**COLOR SUBSTANTIATION OF PASSWORD**

***A Mini Project report submitted in partial fulfillment***

***of the requirements for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

IN

***INFORMATION TECHNOLOGY***

|  |  |
| --- | --- |
| ***By*** |  |
| **M SAISH DATTA** | **12241A1287** |
| **P MADHU VAIBHAV** | **12241A1293** |
| **N SAI KAUSHIK** | **12241A1288** |
|  |  |

***Under the Esteemed Guidance of***

**Dr. PADMALAYA NAYAK**

**Professor**

**Department of IT**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY(AUTONOMOUS)**

**HYDERABAD 500 090**

**MAY 2015**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND**

**TECHNOLOGY**



**CERTIFICATE**

This is to certify that it is a bonafide record of dissertation work entitled **COLOR**

**SUBSTANTIATION OF PASSWORD ”** done by **M.SAISH DATTA(12241A1287), P. MADHU VAIBHAV (12241A1293), N SAI KAUSHIK (12241A1288),**students of **B. Tech (IT)** in the Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2014-2015 in the partial fulfillment of the requirements for the award of degree of B. Tech in Information Technology.

This work is not submitted to any other university for the award of any Degree/Diploma.

**Dr. Padmalaya Nayak ( Prof)** **Dr. Y. Vijayalatha**

Internal Guide Professor & Head

Department of IT Department of IT

GRIET, Hyderabad GRIET, Hyderabad

**External Examiner**

**ACKNOWLEDGEMENT**

We take the immense pleasure in expressing gratitude to our Internal Guide Dr. Padmalaya Nayak, Professor, Information Technology, GRIET . I express my sincere thanks for her encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work .

I am grateful to Dr.Y.Vijayalata, Head of the Department of Information Technology and the Members of the Project Coordinators : Dr.Y. Vijayalata , V.Padma , K.Anusha Nagina for their valuable suggestions.

We express our sincere thanks to Prof. .P. S. Raju, Professor and Director of Gokaraju Rangaraju Institute of Engineering & Technology College, and Dr. Jandhyala N Murthy, Principal of GRIET College, for providing us the conductive environment for carrying through our academic schedules and project with ease

I also take this opportunity to convey my sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad, for their kind cooperation throughout.

Thanking you.

M.Saish Datta P. Madhu Vaibhav

saishdatta@gmail.com madhu.p.vaibhav@gmail.com

9394156894 9700774087



N.Sai Kaushik

saikaushik@gmail.com

9441209450

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

**ABSTRACT**

Access to computer systems is most often based on the use of alphanumeric passwords. However, users have difficulty remembering a password that is long and random-appearing. Instead, they create short, simple, and insecure passwords. Graphical passwords have been designed to try to make passwords more memorable and easier for people to use and, therefore, more secure. Using a graphical password, users click on images rather than type alphanumeric characters. We have designed a new and more secure graphical password system. We describe its security characteristics, and the empirical study we carried out comparing with alphanumeric passwords. In the empirical study participants learned either an alphanumeric or graphical password and subsequently carried out l trials to input their passwords over a period. The results show that the graphical group took longer and made more errors in learning the password, but that the difference was largely a consequence of just a few graphical participants who had difficulty learning to use graphical passwords. In the longitudinal trials the two groups performed similarly on memory of their password, but the graphical group took more time to input a password.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | | **CONTENTS** | | | | |  | | | | |
| **1.** | | | **CERTIFICATE**  **ACKNOWLEDGEMENT**  **ABSTRACT**  **LIST OF FIGURES & SCREEN SHOTS**  **LIST OF TABLES**  **Introduction** | | | | | | | | **1** | | | | |
|  | | | 1.1 Motivation | | | | | | | | X | | | | |
|  | | | 1.2 Problem Definition | | | | | | | | 2x | | | | |
|  | | | 1.3 Objective Of The Project | | | | | | | | 3x | | | | |
|  | | | 1.3 Main contributions | | | | | | | | Xx | | | | |
| **2.** | | | **Background** | | | | | | | | **5x** | | | | |
|  | | | 2.1 | | | Usable Security | | | | | 5 | | | | |
|  | | | 2.2 | | | Authentication | | | | | 7 | | | | |
|  | | | 2.3 | | | Graphical Passwords | | | | | 9 | | | | |
| **3.** | | | **Architecture And Literature Survey** | | | | | | | | **27** | | | | |
|  | | | 3.1 | | | Software Requirements | | | | | 27 | | | | |
|  | | | 3.2 | | | Hardware Requirements | | | | |  | | | | |
|  | | | 3.3 | | | Methodology | | | | | 28 | | | | |
|  | | | 3.4 | | | Proposed Systems | | | | | X | | | | |
|  | | |  | | |  | | | | |  | | | | |
| **4.** | | | **Design** | | | | | | | | **5** | | | | |
|  | | | 4.1 | | | Approaches | | | | | 5 | | | | |
|  | | | 4.2 | | | Coding | | | | | 7 | | | | |
|  | | |  | | |  | | | | |  | | | | |
| **5.** | | | **Implementation and Services** | | | | | | | | **5** | | | | |
|  | | | 5.1 | | | Implementation | | | | | 5 | | | | |
|  | | | 5.2 | | | Steps of Deployment and Use | | | | | 7 | | | | |
|  | | | 5.3 | | | Results Observed | | | | | 9 | | | | |
|  | | | 5.4 | | | Advantages and Disadvantages | | | | | X | | | | |
|  | | |  | | |  | | | | |  | | | | |
|  | | |  | | |  | | | | |  | | | | |
|  | | |  | | |  | | | | |  | | | | |
| **6.** | | | | **Security Discussion** | | | | | | | | **5** | | | |
|  | | | | 6.1 | | | Dictionary Attacks | | | | | 5 | | | |
|  | | | | 6.2 | | | Shoulder-Surfing Attacks | | | | | 7 | | | |
|  | | | | 6.3 | | | Phishing Attacks | | | | | 9 | | | |
|  | | | | 6.4 | | | Social Engineering Attacks | | | | | X | | | |
|  | | | | 6.5 | | | Malware Attacks | | | | | F | | | |
|  | | | |  | | |  | | | | |  | | | |
| **7.** | | **Strategies** | | | | | | | | | **5** | | | | |
|  | | 7.1 | | | | Major Contributions | | | | | 5 | | | | |
|  | | 7.2 | | | | Minor Contributions | | | | | 7 | | | | |
|  | | 7.3 | | | | Conclusions | | | | | 9 | | | | |
|  | |  | | | |  | | | | |  | | | | |
| **8.** | | **Test Suite** | | | | | | | | | **5** | | | | |
|  | | 8.1 | | | | Testing | | | | | 5 | | | | |
|  | |  | | | |  | | | | |  | | | | |
|  | |  | | | |  | | | | |  | | | | |
| **9.** | | **Bibliography** | | | | | | | | | **5** | | | | |
|  | |  | |  | | | | | | |  | | | | |
|  | |  | | **APPENDIX**  **LIST OF FIGURES AND TABLES**  ABSTRACT  **LIST OF FIGURES & SCREEN SHOTS**  **LIST OF TABLES** | | | | | | |  | | | | |
|  | |  | |  | | | | | | |  | | | | |
|  | |  | | | | | | | | |  | | | | |
|  | |  | | | |  | | | | |  | | | | |
|  | |  | | | |  | | | | |  | | | | |
|  | |  | | | |  | | | | |  | | | | |
|  | |  | | | | | |  | | |  | | | | |
|  | |  | | | | | |  | | |  | | | | |
|  | |  | | | | | |  | | |  | | | | |
|  | | | |  | | |  | | | | |  | | | |
|  | | |  |  |  | | --- | --- | --- | |  |  |  | |  | |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  | |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | | | | | | | | |  | | | |
|  | |  | | | |  | | | | |  | | | |
|  | |  | | | |  | | | | |  | | | |
|  | |  | | | |  | | | | |  | | | |
|  | | | | | | | | | |  | | | |  | |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  | |  | | | |  | |  | |  | |  | | | |  | | | | | | | | | | |  | | | |  | |
|  | | | | | | | | | |  | | | |  | |
|  | | | | | | | | | |  | | | |  | |
|  | | | | | | | | | |  | | | |  | |

**INTRODUCTION**

"Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)" .User authentication involves issues of both usability and security. Too often, one or the other is ignored even though both are important and necessary. This problem is evident in knowledge-based authentication systems. For example, passwords are often either memorable-but-insecure or secure-but-difficult-to-remember when they should be memorable and secure. Graphical passwords are potentially more memorable and secure than traditional text passwords because they harness the human ability to easily recognize and recall images. In this thesis, we advance research in the area of knowledge-based authentication through usability and security evaluations of graphical password schemes, the creation of novel schemes that offer improved memory and security, and the identification of some underlying design strategies to inform the design of other knowledge-based authentication schemes.

**1.1 Motivation**

Computer security has traditionally focused on low-level, technical design and implementation details. Security experts often refer to humans as the \weakest link in the security chain, asserting that the problem lies not with the security systems themselves, but with users who are unable or unwilling to comply with security protocols. This approach of separating system design from user behavior is doomed to fail because it ignores the larger context in which security systems are inevitably used.

The shift towards usable security and including human factors as part of system design is an important one that has a direct impact on the security of the system. When users misunderstand how to use security mechanisms, circumvent them because they are too obtrusive, or do not even realize the need for such systems, then the systems are far more likely to result in overall security failures regardless of the systems' technical soundness. People encounter security mechanisms daily, such as physical keys to unlock doors or security alarms intended to alert them of intruders. With respect to computer security mechanisms, people are most often required to authenticate themselves using knowledge-based schemes such as passwords. Even though these are commonly used, and perhaps because they are so prevalent, passwords are plagued with security and usability problems. Technical solutions such as imposing minimum password requirements, and encryption and communication algorithms, for protecting passwords in transit and storage, have not resolved the human factors problems with passwords: usability, memorability, memory interference from having multiple passwords, and predictability in user choice. The \password problem" has been defined as the current situation where many passwords used in practice are either weak-and-memorable or secure-but-difficult-to-remember, despite the need for secure and memorable passwords. Security and usability are often viewed by security experts as opposite extremes, and one must necessarily be sacrificed for the other. We investigate whether it is possible to increase both security and usability at the same time. In this thesis, we focus on one particular aspect of security, namely user authentication. While alternative authentication mechanisms such as biometrics [59] are widely known, these have their own security, privacy, and usability problems [22] that limit their use to specific applications. Due to their widespread usage and relatively low cost, knowledge-based schemes such as passwords are unlikely to disappear; and they may well become even more popular as more day-to-day tasks are computerized. For these reasons, we focus on improving knowledge-based authentication scheme.

**1.2 Problem Definition**

Despite the large number of options for authentication, text passwords remain the most common choice [96] for several reasons. Text passwords are easy and inexpensive to implement, and are familiar to most users. Passwords allow users to authenticate themselves without violating their privacy, as biometrics could, since users can select passwords that do not contain personal information. And finally, passwords are portable since users simply have to recall them, as opposed to tokens which must be carried. However, text passwords also have a number of the inadequacies from both security and usability viewpoints, such as being difficult to remember and being predictable if user-choice is allowed [27, 66, 103, 141]. Passwords are only secure if they are di\_cult for attackers to guess, yet are only usable if users can remember them. The \password problem" is defined [136] as the current situation where many passwords are either weak-and-memorable or secure-but-difficult-to-remember, despite the need for secure and memorable passwords.

Systems sometimes provide on-screen advice on how to create more secure passwords (e.g., select something memorable that would be difficult for others to guess), give feedback about password choice (e.g., with a password strength meter), or force users to create passwords that comply with specific system-defined rules (e.g., the password must include both letters and numbers). Despite these strategies, users often select weak passwords [41] that are predictable and are easy for attackers to guess. This occurs partially because users misunderstand the advice or requirements, underestimate the risks, and because limitations of human memory mean that they must employ coping mechanisms in order to reduce the burden of remembering so many passwords [1]. These coping mechanisms may include reusing passwords across

several accounts, using predictable alphanumeric combinations, or storing passwords in an easily accessible, insecure location [1, 41, 103, 130]. Although they have appealing characteristics, only limited success has been achieved through encouraging the use of passphrases [63] (passwords are longer phrases) or mnemonic passwords [69]

(passwords are abbreviated from a longer word or phrase, for example by using the \_rst letters of the words in a phrase, or including common character substitutions such as \I<3c@s" for \I love cats"). At least in their basic form, both suffer from predictability problems because users choose common character substitutions or well known phrases. Such approaches also do not mitigate the problem of remembering which password corresponds to which account when users have multiple accounts. Furthermore, phishing [33, 60] and other social engineering [140] attacks on passwords have increased dramatically over the past few years since text passwords are easy for users to unintentionally reveal to attackers, complicating matters further. A proposed solution to these password problems is to use password managers. One class of these managers maps easy to remember (weak, low-entropy) user passwords onto stronger passwords (more resistant to guessing attacks), and may also generate site-specific passwords (protecting against some phishing attacks). Password managers exist in different formats: stand-alone applications, browser plug-ins, and browser scripts. As preliminary work to this thesis, we investigated two password managers [20].

Our work shows that while the idea of password managers is promising, in their present form these systems have a number of usability problems that lead to decreased security. We conducted a user study of two browser plug-ins: Pwd Hash [98] and Password Multiplier [53]. We found that the most significant problems arose from users having inaccurate or incomplete mental models of the software. Our study revealed many interesting misunderstandings, such as users who reported that a task was easy even when they were unsuccessful at completing that task, and users who believed that their passwords were being strengthened when in fact they had failed to engage the appropriate protection mechanism. Such \dangerous errors" are especially

concerning because they may have serious security consequences. Our findings also suggest that in the absence of additional education or other means of encouragement, ordinary users would be reluctant to opt-in to using these managers: users were uncomfortable with \relinquishing control" of their passwords to a manager, did not feel that they needed the password managers, and did not believe that these password managers provided greater security.

**1.3 Objective of the Project**

The work began with a general investigation, with new ideas being formed and tested as we progressed with the research. Four main research objectives of this project are described below.

Objective 1: Catalogue existing graphical password schemes, focusing equally on usability and security characteristics, and identify the existing graphical password scheme that appears most promising and that warrants closer evaluation. Objective 2: With respect to security and usability, empirically evaluate the most promising scheme identified through our cataloguing. (This turned out to be the Pass Points scheme.)

Objective 3: Create and empirically test new designs that address any usability and security problems identified in the scheme identified in Objective 2. (Given that Pass Points was the identified scheme, the resulting goal ended up being to increase security and memorability of click-based graphical passwords while maintaining usability.)

Objective 4: Identify the key underlying design characteristics responsible for success of the newly proposed system(s), and generalize these to develop design strategies that can be applied to other types of knowledge-based authentication schemes.

**1.4 Main Contributions**

This research contributes original ideas and knowledge to the field of usable security. We design and test two novel graphical password schemes and a novel algorithm for implementation of click-based graphical passwords. We conducted usability and security analysis of both a pre-existing scheme and newly proposed graphical password systems. As part of our work, we examined how design choices affect user behaviour, as well as the interaction between usability and security.

The main contributions of this research are enumerated below.

1. We reviewed existing graphical password schemes by cataloguing them accord-

ing to several usability and security characteristics. We discovered that there was little consistency in the types of evaluations conducted on graphical passwords, with most evaluations focusing on either usability or security but not both. We identified the most promising scheme in terms of memorability and potential security, and decided that it was worth further evaluation.

2. We conducted two empirical user studies [15] of Pass Points, one controlled experiment conducted in the lab and one large field study where the system was deployed for real usage over several months. In our initial analysis, we show that image choice impacts the usability of Pass Points, that users are extremely accurate in entering their click-points, and that login times and success rate sare generally good. In later analysis of the Pass Points datasets, we show that passwords with certain characteristics have a much higher likelihood of being chosen by users, making them vulnerable to guessing attacks.

3. We proposed Cued Click-Points and Persuasive Cued Click-Points [16].These were prototyped and evaluated with empirical user studies conducted. We show

that these new schemes have usability and memorability advantages over Pass Points. They also significantly increase security with respect to known attacks by reducing the predictability of user selected passwords [17] and increasing the effort required by attackers to launch successful attacks.

4. We proposed centered discretization , a new method for improved implementation of click-based graphical passwords. We evaluated our method using post-hoc analysis of the empirical data collected in the Pass Points field study. Compared to the scheme proposed by the original Pass Points authors , centered discretization allows for smaller tolerance areas, which increases the theoretical password space, and better usability because the system behaves in a manner consistent with user expectations.

5. We extracted and generalized the main design characteristics of our new schemes that led to significant usability and security improvements.

**2.BACKGROUND**

This background chapter provides an introduction to the field of usable security, with a focus on usable authentication, and summarizes relevant methodology in conducting empirical studies. It concludes with an overview of graphical passwords and a summary of published results related to their usability and security evaluations.

**2.1 Usable Security**

Designing user interaction for security applications, and user authentication systems specifically, raises some interesting challenges. The area of usable security can draw from existing Human-Computer Interaction (HCI) knowledge, but some fundamental differences must be taken into account. Properties of security systems that set them apart include:

1.In addition to legitimate users of a security system, there is a second group of users who are actively trying to attack the system. Such attackers will exploit any information leaked by, or that can be extracted through, the interface. They will also leverage any way that the system can be misused or any means to trick legitimate users into revealing confidential information. This makes it difficult to provide some forms of helpful feedback in the user interface, for example to help guide users towards correct passwords, as it may also help attackers.

2. Security is typically a secondary task [134]; if security impedes users' primary goals, users will often try to circumvent the security measures [5, 26, 104].

3.Users have poor mental models of security [20, 134] and they may not even realize that their actions are insecure in the first place. Furthermore, they often misunderstand or underestimate the consequences of insecure actions.

4.Computer security suffers from the \barn door" property [134]: if information or a system is exposed even for a brief time, there is no guarantee that it has not been compromised in an irrecoverable way. The information may have been externally leaked to attackers, or available to malware resident on the system.

While these represent security concerns, they are all directly related to users of the system and as such, solutions must focus as much on the HCI aspects of the system as on the technical security components. Usability problems may significantly impact the real-world security of the system. User interface design decisions may unintentionally sway user behaviour, often towards less secure behaviour.

This may be a direct result of the particular interface, or may be compounded by external influences such as when users reveal their passwords to others due to social expectations. Furthermore, the easiest way

of using a system is often also the least secure way. For example, users may choose very short, simple text passwords because these are easier to remember and enter than longer, more complex sequences of characters.

**2.2 Authentication**

The authentication process can be described as three phases: identification, authentication, and authorization. Users must first make some claim of their identity, provide evidence to substantiate this claim, and if successfully authenticated by the system, access rights are granted to the user.

We classify authentication mechanisms according to the following categories

Something you know (recall): A secret is shared between the user and the system. Users must recall and correctly enter their secret to authenticate themselves. Anyone who knows or guesses the secret will also be able to authenticate as the original user. Examples include passwords and PINs (Personal Identification Numbers).

Something you recognize (recognition): The user and the system share a secret. The system provides cues and the user must correctly recognize the secret. Anyone able to recognize the secret will be able to authenticate as the original user. Graphical passwords where users must recognize pre-selected images from a set of decoys fall into this category. Cued recall systems combine recall and recognition. Users must recognize the cue presented by the system and then use this cue to recall the secret shared with the system.

Something you are (static biometrics): Biometrics measure some unique physical characteristic of the user. These are more difficult to forge than the \_rst two categories but introduce additional concerns. They may require specialized equipment, are di\_cult or impossible to change if compromised, and have potential privacy implications (e.g., they may make it difficult to create different identities for various purposes, and they enable organizations to cross-reference information about a user). Static biometrics include fingerprint, iris, and facial scans, among others.

Something you do (behavioral biometrics): Some unique behavioral characteristic of the user can also be measured. Users authenticate by repeating the required action. Examples include handwritten signatures and keystroke dynamics.

Something you have (tokens): Users must carry a token to be presented for authentication. Anyone who gains access to the token will be able to authenticate as the original user. These are often combined with a PIN or password to offer some protection in case the token is lost or stolen. A smart card, i.e., a card with embedded microprocessor chip, is an example of a token used for authentication.

Where you are (location-based authentication) [29]: Location information can be used to determine if a user is attempting to authenticate from an approved location. This is typically used as a secondary check to identify suspicious login activities. Approved locations may be specific, such as a user's office, or more general, such as identifying the city or country of origin.

**2.3 Graphical Passwords**

For over a century, psychology studies have recognized the human brain's superior memory for recognizing and recalling visual information as opposed to verbal or textual information. The most widely accepted theory explaining this difference is the \dual-coding theory" [85], suggesting that verbal and non-verbal memory (i.e., word-based or image-based) are processed and represented differently in the mind. Images are mentally represented in way that retains the perceptual features being observed and are assigned perceived meaning based on what is being directly observed. Text is a form of knowledge representation. Text is represented symbolically, where symbols are given arbitrary meaning that describes the object represented by the text, as opposed to perceived meaning. For example, `X' may represent the roman numeral 10 or the multiplication symbol; the exact meaning is assigned based on some deeper concept. Furthermore, images may be encoded twice, perceptually and symbolically, if meaning is assigned to the image. Graphical passwords are intended to capitalize on this human characteristic in hopes that by reducing the memory burden on the user, more secure (e.g., longer or more complex) passwords can be produced and users will not resort to unsafe practices in order to cope .

**3.ARCHITECTURE AND**

**LITERATURE SURVEY**

**3.1 Software Requirements**

Operating Systems : Windows/Linux

Language : Core java, Jsp servlet

IDE : Net Beans

Database : MySQL

**3.2 Hardware Requirements**

Processor : Pentium based systems

RAM :512MB (minimum)

Hard Disk : 80 GB

Monitor : 15 VGA Color

**3.3 Methodology**

An authentication system which applies Persuasive Technology should guide and encourage users to select stronger passwords, but not impose system-generated passwords. To be effective, the users must not ignore the persuasive elements and the resulting passwords must be memorable. The proposed system accomplishes this by making the task of selecting a weak password more tedious and time-consuming. The goal is to encourage compliance by making the less secure task more time consuming and awkward. A sound signature is added to this system in order to help the user in recalling the click point on an image during login phase. During registration phase, user is asked to select a sound signature or music. That is, during login phase, if the user clicks the appropriate point on the image then the sound that the user selected during registration phase will be played.

**3.4 Proposed Systems**

A proposed solution to these password problems is to use password managers.

One class of these managers maps easy to remember (weak, low-entropy) user passwords onto stronger passwords (more resistant to guessing attacks), and may also generate site-specific passwords (protecting against some phishing attacks). Password managers exist in different formats: stand-alone applications, browser plug-ins, and browser scripts.

**4.DESIGN**

**4.1 Approaches**

Pair Based Authentication Schemes

During registration user submits his password. Minimum length of the password is 8 and it can be called as secret pass. The secret pass should contain even number of characters. Session passwords are generated based on this secret pass. During the login phase, when the user enters his username an interface consisting of a grid is displayed. The grid is of size 8 x 8 and it consists of alphabets and numbers. These are randomly placed on the grid and the interface changes every time.



Figure: Pair based approach

Figure shows the login interface. User has to enter the password depending upon the secret pass. User has to consider his secret pass in terms of pairs. The session password consists of alphabets , digits and special characters. The first letter in the pair is used to select the row and the second letter is used to select the column. The intersection letter is part of the session password. This is repeated for all pairs of secret pass. Fig 3.1 shows that x is the intersection symbol for the pair “xn”. The password entered by the user is verified by the server to authenticate the user. If the password is correct, the user is allowed to enter in to the system.

Hybrid Textual Authentication Schemes

During registration, user should rate colors as shown in figure

****

Figure 3.2: Registration phase

The User should rate colors from 1 to 8 and he can remember it as “RLYOBGIP”. Same rating can be given to different colors. During the login phase, when the user enters his username an interface is displayed based on the colors selected by the user. The login interface consists of grid of size 8×8. This grid contains digits 1-8 placed randomly in grid cells. The interface also contains strips of colors as shown in figure 10. The color grid consists of 4 pairs of colors. Each pair of color represents the row and the column of the grid.

****

Figure : Login interface

Figure shows the login interface having the color grid and number grid of 8 x 8 having numbers 1 to 8 randomly placed in the grid. Depending on the ratings given to colors, we get the session password. As discussed above, the first color of every pair in color grid represents row and second represents column of the number grid. The number in the intersection of the row and column of the grid is part of the session password. Consider the figure 9 ratings and figure 3.2 login interface for demonstration. The first pair has red and yellow colors. The red color rating is 1 and yellow color rating is 3. So the first letter of session password is 1st row and 3rd column intersecting element i.e **5**. The same method is followed for other pairs of colors. For figure 3.3 the password is “ **5121”.**”

**4.2 CODING**

**Registration.html**

|  |  |
| --- | --- |
|  | <html> |
|  |  |
|  | <body bgcolor="skyblue"> |
|  | <h1></h1> |
|  |  |
|  | <form name="f" action="registration.jsp"> |
|  |  |
|  | <center><h3> |
|  | AA USERNAME<input type="text" name="t1"/><br><br> |
|  | MOBILE NO<input type="text" name="t2"/><br><br> |
|  | EMAIL ID<input type="text" name="t3"/><br><br> |
|  | what is your favourite fooditem? |
|  |  |
|  | <input type="text" name="t4"/> |
|  | </h3><table> |
|  | <tr> |
|  | <td bgcolor="black" height="20" width="20"></td> |
|  | <td bgcolor="blue" height="20" width="20"></td> |
|  | <td bgcolor="red" height="20" width="20"></td> |
|  | <td bgcolor="green" height="20" width="20"></td> |
|  | </tr> |
|  | <tr> |
|  | <td><input type="text" name="1" maxlength="1" size="1"/></td> |
|  | <td><input type="text" name="2" maxlength="1" size="1"/></td> |
|  | <td><input type="text" name="3" maxlength="1" size="1"/></td> |
|  | <td><input type="text" name="4" maxlength="1" size="1"/></td> |
|  | </tr> |
|  | </table> |
|  | <h1><input type="submit" value="submit"/></h1> |
|  | </center> |
|  |  |
|  | </form> |
|  | </body> |
|  |  |
|  |  |
|  | </html> |
|  |  |

**Registration.jsp**

<%@page import="sun.misc.Regexp"%>

<%@page contentType="text/html" pageEncoding="UTF-8" import="java.sql.\*,java.io.\*,java.util.\*"%>

<%

Connection con = null;

Statement st = null;

ResultSet rs = null;

try{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/project","root","database");

System.out.println("connected");

st=con.createStatement();

String un=request.getParameter("t1");

String eid=request.getParameter("t3");

String qst=request.getParameter("t4");

String pno=request.getParameter("t2");

int bclr=Integer.parseInt(request.getParameter("1"));

int blclr=Integer.parseInt(request.getParameter("2"));

int rclr=Integer.parseInt(request.getParameter("4"));

int gclr=Integer.parseInt(request.getParameter("3"));

int i=st.executeUpdate("INSERT INTO `project`.`users` (`uname`, `eid`, `pno`, `quest`, `bclr`, `blclr`, `rclr`, `gclr`) VALUES ('"+un+"', '"+eid+"', '"+pno+"', '"+qst+"', '"+bclr+"', '"+blclr+"', '"+rclr+"', '"+gclr+"')");

if( un==""|| eid=="" || qst=="" )

{ out.println("enter all textfields");}

if(i>0){

out.println("<h1 >Hello "+un+" You Are Registered Success Fully</h1>");

out.print("<h3><a href='login1.jsp'>Login</a></h3>");

}

else {%>

out.println("Sory"); <%}

}

catch (Exception e)

{out.println(e);}

%>

**Login.jsp**

<%--

Document : Login

Created on : Apr 8, 2015, 11:23:55 AM

Author : admin

--%>

<%@page contentType="text/html" pageEncoding="UTF-8" import="java.sql.\*,java.io.\*"%>

<% try{

Class.forName("com.mysql.jdbc.Driver");//Type4

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project","root","database");

System.out.println("connected");

Statement st=con.createStatement();

String un=request.getParameter("u");

String bc=request.getParameter("a");

String bl=request.getParameter("b");

String re=request.getParameter("c");

String gr=request.getParameter("d");

String c0=request.getParameter("00");

String c1=request.getParameter("01");

String c2=request.getParameter("02");

String c3=request.getParameter("03");

String c4=request.getParameter("04");

String c5=request.getParameter("05");

String c6=request.getParameter("06");

String c7=request.getParameter("07");

String c8=request.getParameter("08");

String c9=request.getParameter("09");

String r0=request.getParameter("0");

String r1=request.getParameter("1");

String r2=request.getParameter("2");

String r3=request.getParameter("3");

String r4=request.getParameter("4");

String r5=request.getParameter("5");

String r6=request.getParameter("6");

String r7=request.getParameter("7");

String r8=request.getParameter("8");

String r9=request.getParameter("9");

String r0c0=request.getParameter("000");

String r0c1=request.getParameter("001");

String r0c2=request.getParameter("002");

String r0c3=request.getParameter("003");

String r0c4=request.getParameter("004");

String r0c5=request.getParameter("005");

String r0c6=request.getParameter("006");

String r0c7=request.getParameter("007");

String r0c8=request.getParameter("008");

String r0c9=request.getParameter("009");

String r1c0=request.getParameter("100");

String r1c1=request.getParameter("101");

String r1c2=request.getParameter("102");

String r1c3=request.getParameter("103");

String r1c4=request.getParameter("104");

String r1c5=request.getParameter("105");

String r1c6=request.getParameter("106");

String r1c7=request.getParameter("107");

String r1c8=request.getParameter("108");

String r1c9=request.getParameter("109");

String r2c0=request.getParameter("200");

String r2c1=request.getParameter("201");

String r2c2=request.getParameter("202");

String r2c3=request.getParameter("203");

String r2c4=request.getParameter("204");

String r2c5=request.getParameter("205");

String r2c6=request.getParameter("206");

String r2c7=request.getParameter("207");

String r2c8=request.getParameter("208");

String r2c9=request.getParameter("209");

String r3c0=request.getParameter("300");

String r3c1=request.getParameter("301");

String r3c2=request.getParameter("302");

String r3c3=request.getParameter("303");

String r3c4=request.getParameter("304");

String r3c5=request.getParameter("305");

String r3c6=request.getParameter("306");

String r3c7=request.getParameter("307");

String r3c8=request.getParameter("308");

String r3c9=request.getParameter("309");

String r4c0=request.getParameter("00");

String r4c1=request.getParameter("401");

String r4c2=request.getParameter("402");

String r4c3=request.getParameter("403");

String r4c4=request.getParameter("404");

String r4c5=request.getParameter("405");

String r4c6=request.getParameter("406");

String r4c7=request.getParameter("407");

String r4c8=request.getParameter("408");

String r4c9=request.getParameter("409");

String r5c0=request.getParameter("500");

String r5c1=request.getParameter("501");

String r5c2=request.getParameter("502");

String r5c3=request.getParameter("503");

String r5c4=request.getParameter("504");

String r5c5=request.getParameter("505");

String r5c6=request.getParameter("506");

String r5c7=request.getParameter("507");

String r5c8=request.getParameter("508");

String r5c9=request.getParameter("509");

String r6c0=request.getParameter("600");

String r6c1=request.getParameter("601");

String r6c2=request.getParameter("602");

String r6c3=request.getParameter("603");

String r6c4=request.getParameter("604");

String r6c5=request.getParameter("605");

String r6c6=request.getParameter("606");

String r6c7=request.getParameter("607");

String r6c8=request.getParameter("608");

String r6c9=request.getParameter("609");

String r7c0=request.getParameter("700");

String r7c1=request.getParameter("701");

String r7c2=request.getParameter("702");

String r7c3=request.getParameter("703");

String r7c4=request.getParameter("704");

String r7c5=request.getParameter("705");

String r7c6=request.getParameter("706");

String r7c7=request.getParameter("707");

String r7c8=request.getParameter("708");

String r7c9=request.getParameter("709");

String r8c0=request.getParameter("800");

String r8c1=request.getParameter("801");

String r8c2=request.getParameter("802");

String r8c3=request.getParameter("803");

String r8c4=request.getParameter("804");

String r8c5=request.getParameter("805");

String r8c6=request.getParameter("806");

String r8c7=request.getParameter("807");

String r8c8=request.getParameter("808");

String r8c9=request.getParameter("809");

String r9c0=request.getParameter("900");

String r9c1=request.getParameter("901");

String r9c2=request.getParameter("902");

String r9c3=request.getParameter("903");

String r9c4=request.getParameter("904");

String r9c5=request.getParameter("905");

String r9c6=request.getParameter("906");

String r9c7=request.getParameter("907");

String r9c8=request.getParameter("908");

String r9c9=request.getParameter("909");

ResultSet rs= st.executeQuery("select \* from users where uname='"+un+"'");

if(rs.next()){

String dbuser = rs.getString("uname");

String aa = rs.getString("bclr");

String ab = rs.getString("blclr");

String ac = rs.getString("rclr");

String ad = rs.getString("gclr");

boolean q;

q=false;

//if bc = r aa c ac

//String sss='r'+aa+'c'+ac;

//String ssss='r'+ac+'c'+ab;

// String sssss='r'+ad+'c'+aa;

//String ssssss='r'+ad+'c'+ab;

//String val=sss;

//if( bc!= "sss" || bl!="ssss" || re!="sssss" || gr!="ssssss")

//{ q=false; } else { q=true; }

// String new="";

// r ac c ab

//r ad c aa

//r ad c ab

if (un.equals(dbuser)){ if( bc!='r'+aa+'c'+ac || bl!='r'+ac+'c'+ab || re!='r'+ad+'c'+aa || gr!='r'+ad+'c'+ab )

{q=false; } else {q=true;}}

if(q==true)

{out.println("cngrts");} else {out.println("sry");}

}

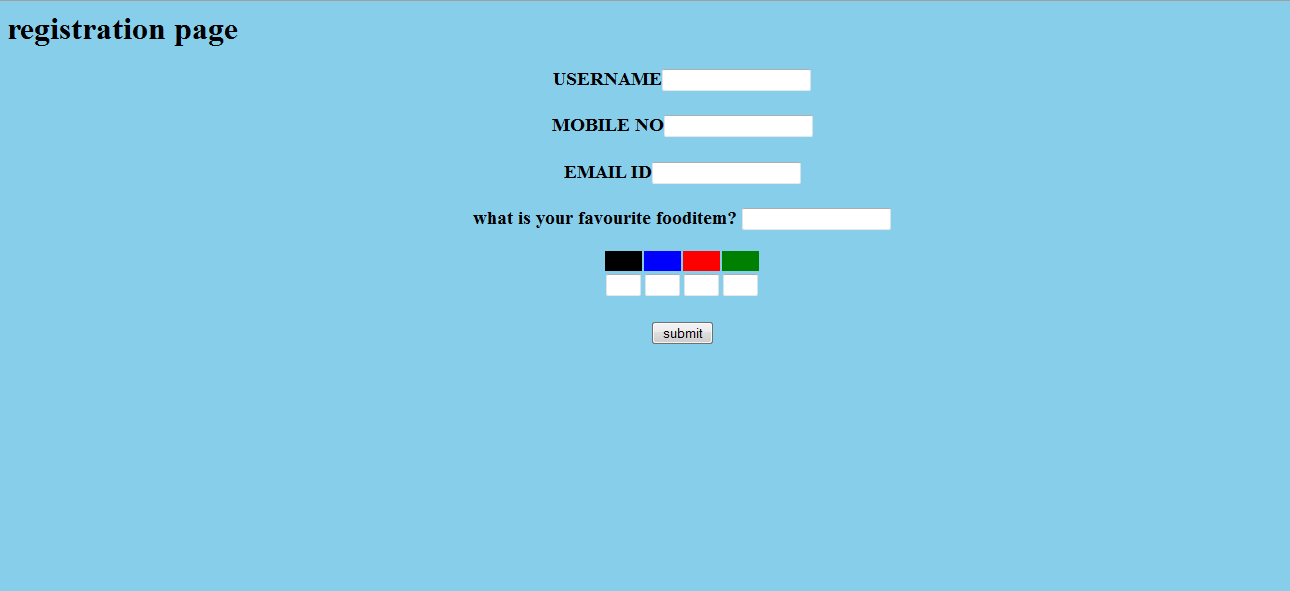
// response.sendRedirect("1.jsp");

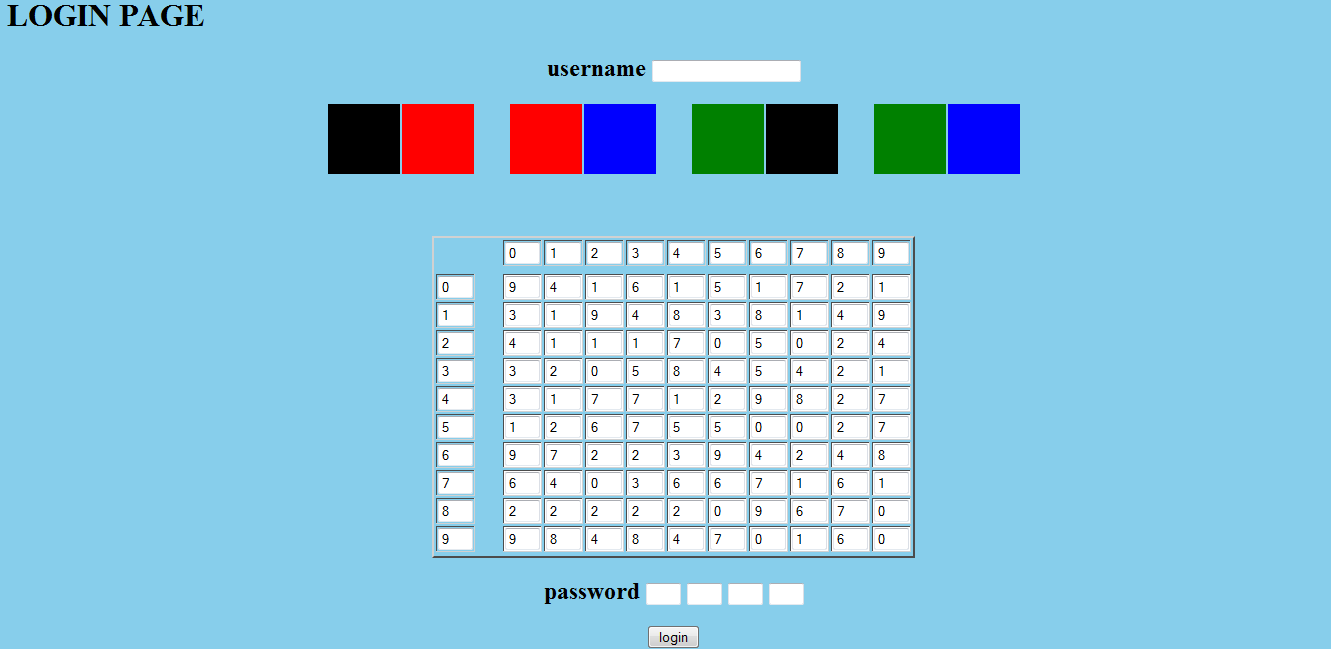
}

catch(Exception e)

{out.println(e);}

%>

****

****

**5.IMPLEMENTATION AND SERVICES**

**5.1 Implementation**

When testing authentication mechanisms with prototypes, it is reasonable for an

implementation to differ from that of a deployable system. The prototype is instrumented to record user behavior and other modifications, such as storing passwords unencrypted, may be necessary to evaluate the usability and security of the system more easily. However, it is also important to consider the impact of the proposed deployable implementation because it may introduce new usability or security problems.

However, for a real implementation, graphical password coordinates should not be stored \in the clear" but rather they are ideally cryptographically hashed to provide an additional layer of security in case the password \_le is compromised, similar as with regular text passwords. A simple solution is to overlay a static grid(potentially invisible to users) onto the image and associate each pixel with the grid square that contains it. The hashed password consists of the identifiers of the grid squares rather than the original pixels. During re-entry, if a click-point falls within the same grid-square as the original point, then the entry is accepted since its hashed value matches the original. However, using a static grid leads to the \edge problem":

if an original click-point is very close to a grid line, then during re-entry a click point

may be within tolerance but fall in an adjacent grid-square, and thus be rejected by the system because the hash values of the two points do not match. Therefore, more sophisticated discretization methods are required.

The graphical password schemes that we propose derive their strength from the following observation: a graphical interface for providing input enables the user to decouple the positions of the inputs from their temporal order. This is in contrast to textual passwords input via a keyboard: here, the temporal order in which the user types characters uniquely determines their position in the password. However, in a graphical password, e.g., consisting of several drawn lines, the final position of each line can be determined independently of the temporal order in which the lines are drawn. We show that this independence between input position and order can be used to build interesting new password schemes, and in some cases obtain authentication that is convincingly stronger than textual passwords but not significantly harder to remember. we present a password selection and input scheme which uses textual passwords augmented by some minimal graphical capabilities that enable the decoupling of temporal order of input and the position in which characters are input. This scheme is interesting because it simply demonstrates the power of graphical input abilities while yielding a scheme that is convincingly stronger than textual passwords are today.

**5.2 Steps of Deployment and Use**

1. Install JSP and SQL Server.

2. Create database and tables on SQL Server.

3. Run the program code.

4. Create a new account as shown in figure.

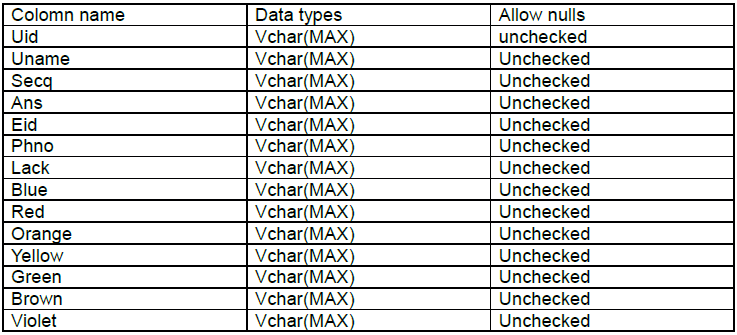
5. A registered user can directly login by typing his/her username and password as shown in figure.

6. If verification is success then, he/she can enter.

7. If the user lost his password they can retrieve it from the password recovery by

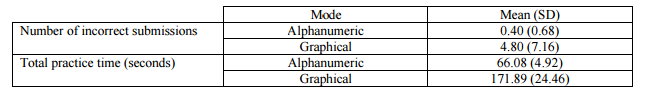
using security question.

Creation of Database:

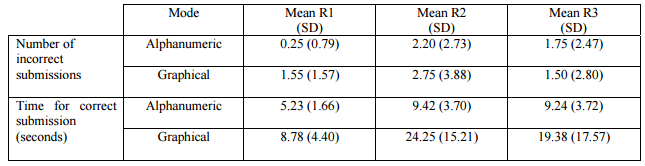


**5.3 Results Observed**

Here we report the quantitative result on the password learning and retention phases. We do not report results from the surveys of user perceptions. After choosing a password, participants practiced their password in the learning phase. The criterion for success was 10 correct logins. The participants continued to input the password until the criterion was met. We measured the number of incorrect password submissions and the total time spent in practice



In the retention phase participants input the password longitudinally three times. In each retention trial the participants had to enter their password correctly only one time. The number of incorrect password submissions and time for the correct submission are shown below.



**5.4 Advantages**

ADVATAGES OF GRAPHICAL PASSWORDS Graphical password schemes provide a way of making more human-friendly passwords while increasing the level of security. Here we use a series of selectable images on successive screen pages, if there are 100 images on each of the 8 pages in an 8-image password, there are 100^8, or 10 quadrillion (10,000,000,000,000,000), possible combinations that could form the graphical password! If the system has a built-in delay of only 0.1 second, it would take (on average) millions of years to break into the system by hitting it with random image sequences. Use of Captcha’s can be eliminated.

**Disadvantages**

DRAWBACKS THE SHOULDER SURFING PROBLEM As the name implies, shoulder surfing is watching over people's shoulders as they process information. Examples include observing the keyboard as a person types his or her password, enters a PIN number, or views personal information. Because of their graphic nature, nearly all graphical password schemes are quite vulnerable to shoulder surfing. Most of the existing schemes simply circumvent the problem by stating that graphical passwords should only be used with handheld devices or workstations set up in such a way that only one person can see the screen at the time of login.

**6.SECURITY DISCUSSION**

**6.1 Dictionary Attacks**

Attackers conduct dictionary attacks by identifying passwords with higher probability of being chosen by users and using this list to systematically try and guess passwords; in effect, attackers try to identify the effective password space (or portion thereof). This can dramatically improve the success ratio compared to an exhaustive attack, by lowering the expected number of guesses required for success. Dictionary attacks can be especially successful if entries are prioritized to test the most probable passwords first. The disadvantage of dictionary attacks is that they require more design and pre-computation than exhaustive attacks since some preliminary work must be done to identify candidate entries for the dictionary. Dictionary attacks can be conducted online or offline, in a similar manner to exhaustive attacks and the same security precautions apply.

When creating text passwords, users typically select real words and use variations such as adding digits to the beginning or end of the word, or replacing some letters with symbols. Forming an attack dictionary that includes entries with these characteristics is likely to yield some success. Programs such as John the Ripper employ these types of \word mangling" rules to create their dictionaries. In incremental mode, John the Ripper allows attackers to define additional rules to help prioritize guesses based on the particular type of passwords being attacked. Automated programs for guessing click-based graphical passwords are not widely available (compared to programs such as John the Ripper for text passwords); this is probably because these types of passwords are not widely deployed. However, as discussed in previous chapters, we have found that many Pass Points and CCP users also behave in a predictable fashion, so it is not unreasonable to expect that such software would be made available if click-based graphical passwords were deployed in practice. We now look at two strategies for creating dictionaries for click-based graphical passwords using hotspots and using geometric patterns.

**6.2 Shoulder Surfing Attacks**

Shoulder-surfing is a targeted attack against a specific user. It can occur when it is possible to observe someone entering a password, either through direct observation or through some external recording device such as a camera or video camera, perhaps with a telephoto lens. Recently published papers discuss the ability to gain information from computer screens through telephoto images of corrections on other items near the computer [7] and the ability to duplicate physical keys based on images from telephoto lenses 195 feet away [70]. Obviously, shoulder-surfing is a general security threat not unique to graphical passwords, but since they use visual output on the computer screen, graphical passwords are also susceptible to shoulder-surfing. Some recognition-based graphical passwords require that multiple successful logins be observed before the full secret can be deduced because only some of the user's portfolio images are displayed at each login or because the scheme does not require that users explicitly reveal the shared secret at login (e.g., as in Weinshall's scheme discussed in Section 2.4.3). However, most other types of graphical passwords can be gathered from observing or recording one successful login; click-based graphical passwords fall into this category. In their present form, we do not recommend that CCP or PCCP be used in environments where shoulder-surfing is a serious threat. With CCP and PCCP, an observer needs to record the images and the precise mouse-clicks on each of these images, then be able to accurately reproduce the series of click-points. Partial information, such as only capturing the images, does not reveal the password but does leak sufficient information to help attackers. Using these captured images, attackers can then mount a divide-and-conquer attack since they now know exactly what sequence of images they are trying to achieve. If conducting an online attack, this presumes that attackers have a sufficient number of guesses available for the particular account before being locked out by the live system. For example, let us assume that an attacker has learned the entire sequence of c = 5 images within a password but not the exact click-point locations. If the number of tolerance squares per image is s = 432, and we assume that all pixels are equally likely to be selected by users, we would expect that an attacker would need to guess 50% of tolerance squares on average before finding the correct one. The total number of guesses, therefore, would be G = :5 \_ c \_ s = 1080. The advantage for attackers is that they know when to stop guessing at each stage, so only need to try as many guesses as necessary to find the correct image before moving on to the next stage. This attack might be made more efficient by using hotspot or pattern information to prioritize their efforts. The sequence of images observed for one user is of little (or no) use to help attackers guess passwords for other accounts. This is because the subset of images and the mapping from one image to the next includes the username as a parameter, so knowledge from one account will not be transferable to other accounts. Pass Points passwords are also susceptible to shoulder-surfing. Attackers must gather information about the image and the precise click-points. We suspect, however, that it may be somewhat more difficult for attackers to gain partial knowledge of passwords from a distance, unless a telephoto camera or video camera is

used. If an attacker is too far away to see the mouse cursor, only the one pass Points image is visible, and this information would be available anyway by entering the username (if known) at the login screen of the live system. In this case, attackers are no further ahead, but can still mount exhaustive or dictionary attacks against this particular image. If an attacker can observe the mouse cursor movements and deduce where the user clicked, then the entire password is known and can be reproduced. Smaller tolerance squares may also reduce the risk of successful shoulder-surfing by either a nearby attacker observing the screen, or an attacker recording the screen using a high-powered telephoto camera lens [7, 70] (since the captured image maybe too blurry to accurately identify the mouse pointer tip). With smaller squares attackers must repeat mouse clicks with greater precision to correctly enter the password. Furthermore, observing mouse cursor movements alone may not reveal exactly where the user clicked since the user may not necessarily stop moving the cursor with every click, especially when familiar with the pattern of mouse clicks. With CCP and

PCCP, attackers who can clearly see the mouse pointer may be able to identify the last position of the mouse immediately before the next image appeared; this could be partially addressed by adding a (short) random delay before the next image appears. While attackers may be able to approximate the password, they are more likely to require several guesses and to run out of login attempts before finding a match than if the system allowed for larger tolerance areas.

Existing shoulder-surfing resistant or shoulder-surfing immune graphical password systems have major usability drawbacks, usually in the amount of time

and effort it takes to log in; as such they are typically not viable alternatives for every-day authentication. Click-based graphical passwords could be made more shoulder-surfing resistant by reducing the size of the images (which, however, consequently also reduces the size of the theoretical password space) or by manipulating the image and cursor on the screen, such as reducing the amount of contrast, to reduce the risk that observers can identify them from far away. These would need to be user-tested to ensure that the usability of the system remains acceptable. Eye tracking has also been proposed as a shoulder-surfing resistant method of user input [68]. By entering a password using only eye gaze, no mouse cursor needs to be visible on the screen. With CCP and PCCP, however, the sequence of images may still be observed even if eye-tracking was used as an input device. Preliminary(unpublished) experiments by members of our group have revealed that eye tracking is not yet sufficiently accurate to be a viable approach. Furthermore, it is un clear if advances in the technology will improve precision enough for graphical password entry using eye tracking, or whether characteristics of human vision make eye tracking inherently imprecise. We are pursuing this line of inquiry, but it is beyond the scope of this thesis.

**6.3 Phishing Attacks**

Phishing is type of attack where attackers convince users to reveal their credentials at a malicious website, typically designed to look like a legitimate site for which the user has an existing account. Attackers can then use these credentials to impersonate the user at the real website. For text passwords, often only a reasonable copy of a website's login page is needed along with a means of luring users to the site. The attacker gathers the username and password from the phishing site and enters it at the legitimate site. Users are typically led to the phishing site by a forged email, appearing to come from the legitimate company.

For CCP and PCCP, a more active role is necessary to capture the user's credentials through phishing. The attackers need to know the correct sequence of images to display in response to user input; something they do not know ahead of time. This is most commonly accomplished through a \man-in-the-middle" attack: the phishing website gets the username from the user, enters this username into the real website, retrieves the user's first image from the real website, displays this image on the phishing website, captures the user's first click-point, transmits that information to the real website, and so on. In effect, the attacker acts as a relay, intercepting all information to and from the user and the real website, and in the process succeeds in logging on to the legitimate website.

Although CCP and PCCP are susceptible to phishing when used in conjunction

with a man-in-the-middle attack as described above, this is a more challenging attack than for text passwords (and Pass Points, as shown next). With Pass Points, the attacker must also know which image to display to the user on a phishing site before the user can log in. However, this image can be retrieved by entering the username (if known) at the legitimate site. The attacker may do this in real-time, as soon as the user enters the username at the phishing site. Although this is also a man-in-the-middle attack, only one contact is needed with the legitimate site during the attack, to retrieve the one Pass Points image. Attackers can then collect the click-points for later (or immediate) use at the legitimate site. Alternatively, a phishing site could pre-fetch the Pass Points images of the users it is targeting, if the usernames are known, and store them on the server. If one of these users is lured to the phishing site, the system can display the correct image immediately, without having to use a man-in-the-middle attack or having to access the legitimate site in real-time.

**6.4 Social Engineering Attacks**

Phishing is a specific type of social engineering attack, but social engineering can include any means of manipulating users into revealing their credentials for malicious purposes, such as phone calls from a fake help desk or credit card company. While these types of calls may require some background work to seem legitimate to users, it is often easier to convince users to reveal their password or other confidential information than it is to break into the system through other means. Text passwords and other types of alphanumeric information are relatively easy to share with attackers (or friends) since they can be spoken or written down. Click-based graphical passwords are more difficult to share, even if a user is tricked into trying to do so. First, the user and the attacker must coordinate a frame of reference describing the image in enough detail so that the attacker (masquerading as a well intentioned associate, in most cases) understands the descriptions of the click-points on the image. With Pass Points, the user must first remember the image, unless it is in front of them, describe the image, and identify the 5 click-points. Dunphy et al. conducted on a preliminary study where the experimenter described a password to a participant, who tried to enter the password based on the description. They report that 4 out of 5 participants were successful. The scenario is somewhat artificial, however, since the experimenter and the participant were looking at the same screen, so had a common frame of reference. CCP and PCCP passwords are more difficult to reveal; users must somehow explain the exact location of their click-points based on characteristics of the images, after first ensuring that the other party is in fact looking at the correct image. The user and attacker must reorient themselves with each image and click-point. And, unless the user is also entering their password, the user must remember 5 images in enough detail to provide accurate descriptions. Although this is a security advantage, it does have usability drawbacks because it also means that users cannot receive a

reset password by phone, for example. If we consider other means of sharing the password, obvious methods include drawing and taking photos or screen shots. It would be difficult to get the required accuracy by drawing, and it assumes that the user somehow shares the drawings with the attacker. A more efficient way of accurately sharing a click-based graphical password is to take screen shots of the images with the mouse cursor (or other indicator) in the correct positions to identify the click-points. These would need to be passed on to the attacker, perhaps by email. If the password must be transferred through electronic means, then a phishing attack is likely to more believable and simpler to accomplish than other types of social engineering attacks. If taken offline, users could print screen shots of their images, mark the click-points with a pen, and share these printed copies (or put them away for backup purposes). Overall, it appears that CCP and PCCP passwords would be moderately more difficult to gather through social engineering attacks than Pass Points, and significantly more difficult than text passwords.

**6.5 Malware Attacks**

Malware includes any unauthorized programs running on a computer. These can

collect information from the hard drive or directly from the user's input, and transmit this information back to attackers, or make it available for retrieval.

Key-loggers can capture and keep a log of the user's typing, and as such can record text passwords. Attackers can then look through the captured data \_le (log), identifying likely usernames and passwords. Key-loggers do not provide enough information to reveal most graphical passwords, unless the scheme exclusively uses keyboard entry. To collect Pass Points, CCP, and PCCP account information, an attacker would need to capture the user's keystrokes to collect the username, screen information for determining the image and its position on the screen, and mouse clicks to know when a click-point has been selected since cursor movement alone may not reveal the exact location of the click-points. This information would then need to be synchronized to accurately determine which mouse clicks correspond to password click-points on specific images. Although feasible, this is a more difficult than simply recording the keyboard input. A screen-scraper would be needed to collect the screen information, a mouse-logger to record mouse clicks for the exact location of the click-points, and then the two would need to be synchronized in time. Alternatively, it may be possible for mouse-loggers to also capture information about the position of windows on the screen and use this information to determine image positions without the need for a screen scraper. We expect that if click-based graphical passwords became popular, then malware collecting the necessary information would soon follow. Compromised computers are a significant threat against all of a user's information and computer resources, not only against a user's login credentials. This is a general security problem that will affect every authentication mechanism if used from an unsecured computer or using insecure communication channels. If there is malware on the end-user computer, then it is safest to assume that all resources and communications are compromised.

**7. STRATEGIES**

**7.1 Major Contributions**

To meet the first objective, we reviewed existing graphical password schemes by

cataloguing them according to several usability and security characteristics. While we uncovered a wide variety of approaches to graphical passwords, we discovered that there was little consistency in how these systems were presented or evaluated. We also found that many were not thoroughly assessed from both usability and security perspectives. To our knowledge, the most recent surveys of graphical passwords in the peer-reviewed literature were published in 2005 [77, 115]; our work provides a more comprehensive summary and includes recent work in the area. To address the second objective, we conducted usability and security analysis of Pass Points. We initially carried out a user study in the lab, followed by a large field study where students from three classes used Pass Points to access online material for approximately two months. In our initial analysis, we show that image choice impacts the usability of Pass Points, that users are extremely accurate in entering their click-points, and that login times and success rates are generally good. In later analysis of the Pass Points datasets, we show that users select passwords that form simple geometric patterns and that the click-point distributions have significant amounts of clustering. Both of these results indicate that attackers may be able to predict

passwords with higher likelihood of being chosen, and then use this information to launch efficient dictionary attacks. The third objective was met by designing, prototyping, and testing two novel click-based graphical password schemes: Cued Click-Points (CCP) and Persuasive Cued Click-Points (PCCP). These were intended to further increase memorability and usability, as well as increase security when compared to Pass Points. We conducted a lab study of each scheme, showing a significant improvement in the randomness of user chosen passwords. Indeed, both showed remarkable decrease in occurrence of geometric patterns, and PCCP additionally showed significant decrease in click-

point clustering. On the measures we used to evaluate patterns and clustering, the PCCP dataset was similar to the randomly generated datasets. The two schemes have additional security benefits, due to the large number of images that attackers would need to discover, collect, and analyze in order to launch successful guessing attacks. The same characteristics that render CCP and PCCP apparently more secure also make them more usable. The use of implicit feedback helps users recognize when, and at which stage, they made a mistake during login. One-to-one cueing helps with memorability of the passwords, as evidenced by the high login success rates and quick password entry times, even though the passwords were more resistant to the attacks considered than Pass Points. As part of meeting the third objective, we also created centered discretization, a new method for the discretization of click-based graphical passwords. Centered discretization ensures a uniform tolerance area around a click-point; this is a feature that we believe is a major improvement over robust discretization. In our post-hoc analysis, we compare centered discretization and robust discretization.

Our results show that centered discretization eliminates what we define as false positives and false negatives that occur with robust discretization. Our algorithm allows for smaller tolerance areas, which increases the theoretical password space, and better usability because the system behaves in a manner consistent with user expectations. To meet the fourth objective, we identify what we believe are the precise mechanisms that lead to increased usability and security in CCP and PCCP. We believe that these four design strategies can be generalized and are applicable to other knowledge-based authentication schemes. The concept of implicit feedback addresses an important issue in usable security: the need to provide feedback to users without also helping attackers. Implicit feedback provides feedback that is only meaningful to users who already have knowledge of the correct password; the same feedback reveals nothing to those who are do not know the password. In one-to-one cueing, the system offers a cue to help users remember each component of their password. Each cue helps to trigger the specific memory associated with that cue. Our third design principle uses concepts from Persuasive Technology to encourage users to select less predictable passwords by making this behavior the safe-path-of-least-resistance. The last design principle addresses the issue of matching the user's expectations of system behavior and discusses how a disconnect between system performance and a user's mental model can lead to usability and security problems.

**7.2 Minor Contributions**

This research also produced several minor contributions. Although these were not directly mandated by our objectives, they provide advancement in the area of graphical password research. With our empirical studies of Pass Points, we provide evidence confirming some of the usability results first reported by the original Pass Points authors . We also provide evidence contradicting some of the earlier findings. Our results suggest better usability than initially thought with respect to accuracy in targeting click points; this property could be harnessed to increase the theoretical password space. We also clarify that the prototype system used by Wieden beck et al. for Pass Points did not implement robust discretization. Their systems instead used a centered tolerance approach to verifying click-points, which means that their results do not take into account the variations in system behavior that could have impacted usability. In Chapter 6, we show that robust discretization would have reduced the reported usability by significantly increasing the number of falsely rejected login attempts. Our field study of Pass Points provides the first look at the memorability effects of multiple password interference. We found that users having two passwords (one or each of two accounts) had lower login success rates than those who only had to remember one password. This raised further questions about whether memorability

was better for graphical passwords than text passwords when multiple Passwords needed to be remembered. One observation of our cataloguing efforts was that there is lack of consistency in user studies conducted to evaluate graphical password schemes. As a result, it is very difficult to get an accurate comparison of the usability and security of the different schemes. In our work, we have described in detail and used the same methodology and evaluation criteria for all three schemes that we evaluated, allowing for more

precise comparison. In our analysis of user choice in password selection, we introduced and utilized point pattern analysis from spatial statistics to determine and compare the clustering in point patterns that arise in graphical passwords. This approach is typically used in earth sciences and biology. These methods allow for statistical comparison of click points from each study, and for comparison with randomly generated click-point data that simulates the click-point distribution found in the theoretical password space.

**7.3 Conclusions**

we present the results of two usability studies of Pass Points. The initial lab study revealed mostly positive results and led to a larger field study to see how Pass Points worked in practice. The lab study confirmed earlier work that the usability of these passwords was good in terms of success rates and password-entry times. We additionally showed that participants were more accurate in targeting their click-points than previously suggested, indicating that smaller tolerance squares may be acceptable. Finally, contrary to previous work, we found that the choice of image significantly influenced success rates. The field study represented the first large-scale, real-world study of click-based graphical passwords presented in the literature. Password entry times were acceptable, accuracy was not quite as high as in-lab but still very good, success rates improved with practice, and participants continued to use the system even though they

could easily have switched to a text password. However, we found several legitimate concerns with adopting Pass Points as a means of authentication. We provided the first empirical evidence that interference from having to remember multiple graphical passwords is problematic. Participants also reported using patterns in selecting their passwords, suggesting increased susceptibility to guessing attacks. The differences between the lab and field studies also raise methodological concerns in usable security. So far, lab studies are the most common form of usability evaluation and while others have cautioned that these were inadequate in providing realistic usability data, our two studies provide empirical evidence of this problem. We do not suggest that lab studies be eliminated. They offer a relatively quick and cost effective way to test new ideas to \_nd which are more promising by allowing for initial usability and security evaluations. However we caution that real usage may vary from lab usage and that field studies are an important second step to confirm results found

in the lab. An important usability and security goal in authentication systems is to help users select better passwords and thus increase the effective password space. Our earlier Pass Points studies revealed memorability issues and security concerns because users selected click-points that formed hotspots, making it possible to conduct successful dictionary attacks with minimal effort. CCP was designed to address these issues by using one-to-one cueing, adding implicit feedback, and increasing the number of images used to proportionally increase the effort required to perform hotspot analysis. However, hotspots were still occurring in CCP. We designed PCCP to encourage and guide users in selecting more random click based graphical passwords. A key feature in PCCP is that creating a secure password is the \safe-path-of-least-resistance", making it likely to be more effective than schemes where behaving securely adds an extra burden on users. The approach has proven effective at reducing the formation of hotspots and avoiding known hotspots, thus increasing the effective password space, without significantly affecting the memorability of passwords.

Due to our interest in usable security, in this thesis we have focused our analysis on dictionary attacks because their success is a direct result of user choice in password selection. Our general intent in designing CCP and PCCP was to find ways of increasing memorability of passwords while decreasing predictability.

The best measure of predictability is to examine the passwords for patterns (as done in Chapter 7) and common traits (such as hotspots) that may reduce the effective password space. we identified and provided an overview of several other threats to authentication mechanisms and discussed how these may affect our proposed click based graphical password schemes. Table 8.2 summarizes CCP and PCCP's features based on the same security characteristics as the other graphical password schemes reviewed . For completeness, we included which covers the usability characteristics also covered in the same section.

We find that CCP and PCCP appear to be more secure against dictionary attack

than Pass Points and text passwords. CCP and PCCP may require more sophisticated strategies than Pass Points for phishing attacks. With respect to other types of attacks, CCP and PCCP appear no more susceptible than other schemes, with the possible exception of shoulder-surfing.

Our general goal in this thesis was to increase the memorability and security of

knowledge-based authentication schemes. We focused on click-based graphical passwords. We were successful at designing innovative schemes that improved memorability and that were more secure than existing alternatives. From this empirical work, we identified the key features of our designs and derived design strategies that we believe are applicable to other knowledge-based authentication schemes. The relationship between usability and security is a complex one; too often, improvements in one lead to a reduction in the other. As we have shown, it is possible to increase both simultaneously through careful design that considers usability and security in combination. We emphasize the need for thorough usability and security evaluations because system design can significantly impact user behavior, sometimes in unanticipated ways, which in turn can significantly impact the security of a system.

**8. TESTING**

**8.1 Testing**

One of these has to do with the “Login page password guessing attack” that is in the module, Html\_Authentication\_Audit.script. This script is truly a brute-force attack against a login page, and does not assume that the hacker knows the username of an authentic user of the website. Therefore, even if you lock out a username that is in your database of acceptable users, this particular script has most likely chosen a username or usernames that are not in your database.

First, consider the case that your website locks out known user accounts after some number of failed password attempts. That is, the username entered belongs to a username in your database. After some number of failed logins, your website locks that user out.

Let us assume the following

Your login page is using TLS or an acceptable variant, so that, for example, if your users are logging in from an unprotected wireless internet café, your login page request is not broadcasting their username and password to the entire café.

After some number of failed password attempts, you are locking out usernames that you know, at least for some time.

Let there be a hacker! I hereby name my hacker, Hackster. I like to give Hackster perhaps more credit than Hackster is due. The first thing Hackster is going to do is attack the ‘forgot password’ page and see if there is a different response for ‘I cannot find that email address in my list of accounts’ and ‘Hello dude. We sent you a new password!’ If Hackster gets the latter, then he no longer needs to brute force the username, and instead he is going to brute force the ‘forgot password’ page until he has a list of usernames. Suppose Hackster tried that and found out the ‘forgot password’ page was perfect.

Hackster is going to try usernames and passwords both chosen randomly hoping that the combination will hit before the known username lockout hits. Alternatively, Hackster is going to try the same random username with random passwords. What are the odds Hackster succeeds?

Suppose your password policy is weak. You allow 8-character passwords and you do not care what they look like. Hackster has a 50:50 chance of guessing the password after trying 267 passwords. Hackster is no fool and he can easily afford to rent 1000 zombies to attack your website. Supposing it takes 1/5 second to get a response, Hackster will have tried all of those 8,000 million combinations in 18.6 days. Hackster won in less than a month.

Therefore, if the username has an account, it might make sense to lockout that account after, say, 10 bad password attempts. That means that on average across his 1000 zombies, Hackster is going to try a new username every couple of seconds. Hackster may not know during the start of his attack that a lockout will occur, but eventually he will discover that some usernames result in a lockout after 10 bad passwords, while others do not. Hackster will therefore conclude that the usernames that are locked out belong to real users. After that, if the username does not belong to an account, Hackster is also going to stop trying passwords after 10 or 11 bad password attempts, or Hackster is dumber than I thought and is going to waste bandwidth on a futile cause. In the latter case, I would let dumb Hackster do that. Smart Hackster is rolling over to a new username, on the average, every 2 seconds, whether he hit a real username or not. Hackster is hoping for the payoff that one of his combinations will hit. Hackster is trying username and password pairs, so really his chance of guessing is for that combination which is 2615.  That means we multiply his 18.6 days by 208 thousand-million to arrive at 3.9 million-million days. Smart Hackster has probably done the math and given up.

Hackster has probably considered whether to hit the site with the 11th password attempt or not. If he does, then he knows whether the account was real or not, but then on the average he has also generated a help-desk call from the real user every 18-36 days. In the latter case, Hackster has a list of real usernames, and can go back after awhile and try another 10 passwords. (Note that there is a clue to your Incident Response Team there. If the user does not call in soon after being locked out, then there is likely an incident in progress or Hackster is also employing social-hacking techniques—those being out of scope of “brute-force”).

Dumb Hackster, however, is generating a help-desk call every 18 days because dumb Hackster forgot to do the math.

Because of the lockout policy, only on real accounts, the website has a reasonable chance of withstanding a brute force attack even with a rather poor password policy.

We assumed that the round-trip time was 1/5 second. The calculated days are linear with that time so if the round-trip time is even another 1/5 or 1/10 of that time then the brute-force time does not get a lot better. It does, however, mean that the help desk call frequency will increase by roughly the same factor. Therefore, some attacks could lock out a user every few days.

The previous analysis assumed usernames were relatively simple. Usernames are often case-insensitive, but applying policies to username lengths is not a common practice. Therefore, the username factor of 268 might be as small as 264 or maybe even 263. If Hackster can target the 3-6 range length of usernames then Hackster could be generating 264 times as many lockouts as described above. Instead of one every few days or few weeks, Hackster is generating 25,000 a day! It probably depends on how many short usernames there are. Hackster might run out of all of them in a few days or weeks, but your help desk is swamped during that time!

If your website has a policy against short usernames or requires full email addresses that are unique as username logins, then a lockout policy on an existing username is quite successful at thwarting a brute-force attack even if the login page simply ignores non-existing username attempts.

Suppose that you have no lockout on accounts. Is there a way to deal with brute-force attacks without a lockout?

As described above, depending on the turnaround to your site, a poor password policy can result in Hackster breaking in as little as 18 days if Hackster already knows the username of the account.

First, we consider hardening the password policy. Suppose you have a password policy of 12 characters, one of them has to be a number, and of mixed case. Hackster has a 50:50 chance of getting in within a window of 6211 possible tries. That comes out to about 52 million-million-million tries.

Suppose we penalize a bad answer with a time-out. Hackster is not going to stick around waiting for a response unless every response (good or bad) takes the same amount of time. Therefore, suppose our website has built-in a login delay of 2 seconds with a standard deviation of a half-second. In other words, 90% of the time a user is waiting at least a second to log in, and only 10% are waiting longer than 3 seconds (you might even truncate the distribution around those 10%-90% marks, or use a constant delay).

Hackster’s machines have to wait 2-1/2 seconds or longer to be more than 70% sure that the login was unsuccessful. We stick with the worst case that Hackster picks 2 seconds because he is not as smart as I thought he was (or your delay is not random). Let us give Hackster a million threads or a million zombies or combination thereof. Since Hackster has to wait at least 2 seconds before moving his thread on to the next try, his thread/machine will wait. Well, with the bad password, Hackster is still going to win in 186 days. With that stronger password, Hackster takes over 3 million years to do the same thing!

For purists, the Gaussian distribution of delay times is better, because your internal algorithms may take slightly longer one way or the other, and the Gaussian delay would tend to hide that time difference a bit better than would a constant time delay. Beyond that, you should monitor your logs and if you are under attack then start looking to see if there are IP addresses that can be blocked.

## Conclusions

If your website has a strong password policy, and pauses a second or two before responding to a login attempt (successful or not) then a brute-force attack has a very low probability of success.

If your website has a lockout policy for known users after some number of sequential login failures, then a brute-force attack has a low probability of success. However, as username complexity decreases, the number of help desk requests to reset accounts increases and could become overwhelming.

If your website has a lockout policy for know users, then unless the login page responds in exactly the same way to the Xth attempt of an unknown user then your website is leaking usernames to hackers.

The lockout did hurt Hackster. After 10 failed passwords, Hackster moves on to the next username on the list. However, your phones are ringing and users are inconvenienced. The next step would be to unload the help desk by resorting to a 2ndary failed Xth-attempt login page that asks for mother’s maiden name or presents a CAPTCHA.

**9.BIBLIOGRAPHY**

[1] A. Adams and M. Sasse. Users are not the enemy. Communication of the ACM,(12):41{46, 1999.

[2] F. Alsulaiman and A. El Saddik. A novel 3D graphical password schema. In

IEEE International Conference on Virtual Environments, Human-Computer

Interfaces and Measurement Systems, July 2006.

[3] American Psychological Association. Publication Manual of the American Psy-

chological Association. American Psychological Association (APA), 5th edition

edition, 2001.

[4] J. Anderson and G. Bower. Recognition and retrieval processes in free recall.

Psychological Review, 79(2):97{123, March 1972.

[5] R. Anderson. Why cryptosystems fail. In 1st ACM Conference on Computer

and Communications Security, December 1993.

[6] D. Andrews, B. Nonnecke, and J. Preece. Electronic survey methodology: A

case study in reaching hard-to-involve Internet users. International Journal

of Human-Computer Interaction, Lawrence Erlbaum Associates, 16(2):185{210,

2003.

[7] M. Backes, M. Durmuth, and D. Unruh. Compromising reections | or | how

to read LCD monitors around the corner. In IEEE Symposium on Security and

Privacy, 2008.

[8] A. Baddeley and R. Turner. R. spatstat: An R package for analyzing spatial

point patterns. Journal of Statistical Software, 12(6):1{42, 2005.

[9] J. Birget, D. Hong, and N. Memon. Graphical passwords based on robust

discretization. IEEE Transactions on Information Forensics and Security,

1(3):395{399, 2006.

[10] G. Blonder. Graphical passwords. United States Patent 5,559,961, 1996.

[11] I. Britton. Freefoto website. http://www.freefoto, accessed February 2007.

[12] A. Brodskiy. Personal communication, September 3 2006.

[13] S. Brosto\_ and M. Sasse. Are Passfaces more usable than passwords? A \_eld

trial investigation. In British Human-Computer Interaction Conference (HCI),

September 2000.

180

181

[14] S. Chakrabarti and M. Singhal. Password-based authentication: Preventing

dictionary attacks. Computer, IEEE Computer Society, 40(6):68{74, June 2007.

[15] S. Chiasson, R. Biddle, and P. van Oorschot. A second look at the usability

of click-based graphical passwords. In 3rd Symposium on Usable Privacy and

Security (SOUPS), July 2007.

[16] S. Chiasson, A. Forget, R. Biddle, and P. van Oorschot. Inuencing users

towards better passwords: Persuasive Cued Click-Points. In Human Computer

Interaction (HCI), The British Computer Society, September 2008.

[17] S. Chiasson, A. Forget, R. Biddle, and P. van Oorschot. User interface design

a\_ects security: Patterns in click-based graphical passwords (Manuscript under

submission). Technical Report TR-08-14, School of Computer Science, Carleton

University, 2008.

[18] S. Chiasson, A. Forget, E. Stobert, P. van Oorschot, and R. Biddle. Multiple

password interference in text and click-based graphical passwords. (Manuscript

under submission). Technical Report TR-08-20, School of Computer Science,

Carleton University, September 2008.

[19] S. Chiasson, J. Srinivasan, R. Biddle, and P. van Oorschot. Centered discretiza-

tion with application to graphical passwords. In USENIX Usability, Psychology,

and Security (UPSEC), April 2008.

[20] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of

two password managers. In 15th USENIX Security Symposium, August 2006.

[21] S. Chiasson, P. van Oorschot, and R. Biddle. Graphical password authentication

using Cued Click Points. In European Symposium On Research In Computer

Security (ESORICS), LNCS 4734, pages 359{374, September 2007.

[22] L. Coventry. Usable biometrics. In L. Cranor and S. Gar\_nkel, editors, Security

and Usability: Designing Secure Systems That People Can Use, chapter 10,

pages 175{197. O'Reilly Media, 2005.

[23] F. Craik and J. McDowd. Age di\_erences in recall and recognition. Journal

of Experimental Psychology: Learning, Memory, and Cognition, 13(3):474{479,

July 1987.

[24] L. Cranor. A framework for reasonning about the human in the loop. In

USENIX Usability, Psychology, and Security (UPSEC), April 2008.

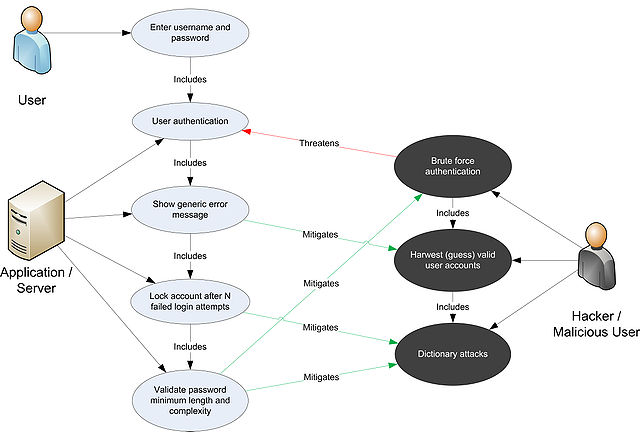
[25] L. Cranor and S. Gar\_nkel. Security and Usability: Designing Systems that

People Can Use. O'Reilly Media, edited collection edition, 2005.

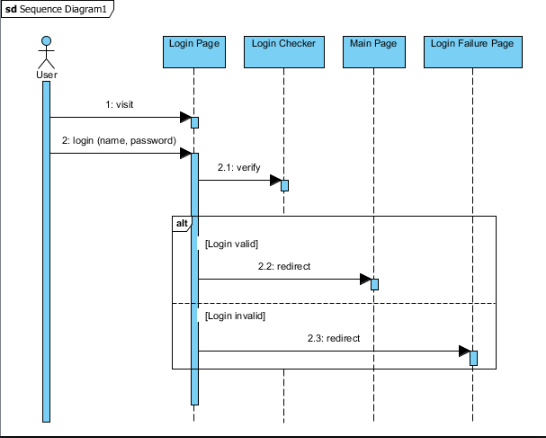
**APPENDIX**

**UML Representations**

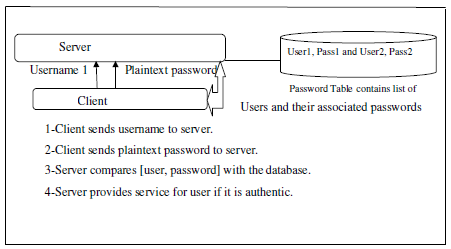
**1. Use Case Diagram**



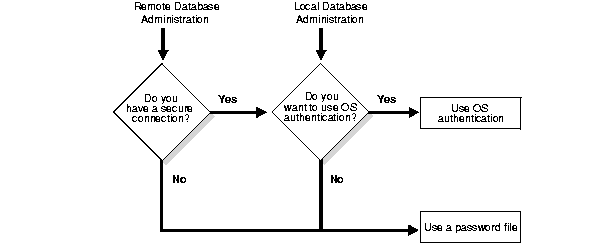
**2. Sequence diagram**



**3.Traditional Approach**



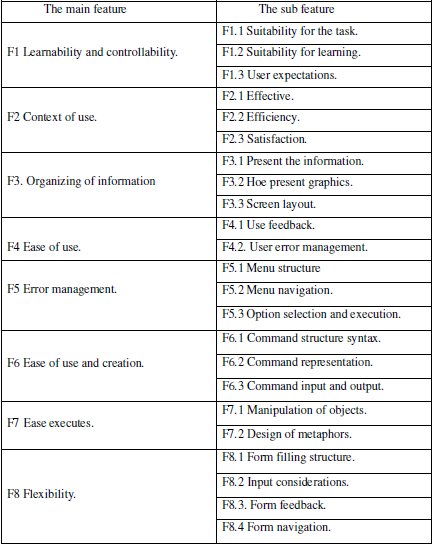
**4.Connection with Database**



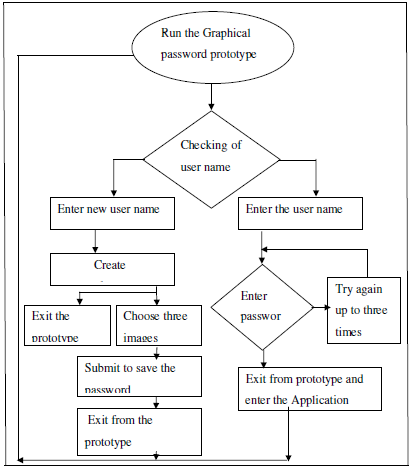
**5.Differences between Qualitative and Quantitative Approach**

|  |  |
| --- | --- |
| **Qualitative Approach** | **Quantitative Approach** |
| Develops theory | Develops theory and tests theory as well. |
| Describes meaning or discovery | Establishes relationship or causation |
| uses communication and observation | Uses instruments |
| Uses unstructured data collection | Uses structured data collection |

**6.Differences Between Main and Sub Features**



**7.Graphical Password Prototype**



**8.Formulation And Stages**

