
    else
        Authentication = Fail;
    end
```

Now adding up execution time of all lines for execution time of complete algorithm for one user. Taking "n" common from all individual terms we will have the following equation

$$T(n) = n(\frac{c4}{3}+\frac{c6}{3}+\frac{c9}{3}+\frac{c12}{3}+\frac{c14}{3}+\frac{c17}{3}+\frac{c18}{3}+\frac{c20}{3}+\frac{c21}{3}+\frac{c22}{3}+\frac{c25}{3}+\frac{c28}{3}) \quad (3.1)$$

Execution time can be written in the form of linear function ax+b. As linear algorithms are closest to ideal which proves that our proposed technique is better than existing approaches in terms of efficiency as well.

Chapter 4

Results & Discussion

4.1 Analysis of Technique

In order to demonstrate the effectiveness of proposed technique and its advantages over existing techniques we define the following notations to use in this section.

- N: Selected Password
- $|N|$: Length of selected password.
- k: $|N|/3$: Number of tokens.
- P: Password space (All printable characters)
- P1: Possible passwords in one grid

Here we would take the worst case scenario and minimum length of password allowed and see how propose system is going to respond.

$$|N| = 12 \tag{4.1}$$

$$k = 4 \tag{4.2}$$

$$P : 96 \tag{4.3}$$

The number of possible passwords, out of 25 characters three taken at a time with one click will be:

$$P1 = 25C3 = \frac{25!}{3! * 22!} = 2300 Passwords \tag{4.4}$$

Every time grid generated will consist of random characters at random positions within the grid. Only characters of first token will be displayed every time for first grid of every log-in attempt. Token characters present in grid will also have random positions for every attempt.

For one grid with the introduction of randomness of selection of characters from password space P considering three characters remain same we will have the following number of possible passwords:

$$P2 = \frac{(96-3)!}{22! * (93-22)!} = 1.210090E + 21 Passwords \tag{4.5}$$

Now for worst case scenario, considering the fact that, even if the attackers get hold of one token from password. They will have to put 'k' times the effort required to crack one password in order to get hold of complete password.

4.2 Comparison with existing methods and techniques

In this section we will be comparing our proposed technique with existing state of the art techniques based on certain factors. In the table below, different methods are evaluated on the basis of resistance they are providing against different type of attacks [23]. Our main focus is to provide a scheme which mainly provides complete protection against shoulder surfing and keyloggers while competing with existing techniques and methods. Table 4.1 provides a generic comparison of existing methods whereas table 4.2 provides the comparison of techniques discussed which will provide better understanding of vulnerabilities and effectiveness [37].

Method	Resistance against attacks	Processing Time	Protection Level	User Acceptance & Adaptability	Extra Cost
Textual Passwords	No	Fast	Low	High	No
Graphical Passwords	Shoulder Surfing	Medium	Medium	Low	No
Bio-metrics	Shoulder Surfing , Brute Force, keyloggers, Phishing	Medium	High	Medium	Yes
Two Factor Authentication	Brute Force	Slow	Low	Low	Yes
Textual-Graphical Passwords	Shoulder Surfing, keyloggers, Brute Force	Fast	High	Medium	No

Table 4.1: Comparison and Evaluation Of Existing Schemes On the Basis of Resistance Against Shoulder Surfing Attacks And User Acceptance

Technique	Advantages	Disadvantages	Key Space	Ease of use Range (1-5)	Randomness Scale (1-5)
Proposed Technique	Scale-able, Good generic security, keyloggers and Shoulder surfing resistant	Effort in Finding correct option	Large	****	*****
Pass-Point	Complexity	Difficulty in remembering pass-points, predictable	Medium	**	**
Shoulder Surfing Resistant Graphical Password	Two step authentication	Image generation at every log in	Medium, hot spots remain an issue	***	***
Click Based Graphical Password	Two step authentication	Image generation at every log in	Large	***	**
Pass-Face	Complexity, faces easy to remember	Predictable, affected by race, gender etc	Medium	**	**
Blonder	Complexity, Images easier to remember	Crowded display makes it hard to find selected images, Image generation at every log in, load on server side	Large	**	****
Captcha as Graphical Password	Effective when used in combination, protection against bots	Extra load on server side, harder to read, not suitable for physically challenged users	Large	*	***
S3PAS	Shoulder surfing and key logger resistant	Image generation at every log-in	Large	****	****

Table 4.2: Authentication Schemes Comparison On The Basis of Key Space, Ease Of Use And Randomness

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Whenever users provide their credentials to access any system on a public device or even on their personal device in public place, they are vulnerable to shoulder surfing, keyloggers and brute force attacks. In this study we have implemented a technique which introduces session password based on original password already registered, these session password will change for every login attempt. Hence providing protection against mentioned capture attacks.

Main advantage of this approach is that, only three password characters i.e. one token from original password will be displayed at one time in a grid. Also with the introduction of randomness in grids possible password combination to crack one token will be 1.210090E +21.

In this technique we are using a grid of 5X5 which provides better visibility to users as it does not cause over crowding, hence increasing the userfriendliness. This tech-

nique provides overall balance in terms of protection against capture attacks, ease of use and it is compatible with all operating systems. Also time complexity of proposed algorithm is of the order O(n) which is better than existing techniques.

5.2 Future Work

First we aim to implement our approach on the server end this would help us to incorporate this scheme in existing applications. Also it will allow us complete login process every time without even calling password.

We will also make extensions and plugins for browsers, where saved passwords can be stored securely and can only be accessed by going through the login process of implemented technique.

In last we aim to make complete password vault with extended features, which can compete with existing systems and will provide users with basic as well as corporate level security for their confidential data access control.

Bibliography

[1] S. S. Greene, "Security policies and procedures," *Principles and practices. Upper Saddle*, 2006.

[2] C. D. Schou and K. J. Trimmer, "Information assurance and security," *Journal of Organizational and End User Computing*, vol. 16, no. 3, p. 1, 2004.

[3] W. V. Maconachy, C. D. Schou, D. Ragsdale, and D. Welch, "A model for information assurance: An integrated approach," in *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, vol. 310. United States Military Academy, West Point. IEEE, 2001.

[4] G. J. Simmons, "A survey of information authentication," *Proceedings of the IEEE*, vol. 76, no. 5, pp. 603–620, 1988.

[5] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 551–560.

[6] E. A. Stobert, "Graphical passwords and practical password management," Ph.D. dissertation, Ph. D. thesis, Carleton University Ottawa, Ottawa, 2015.

[7] A. H. Lashkari, S. Farmand, D. Zakaria, O. Bin, D. Saleh *et al.*, "Shoulder surfing attack in graphical password authentication," *arXiv preprint arXiv:0912.0951*, 2009.

[8] R. Biddle, S. Chiasson, and P. C. Van Oorschot, "Graphical passwords: Learning from the first twelve years," *ACM Computing Surveys (CSUR)*, vol. 44, no. 4, p. 19, 2012.

[9] F. Tari, A. Ozok, and S. H. Holden, "A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords," in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 56–66.

[10] S. Sagiroglu and G. Canbek, "Keyloggers," *IEEE technology and society magazine*, vol. 28, no. 3, 2009.

[11] M. Bishop and D. V. Klein, "Improving system security via proactive password checking," *Computers & Security*, vol. 14, no. 3, pp. 233–249, 1995.

[12] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proceedings of the 2nd USENIX Security Workshop*, 1990, pp. 5–14.

[13] R. Morris and K. Thompson, "Password security: A case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359172

[14] D. Coppersmith, "The data encryption standard (des) and its strength against attacks," *IBM journal of research and development*, vol. 38, no. 3, pp. 243–250, 1994.

[15] M. Bishop, "Proactive password checking," in *4th Workshop on Computer Security Incident Handling*. Citeseer, 1992, pp. 1–9.

[16] M. Garcia, "2006-331: Password auditing tools," *age*, vol. 11, p. 1, 2006.

[17] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the hash functions md4 and ripemd," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 1–18.

[18] N. Provos and D. Mazieres, "A future-adaptable password scheme." in *USENIX Annual Technical Conference, FREENIX Track*, 1999, pp. 81–91.

[19] C. Walker, "Password security thirty five years later," *SANS Info Sec*, 2014.

[20] T. Petsas, G. Tsirantonakis, E. Athanasopoulos, and S. Ioannidis, "Two-factor authentication: Is the world ready?: Quantifying 2fa adoption," in *Proceedings of the Eighth European Workshop on System Security*, ser. EuroSec '15. New York, NY, USA: ACM, 2015, pp. 4:1–4:7. [Online]. Available: http://doi.acm.org/10.1145/2751323.2751327

[21] W. C. Summers and E. Bosworth, "Password policy: the good, the bad, and the ugly," in *Proceedings of the winter international synposium on Information and communication technologies*. Trinity College Dublin, 2004, pp. 1–6.

[22] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password strength: An empirical analysis," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[23] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[24] A. De Luca and J. Lindqvist, "Is secure and usable smartphone authentication asking too much?" *Computer*, vol. 48, no. 5, pp. 64–68, 2015.

[25] H. K. Sarohi and F. U. Khan, "Graphical password authentication schemes: current status and key issues," *Int. J. Eng. Innovative Technol.(IJEIT)*, vol. 10, no. 2, 2013.

[26] A. Fulkar, S. Sawla, Z. Khan, and S. Solanki, "A study of graphical passwords and various graphical password authentication schemes," *World*, vol. 1, no. 1, pp. 04–08, 2012.

[27] D. Davis, F. Monrose, and M. K. Reiter, "On user choice in graphical password schemes." in *USENIX Security Symposium*, vol. 13, 2004, pp. 11–11.

[28] X. Suo, "A design and analysis of graphical password," 2006.

[29] T. Valentine, "Memory for passfaces after a long delay," Technical Report, Goldsmiths College, University of London, Tech. Rep., 1999.

[30] ——, "An evaluation of the passface personal authentication system," Tech. Rep., 1998.

[31] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon, "Authentication using graphical passwords: Effects of tolerance and image choice," in *Proceedings of the 2005 symposium on Usable privacy and security*. ACM, 2005, pp. 1–12.

[32] M. A. S. Gokhale and V. S. Waghmare, "The shoulder surfing resistant graphical password authentication technique," *Procedia Computer Science*, vol. 79, pp. 490–498, 2016.

[33] V. S. Borkar and P. C. Golar, "Click based graphical passward with text password authentication," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 15, no. 11, p. 76, 2015.

[34] B. B. Zhu, J. Yan, G. Bao, M. Yang, and N. Xu, "Captcha as graphical passwords—a new security primitive based on hard ai problems," *IEEE transactions on information forensics and security*, vol. 9, no. 6, pp. 891–904, 2014.

[35] L. Sobrado and J. Birget, "Graphical passwords. the rutgers scholar: An electronic bulletin of undergraduate research, volume 4, 2002," 2004.

[36] H. Zhao and X. Li, "S3pas: A scalable shoulder-surfing resistant textual-graphical password authentication scheme," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, vol. 2. IEEE, 2007, pp. 467–472.

[37] M. Raza, M. Iqbal, M. Sharif, and W. Haider, "A survey of password attacks and comparative analysis on methods for secure authentication," *World Applied Sciences Journal*, vol. 19, no. 4, pp. 439–444, 2012.

Appendices

Appendix A

Main

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Authenticate</title>
<link
    href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    rel="stylesheet">
<link
    href="//maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css"
    rel="stylesheet">
<link href="css/main.css" rel="stylesheet">
<link href="css/grid.css" rel="stylesheet">
```

```
<script type="text/javascript" src="js/functions.js"></script>
<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
    media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
<script
    src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
<script
    src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->
</head>

<body>
<div id="pre_auth" style="display: none;">
   <div class="container">
      <div class="row">

         <div class="col-sm-12">
            <h2 class="main-title"><i class="fa fa-cube"></i> Textual
                Graphical Authentication</h2>
         </div>

         <div class="col-sm-6 col-sm-offset-3 col-md-4 col-md-offset-4">

            <div id="user_signup_div" class="dark-box">

               <form class="user_signup">
```

```
<h3>Signup</h3>
<p>Fill in the form below and click Signup button:</p>

<div id="signup_error" class="form-group text-danger
    errors"></div>

<div class="form-group">
   <input type="email" name="thisUsername"
       placeholder="Username" class="form-control"
       required>
</div>

<div class="form-group">
   <input type="password" name="password"
       placeholder="Password" class="form-control"
       required>
</div>

<div class="form-group">
   <input type="text" name="display_name"
       placeholder="Full Name" class="form-control"
       required>
</div>

<div class="form-group">
   <button type="submit" name="login" class="btn
       btn-primary"><i class="fa fa-user"></i>
```

```
                Signup</button>
            <button type="button" id="btn_cancel_signup"
                name="btn_cancel_signup" class="btn
                btn-default">Cancel</button>
        </div>
    </form>

</div>



<div id="user_signin_div" class="dark-box">

    <form class="user_login">
        <h3>Login</h3>
        <p>Enter your username to continue...</p>

        <div id="login_error" class="form-group text-danger
            errors">

        </div>

        <div class="form-group">
            <input type="text" name="thisUsername"
                placeholder="Username" class="form-control"
                required>
        </div>
```

```
<div id="pass-grid" class="form-group" style="display:
    none">
   <table class="table table-bordered p-grid">
      <tbody>
         <tr>
            <td><button type="button" id="b0"
               value="0"></button></td>
            <td><button type="button" id="b1"
               value="1"></button></td>
            <td><button type="button" id="b2"
               value="2"></button></td>
            <td><button type="button" id="b3"
               value="3"></button></td>
            <td><button type="button" id="b4"
               value="4"></button></td>
         </tr>
         <tr>
            <td><button type="button" id="b5"
               value="5"></button></td>
            <td><button type="button" id="b6"
               value="6"></button></td>
            <td><button type="button" id="b7"
               value="7"></button></td>
            <td><button type="button" id="b8"
               value="8"></button></td>
            <td><button type="button" id="b9"
               value="9"></button></td>
```

```
</tr>
<tr>
   <td><button type="button" id="b10"
       value="10"></button></td>
   <td><button type="button" id="b11"
       value="11"></button></td>
   <td><button type="button" id="b12"
       value="12"></button></td>
   <td><button type="button" id="b13"
       value="13"></button></td>
   <td><button type="button" id="b14"
       value="14"></button></td>
</tr>
<tr>
   <td><button type="button" id="b15"
       value="15"></button></td>
   <td><button type="button" id="b16"
       value="16"></button></td>
   <td><button type="button" id="b17"
       value="17"></button></td>
   <td><button type="button" id="b18"
       value="18"></button></td>
   <td><button type="button" id="b19"
       value="19"></button></td>
</tr>
<tr>
```

```
                    <td><button type="button" id="b20"
                        value="20"></button></td>
                    <td><button type="button" id="b21"
                        value="21"></button></td>
                    <td><button type="button" id="b22"
                        value="22"></button></td>
                    <td><button type="button" id="b23"
                        value="23"></button></td>
                    <td><button type="button" id="b24"
                        value="24"></button></td>
                </tr>
            </tbody>
        </table>
    </div>

    <div class="form-group">
        <button type="submit" name="login" class="btn
            btn-success">Login</button>
        <button type="button" id="btn_signup"
            name="btn_signup" class="btn
            btn-info">Signup</button>
        <button type="button" id="btn_cancel_login"
            name="btn_cancel_login" class="btn
            btn-default">Cancel</button>
    </div>
</form>
```

```
            </div>


         </div>


         <div class="clearfix"></div>
      </div>
   </div>
</div>



<div id="post_auth" class="container" style="display: none;">
   <div class="row">
      <div class="col-sm-6 col-sm-offset-3">
         <h2 class="text-center">Welcome <span
             id="display_name"></span></h2>
         <ul class="my-link-list">
            <li><a href="http://google.com" target="_blank"><i class="fa
                fa-google"></i> Google</a></li>
            <li><a href="http://yahoo.com" target="_blank"<i class="fa
                fa-yahoo"></i> Yahoo</a></li>
         </ul>
         <p class="text-center"><a href="#" id="signout" class="btn
             btn-success">Sign Out</a></p>
      </div>


      <div class="clearfix"></div>
   </div>
```

```
</div>
```

```
<script type="text/javascript">
thisUser = null;
rand1 = null; // first random number
x = -3, incr = 3; // for increment till password length
c1=null,r1=null;
c2=null,r2=null;
c3=null,r3=null;
loc = null;
loc2 = null;
maxRow = null;
minRow = null;
maxCol = null;
minCol = null;
password = null;
pArray = new Array(); // password.split('');
plength = null; // password.length;
randomNumbers = new Array();
p1=null, p2=null, p3=null;
authflag = true, processStarted = processCompleted = false;
</script>

<script
    src="//ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

```
<script
    src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<script src="//www.gstatic.com/firebasejs/3.4.1/firebase.js"></script>
<script src="//www.gstatic.com/firebasejs/3.4.0/firebase-app.js"></script>
<script
    src="//www.gstatic.com/firebasejs/3.4.0/firebase-auth.js"></script>
<script
    src="//www.gstatic.com/firebasejs/3.4.0/firebase-database.js"></script>
<script src="js/functions.js"></script>


<script>
// Initialize Firebase
var config = {
   apiKey: "AIzaSyAnaNMCe5ul4SOr9ZLinnc-DCtB5u2pGKI",
   authDomain: "sohaibk-b183a.firebaseapp.com",
   databaseURL: "https://sohaibk-b183a.firebaseio.com",
   storageBucket: "sohaibk-b183a.appspot.com",
   messagingSenderId: "130990933079"
};
firebase.initializeApp(config);
</script>


<script type="text/javascript">
$(document).ready(function(){

   firebase.auth().onAuthStateChanged(function(user) {
      if (user)
```

```
    {
        thisUser = user;
        postLogin();
    }
    else
        preLogin();
});


$("#btn_signup").click(function(e){
    $("#user_signin_div").fadeOut(function(){
        $("#user_signup_div").show();
    });
});


    $(".user_signup").submit(function(e){
        e.preventDefault();

        var thisElem = $(this);
        var email = $('input[name=thisUsername]', thisElem).val();
        var password = $('input[name=password]', thisElem).val();
        var displayName = $('input[name=display_name]', thisElem).val();

        $("#signup_error").html('');
```

```
    firebase.auth().createUserWithEmailAndPassword(email,
        password).then(function(user){

      thisUser = user;

      user.updateProfile({
        displayName: displayName
      }).then(function(user){
        firebase.database().ref('users/' +
            Base64.encode(email)).set({
          password: password
        });

        postLogin();
      });

    }).catch(function(error) {

      var errorCode = error.code;
      var errorMessage = error.message;

      $("#signup_error").html(error.message);

    });
});

/**
```

```
     * Handle cancel signup
     */
    $("body").on("click", "#btn_cancel_signup", function(e){

       $("#user_signup_div").fadeOut(function(){
          $("#user_signin_div").show();
       });

    });



$(".user_login").submit(function(e){
    e.preventDefault();

    var thisElem = $(this);
    var thisUsername = $('input[name=thisUsername]', thisElem).val();



    if ( !processCompleted )
    {
       var decodedUsername = Base64.encode(thisUsername);
       var userPath = 'users/' + decodedUsername;

       firebase.database().ref(userPath).once('value').then(function(snapshot)
          {

          var snap = snapshot.val();
```

```
        if ( snap == null )
        {
           tryAgain('Invalid username or password', false);
           return false;
        }

        $("#login_error").html('');
        password = snapshot.val().password;

        pArray = password.split('');
        plength = password.length;

        determineLogic();

        $("#pass-grid").slideDown();
        thisElem.addClass("challenged");
        $("input[name=thisUsername]").attr('readonly', true);

        processStarted = true;
     });
  }
  else
  {
     $("#login_error").html('');

     if ( authflag == false )
```

```
        {

           tryAgain('Invalid username or password', true);
           return false;
        }

        firebase.auth().signInWithEmailAndPassword(thisUsername,
            password).then(function(user){

           thisUser = user;
           postLogin();

        }).catch(function(error) {

          var errorCode = error.code;
          var errorMessage = error.message;

          $("#login_error").html(error.message);

        });
    }
});

    /**
     * Handle cancel login
     */
    $("body").on("click", "#btn_cancel_login", function(e){
```

```
        e.preventDefault();

        resetProcess();
    });


$("#signout").click(function(e){
    e.preventDefault();

    firebase.auth().signOut().then(function() {

      thisUser = null;
      preLogin();
    }, function(error) {

      alert(error.message);

    });
});


/**
 * Handle the button click on the grid
 */
$('.p-grid button').click(function(){

    ClickedButtonValue = $(this).val();
```

```
      authflag = checkAuth();
      console.log(authflag);

      if ( x == (plength - 3) )
         processCompleted = true;

      determineLogic();
   });


   /**
    * Handle try again request
    */
   $("body").on("click", "#link-try-again", function(e){
      e.preventDefault();

      resetProcess();
   });


});
</script>
</body>
</html>
```

Appendix B

Function

```
var Base64 =
    {_keyStr:"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="
,encode:function(e){var t="";var n,r,i,s,o,u,a;var
    f=0;e=Base64._utf8_encode(e);while(f<e.length)
{n=e.charCodeAt(f++);r=e.charCodeAt(f++);
i=e.charCodeAt(f++);s=n>>2;o=(n&3)<<4|r>>4;
u=(r&15)<<2|i>>6;a=i&63;if(isNaN(r)){u=a=64}else
    if(isNaN(i)){a=64}t=t+this._keyStr.charAt(s)+this
._keyStr.charAt(o)+this._keyStr.charAt(u)+this._keyStr.charAt(a)}return
    t},decode:function(e){var t="";
var n,r,i;var s,o,u,a;var f=0;e=e.replace(/[^A-Za-z0-9+/=]/g,"");
while(f<e.length){s=this._keyStr.indexOf(e.charAt(f++));
o=this._keyStr.indexOf(e.charAt(f++));
u=this._keyStr.indexOf(e.charAt(f++));
a=this._keyStr.indexOf(e.charAt(f++));n=s<<2|o>>4;
r=(o&15)<<4|u>>2;
```

```
i=(u&3)<<6|a;t=t+String.fromCharCode(n);
if(u!=64){t=t+String.
fromCharCode(r)}if(a!=64){t=t+String.fromCharCode(i)}}t=Base64._utf8_decode(t);
return t},_utf8_encode:function(e){e=e.replace(/rn/g,"n");var t="";
for(var n=0;n<e.length;n++){var r=e.charCodeAt(n);
if(r<128){t+=String.fromCharCode(r)}else
    if(r>127&&r<2048){t+=String.fromCharCode(r>>6|192);
t+=String.fromCharCode(r&63|128)}else{t+=String.fromCharCode(r>>12|224);
t+=String.fromCharCode(r>>6&63|128);
t+=String.fromCharCode(r&63|128)}}return t},_utf8_decode:function(e){var
    t="";
var n=0;var r=c1=c2=0;while(n<e.length){r=e.charCodeAt(n);
if(r<128){t+=String.fromCharCode(r);
n++}else if(r>191&&r<224){c2=e.charCodeAt(n+1);t
+=String.fromCharCode((r&31)<<6|c2&63);
n+=2}else{c2=e.charCodeAt(n+1);
c3=e.charCodeAt(n+2);
t+=String.fromCharCode((r&15)<<12|(c2&63)<<6|c3&63);
n+=3}}return t}}


/**
 * This will return random string
 *
 * @return String, this will be the random number
 */
function getRand()
{
```

```
   return Math.floor(Math.random() * 24) + 1;
}



function getchar()
{

   return Math.floor(Math.random() * (126 - 32));
}



function getRow(indexPoint)
{
   if(indexPoint >=0 && indexPoint <= 4)
   { return 0;}

   if(indexPoint >=5 && indexPoint <= 9)
   { return 1;}

   if(indexPoint >=10 && indexPoint <= 14)
   { return 2;}

   if(indexPoint >=15 && indexPoint <= 19)
   { return 3;}

   if(indexPoint >=20 && indexPoint <= 24)
   { return 4;}
```

```
}



function getMax(num1,num2,num3)
{
   maxnum = num1;
   if(num2 > maxnum)
   { maxnum = num2;}
   if(num3 > maxnum)
   { maxnum = num3;}

   return maxnum;
}



function getMin(num1,num2,num3)
{
   maxnum = num1;
   if(num2 < maxnum)
   { maxnum = num2;}
   if(num3 < maxnum)
   { maxnum = num3;}

   return maxnum;
}
```

```
function dot()
{
   var inc = 0;
   var y = null;

   rand1 = getRand();

   $("#b" + rand1).text( pArray[0 + x] );

   while(inc < 25) // filling rest of the grid
   {
      y= String.fromCharCode(getchar() + 32);

      while(y == pArray[0 + x] ) // making sure that pword characters
          dont repeat.
         y= String.fromCharCode(getchar() + 32);

      if (inc == rand1 )
         inc = inc +1;



      else
      {
         $("#b" + inc).text(y);
         inc = inc + 1;
      }
   }
```

```
}



function line ()
{
   var inc = 0;
   var y = null;

   rand1 = getRand();
   loc = rand1;
   loc2 = null;
   $("#b" + rand1).text(pArray[0 + x]);



   rand1 = getRand();
   loc2 = rand1
   while (loc == rand1 || loc == (rand1 + 5) || loc == (rand1 - 5) || loc
       == (rand1 + 1) || loc == (rand1 - 1) || loc == (rand1 - 4)|| loc ==
       (rand1 - 6) || loc == (rand1 + 4)|| loc == (rand1 + 6))
   {
       rand1 = getRand();
       loc2 = rand1;
   }

   if (pArray[0 + x] == pArray[1 + x]) // checking which two are same
       then printing the two different ones
   {
```

```
        $("#b" + rand1).text(pArray[2 + x]);
    } else {
        $("#b" + rand1).text(pArray[1 + x]);
    }
    ////////////////////////////////

    while (inc < 25)
    {
        y = String.fromCharCode(getchar() + 32);
        while (y == pArray[0 + x] || y == pArray[1 + x] || y == pArray[2 +
            x]) // making sure that pword characters dont repeat.
        {
            y = String.fromCharCode(getchar() + 32);
        }
        if (inc == loc || inc == loc2) {
            inc = inc + 1;
        } else {
            $("#b" + inc).text(y);
            inc = inc + 1;
        }
    }

}

function triangle ()
{
    loc = null;
```

```
    loc2 = null;
    rand1 = getRand(); //random number
    loc = rand1;
    $("#b" + loc).text(pArray[0 + x]); //assignment
    p1 = loc;
    rand1 = getRand(); //2nd random number
    console.log("rand1 : " + loc);
    while (loc == rand1 || loc == (rand1 + 5) || loc == (rand1 - 5) ||
        loc == (rand1 + 1) || loc == (rand1 - 1))
   {
        rand1 = getRand();
        loc2 = rand1;
    }
    loc2 = rand1;
    console.log("rand2 : " + loc2);
    $("#b" + loc2).text(pArray[1 + x]); //2nd assignment
    p2 = loc2;
    rand1 = getRand(); // 3rd random number
    p3 = rand1; // edge cases can also be checked here as well leaving
        this to do later
    // finding rows and coloumns
    c1 = p1 % 5;
    r1 = getRow(p1);
    c2 = p2 % 5;
    r2 = getRow(p2);
    c3 = p3 % 5;
    r3 = getRow(p3);
```

```
    maxRow = getMax(r1, r2, r3);
    minRow = getMin(r1, r2, r3);
    maxCol = getMax(c1, c2, c3);
    minCol = getMin(c1, c2, c3);
    // edge cases check
    while (minRow == maxRow || minCol == maxCol || loc == rand1 || loc ==
        (rand1 + 5) || loc == (rand1 - 5) || loc == (rand1 + 1) || loc ==
        (rand1 - 1) ||
        loc2 == rand1 || loc2 == (rand1 + 5) || loc2 == (rand1 - 5) ||
            loc2 == (rand1 + 1) || loc2 == (rand1 - 1))
   {
        rand1 = getRand();
        p3 = rand1;
        // finding rows and coloumns
        c1 = p1 % 5;
        r1 = getRow(p1);
        c2 = p2 % 5;
        r2 = getRow(p2);
        c3 = p3 % 5;
        r3 = getRow(p3);
        maxRow = getMax(r1, r2, r3);
        minRow = getMin(r1, r2, r3);
        maxCol = getMax(c1, c2, c3);
        minCol = getMin(c1, c2, c3);
    }
    console.log("rand3 : " + rand1);
    $("#b" + rand1).text(pArray[2 + x]);
```

```
    //p3=rand1;

    // function for filling out the rest of the grid
    var inc = 0;
    var y = null;
    while (inc < 25)
   {
        y = String.fromCharCode(getchar() + 32);
        while (y == pArray[0 + x] || y == pArray[1 + x] || y == pArray[2 +
            x]) // making sure that pword characters dont repeat.
      y = String.fromCharCode(getchar() + 32);

        if (inc == p1 || inc == p2 || inc == p3)
            inc = inc + 1;
        else
      {
            $("#b" + inc).text(y);
            inc = inc + 1;
        }
    }

    console.log("maxRow : " + maxRow);
    console.log("maxCol : " + maxCol);

    console.log("MinRow : " + minRow);
    console.log("MinCol : " + minCol);
}
```

```
/**
 * This will determine the logic either dot, triable or line
 *
 * @return void
 */
function determineLogic()
{
   x = x + 3;

   if( pArray[0 + x ] == pArray[1 + x ] && pArray[1 + x ] == pArray[2 + x
       ] && pArray[0 + x ] == pArray[2 + x ] ) // Dot Logic
      dot();


   else if( pArray[0 + x ] == pArray[1 + x ] || pArray[1 + x ] ==
       pArray[2 + x ] || pArray[0 + x ] == pArray[2 + x ] ) // Line Logic
      line();


   else // Triangle
      triangle();
}


function preLogin()
{
```

```
   jQuery(".errors").html('');
   $("#pre_auth").show();
   $("#post_auth").hide();
}



function postLogin()
{
   $(".errors").html('');
   $("#pre_auth").hide();
   $("#post_auth").show();

   $("#display_name").html(thisUser.displayName);
}



/**
 * This will reset the whole process
 *
 * @return void
 */
function resetProcess()
{
   pArray = new Array();
   x = -3;

   password = pLength = rand1 = null;
```

```
   processStarted = processCompleted = false;
   authflag = true;

   $("#pass-grid").slideUp();
   $("input[name=thisUsername]").val('').attr('readonly', false);
   $("#login_error").html('');

   $("form.user_login").removeClass("challenged");
}

/**
 * Display try again login error
 * @param String, message to display
 * @param Boolean, display "Try Again" link if true
 *
 * @return void
 */
function tryAgain(msg, link)
{
   if ( link )
      msg = msg + ' <a href="javascript:;" id="link-try-again">try
          again</a>';

   $("#login_error").html( msg );
}
```

```
/**
 * This will check the authentication
 *
 * @return Boolean
 */
function checkAuth()
{
   if( pArray[0 + x ] == pArray[1 + x ] && pArray[1 + x ] == pArray[2 + x
       ] && pArray[0 + x ] == pArray[2 + x ] ) // Dot Logic
      return case1();


   else if( pArray[0 + x ] == pArray[1 + x ] || pArray[1 + x ] ==
       pArray[2 + x ] || pArray[0 + x ] == pArray[2 + x ] ) // Line Logic
      return case2();


   else // Triangle
      return case3();
}



function case1()
{
   if (authflag == false)
      return false;
```

```
   else
   {
      if(ClickedButtonValue == (rand1 + 1) || ClickedButtonValue ==
          (rand1 - 1) || ClickedButtonValue == (rand1 + 5) ||
          ClickedButtonValue == (rand1 - 5))
         return true;
      else
         return false;
   }
}

function case2 ()
{
   if (authflag == false)
      return false;
   else
   {
      c1 = loc % 5;
      r1 = getRow(loc);

      c2 = loc2 % 5;
      r2 = getRow(loc2);

      maxRow = getMax(r1, r2);
      minRow = getMin(r1, r2);
      maxCol = getMax(c1, c2);
      minCol = getMin(c1, c2);
```

```
// after getting rows and coloumns checking for same row or coloumns

if (maxRow == minRow) // same row
{
   clickedRow = getRow(ClickedButtonValue);
   clickedCol = ClickedButtonValue % 5;

   if (clickedRow == minRow && clickedCol < maxCol && clickedCol >
       minCol)
      return true;

   else
      return false;


}

else if (maxCol == minCol) // same colounm
{
   clickedRow = getRow(ClickedButtonValue);
   clickedCol = ClickedButtonValue % 5;

   if (clickedCol == minCol && clickedRow < maxRow && clickedRow >
       minRow)
```

```
            return true;
         else
            return false;



      }


      else
      {
         var mid = c1 - c2;
         console.log("m1:" + mid);
         var mid2 = r1 - r2;
         console.log("mid2:" + mid2);
         var diag = 0;

         if ((mid < 0 && mid2 > 0) || (mid > 0 && mid2 < 0))
            diag = mid + mid2;

         else
         {
            diag = mid - mid2;

            if (diag < 0)
               diag = diag * (-1);
         }
```

```
if (diag == 0)
{
   //diag == 0 means both password characters are in diagonal
       position
   if (diag == 0 && loc < loc2 && c1 < c2)
   {
      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) &&
          ((ClickedButtonValue == (loc + 6)) ||
          (ClickedButtonValue == (loc + 12)) ||
          (ClickedButtonValue == (loc + 18))))
         return true;
      else
         return false;
   }
   else if (diag == 0 && loc < loc2 && c1 > c2)
   {
      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) &&
          ((ClickedButtonValue == (loc + 4)) ||
          (ClickedButtonValue == (loc + 8)) || (ClickedButtonValue
          == (loc + 12))))
         return true;
```

```
   else
      return false;



}
else if (diag == 0 && loc > loc2 && c1 > c2)
{
   clickedRow = getRow(ClickedButtonValue);
   clickedCol = ClickedButtonValue % 5;
   if (((clickedRow > minRow && clickedRow < maxRow) &&
       (clickedCol > minCol && clickedCol < maxCol)) &&
       ((ClickedButtonValue == (loc2 + 6)) ||
       (ClickedButtonValue == (loc2 + 12)) ||
       (ClickedButtonValue == (loc2 + 18))))
      return true;
   else
      return false;



}
else if (diag == 0 && loc > loc2 && c1 < c2)
{
   clickedRow = getRow(ClickedButtonValue);
   clickedCol = ClickedButtonValue % 5;
   if (((clickedRow > minRow && clickedRow < maxRow) &&
       (clickedCol > minCol && clickedCol < maxCol)) &&
       ((ClickedButtonValue == (loc2 + 4)) ||
```

```
            (ClickedButtonValue == (loc2 + 8)) ||
            (ClickedButtonValue == (loc2 + 12))))
            return true;
         else
            return false;
      }
   }
      else
      {
         if ((c1 - c2 > 1 || c1 - c2 < -1) && (r1 - r2 > 1 || r1 -
             r2 < -1)) // far apart but not diagonal
         {
            if (c1 > c2 && r1 > r2)
            {
               console.log("1");
               return false;
            }
            else if(c1 > c2 && r1 < r2)
            {
               console.log("2");
               return false;
            }
            else if(c1 < c2 && r1 > r2)
            {
               console.log("3");
               return false;
            }
```

```
            else if(c1 < c2 && r1 < r2)
            {
               console.log("4");
               return false;
            }
            /* clickedRow = getRow(ClickedButtonValue);
            clickedCol = ClickedButtonValue % 5;
            if (clickedRow > minRow)
               return true;
            else
               return false; */

         }
         else if ((c1 - c2 == 1 || c1 - c2 == -1) && (r1 - r2 > 1 ||
             r1 - r2 < -1)) // rows are far apart
         {
            console.log("5");
            return false;
         }
         else if ((c1 - c2 > 1 || c1 - c2 < -1) && (r1 - r2 == 1 ||
             r1 - r2 == -1))
         {
            console.log("6");
            return false;
         }
      }
   }
```

```
   }
}

function case3()
{
    if (authflag == false)
      return false;
   else
   {
   ////////////////////// checking for same rows and coloummns

      if (r1 == r2)
      {
         if (c1 == c3)
         {
            // finding position of clicked button
               clickedRow = getRow(ClickedButtonValue);
               clickedCol = ClickedButtonValue % 5;
               if (((clickedRow > minRow && clickedRow < maxRow) &&
                   (clickedCol > minCol && clickedCol < maxCol)) ||
                   (clickedRow == r1 && clickedCol > minCol && clickedCol <
                   maxCol) || (clickedCol == c1 && clickedRow > minRow &&
                   clickedRow < maxRow))
                   return true;
               else
                  return false;
         }
```

```
else if (c2 == c3)
{

      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) ||
          (clickedRow == r1 && clickedCol > minCol && clickedCol <
          maxCol) || (clickedCol == c2 && clickedRow > minRow &&
          clickedRow < maxRow))
         return true;
      else
         return false;
}
else
{
      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) ||
          (clickedRow == r1 && clickedCol > minCol && clickedCol <
          maxCol))
          return true;
      else
         return false;
}
```

```
}
else if (r1 == r3)
{
   if (c1 == c2)
   {
         clickedRow = getRow(ClickedButtonValue);
         clickedCol = ClickedButtonValue % 5;
         if (((clickedRow > minRow && clickedRow < maxRow) &&
             (clickedCol > minCol && clickedCol < maxCol)) ||
             (clickedRow == r1 && clickedCol > minCol && clickedCol <
             maxCol) || (clickedCol == c2 && clickedRow > minRow &&
             clickedRow < maxRow))
             return true;
         else
            return false;
   }
   else if (c2 == c3)
   {
         clickedRow = getRow(ClickedButtonValue);
         clickedCol = ClickedButtonValue % 5;
         if (((clickedRow > minRow && clickedRow < maxRow) &&
             (clickedCol > minCol && clickedCol < maxCol)) ||
             (clickedRow == r1 && clickedCol > minCol && clickedCol <
             maxCol) || (clickedCol == c2 && clickedRow > minRow &&
             clickedRow < maxRow))
             return true;
         else
```

```
            return false;
    }
    else
    {
          clickedRow = getRow(ClickedButtonValue);
          clickedCol = ClickedButtonValue % 5;
          if (((clickedRow > minRow && clickedRow < maxRow) &&
              (clickedCol > minCol && clickedCol < maxCol)) ||
              (clickedRow == r1 && clickedCol > minCol && clickedCol <
              maxCol))
              return true;
          else
             return false;
    }

}
else if (r2 == r3) ////////////////////////////////
{
   if (c1 == c3)
   {
          clickedRow = getRow(ClickedButtonValue);
          clickedCol = ClickedButtonValue % 5;
          if (((clickedRow > minRow && clickedRow < maxRow) &&
              (clickedCol > minCol && clickedCol < maxCol)) ||
              (clickedRow == r2 && clickedCol > minCol && clickedCol <
              maxCol) || (clickedCol == c1 && clickedRow > minRow &&
              clickedRow < maxRow))
```

```
            return true;
      else
         return false;
}
else if (c1 == c2)
{
      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) ||
          (clickedRow == r2 && clickedCol > minCol && clickedCol <
          maxCol) || (clickedCol == c2 && clickedRow > minRow &&
          clickedRow < maxRow))
          return true;
      else
         return false;
}
else
{
      clickedRow = getRow(ClickedButtonValue);
      clickedCol = ClickedButtonValue % 5;
      if (((clickedRow > minRow && clickedRow < maxRow) &&
          (clickedCol > minCol && clickedCol < maxCol)) ||
          (clickedRow == r2 && clickedCol > minCol && clickedCol <
          maxCol))
          return true;
      else
```

```
            return false;
      }


}
else
{
   if (c1 == c3)
   {
         clickedRow = getRow(ClickedButtonValue);
         clickedCol = ClickedButtonValue % 5;
         if (((clickedRow > minRow && clickedRow < maxRow) &&
             (clickedCol > minCol && clickedCol < maxCol)) ||
             (clickedCol == c1 && clickedRow > minRow && clickedRow <
             maxRow))
             return true;
         else
            return false;
   }
   else if (c1 == c2)
      {
         clickedRow = getRow(ClickedButtonValue);
         clickedCol = ClickedButtonValue % 5;
         if (((clickedRow > minRow && clickedRow < maxRow) &&
             (clickedCol > minCol && clickedCol < maxCol)) ||
             (clickedCol == c2 && clickedRow > minRow && clickedRow <
             maxRow))
             return true;
```

```
            else
               return false;
         }

      else if (c3 == c2)
         {
            clickedRow = getRow(ClickedButtonValue);
            clickedCol = ClickedButtonValue % 5;
            if (((clickedRow > minRow && clickedRow < maxRow) &&
                (clickedCol > minCol && clickedCol < maxCol)) ||
                (clickedCol == c2 && clickedRow > minRow && clickedRow <
                maxRow))
                return true;
            else
               return false;
         }

      else
         {
         // finding position of clicked button
            clickedRow = getRow(ClickedButtonValue);
            clickedCol = ClickedButtonValue % 5;
            if ((clickedRow > minRow && clickedRow < maxRow) &&
                (clickedCol > minCol && clickedCol < maxCol))
                return true;
            else
               return false;
```

```
            }


        }
    }
}
```

Appendix C

Grid

```
/* *** P GRID *** */
table.p-grid{
   width: 350px;
   margin: 15px auto;
}
   table.p-grid td{
      width: 20%;
      height: 70px;
      padding: 1 !important;
   }
      table.p-grid td button{
         background: rgba(0, 0, 0, 0);
         color: #fff;
         font-size: 21px;
         border: 0;
         border-radius: 0 !important;
```

```
    display: block;
    width: 100%;
    height: 100%;
}
table.p-grid td button:hover{
    background-color: #000;
}
```

Appendix D

Interface

```
html,body{/*height: 100%;*/}


body{
   background: #fff url('../images/texture.jpg') no-repeat left top;
   background-attachment: fixed;
}


a, a:visited{
   color: #fff;
   text-decoration: none;
   border-bottom: 1px dashed #fff;
   display: inline-block;
}
   a:hover{
      color: #fff;
      text-decoration: none;
```

```
      border-bottom: 1px solid #fff;
   }

#pre_auth{
   /*background: rgba(0, 0, 0, 0.5) none;*/
   /*height: 100%;*/
}

   h2.main-title{
      color: #fff;
      font-size: 30px;
      text-align: center;
      margin: 50px 0;
   }

   .dark-box{
      background-color: rgba(0, 0, 0, 0.4);
      color: #f8f8f8;
      border-radius: 5px;
      padding: 6px;
   }
      .dark-box > form > h3{
         color: #ffe900; /* #fff for white */
         margin-top: 0;
      }

#post_auth{color: #fff;}
```

```
ul.my-link-list{
   padding: 0;
   margin: 0;
}
   ul.my-link-list li{
      font-size: 17px;
      text-align: center;
      list-style: none;
      margin-bottom: 15px;
   }

/* AUTHENTICATION */
.user_login,
.user_signup{/*margin-top: 60px;*/}

#display_name{
   color: #fff;
   font-style: italic;
}

#user_signup_div{
   display: none;
}

form.user_login{
}
```

```
form.user_login #btn_cancel_login{display: none}


form.user_login.challenged #btn_signup{display: none}
form.user_login.challenged #btn_cancel_login{display: inline-block}
```