3-2018

Implementation of Captcha as Graphical Passwords For Multi Security

Babu Rajeev Maddipati
bmaddipati@stcloudstate.edu

Implementation of Captcha as Graphical Passwords for Multi Security

by

Babu Rajeev Maddipati

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Information Assurance

February, 2018

Abstract

To validate human users, passwords play a vital role in computer security. Graphical passwords offer more security than text-based passwords, this is due to the reason that the user replies on graphical passwords. Normal users choose regular or unforgettable passwords which can be easy to guess and are prone to Artificial Intelligence problems. Many harder to guess passwords involve more mathematical or computational complications. To counter these hard AI problems a new Captcha technology known as, Captcha as Graphical Password (CaRP), from a novel family of graphical password systems has been developed. CaRP is both a Captcha and graphical password scheme in one. CaRP mainly helps in hard AI problems and security issues like online guess attacks, relay attacks, and shoulder-surfing attacks if combined with dual view technologies. Pass-points, a new methodology from CaRP, addresses the image hotspot problem in graphical password systems which lead to weak passwords. CaRP also implements a combination of images or colors with text which generates session passwords, that helps in authentication because with session passwords every time a new password is generated and is used only once. To counter shoulder surfing, CaRP provides cheap security and usability and thus improves online security. CaRP is not a panacea; however, it gives protection and usability to some online applications for improving online security.

Keywords: Graphical password, password, CaRP, Captcha, dictionary attack, password guessing attack

Table of Contents

List of Figures

Figure Page

Figure Page

Chapter I: Introduction

Using hard Artificial Intelligence is a problem for security, Captcha, creates a new paradigm which differentiates human users from computers (bots) (von Ahm, Blum, Hopper, & Langford, 2003) i.e., through the use of a puzzle, which bots cannot identify but a human user can. This Captcha is mainly used to provide internet security for email from bots. But this has not gotten much success with cryptographic primitives based on hard mathematical problems. To ensure security there are different ways, one is CaRP (Captcha as Graphical Password) from the family of graphical password schemes which mainly deal with hard AI problems.

CaRP is a click-based graphical password, where a sequence of random clicks on an image is used to derive a password, and the images generated by CaRP are a challenge and a new image is generated for every login attempt. Users are asked to click on the image or any part of that image as a password and these points or images are then stored as a graphical password. These images are different for every user. The newly generated graphical password is used along with the regular user password. Any Captcha can be converted to a CaRP, although only if the Captcha implies multiple-object classification. CaRPs can be implemented on both text and image recognition Captcha. The one with text CaRP, where the password is a text password, is a sequence of characters but only by clicking on the right character sequence on a CaRP image is the password authenticated. CaRP provides protection from online dictionary attacks on passwords which have been used for a long time and present a serious security threat (HP TippingPoint DVLabs, 2010). Online dictionary attacks are considered a main cybersecurity risk. To countermeasure this logon attempts do not work for two reasons: (a) it causes denial-of-service attacks (Wolverton, 2002), and (b) it is vulnerable to global password attacks which mean hackers try to break into any account rather than one (Pinkas & Sander, 2002) by trying each password on multiple accounts while they make sure that the attempts are under the threshold to avoid an account lock.

CaRP can be used to prevent relay attacks where Captcha challenges were meant to be solved by humans. CaRP also can prevent shoulder surfing attacks if it is combined with dual-view technologies.

CaRP applications include:

- CaRP can be applied on touch-screen devices where typing passwords are difficult especially for e-banks to secure internet applications. Captcha is mainly used for user logins in major e-banking systems (Li et al., 2010).
- CaRP can protect from spammers, which increases the spammer's operational costs and also reduces spam emails. CaRP helps to protect against spam bots from entering an email account even if they know the password.

Graphical Passwords

All the graphical password systems are classified into three categories, based on memorizing and entering passwords: identification, recall, and cued recall

- Recognition-based scheme
- Recall-based scheme
- Cued-Recall based scheme

Recognition-based scheme. A recognition-based scheme focuses on identifying, which are decoys or the visual objectives belonging to a password portfolio. PassFaces (Real User Corporation, 2012) is one scheme where the user selects a portfolio of faces from the database while creating a password. During authentication, the user asks to select a face from the panel of faces from his/her portfolio. This will be done over several rounds, for every round a different panel is utilized. To successfully login, a user must select the correct face in each round. The images are the same for logins, but their locations are changed for each round. Story (Davis, Monrose, & Reiter, 2004) is another schema, which is like PassFaces, but the images a user selects while creating the password are in order and the user must provide the images in the correct order. Another scheme, Déjà Vu (Dhamija & Perrig, 2003), is also similar but instead of a portfolio, the computer generates “random-art” images. Here the user should provide a path through a panel of images, for example: starting from the top-left image, moving down if the image is in her portfolio, or to the right otherwise (Weinshall, 2006). This process repeats with a different panel each time. To perform a successful login, the user needs to give correct answers with the cumulative probability that they were not entered by chance and exceeds a threshold within a given number of rounds.

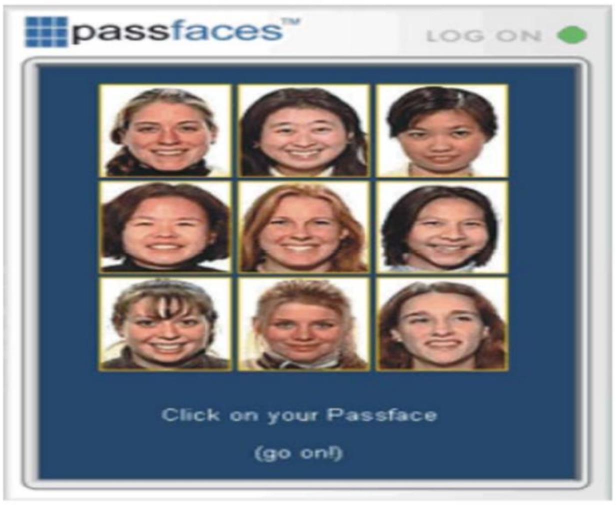


Figure 1. Passfaces

Recall-based scheme. In a recall-based scheme, it asks the user to regenerate the same interaction result without cueing. The first recall-based scheme proposed was Draw-A-Secret (DAS) (Jermyn, Mayer, Monrose, Reiter, & Rubin, 1999) which asks the user to draw password on a 2D grid (see Figure 2). The system analyses the grid cells and drawing path of user-drawn password. Pass-Go (Tao & Adams, 2008) is an improved version of DAS, where this analysis grid intersects points instead of grid cells. BDAS (Background DAS) (Dunphy & Yan, 2007) is also an improved version where is adds background images to DAS, which helps to create more complex passwords.

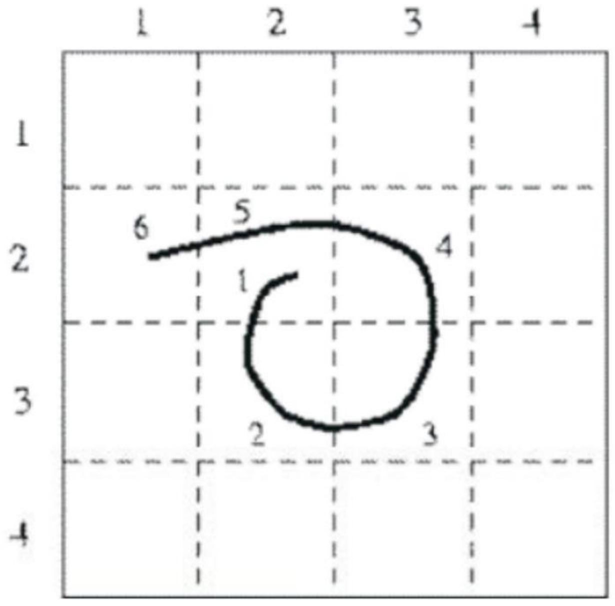


Figure 2. Draw-A-Secret (DAS)

Cued recall-based scheme. In the cued-recall scheme, the system provides some external clue which helps the user to memorize and enter a password. PassPoints (Wiedenbeck, Waters, Birget, Brodskiy, & Memon, 2005) is one of the cued-recall schemes which are also known as click-based cued-recall schemes, where a password is created by clicking at a sequence of points anywhere on an image and during authentication repeats the same sequence. Cued Click Points (CCP) (Chiasson, van Oorschot, & Biddle, 2007) is like PassPoints though instead of clicking all clicks on one image, here each click will be on different images, where the next image is selected by a deterministic function. Persuasive Cued Click Points (PCCP) (Chiasson, Forget, Biddle, & van Oorschot, 2008) is an improved version of CCP, where the password is generated by selecting a point inside a randomly positioned viewport, resulting in a more randomly distributed click-points in a password.

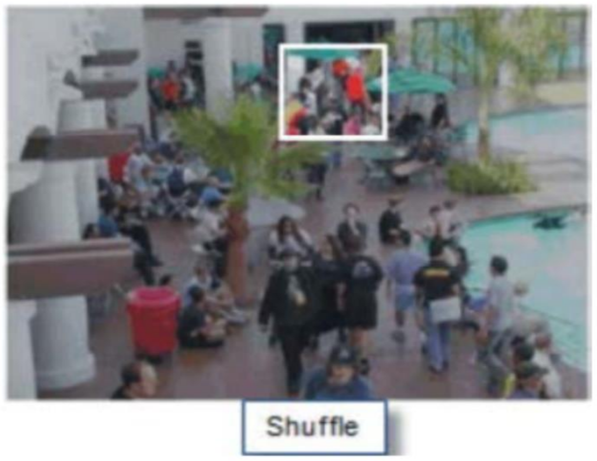


Figure 3. PassPoints

From the above three types of password schemes, the recognition-based scheme is easiest for users to memorize and pure recall is the hardest (Biddle, Chiasson, & van Oorschot, 2012). Guessing attacks are easily performed with recognition-based schemes. For recognition-based schemes there is a password space range of 2^{13} to 2^{16}, DAS and Pass-Go (Tao & Adams, 2008) were broken by a guessing attack using dictionaries of 2^{31} to 2^{41} with full password space of 2^{58} entries and similarly with PassPoints are 2^{26} to 2^{35} with full password space of 2^{43}.

In solving hard AI problems, Captcha helps to separate humans from bots. Mainly there are two types of visual Captcha: text Captcha and Image-Recognition Captcha (IRC). Text Captcha deals with character recognition, and it depends on the difficulty of character segmentation, which is computationally expensive and combinatorically difficult. While IRC deals with the recognition of non-character objects, but non-character object recognition is good when compared to character recognition. IRCs mainly depend on the difficulty in identification or classification of an object, combined with object segmentation. Asirra (Elson, Douceur, Howell, & Saul, 2007) is a scheme from IRC which depends on binary object classification: where a user is asked to find all the cats from a panel of 12 images of cats and dogs. Asirra is prone to machine-learning attacks. IRCs which are based on object identification or classification of objects are insecure (Zhu et al., 2010). Multi-label classification problems are considered more difficult than binary classification problems. Captcha can overcome relay attacks where Captcha challenges relayed to human solvers and the answers were given back to the target application.

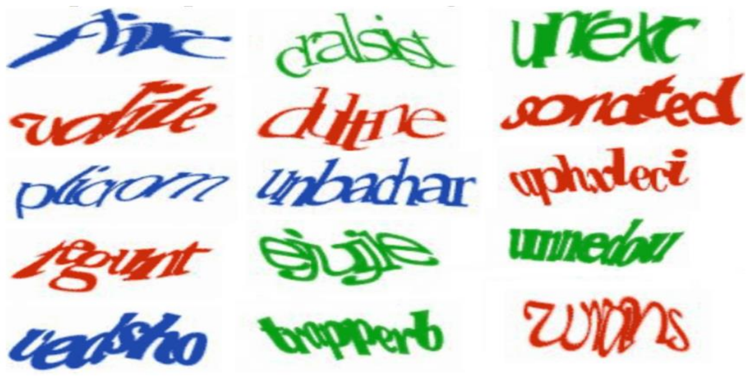


Figure 4. Captcha

Captcha in Authentication

To counter online dictionary attacks, the use of both Captcha and passwords in user authentication protocol known as Captcha-based Password Authentication (CbPA) protocol (Pinkas & Sander, 2002). The CbPA-protocol asks the user to input a user ID and password and then it asks for a Captcha challenge, if the user provides the correct Captcha, a valid browser cookie is received. If the user enters an invalid user ID and password, then before being denied it asks to solve a Captcha. Improved CbPA-protocol (van Oorschot & Stubblebine, 2006) stores cookies only on user-trusted computers and it prompts a Captcha challenge only when the failed attempts exceeded a certain threshold. Further improved, the threshold limit varies from user trusted and untrusted machines, for trusted machines the threshold is small and for untrusted machines the threshold is large (Alsaleh, Mannan, & van Oorschot, 2012).

To address spyware, user used Captcha with recognition-based graphical passwords, where below every image a text Captcha is placed and the user clicks on their own pass-image from decoy images and solves the Captcha during authentication (Gao, Liu, Wang, & Dai, 2009). These the user clicks, or specific locations are selected while creating the password.

In the above schemes, Captcha is an independent entity, which means either a text or graphical password is used at a time. But a CaRP is both a Captcha and a graphical password scheme, which are combined into a single entity.

PGRP Protocol

The proposed PGRP scheme is more restrictive against attackers than commonly used countermeasures. At the same time, PGRP requires answering fewer ATTs for all legitimate users, including those who occasionally require multiple attempts to recall a password. Presented is a login protocol based on ATTs to protect against online password guessing attacks. It reduces the number of ATTs that legitimate users must correctly answer so that a user with a valid browser cookie will rarely be prompted to answer an ATT (Buvanesvari & Prasath, 2013).

A deterministic capacity (AskATT) of the entered client qualifications is utilized to choose whether to ask the client an ATT. To enhance the security of the PS convention, a protected non-deterministic keyed hash capacity as AskATT() so that each username is connected with one key that ought to be changed at whatever point the relating secret key is changed. The proposed capacity requires additional server-side stockpiling per username and no less than one cryptographic hash operation for each login attempt. In this proposed work, coordinated sound mark to help in reviewing the secret key. In day-to-day life, there are different illustrations of reviewing an item by the sound identified with that question. When a user enters their user ID and selects one sound recurrence, which he needs to be played at login time, a resistance worth is additionally chosen which will choose that the client is genuine or a fake (Buvanesvari, & Prasath, 2013).

Chapter II: Background and Review of Literature

According to Zhu et al. (2014), many security primitives are based on hard mathematical problems. Using hard AI problems for security is emerging as an exciting new paradigm but has been under-explored. In this paper, a new security primitive based on hard AI problems. Namely, a novel family of graphical password systems built on top of Captcha technology, which call Captcha as gRaphical Passwords (CaRP). CaRP is both a Captcha and a graphical password scheme. CaRP addresses several security problems at the same time such as: online guessing attacks, relay attacks, and (if combined with dual-view technologies) shoulder-surfing attacks. Notably, a CaRP password can be found only probabilistically by automatic online guessing attacks, even if the password is in the search set. CaRP also offers a novel approach to address the well-known image hotspot problem in popular graphical password systems, such as PassPoints, that often lead to weak password choices. CaRP is not a panacea; however, it offers reasonable security and usability, and appears to fit well with some practical applications for improving online security.

According to Zhu et al. (2014), a security module produces an irregular picture having a majority of password-component markers in it. The irregular picture is then given to a client, and the client chooses parts of the arbitrary picture. The security module decides if they chose segments of the irregular picture and compare it to a password for the client. Be that as it may, if they chose segments of the irregular picture that don't compare to the client's password, the security module may create another arbitrary picture having more password-component pointers in it, wherein each of the irregular pictures are computationally de-corresponded.

According to Dailey and Namprempre (2004), a new construct, the Text-Graphics Character (TGC) Captcha, for preventing dictionary attacks against password authenticated systems allowing remote access via dumb terminals. Password authentication is commonly used for computer access control. But password authenticated systems are prone to dictionary attacks, in which attackers repeatedly attempt to gain access using the entries in a list of frequently-used passwords. CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) are currently being used to prevent automated “bots” from registering for email accounts. They have also been suggested as a means for preventing dictionary attacks. However, current CAPTCHAs are unsuitable for text-based remote access. Our TGC CAPTCHA fills this gap. We believe that the system will not only help improve the security of servers allowing remote terminal access, but also encourage a healthy spirit of competition in the fields of pattern recognition, computer graphics and psychology (Dailey & Namprempre, 2004).

According to Ghorpade, Mukane, Patil, Poal, and Prasad (2014), a CAPTCHA is a program that protects websites from bots by generating as well as grading tests that humans can pass that test but current computer programs cannot. Humans read the distorted text, but the computer cannot read the distorted text. A good CAPTCHA must be robust as well as human friendly. CaGP (CAPTCHA as a Graphical Password) is a graphical password where a password is derived by using a sequence of clicks on an image. An image which is used in CaGP is a CAPTCHA challenge, and for every login attempt a new CaGP image is generated. Many e-banking systems have used CAPTCHA for user logins. The operating cost of spammer is increased by CaGP and it helps to reduce spam emails. For an email service provider which uses

CaGP, a spam bot cannot log into an email account if it doesn’t know the password (Ghorpade et al., 2014).

According to Yan and Dunphy (2007), Draw a secret (DAS) is a representative graphical password scheme. This investigate the novel idea of introducing background images to the draw a secret scheme, in that users were initially supposed to draw passwords on a blank canvas overlaid with a grid. Encouraging results from their two user studies had shown that people aided with background images tended to set significantly more complicated passwords than their counterparts using the original scheme. The background images also concentrated other conventional characteristics in DAS passwords such as symmetry and centering within the drawing grid, further improving the strength of the passwords.

According to Murugavalli, Jainulabudeen, Senthil Kumar, and Anuradha (2016), data and PC security are bolstered by passwords, as passwords assume an essential part in the verification process. The customary confirmation strategy utilizes content-based passwords, which is also called alphanumeric passwords, and is not considered to be solid in information security. To defeat these disadvantages, the graphical password was conspired, and is presented as a contrasting option to content-based passwords. Yet, the graphical password conspire is helpless against shoulder surfing attacks and spyware attacks. To conquer this weakness of graphical passwords, the rising strategy that is Completely Automated Open Turing Tests to differentiate Computers and Humans Apart (Captcha), as a test reaction is created to recognize people from bots in confirmation. To guarantee security, an optional strategy to printed CAPTCHA is supplanted by Captcha as gRaphical Password (CaRP). As the CaRP plot has an extent of refinements in digital security, a two-way validation strategy is proposed in one of the CaRP procedures of the Recognition-based plan. The graphical password conspires when looked at and provides uncommon nascent results when it mixes both Captcha and graphical passwords. With dual-view technology it is very hard to decrypt.

According to Jo, Kim, and Lee (2014), validation to a registering framework is made from two sections, recognizable proof and confirmation. We propose a strategy that can expand the present password-based framework by fortifying the recognizable proof process. It uses individual mystery information rather than a login ID to distinguish a client extraordinarily, henceforth to be called mindmetrics. It at that point asks the client to pick the right login ID among numerous decisions of incompletely clouded IDs. Since it doesn't acknowledge a login ID amid the confirmation procedure, a stolen or split password cannot be utilized for picking up an entrance to the processing framework unless the assailant gives a right distinguishing proof material, i.e., a mindmetrics token. This extra step raises the security of a validation framework extensively over finished single or twofold password frameworks. Since the stolen passwords cannot be utilized instantly by the aggressors, account holders can have an additional opportunity to change their passwords prior to the assailants gaining entrance. This plan does not require any specific equipment and can be actualized rather easily. It might be utilized where biometric plans cannot be utilized cost-successfully, e.g., on open internet business sites. Mindmetrics plot isolates the ID server and the confirmation server, subsequently it is versatile enough to be used on an extensive framework. We executed a proof-of-idea framework and assessed it with test clients. The review demonstrates that the framework is not as meddlesome as different plans, clients feel better secured, and they will utilize the plan on open sites (Yan & Dunphy, 2007).

SHA-1 Algorithm

SHA stands for Secure Hash Algorithm. It was published in 1993 with SHA0 version but later in 1995 with small change it was revised and published by the United States NSA (National Security Agency). SHA-1 differs from the older one only by a single bitwise rotation in its message compression functionality. The structure of this algorithm is like MD4 and MD5 with small changes. The SHA-1 algorithm is a hash function that is commonly used in cryptography for the methods like message authentication, digital signatures, etc. SHA was secure for a long time, but from 2005 on, there were valid attacks against it and now all major companies reject the certificates no will no longer be used after 2017. The SHA-1 algorithm produces a 160-bit value known as a message digest. The hash value is represented as a hexadecimal number which is 40 digits long.

In general, the hash function is nothing but the change of any size data into a fixed size of data. The key or algorithm that is used to change the meaning of the data is known as the hash algorithm. The value obtained after changing the original data by the hashing algorithm is a hash value. Without the knowledge of the algorithm, it is impossible to get the original data file.
For hash algorithms to survive longer they should be able to meet these three requirements. First is speed, it should be able to convert large files into hexadecimal code within a short time, but not too quick so that it can be broken. The second is that if you change any bit of data, anywhere right from beginning to the end, the entire hash value needs to be changed. This means change in single letter it should create a new hash value which is completely different from the past hash value. If the value does not change there may be a chance of finding the sequences and potentially break the hash algorithm, so that the original data file may be corrupted. The third is

that needs to avoid hash collisions, like having two documents with the same hash value. For instance, create an artificial hash collision, like creating a new file and naming it with the existing hash value, then the problem arises and security comes into the picture. So that can fake the documents, change the data, and a threat to data integrity arises.

Generally, in practice it takes a lot of time and large data files to create artificial collisions. But examples of the MD5 hash algorithm which had several such collisions in the past. If the hash algorithm is slow, no one is willing to use it, but if it is too quick then it is easier to create several artificial collisions within a lesser amount of time. There are examples of sending a document inside which is malicious and get the same hash code using MD5, which is why it was considered broken. Also, there is one more reason for this hash algorithm to be broken, it is the amount of usage. MD5 had been used for so long and so frequently that if any word that type in google gets the hash value for that word. Computers are now fast enough to create the artificial collisions deliberately. That is why the SHA-1 algorithm comes into the picture. This algorithm also is a hash algorithm which has been used for a long period of time for encryption, and as the time passes, there were collisions on SHA-1 as well and now SHA-1 has been blocked by major companies. Now companies have started using the latest versions of it like SHA2 and SHA3 which are more secure.

How SHA-1 works. As discussed above, SHA-1 has a message digest value of 160 bits. Later SHA2 and SHA3 were published with more message digest lengths of 256, 384 and 512 bits. SHA-1 has a message size of <2^64. It has a block size of 512 bits which is like SHA2 but SHA3 has more message size than these hash algorithms. The search space for the attacker in SHA-1 which is of 160 MD value is 80 bits, that means it is less secure than the remaining hash algorithms SHA2, SHA3, and SHA5. For the remaining hash algorithms, the search space varies from 80 to 128, 192, and 256 respectively. This means the attacker has to get more space to search than the SHA-1 algorithm. Therefore, SHA-1 is less secure than the remaining versions.

Let us see how SHA-1 works. SHA-1 converts any length of value to a 160-bit value as discussed. This means SHA-1 converts an entire large data set or just a single word file into that fixed set of value. As it has a block size of 512, once at a time it takes 512-bit blocks of data and loops it till the file is extended. Suppose if the data file has 512 bits of data and then it needed only one loop to convert all the data into the hash value. If the file size is larger the loop iterates with 512 bits of data and converts it into the hash value. SHA-1 starts with the internal state bringing the bits of data one at a time and then the internal state is changed. Once the data in the file is completed the internal state value and that is our hash value.

In detail, the first step is appending the padding bits (1 followed by 0's). We need to pad up to 448 mod 512. The second step is the remaining 64 bits of padding length. We need to get the message in multiples of 512 bits. The third step is initializing the message digest buffer. In-order to hold the intermediate and final message digest value to initialize the MD value. The size is 160-bit, so in-order to initialize it has 5 registers each of 32 bits total of the 160-bit value. We need to initialize these five registers with hexadecimal values initially. In step 4 there are 4 rounds with each round it is similar to the individual primitive functions internally. In each round, there are 20 steps, which in total it takes 80 steps to process the data. So, beginning with the compression functions the registers are initialized with internal states. Then the message or the data file that was presented or input mix or combined and shuffled with these registers or internal states and perform 80 char compressions to look more and more random. For the same

message, it performs the same algorithm, so if the same input file inserted again, it produces the same result. So now, after shuffling with the registers, the data will have new registers.
Then with the recent internal states, get the new 160 message digest values of the SHA-1 algorithm, also known as the hash value.

SHA-1 has existed for a longer period of time, but once artificial collision starts increasing the use of this algorithm has stopped. Also, newer versions with more security space and message digest values were introduced like SHA2, SHA3, and SHA5. SHA5 is one of the most recent hashing algorithms, with 512 bits of MD values and with more security space of 256 bits, which means for an attacker it takes a greater amount of time to create a collision compared to SHA-1.

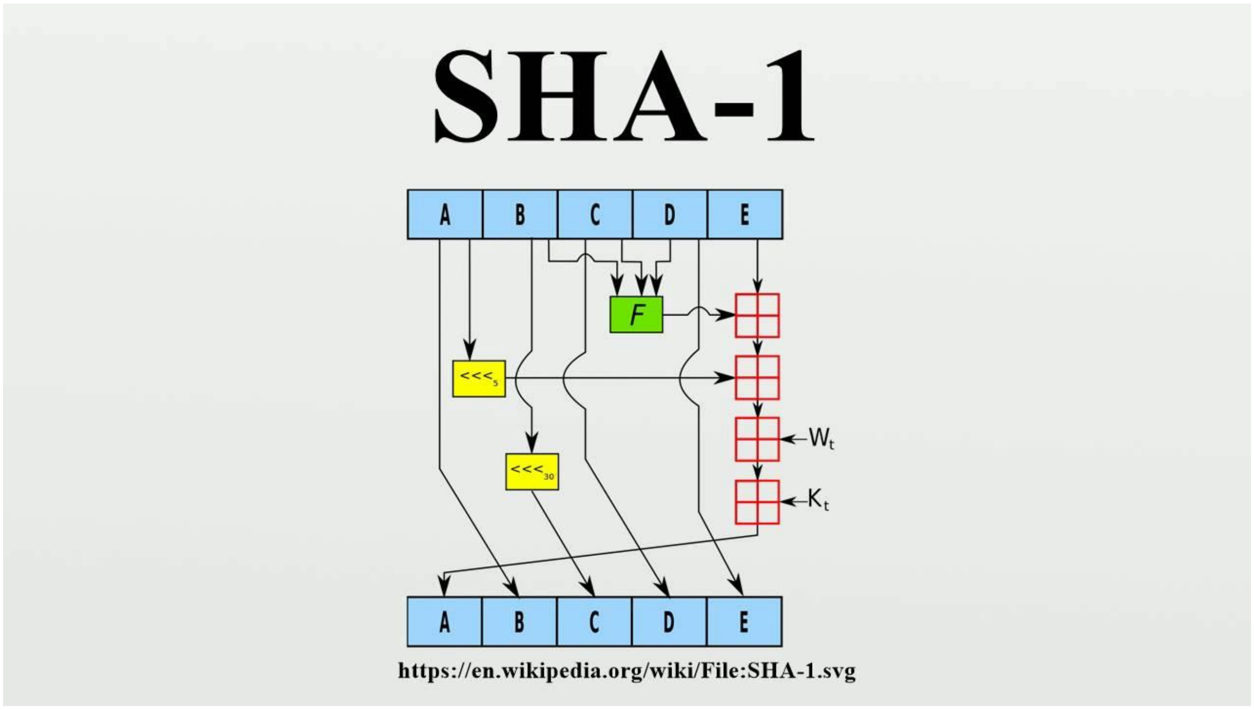


Figure 5. SHA-1 Flowchart.

Diffie–Hellman Key Exchange (D–H)

Transport Layer Security (TLS) (n.d.) protocol has been developed by the Internet Engineering Task Force (IETF) as a standard tool for providing security. This paper provides an overview of the design and workings of the TLS protocol and how it enables network security for the exchange of information. This paper uses different algorithms to interchange data securely by using key exchange or key agreement. Before implementing key exchange, this paper gives a concise description and clear understanding of transport layer protocol. It also uses different block ciphers and stream ciphers. TLS is implemented on top of the transport layer protocol. This paper also deals with attacks on TLS.

TLS-Transport Layer Service (n.d.) is a cryptographic protocol designed to provide security for communication over the Internet. It uses X.509 certificate and asymmetric cryptography to authenticate the communicating parties so as to exchange a symmetric key. This session key is used to encrypt data flowing between the two parties. In asymmetric cryptography, there are two keys one is a (secret) or private key and the other is a public key. The Internet protocol suite is the Computer Networking model and Communication Protocols used on the Internet and similar computer networks.

TLS encrypts the data of network connections in the application layer. In OSI model equivalences, TLS is initialized at Layer 5 (Session Layer) and works at Layer 6 (Presentation Layer). The session layer has a handshake using an asymmetric cipher in order to establish cipher settings and a shared key for that session; then the presentation layer encrypts the rest of the communication using a symmetric cipher and that session key (Dierks & Allen, 2003). In the

TLS and SSL models, they work instead of the transport layer, which carries encrypted data in segments.

Transport Layer Service is used to implement the Key Exchange or Key Agreement so as to provide keys to the authenticated users and transport data between the communicating parties by encapsulating data in HTTP/FTP/SMTP.

For email transmission, using the TLS protocol in communication then there is no need to do encryption for emails.

Implementation of the TLS protocol occurs in two levels: TLS Record Protocol and TLS Handshake Protocol

TLS record protocol. This ensures a secure communication and private channel between the client and server. This encapsulates the TLS Handshake Protocol. For the private channel, this uses symmetric cryptographic keys, i.e. the same key for both the client and server. This protocol also uses Message Authentication Code which generates hash functions for the secure channel.

TLS handshake protocol. This protocol is used to initiate the authenticated communication between the client and server. Before sending data, this protocol allows the client and server to use the same encryption algorithm and keys. This will determine the authentication for the web browser.

How TLS works. TLS contains four protocol layers to communicate between the client and server: (a) Alert Protocol, (b) Record Layer, (c) ChangeCipherSpec Protocol, and (d) Handshake Protocol.

Alert protocol. The Alert protocol is responsible for connectivity issues between the client and server, like sending errors, problems or warnings. There are two types of alert protocols—Severity Level and Alert Description.

1. Severity Level has two values based on its concern as a 1 or 2. The value of '1' is for a warning message and indicates two parties to disconnect from their session and restart a new handshake. The value '2' is for a fatal alert message that indicates that the two parties are to disconnect from the current session.
2. The Alert Description gives the description of the specific error from an alert message. It is composed of one byte, which gives the description that follows with the "fatal" message and are: CloseNotify, UnexpectedMessage, BadRecordMAC, DecompressionFailure, HandshakeFailure, NoCertificate, BadCertificate, UnsupportedCertificate, CertificateRevoked, CertificateExpired, CertificateUnknown, IllegalParameter.

Record layer. This layer documents the messages from Alert, ChangeCipherSpec, Handshake and the application protocol. The document provides header and MAC produced hash values for each message. The header is composed of five bytes: Protocol Definition (1byte), Protocol Version (2 bytes) and the Length (2 bytes). The protocol messages that follow the header cannot be longer than 16,384 bytes, as specified by the TLS protocol. (Thomas, 2000).

ChangeCipherSpec protocol. This layer consists of one message that gives instructions at the beginning of the secure communications. The ChangeCipherSpec message is only one byte which is '1' for the change in the communications protocol.

Handshake Protocol

The client and server pass messages which establish a handshake connection. The TLS handshake is as follows: ClientHello, ServerHello, ServerKeyExchange, ServerHelloDone, ClientKeyExchange, ChangeCipherSpec, and Finished.

- ClientHello. The client is requesting the connection to the server by passing the message “ClientHello” and with this the client also sends a 32-byte random number and SessionID which describes that the client is in encrypted communications and blank field.
- ServerHello. Based on the ClientHello the server makes choices which are of five fields: SessionID, Version of TLS used, the CompressionMethod and CipherSuite, along with a date and time stamp.
- ServerKeyExchange. From the above two steps, both the client and server agree that the transmission of data will be encrypted. Since no algorithm is decided upon yet, data is sent with no encryption, and communication is in the clear or public domain. The public key of the server is used for the session key encryption and both the client and server use the same key to encrypt data that is to be transmitted. Digital certificates are used for authentication which combines with the public key. These certificates contain RSA Security or VeriSign, and have an expiration date which helps the client to verify the certificate owner and certificate authority. The only public key is available in the certificate but no private key, it includes the private key then there is no use of a digital signature.

- ServerHelloDone. After the ServerKeyExchange message, the client gets the ServerHelloDone message which indicates acknowledgment from the server that it has received the message.
- ClientKeyExchange. For the TLS session, the client doesn’t need any public or private keys, in this ClientKeyExchange message it has key information that the two parties use for communication. At this point, mitigate “man in the middle attacks” because to decrypt the message a masquerader would need to have the server’s private key. By the end of this process, the client and server negotiate communication between themselves.
- ChangeCipherSpec. This message gives information about the change of state, i.e. from insecure to a secure of state of data. Both the client and server machines send this ChangeCipherSpec to agree to the secure state.
- Finished. Before completion of the TLS handshake, three things should be verified, they are: (a) information of key; (b) audit information, i.e., previous stored messages of TLS handshakes; and (c) the value, which indicated who is the sender either the client or server (Biddle et al., 2012).

In the case of an e-mail server, the TLS handshake session verification will be indicated by a lock icon, which indicates that the secure protocol is in use by both the client browser and web e-mail server.

The name itself specifies that it is used for exchanging keys in cryptography. The Diffie–Hellman key exchange method allows two parties who have no prior knowledge of either’s keys to jointly establish a shared secret key over an unsecured channel. This can also be used to encrypt subsequent communications using a symmetric key.

Diffie–Hellman key agreement is an anonymous (non-authenticated) Key-Agreement Algorithm. It provides perfect forward secrecy in the Transport Layer Service ephemerals modes. Diffie–Hellman establishes a shared secret key that can be used for secret communications while exchanging data over a public network.

The Diffie-Hellman key exchange (n.d.) algorithm is used to exchange the data between two or more authenticated users. This algorithm is simple in implementation to exchange the keys in a possible secure way. The output of this is computationally secure and it so because by the time an attacker decodes the private shared key the value of the key is useless.

To make the algorithm operation simple, instead of using very large numbers let us use a diagram of colors, to demonstrate the key exchange. The part of the process that is important is the exchange of their secret colors. ‘A’ and ‘B’ exchange these colors as a mix. This mix helps to generate a key common to both the parties that cannot be cracked computationally by supercomputers in a reasonable amount of time. ‘A’ and ‘B’ both use this common secret to encrypt and decrypt their sent and received data. Yellow is the starting color and is publicly declared that is it is known by the two authenticated parties and by the eavesdropping opponent.

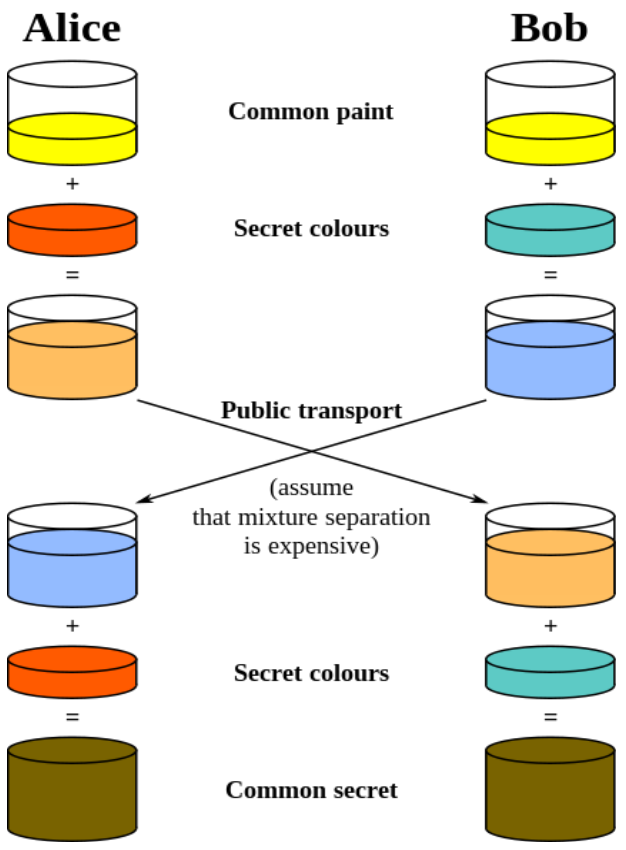


Figure 6. Basic Operation of this Algorithm

Skeletal Operation for Cyclic Groups (Finite)

Two important terms that are used as inputs for the working of the algorithm are:

- A Generator ‘g.’ A public element known by everyone including the attacker.
- A prime number ‘p.’

The environment of operation is a Cyclic Group ‘G’ and it is Finite in length. The multiplicative aspect of group G is written as:

My examples the two authorized parties ‘A,’ and ‘B,’ ‘C’ is the intruder.

- ‘A’ selects a natural number, which feels lucky to him and sends it to ‘B’ as g^a.
- ‘B’ in the same way picks his lucky or random natural number b and sends it to ‘A’ as g^b.

- ‘A’ after receiving from ‘B’ computes $(g^b)^a$.
- In the same way, ‘B’ computes $(g^a)^b$ after receiving from ‘A’.

‘A’ and ‘B’ both possess a group element g^{ab}, which operates as a shared secret key.

Now in order to encrypt $e = mg^{ab}$ where ‘m’ is the message and ‘g’ is the element.

Next comes the decryption of ‘m’ from ‘e’ as shown:

We compute $(g^{ab})^{-1}$, using |G|:

‘A’ knows G, b, and g^a. As g generates G it follows that $g^{|G|} = 1$ (the group identity).

‘A’ calculates

$(g^a)^{|G|-b} = g^{a(|G|-b)} = g^{a|G|-ab} = g^{a|G|}g^{-ab} = (g^{|G|})^a g^{-ab} = 1^a g^{-ab} = g^{-ab} = (g^{ab})^{-1}$.

When ‘A’ sends ‘B’ the encrypted message, $e = mg^{ab}$, ‘B’

computes $e(g^{ab})^{-1} = mg^{ab}(g^{ab})^{-1} = m(1) = m$.

For larger tables that are required for multiplication, a multiplication algorithm and exponentiation of Finite Cyclic groups is required [4].

The Table of Secrets

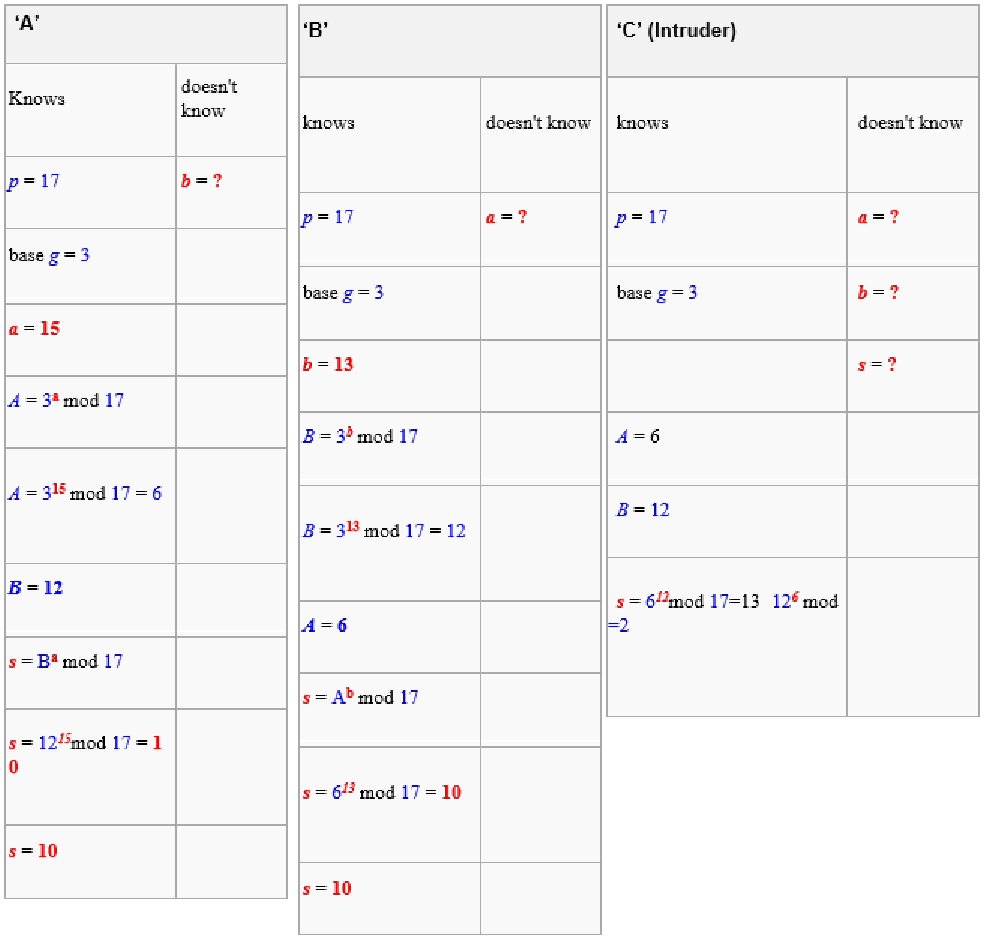
The table specifies values, which are declared as global, open values, and values. These are secret as selected by the two exchanging parties. The global values are in blue and the secret values are in **red**.

Let us set up a few characters so as to know how this algorithm works:

- Here ‘C’ is the user who tries to break in, so as to know what ‘A’ and ‘B’ are trying to exchange. ‘C’ tries to know what they are communicating but does not alter the contents used in the communication.

g = public (prime) base, known to ‘A’, ‘B’ and ‘C’. $g = 3$

1. p = public (prime) number, known to ‘A’, ‘B’ and even $p = 17$
2. a = ‘A’’s private key, known only to ‘A’ $a=15$
3. b = ‘B’’s private key known only to ‘B’. $b = 13$
4. A = ‘A’’s public key, known to ‘A’, ‘B’ and ‘C’. $A = g^a \bmod p = 6$



‘A’	
Knows	doesn't know
$p = 17$	$b = ?$
base $g = 3$	
$a = 15$	
$A = 3^a \bmod 17$	
$A = 3^{15} \bmod 17 = 6$	
$B = 12$	
$s = B^a \bmod 17$	
$s = 12^{15} \bmod 17 = 1$ 0	
$s = 10$	

‘B’	
knows	doesn't know
$p = 17$	$a = ?$
base $g = 3$	
$b = 13$	
$B = 3^b \bmod 17$	
$B = 3^{13} \bmod 17 = 12$	
$A = 6$	
$s = A^b \bmod 17$	
$s = 6^{13} \bmod 17 = 10$	
$s = 10$	

‘C’ (Intruder)	
knows	doesn't know
$p = 17$	$a = ?$
base $g = 3$	$b = ?$
	$s = ?$
$A = 6$	
$B = 12$	
$s = 6^{12} \bmod 17 = 13$ $12^{6} \bmod$ $=2$	

Figure 7. Demonstration of Algorithm Operation Involving Only Two Authenticated Key Exchanging Parties.

Operations Involving More Than Two Parties

The scope of Diffie–Hellman key agreement is not limited to sharing of keys between two parties. It has a broader scope where any number of users can take part in an agreement by performing iterations of the agreement protocol and exchanging intermediate data. For example: This time ‘A,’ ‘B,’ and ‘C’ are the authenticated parties who agree to Diffie–Hellman Key Exchange, with modulo p taken for all operations

Working:

1. The parties agree on the algorithm parameters p and g.
2. The parties generate their private keys, named a, b, and c.
3. ‘A’ computes g^a and sends it to ‘B’.
4. ‘B’ computes $(g^a)^b = g^{ab}$ and sends it to ‘C’.
5. ‘C’ computes $(g^{ab})^c = g^{abc}$ and uses it as his secret.
6. ‘B’ computes g^b and sends it to ‘C’.
7. ‘C’ computes $(g^b)^c = g^{bc}$ and sends it to ‘A’.
8. ‘A’ computes $(g^{bc})^a = g^{bca} = g^{abc}$ and uses it as his secret.
9. ‘C’ computes g^c and sends it to ‘A’.
10. ‘A’ computes $(g^c)^a = g^{ca}$ and sends it to ‘B’.
11. ‘B’ computes $(g^{ca})^b = g^{cab} = g^{abc}$ and uses it as his secret.

An attacker let’s say ‘C’ has been able to see g^a, g^b, g^c, g^{ab}, g^{ac}, and g^{bc}, but cannot use any combination of these to reproduce g^{abc} (Rashmi & Maheshwarappa, 2003).

Issues Involved in Security

The security of the protocol depends on the values selected for ‘G’ and ‘g.’ To obtain the value of g^{ab}, the intruder must understand the working of the Diffie-Hellman algorithm, which is not possible at the time he tries to break in. It is efficient in solving the problem of the discrete algorithm to make it easy to compute a or b that are used in the key exchange protocol.

The Pohlig-Hellman algorithm that is used to obtain the values of a and b can be prevented by selecting a large prime factor of G. To avoid this a Safe Prime is selected by using the expression $p = 2q + 1$.

If the parties use a random number generator, it would be easy for the intruder to break in. At the end of the session, the secret keys are discarded.

Other Practical Uses

Password-authenticated key agreement. Two users in our case ‘A’ and ‘B’ use Password-Authenticated Key Agreement (PAKA) from Diffie-Hellman to safeguard from a Man-in-the-Middle attack. The simplest option is to compare the hash functions of s combined with the password and independently calculated on both ends of the channel. The best part is that the attacker can test only one password per iteration and by this way weak passwords provides relatively better security (Modern Cryptography, n.d.).

Public key implementation. This algorithm can be used as an integral part of Public Key Infrastructure (PKI).

The public key that ‘A’ makes use of is $(g^a \bmod p, g, p)$.

For ‘B’ to send a message to ‘A’, he selects a random number ‘b’ and sends it as $g^b \bmod p$ without encryption along with the message and the symmetric key $(g^a)^b \bmod p$ encrypted.

Only ‘A’ can decrypt the message because only she has *a* (the private key). A shared public key can also answer a Man-in-the-Middle attack.

An attack-cryptocurrency. The way a sender and receiver operate are that the sender creates the public part of the key; on the other hand, the receiver can compute the private part. Due to this, the receiver is the only one who can complete a transaction. So, for every transaction they perform a check for a single formula to ensure that it belongs to them.

Data Encryption Standard (DES)

The symmetric key algorithm is a class of algorithm that uses the same Cryptographic keys for both encryption of plaintext and decryption of cipher text for cryptography. To go between the two keys the keys may be identical or a simple transformation. These keys represent a shared secret between multiple parties that is used to maintain a private link. In types of symmetric key algorithms, it uses either stream ciphers or block ciphers.

Encrypting the digits of a message one at a time is a stream cipher. Stream ciphers attempt to simulate the one-time pad by using short keys to generate longer keys that can then be used in a one-time pad encryption.

In block ciphers, it takes several bits and encrypts them as a single unit, packing the plaintext so it will be a multiple of the block size. Normally the block size is fixed, and the block of cipher text produced by the block cipher is usually also the same length as the plaintext block size. Typical block sizes are 64 (DES) and 128 (AES)

The Data Encryption Standard (DES) (n.d.) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem. A proposal from IBM, a modification of a project called Lucifer, was accepted as DES. DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS) (Data Encryption Standard, n.d.).

DES is a block cipher, as shown

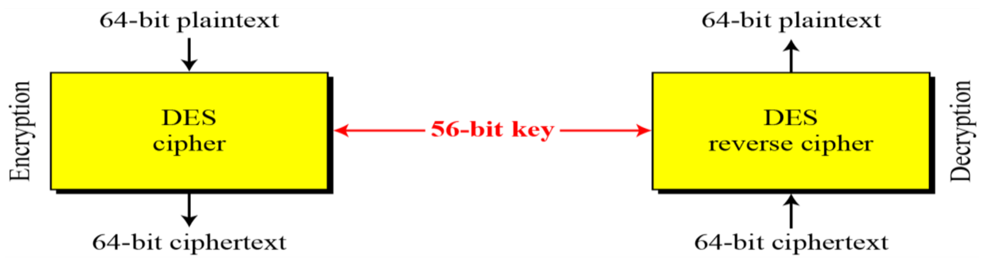


Figure 8. DES Block Cipher

DES is a block cipher that takes a fixed length of plaintext and transfers to a series of complicated operations into another cipher text of the same length. In DES, it has 64 bits of block size. To customize the transformation, DES uses a key so that only decryption is supposed to be performed. Although it has 64 bits, only 56bits are used by the algorithm and the rest of the eight bits are used for checking parity so they are discarded. Therefore, the effective key length is 56 bits.

The encryption process is made up of two permutations (P-boxes), which implements initial and final permutations, and 16 Feistel rounds.

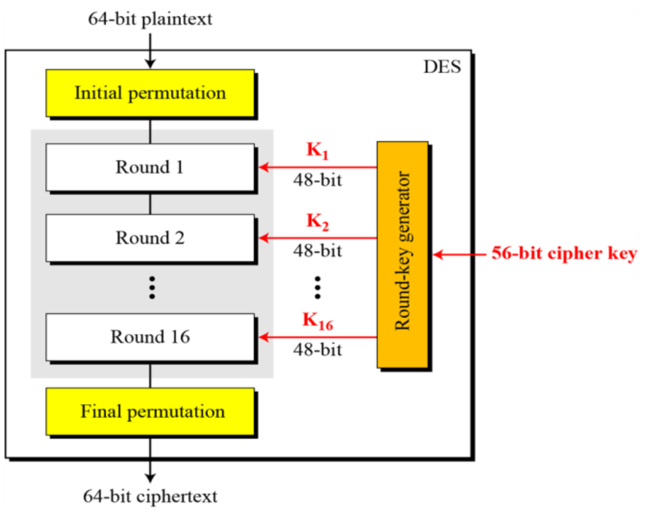


Figure 9. Initial Permutation Septs in DES

Initial and final permutation steps in DES:

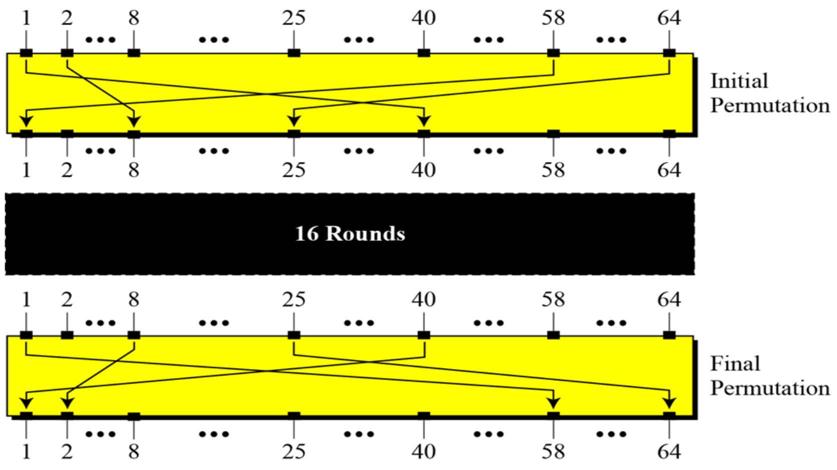
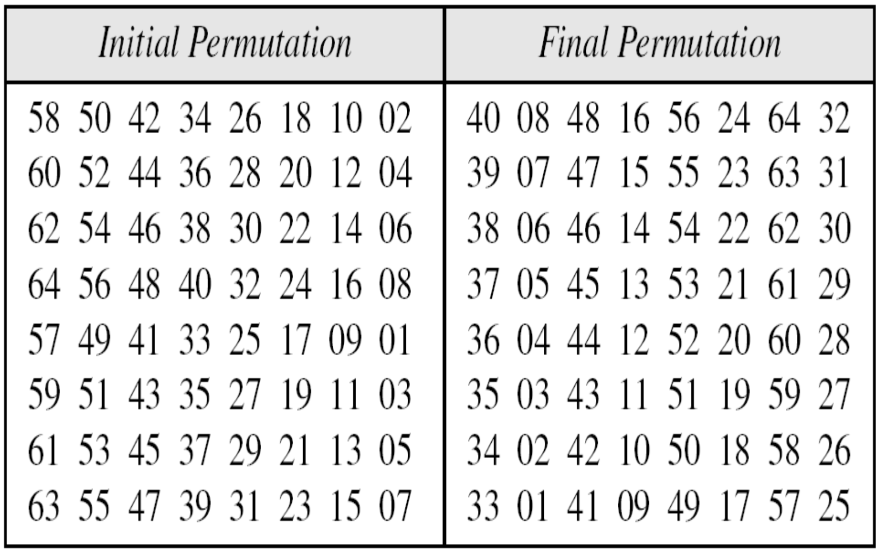


Figure 10. Final Permutation Septs in DES



Initial Permutation								Final Permutation							
58	50	42	34	26	18	10	02	40	08	48	16	56	24	64	32
60	52	44	36	28	20	12	04	39	07	47	15	55	23	63	31
62	54	46	38	30	22	14	06	38	06	46	14	54	22	62	30
64	56	48	40	32	24	16	08	37	05	45	13	53	21	61	29
57	49	41	33	25	17	09	01	36	04	44	12	52	20	60	28
59	51	43	35	27	19	11	03	35	03	43	11	51	19	59	27
61	53	45	37	29	21	13	05	34	02	42	10	50	18	58	26
63	55	47	39	31	23	15	07	33	01	41	09	49	17	57	25

Figure 11. Initial and Final Permutation Tables

DES uses 16 rounds. Each round of DES is a Feistel cipher which is used in the construction of block ciphers. It is also commonly known as the Feistel network. One of the advantages of a Feistel structure is that both encryption and decryption has similar operations also identical in some cases, requiring a reversal of the key schedule. A round in DES (encryption site):

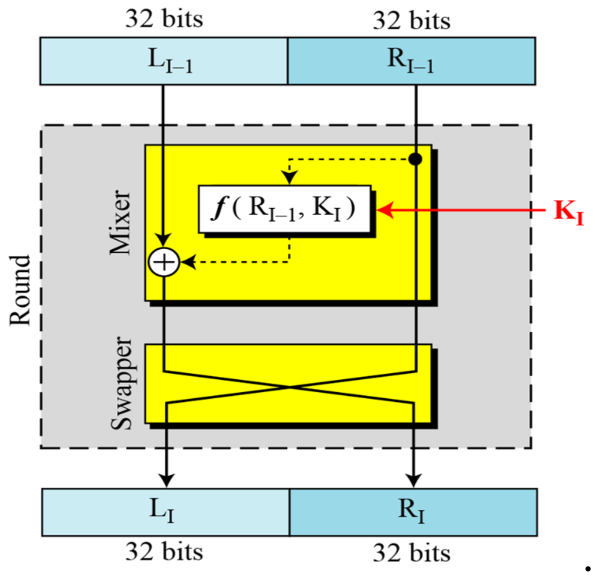


Figure 12. Single DES Round

The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output. This DES function is very important in the DES algorithm.

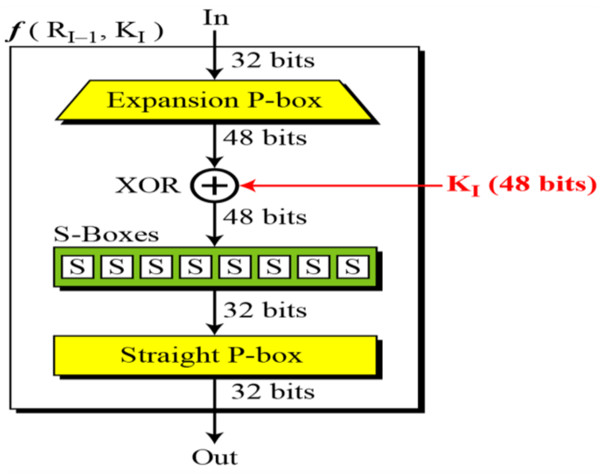


Figure 13. Block Diagram for DES Function

DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be broken using 2^{43} pairs of known plaintexts. However, from the practical point of view, finding so many pairs is very unlikely.

Chapter III: Methodology

CaRP: An Overview

In CaRP, for every login attempt a new image is generated even for the same user, it uses an alphabet of visual objects (alphanumerical characters, similar animals) to generate a CaRP image which is a Captcha challenge. The main difference between CaRP images and Captcha images is the visual objects in the alphabet should be in the CaRP image which requires the user to input any password but, in a Captcha image it is not necessary. As mentioned earlier CaRP schemes are clicked-based graphical passwords. These are based on memorizing and entering a password and CaRP schemes are classified into two categories: Recognition and Recognition-recall. The latter asks to recognize an image based on recognized objects as cues to enter the password. This is a combination of recognition and cued-recall, and due to this it contains both advantages of the recognition-based system of being easy for human memory and the cued-recall advantage of a large password space.

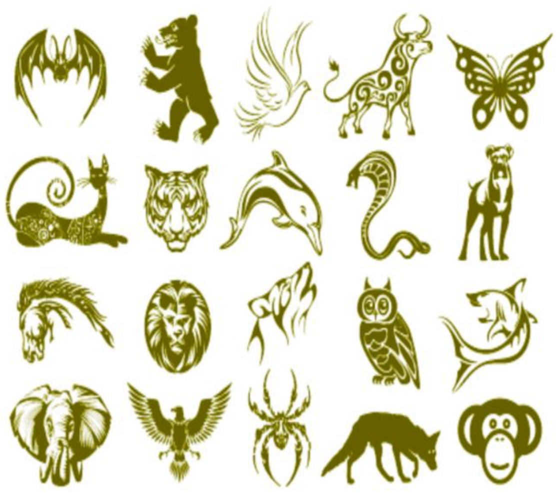


Figure 14. CaRP

Captcha as a gRaphical Password (CaRP). CaRP is another approach to impede online guessing attacks. In a guessing attack, a secret key conjecture is tried in the fizzled trial and is resolved wrong and then avoided from ensuing trials. If the number of trails increase, then the complexity of the password increases. Mathematically, let P be the arrangement of secret key estimates before any trial, ρ be the password to discover, A mean the attempt though A signifies the n-th trial, and p(A = ρ) be the likelihood that ρ is tried in endeavor A. Given Sn a chance to be the arrangement of password guess tried in trials up to An. The secret key supposition to be tried in n-th attempt is from set P|Sn−1, i.e., the relative supplement of Sn−1 in P. On the off chance that ρ ε P, then

$$p(A=\rho\,|\,A1!=\rho\ldots\ldots\ldots\ldots An-1!=\rho) > p(A=\rho)\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

and

$$S_n \to P$$

$$p(A=\rho\,|\,A1\_=\rho\ldots\ldots An-1\_=\rho)\to 1 withn \to |\,P\,|\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2) S_n \to P$$

Following are two types of guessing attacks:

- Automatic Guessing Attacks apply an automatic attempt and error process, but P can be manually constructed.
- Human Guessing Attacks apply a manual attempt and error process. CaRP adopts a completely different approach to counter automatic guessing attacks. It aims at realizing the following equation in an automatic guessing attack.

$$p(A=\rho\,|\,A1\ldots\ldots\ldots\ldots An-1) = p(A=\rho), \forall n\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Eq.(iii) means that each attempt is computationally independent of another attempt. Specifically, no matter how many attempts run previously, the chance of finding the password in the current attempt always remains the same. That is, a password in P can be found only probabilistically by automatic guessing (including brute-force) attacks, in contrast to existing graphical password schemes where a password can be found within a fixed number of trials.

Recognition-Based CaRP

In Recognition-based CaRP, a password is a sequence of visual objects in the alphabet. Recognition-based CaRP has access to an infinite number of different visual objects. Presented below are three forms of two recognition-based CaRP schemes and another variation.

ClickText. ClickText is a recognition-based CaRP scheme built on top of text Captcha. ClickText alphabets should be clear without visually confusing characters. Consider the letter "O" and the digit "0" which may cause confusion in CaRP images, thus these characters must be removed from the alphabet. A ClickText password is like a text password, a sequence of characters in the alphabet, e.g., ρ = "AB#9CD87." From this, a ClickText image is generated by the Captcha engine. While generating a ClickText image all the alphabet characters positions or locations are tracked to produce a ground truth for the location of the character in the generated image. Based on user- clicked points the authentication server relies on the ground truth to identify the characters. In this way ClickText images or characters can be arranged randomly on a 2D space (Elson et al., 2007). Whereas, in text Captcha characters are aligned from left to right for users to type them sequentially. A ClickText image of an alphabet of 33 characters is shown in Figure 15. Alphabets in user passwords are clicked on the image with the password order, for example "A," "B," "#," "9," "C," "D," "8," and then "7" for password ρ = "AB#9CD87".

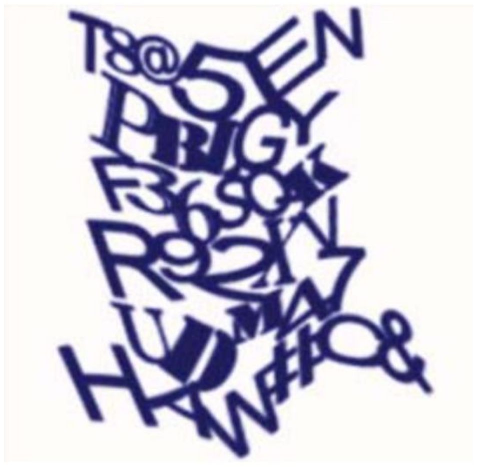


Figure 15. A ClickText Image with 33 Characters

ClickAnimal. Captcha Zoo (Lin, Huang, Bell, & Lee, 2011) is a Captcha scheme where different colors, poses, textures and lightings of 3D models of a horse and dog are used to generate 2D animals and arrange them on the cluttered background. To pass the test, it asks the user to click all the horses as shown in Figure 16 where all of the horses are circled in red. ClickAnimal is one more recognition-based CaRP scheme which should be built on Captcha Zoo, here there is an alphabet of similar animals like a dog, horse, pig, etc. The password is some sequence of animal names like ρ = "Turkey, Cat, Horse, Dog . . ." For every animal, the system generates one or more 3D models. ClickAnimal images are generated from Captcha generation processes, and 3D models are used to generate 2D animals by applying different colors, views, lightning effects, textures, and optionally distortions. These are placed on cluttered background like a grassland. In a ClickAnimal image, some animals may appear to be close in appearance to other animals, but the core parts are not too close, and humans can identify them fairly easily. In Figure 10, the ClickAnimal image is of 10 alphabet animals. From a 3D model, the user gets different views. When this image is combined with additional anti-recognition mechanisms it makes it difficult for machines to identify animals, but humans can identify them easily.

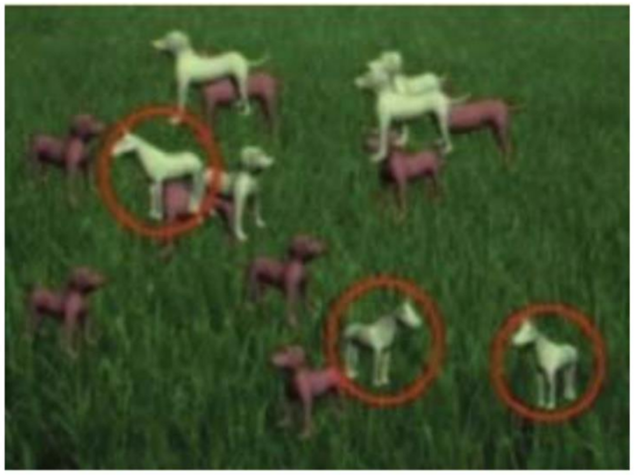


Figure 16. Captcha Zoo with Horses Circled Red

AnimalGrid. The number of similar animals is much less than the number of available characters. ClickAnimal has a smaller alphabet size, thus a smaller password space when compared to ClickText. To restrict guessing attacks, CaRP should have a large effective password space. The user can increase AnimalGrid's password space by mixing it with the grid-based graphical password, the grid depends on the size of the selected animal. DAS (Gyorffy, Tappenden, & Miller, 2011) requires drawing on the grid, and a new scheme known as Click-A-Secret(CAS) differs by changing the requirement from drawing to clicking. AnimalGrid is a combination of ClickAnimal and CAS, where the grid-cells count should be much greater than the alphabet size. Here the advantage is based on this grid, and it generates a follow up grid. If a wrong animal is clicked, the next grid is then wrong. If the user clicks on a correct labeled grid-cell of the wrong grid, this guides to a wrong grid-cell at the authentication server side when the correct grid is used.

To enter a password for AnimalGrid, first ClickAnimal is displayed and, once the user selects the animal, an image of n x n grid appears where the selected animal bounding rectangle size is equal with the grid-cell. Every grid is labeled which helps the users to identify them. First, from Figure 17, the left image shows a selected animal and the 6 x 6 grid in Figure 17 where the red turkey is shown there. A user can select zero to multiple grid-cells matching her password. Therefore, a password is a sequence of animals interweaving with grid-cells, e.g., ρ = "Dog, Grid⟨2⟩, Grid⟨1⟩; Cat, Horse, Grid⟨3⟩," where Grid⟨1⟩ means the grid-cell indexed as 1, and grid-cells after an animal means that the grid is determined by the bounding rectangle of the animal. A password must begin with an animal.

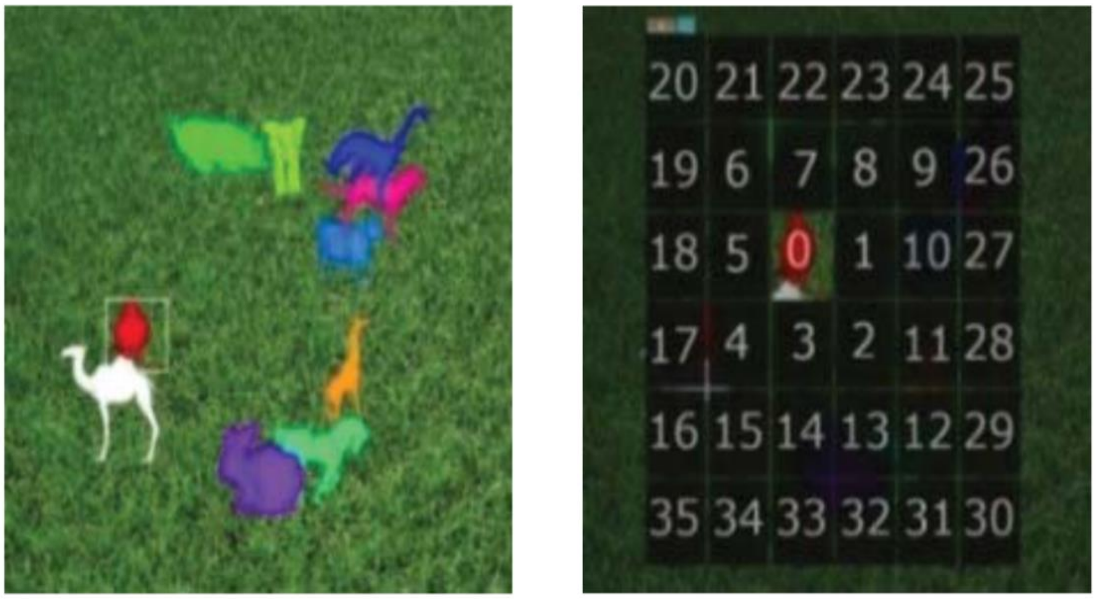


Figure 17. A ClickAnimal Image (left) and 6 x 6 Grid (right) Determined by Red Turkey's Bounding Rectangle

The user selects an animal from a ClickAnimal image when it matches the first animal in the password. The locations of clicked points are then recorded. A bounding rectangle is calculated and displayed in the Figure 17 white rectangle. Based on image selection, the user modifies inaccurate edges. Once the boundary rectangle is finalized, an image of n × n grid with the identified bounding rectangle as its grid-cell size is generated and displayed. It automatically fits the size even if it is too large or too small. The user then clicks a sequence of zero to multiple grid-cells that match the grid-cells following the first animals in her password and again back to the ClickAnimal image. For the example password ρ given previously, she clicks a point inside grid-cell⟨2⟩, and then a point inside grid-cell⟨1⟩ to select the two grid-cells. The coordinates of user-clicked points on the grid image (the original one before scaling if the grid image is scaled) are then recorded. The above process is repeated until the user has finished entering her password. The resulting sequence of coordinates of user-clicked points, e.g., "AP⟨150,50⟩, GP⟨30,66⟩, GP⟨89,160⟩, AP⟨135,97⟩..." where "AP⟨x,y⟩" denotes the point with coordinates ⟨x,y⟩ on a ClickAnimal image, and "GP⟨x,y⟩" denotes the point with coordinates ⟨x,y⟩ on a grid image, is sent to the authentication server.

Recognition-Recall CaRP

In recognition-recall CaRP, the password is created from a sequence of some invariant points of objects. An invariant point of an object (letter "A") is a point that has a fixed relative position in different incarnations (fonts) of the object, and with this, it can be uniquely identified by humans irrespective of how the object appears in CaRP images. To enter a password, a user should identify the objects in the CaRP image, and after that use these identified objects as cues which then helps to locate and click the invariant points matching user's password. Every password point has some tolerance range so that if the click is within the tolerance range, it accepts it as a password point. Three pixels or less is the ideal tolerance range (Chiasson et al., 2007).

TextPoints. TextPoints is a recognition-recall CaRP scheme with an alphabet of characters. Figure 18 shows some invariant points of letter “A,” which provides a strong cue to locate its invariant points from memory. If the distance to the closest boundary of the object exceeds a threshold, that point is known as the internal point of an object. For TextPoints, to form a set of clickable points, a set of internal points of characters are selected. Here there is a chance of overlap between clickable points and neighboring characters and their tolerance range also occluded on the image generated by the underlying Captcha engine. In determining clickable points to avoid overlap on CaRP images, the distance between any pair of clickable points in a character must exceed a threshold so both can be separated. Consider an example, if the center of a stroke segment in one character is selected, the user cannot select the center of a similar stroke segment in another character. To avoid this, the user must select a different point from the stroke segment. By this requirement users can ensure that a clickable point is context-dependent, however depending on the character that the point lies in the same type of structured point may or may not be the clickable point. Even though the clickable points are known for each character, character recognition is necessary for locating clickable points on a TextPoints image. These things cannot be achieved by bots.

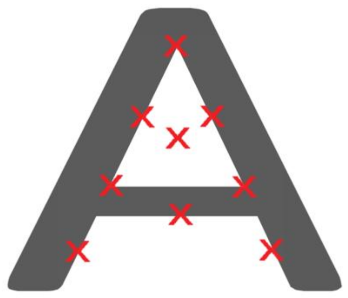


Figure 18. Some Invariant Points (red crosses) of “A”

Here the password is a sequence of clickable points, where a character can contribute multiple clickable points, in this way TextPoints has a much larger password space than ClickText.

- Image Generation: Both ClickText images and TextPoints images look the same but to ensure any collision or tolerance region occlude with another clickable point's, locations of clickable points are checked. If this fails, the system generates another image. Failures are rare because the clickable points are all internal points.
- Authentication: While creating a password, the user asks to select a CaRP image where all clickable points are marked on respective characters. In authentication, first the user identifies chosen characters and clicks the password points on the right characters. The server checks each user-clicked point on the image to find the nearest clickable point. Login fails if their distance exceeds the tolerable range. Otherwise, the hash value is calculated based on the recovered sequence of clickable points to verify it with the stored value.

Comparing password points is similar between TextPoints and click-based graphical passwords (PassPoints). In PassPoints, salient points should be avoided because they are mounted to dictionary attacks, but in avoiding salient points leads to memorizing the password (Wiedenbeck, Waters, Birget, Brodskiy, & Memon, 2005). But when considering TextPoints, this problem will not occur. ClickPoints in TextPoints refers to salient points of their characters and helps to memorize a password, but bots cannot exploit this because both are dynamic and contextual:

- Dynamic: For every image, the locations of clickable points and their characters differ from one another. Clickable points of one image are independent of clickable points of another image.
- Contextual: The similar structured point does not depend on its context or a clickable point, this is only with the right context, i.e., at the right location of a right character.

Bots or machines cannot exploit this to mount dictionary attacks on TextPoints because both dynamic and contextual requirements correct context, i.e., characters, first. Identifying characters in a Captcha image is a task beyond a computer's capability.

Technical measures like login passwords and anti-virus protection are also essential. However, a secure physical space is the first and most important line of defense.

Is the place you keep your work environment PC sufficiently secure to anticipate burglary or access to it while you are away? While the Security Department gives scope over the Medical focus, it just takes seconds to take a PC, especially some compact gadgets like a tablet or a PDA. A PC ought to be secured like any other important owned item when you are absent. Human dangers are not by any means the only concern. PCs can be damaged by natural accidents (e.g., water, espresso) or physical injury. Ensure the physical area of your PC assesses those dangers as well**.**

From Figure 19, the registration process starts with the generation of a unique ID for a user and then on to the next step, it prompts to select a sound signature with a tolerance level and based on the tolerance level it generates an image. If the user failed to pass the Captcha test then a more complex Captcha will generate, after 'n' number of failed attempts it blocks the user. Once the Captcha is authenticated correctly the user can create his profile.

After creating a profile, the user can login with the same image profile that was used while creating the profile. Based on the mouse clicks or Captcha challenge, if these mouse clicks match with the database clicks then the user can log in, if not it asks the user to solve the Captcha again. With three failed attempts it will block the user from logging in.

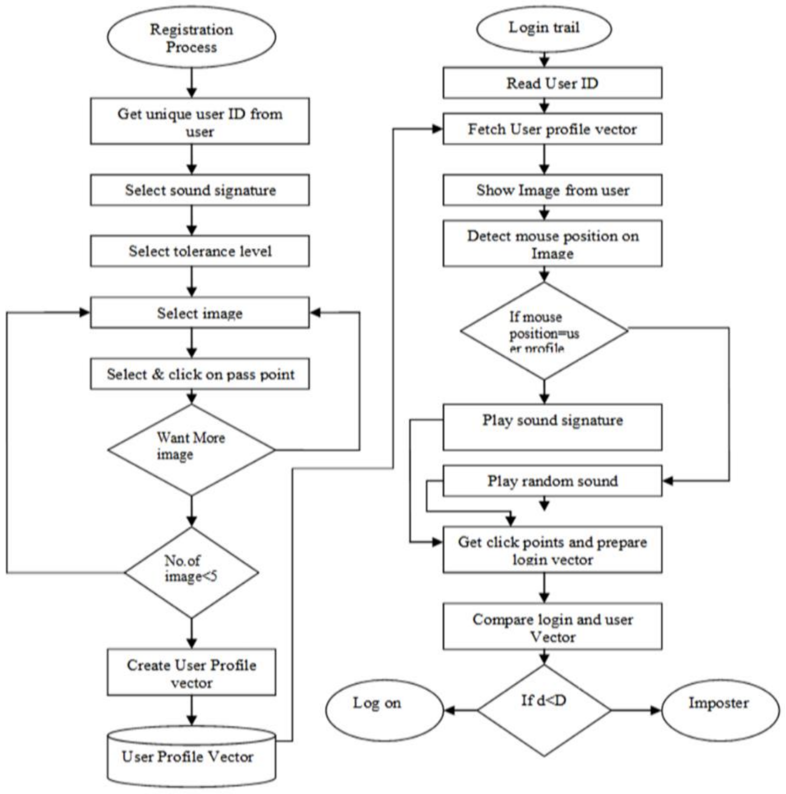

Registration Process
Get unique user ID from user
Select sound signature
Select tolerance level
Select image
Select & click on pass point
Want More image
No.of image<5
Create User Profile vector
User Profile Vector
Login trail
Read User ID
Fetch User profile vector
Show Image from user
Detect mouse position on Image
If mouse position=user profile
Play sound signature
Play random sound
Get click points and prepare login vector
Compare login and user Vector
If d<D
Log on
Imposter

Figure 19. Flow Chart

Chapter IV. Security Analysis

Security of Underlying Captcha

There is no theoretic security model available yet. All existing models on Captcha Security are the only approximate process. All the modern Captcha schemes depend on segmentation of the object which is computationally expensive, and a combinatorically difficult problem.

The complexity of object segmentation, C, is exponentially dependent on the number M of objects contained in a challenge, and the polynomial is dependent of the size N of the Captcha alphabet: $C=\alpha^{M}P(N)$, where $\alpha > 1$isaparameter, and P () is a polynomial function (Chellapilla, Larson, Simard, & Czerwinski, 2005). A Captcha challenge typically contains 6 to 10 characters, whereas a CaRP image typically contains 30 or more characters. The complexity to break a Click-Text image is about α^{30} P(N)/(α^{10} P(N)) = α^{20} times the complexity to break a Captcha challenge generated by its underlying Captcha scheme.

From above, ClickText is much harder to break than its underlying Captcha scheme. CaRP characters are 2D, if the user changes to one more dimension, it increases its segmentation difficulty. By doing this for improved usability, the user can decrease the distortions in ClickText images. Whereas ClickAnimal depends on both multiple-label classification and object segmentation.

CaRP does not depend on a specific Captcha, if one Captcha fails a new and more robust Captcha scheme may appear and be used to construct a new CaRP scheme.

Automatic Online Guessing Attacks

In automatic online guessing attacks, dictionaries are constructed manually, and then a trial and error method is automatically implemented. CaRP has some CPA-secure Captcha if the user ignores some probabilities. It has the following properties:

1. Internal object-points on one CaRP image are computationally-independent of internal object-points on another CaRP image. Particularly, clickable points on one image are computationally-independent of clickable points on another image.
2. Trials in guessing attacks are mutually independent.

The first property can be proven by contradiction. Assume that the property does not hold, i.e., there exists an internal object-point α on one image A that is non-negligibly dependent of an internal object-point β on another image B. An adversary can exploit this dependency to launch the following chosen-pixel attack. In the learning phase, image A is used to learn the object that contains point α. In the testing phase, point β on image B is used to query the oracle. Since point α is non-negligibly dependent of point β, this CPA-experiment would result in a success probability non-negligibly higher than a random guess, which contradicts the CPA-secure assumption. User concludes that the first property holds.

The second property is a consequence of the first property since user-clicked internal object-points in one trial are computationally-independent of user-clicked internal object-points in another trial due to the first property. The user has ignored background and boundary object-points since clicking any of them would lead to authentication failure.

User can say that the only way to guess a password in automatic online guessing attacks is by probabilistically, regardless of how many trials and errors. Even if you know the correct password which is to be tested, the trial has very little chance to pass because bots do not have the capability to identify the objects in the CaRP image. Here the number of trials is also limited. In a brute-force or dictionary attack, if you know the correct password, this would succeed in compromising existing graphical passwords.

Human Guessing Attacks

In human guessing attacks, humans need to enter a password manually, which is slower compared to machine. For 8-character passwords, the password space is $33^8 \approx 2^{40}$ for ClickText with an alphabet of 33 characters, $10^8 \approx 2^{26}$ for ClickAnimal with an alphabet of 10 animals, and $10 \times 46^7 \approx 2^{42}$ for AnimalGrid with the setting as ClickAnimal plus 6×6 grids.

If the user assumes that 1000 people are employed to work eight hours per day without any breaks in a human guessing attack, and that each person takes 30 seconds to finish one trial. It would take them on average $0.5 \cdot 33^8 \cdot 30/ (3600 \cdot 8 \cdot 1000 \cdot 365) \approx 2007$ years to break a ClickText password, $0.5 \cdot 10^8 \cdot 30/(3600 \cdot 8 \cdot 1000) \approx 52$ days to break a ClickAnimal password, or $0.5 \cdot 10 \cdot 46^7 \cdot 30/(3600 \cdot 8 \cdot 1000 \cdot 365) \approx 6219$ years to break an AnimalGrid password (Bonneau, 2012)..

Because of larger password space for TextPoints, it requires a much longer time than those on ClickText.

A recent study on text passwords indicates that users tend to choose passwords of 6–8 characters and have a strong dislike of using non-alphanumeric characters, and that an acceptable benchmark of effective password space is the expected number of optimal guesses per account needed to break 50% of accounts, which is equivalent to 21.6 bits for Yahoo! Users (Bonneau, 2012).

Assume that ClickText has roughly the same effective password space as text passwords, it requires on average of 1000 people to work 1.65 days or one person to work 4.54 years to find a ClickText password.

Relay Attacks

Human surfers act as the relay to solve Captcha challenges for further surfing websites, or sweatshops where humans are hired with small payments to pass Captcha challenges. But in CaRP it has the CbPA-protocol which is robust for relay attacks (von Oorschot & Stubblebine, 2006). Methods used by humans for Captcha challenge are very different from CaRP. So, for CaRP, it needs a large number of unwitting people to mount human guessing attacks.

Let us assume that sweatshops are hired to mount human guessing attacks and the cost to click one password on CaRP is $1, at the lowest. If there are 1000 Captcha Challenges the estimated cost is the average cost to break a 26-bit password is $0.5 \cdot 2^{26} \cdot 1/1000$ or about $33,600 US dollars (Motoyama et al., 2012).

Shoulder-Surfing Attacks

CaRP is not robust to shoulder-surfing attacks, but when CaRP is combined with dual-view technology, CaRP can thwart shoulder-surfing attacks. Shoulder-surfing attacks are dangerous for instance when entering graphical passwords in bank ATM machines. Normal or commonly used LCDs are limited to brightness and color depending on view angles, but a software is used in dual-view technology so that it displays on two LCD screens concurrently, where one LCD has one public image in most viewable angles and another private image in a specific view-angle (Kim, Cao, Zhang, & Tan, 2012). The CaRP image is on the private image, a shoulder-surfing attacker can collect user-clicked points on the screen but cannot capture the private CaRP image the user clicked points because only the user can see it. The user-clicked points collected by an attacker are useless because for every login attempt CaRP generates a new computationally-independent image, so captured points will not be useful.

Whereas, for a traditional implementation of graphical passwords such as PassPoints which uses a static image for every login attempt, even though the image is on private LCD by dual-view technology, the captured user click points can be used to log in successfully.

Chapter V: System Analysis and Design

Feasibility Study

In this level based on the cost estimates and other aspects, a feasibility report will be generated for a business proposal. This report gives an overall picture of the project so that they can conclude whether to go forward with this research or not. For a better understanding of this feasibility study, needs major requirements for the system. Three key considerations involved in the feasibility analysis are Economical Feasibility, Technical Feasibility, and Social Feasibility.

Economical feasibility. This study is most essential for any organization because this explains the economic impact on the organization. Based on this report, the organization will conclude whether to invest in research and development or not. Unless the clear picture, they won’t go any further. With open sources, the cost is often less to invest in software products.

Technical feasibility. Technical feasibility is to verify whether the equipment or software tools that are used won't negatively impact the future, i.e., there should not be any huge changes for implementation. If not, there will be a high demand for resources in the future.

Social feasibility. Social feasibility also plays a vital role in any organization. A higher level of user acceptance is required to continue research. The user requires some knowledge of the subject before using it, so this needs to be taken into account to invest money in training the user for using the system effectively. The methods that are used to educate the user will determine the level of acceptance which also helps them to get familiar with the system.

Existing System

There are several types of Captchas being used in the real world and out of those the most commonly used technique to differ humans and bots are graphical based Captcha. It is hard to crack, but not impossible as bots are becoming more intelligent. Captcha is currently a standard Internet security system to shield online email and different administrations from being mishandled by bots.

However, there are several drawbacks such as they are difficult to read, not compatible for users with disabilities, can be time-consuming to decipher, and there are technical difficulties with certain browsers which may be greatly enhanced by the use of artificial intelligence. Some of the images are not readable due to high distortion or a greater complexity level on the image, which makes letters overlapped on the image.

Proposed System

In this paper, another security primitive considering the difficult AI issues mentioned above and a novel group of graphical secret key frameworks based on Captcha innovations, which call Captcha as graphical passwords (CaRP). To add more security, implementing ClickText is used, which is a recognition-based CaRP scheme. As ClickText is an advanced version of CaRP, it provides more security to clients.

In addition to distorted images, if the click-based mechanism is used to select a password, it is more complex to solve the puzzle. ClickText has roughly the same effective password space as text passwords, and it requires on average 1000 people to work 1.65 days or one person to work 4.54 years to find a ClickText password.

Hardware Requirements:

System	Pentium IV 2.4 GHz
Hard Disk	40 GB
Monitor	Any (1)
Mouse	Any (1)

Ram	512 Mb

Software requirements:

Operating System	Windows XP/7/Linus
Coding Language	JAVA/J2EE
IDE	Elipse/NetBeans 7.4
Database	MYSQL
Server	Apache Tomcat 7/Glassfish 4.1.1
Browser	Chrome/IE/Firefox

System Design

System architecture.

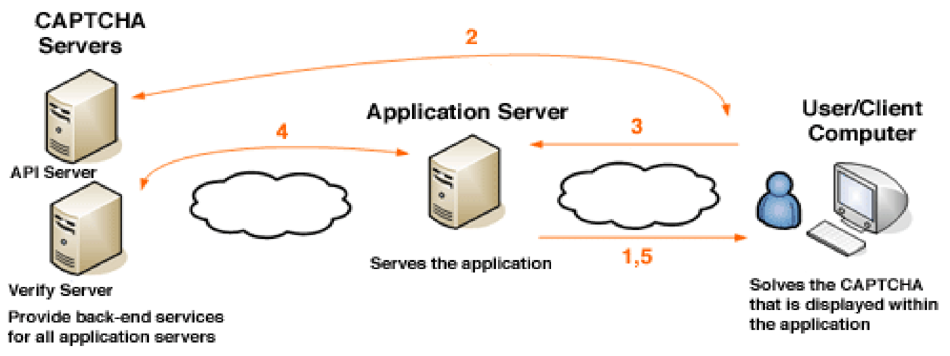

CAPTCHA Servers
2
API Server
4
Application Server
3
User/Client Computer
Verify Server
Provide back-end services for all application servers
Serves the application
1,5
Solves the CAPTCHA that is displayed within the application

Figure 20. System Flow

Data flow diagram.

1. The DFD is also called a bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data that is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools currently used. It is used to model the system components. These components are the system

processes, the data used by the processes, an external entity that interacts with the system and the information that flows in the system.

3. The DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

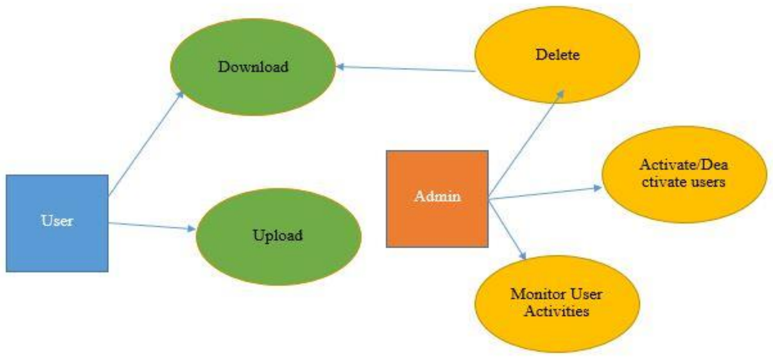


Figure 21. Input Design

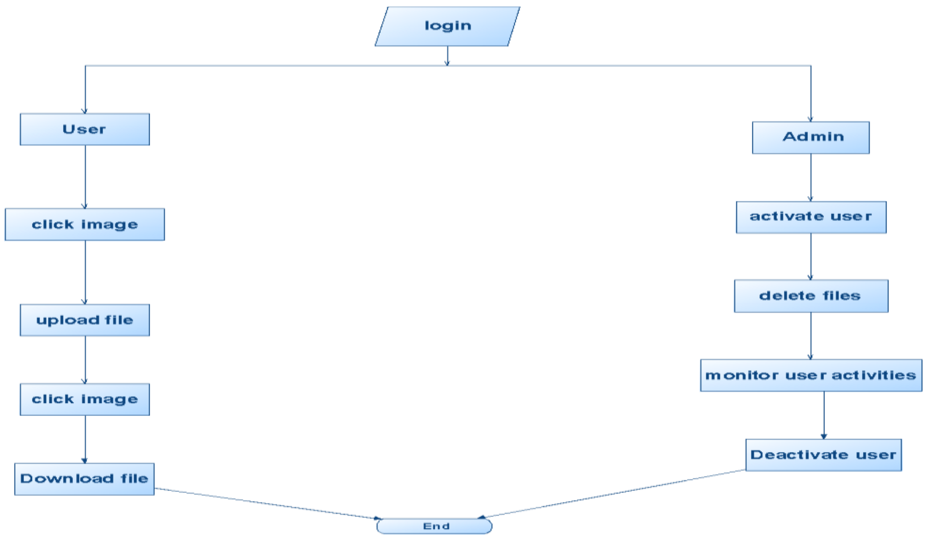


Figure 22. Data Flow

UML diagrams. UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form, UML is comprised of two major components—a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying visualization, constructing and documenting the artifacts of a software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of programming languages and the development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.

6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Class diagram. In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

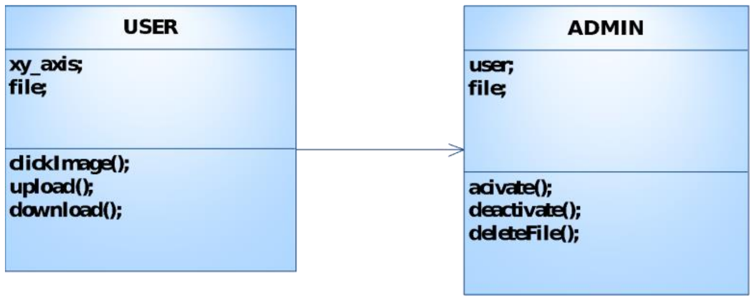


Figure 23. Class Diagram

Use case diagram. A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

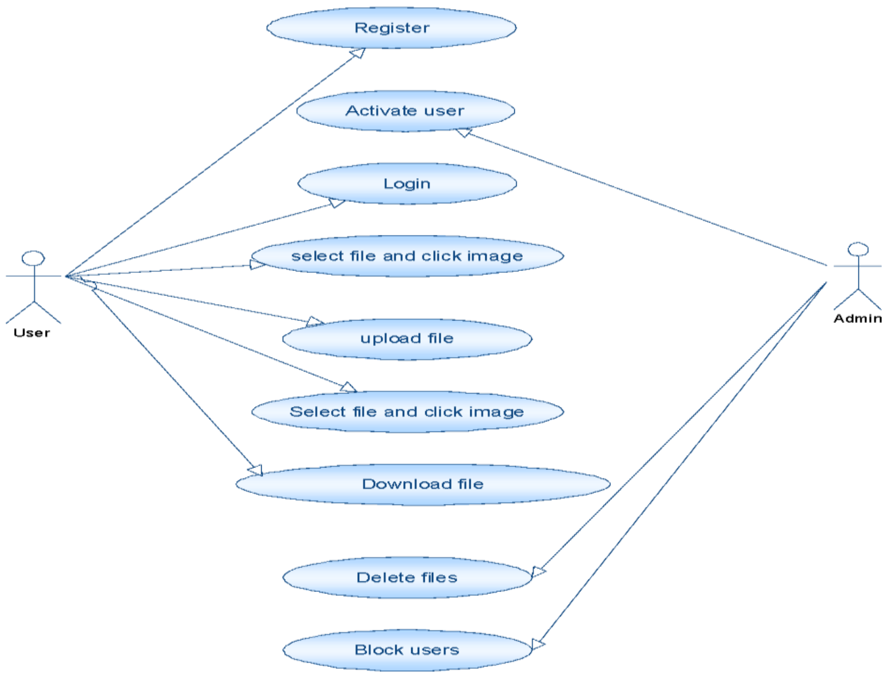


Figure 24. Use Case Diagram

Sequence diagram. A sequence diagram in the Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

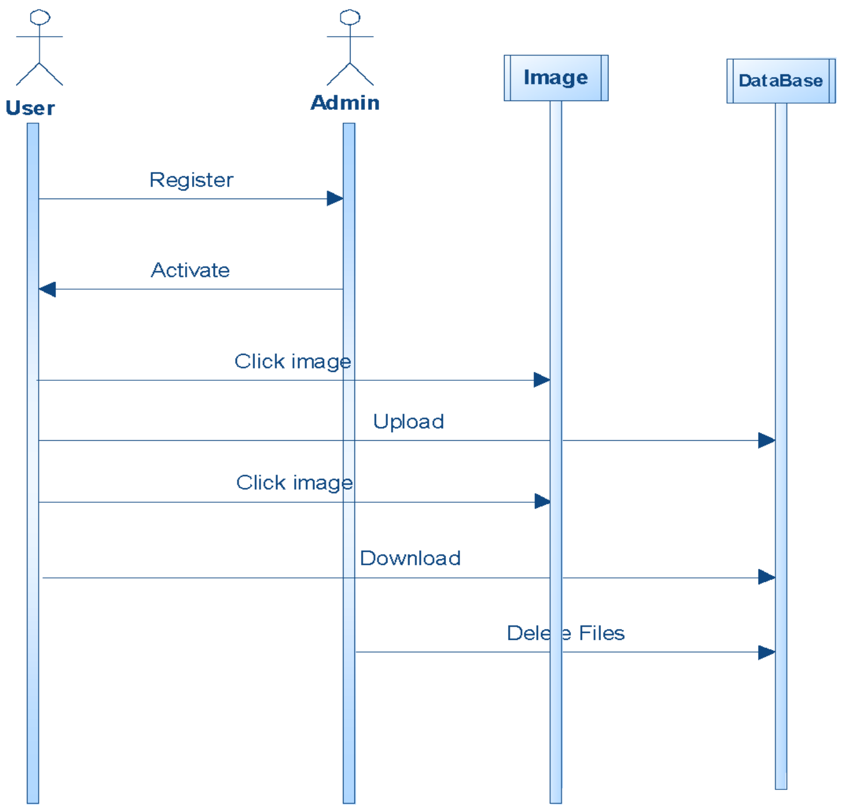


Figure 25. Sequence Diagram

Activity diagram. Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

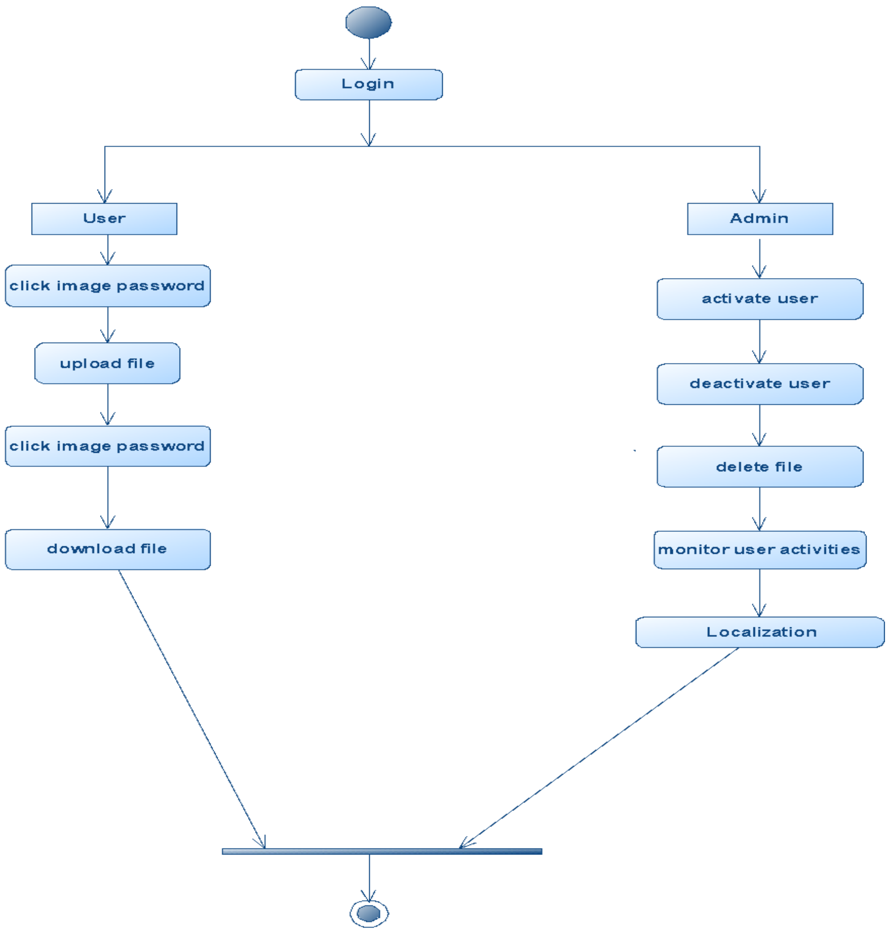


Figure 26. Activity Diagram

Chapter VI: System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is also the process of exercising software with the intent of ensuring that the it meets its requirements, user expectations and does not fail in an unacceptable manner. There are various types of test that can be performed. Each type of test addresses a specific testing requirement.

Types of Tests

Unit testing. Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that the program inputs produce valid outputs. All decision branches and internal code flow should be validated by this test. It is the testing of individual software units of the application. It is done after the completion of an individual unit but before integration. This is a structural test, that relies on knowledge of its construction and it is invasive. Unit tests perform basic tests at the component level and test a specific business process, application, and/or system configuration. Unit tests also ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted in two distinct phases.

- Test strategy and approach. Field testing will be performed manually, and functional tests will be written in detail.

- Test objectives
    - All field entries must work properly.
    - Pages must be activated from the identified link.
    - The entry screen, messages and responses must not be delayed.
- Features to be tested
    - Verify that the entries are of the correct formats
    - No duplicate entries should be allowed.
    - All links should take the user to the correct page.

Integration testing. Integration tests are designed to test integrated software components to determine if they run as one cohesive program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is performing correctly and consistently. Integration testing is specifically aimed at exposing the problems that arise from the combination of these components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level interact without error.

All the test cases mentioned above passed successfully, and no defects were encountered.

Acceptance testing. User acceptance testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

All the test cases mentioned above passed successfully, and no defects were encountered.

Functional testing. Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input: identified classes of valid inputs must be accepted.
- Invalid Input: identified classes of invalid inputs must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

The organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, the systematic coverage pertaining to identifying business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System testing. System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

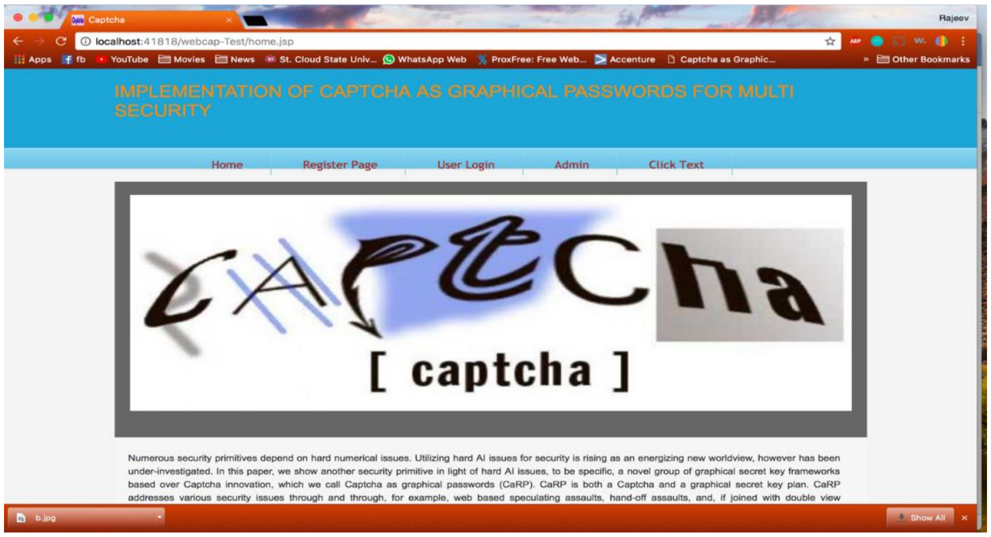
There will be different layers to test and this system testing comes under a higher environment and will have restricted access to normal testers.

White box testing. White box testing is a test in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. Its purpose is used to test areas that cannot be reached from a black box level.

Black box testing. Black box testing is a test of the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as with most other kinds of tests, must be written from a definitive source document, such as a specification or requirements document. It is a test in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Chapter VII: Implementation

The main modules of the application are Registration, Login, Upload, Download, and ClickText. Each module uses a different type of mechanism. All of these modules functionalities will be described below.



IMPLEMENTATION OF CAPTCHA AS GRAPHICAL PASSWORDS FOR MULTI SECURITY

Numerous security primitives depend on hard numerical issues. Utilizing hard AI issues for security is rising as an energizing new worldview, however has been under-investigated. In this paper, we show another security primitive in light of hard AI issues, to be specific, a novel group of graphical secret key frameworks based over Captcha innovation, which we call Captcha as graphical passwords (CaRP). CaRP is both a Captcha and a graphical secret key plan. CaRP addresses various security issues through and through, for example, web based speculating assaults, hand-off assaults, and, if joined with double view

Figure 27. Home Page

Registration

Registration uses the normal Captcha technique which creates some discrete images with random texts and alphabets. These are generated at runtime; this page then asks the user to memorize the Captcha text which is a simple word along with the password. The generated Captcha text is used to login along with the password. Each registration should be different for each user, otherwise it will throw an error stating that the user already exists. The successful login is saved to a database along with the image that is generated during the registration. Below, Figure 28 shows that using the same Captcha that was generated while registering.

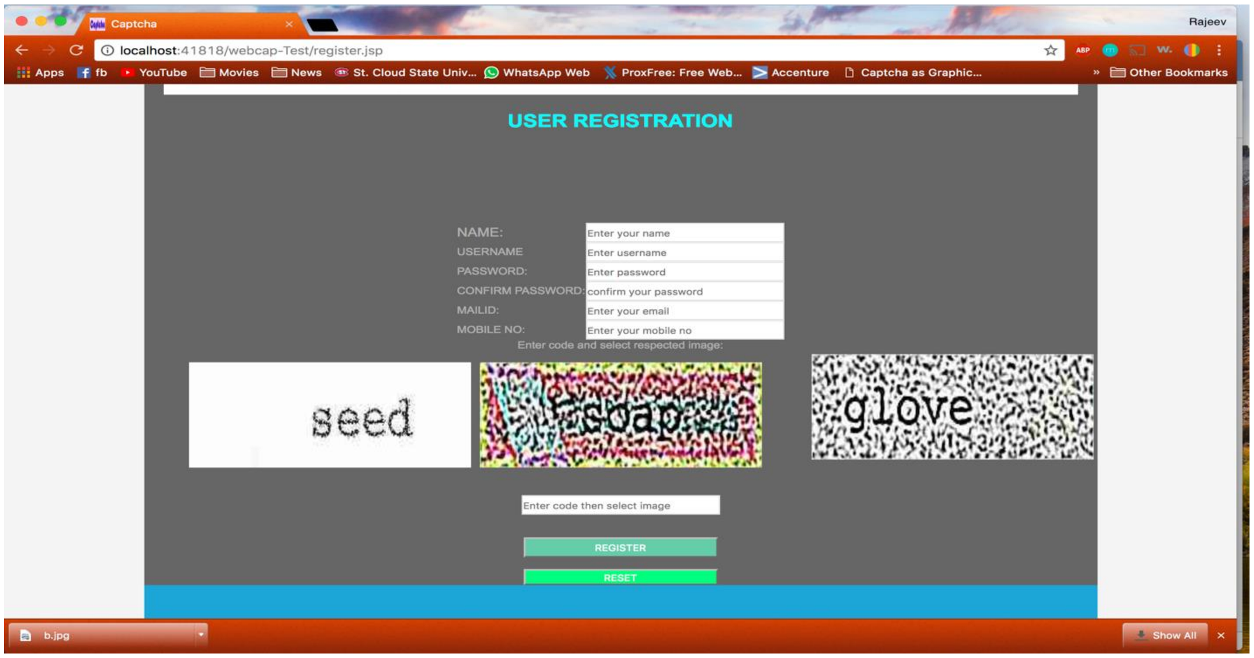


Figure 28. Registration Page

Login

Login functionality is an extension of the registration page. It pulls all the data associated with a specific user at the time of login. It displays three Captcha images, out of those one image is from the registration page. To log in successfully the user must provide their username, password and Captcha image (from registration page). After three failed login attempts, the user is blocked and needs an admin to reactivate the user's account and an email will be sent to the email address specified at the time of registration. There are two different types of logins—one with using the same Captcha used while registering and the other is just to validate the user for the registration page not carrying the same Captcha to the login page. Admin login is basic login page used for monitoring user activities.

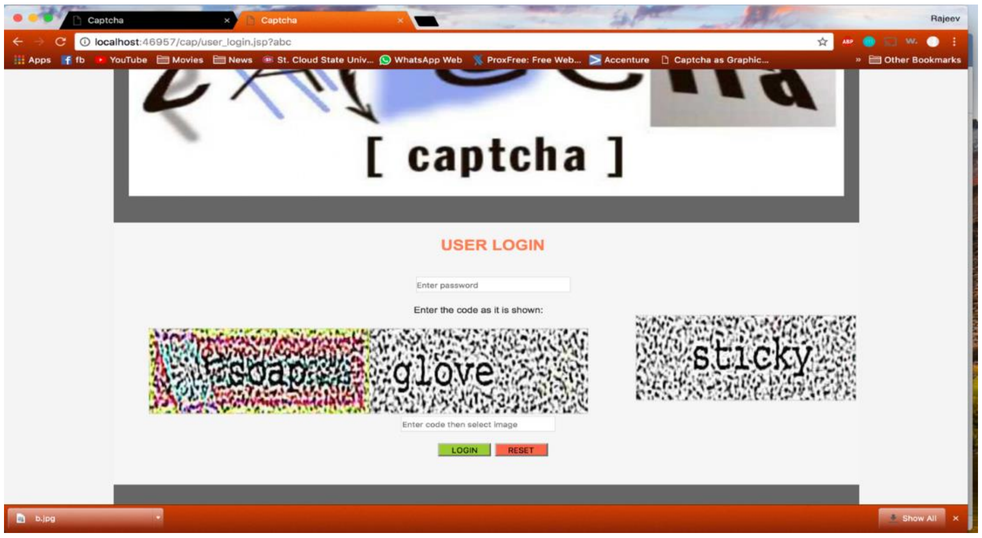


Figure 29. Login Page

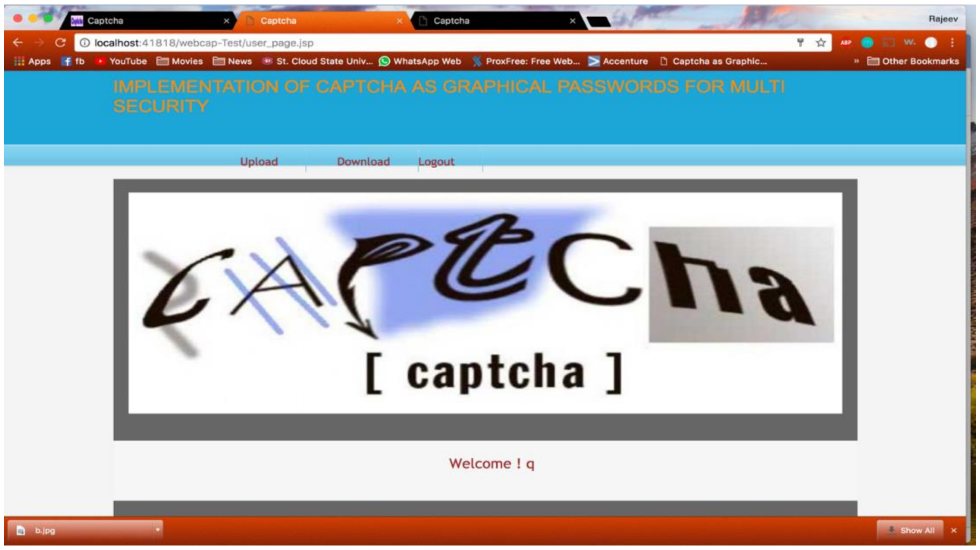
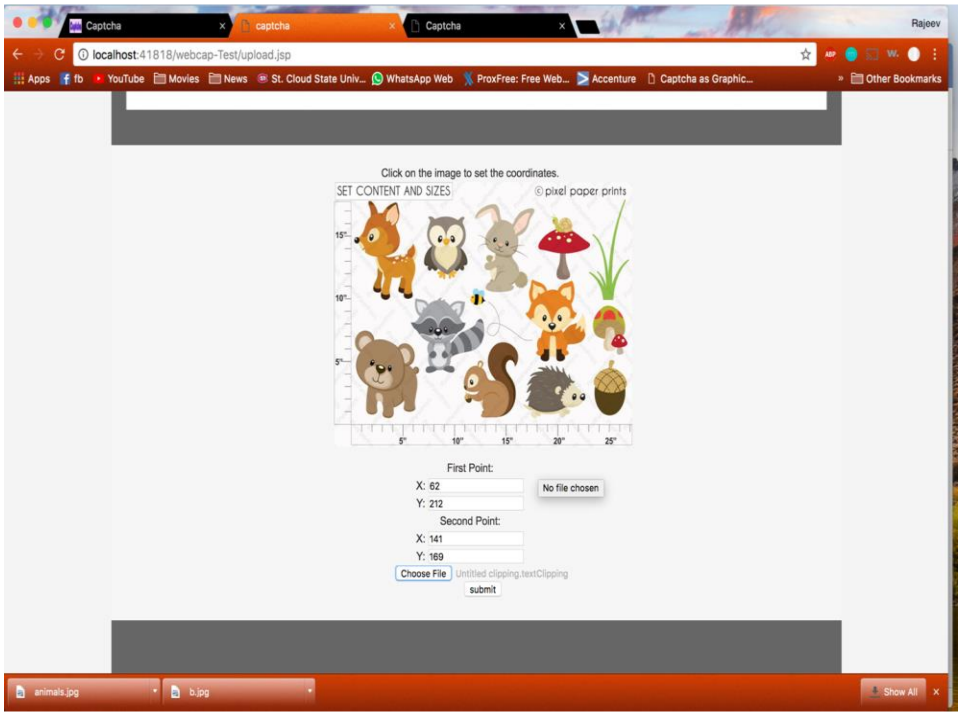

IMPLEMENTATION OF CAPTCHA AS GRAPHICAL PASSWORDS FOR MULTI SECURITY
Upload
Download
Logout
Welcome ! q

Figure 30. Welcome Page

Upload

After successful login, the user has different options and upload is one among them. In this page, an image is shown to the user and asks to select two locations or coordinates on the image to upload any file. These coordinates will be stored in the database only when there is a proper upload.

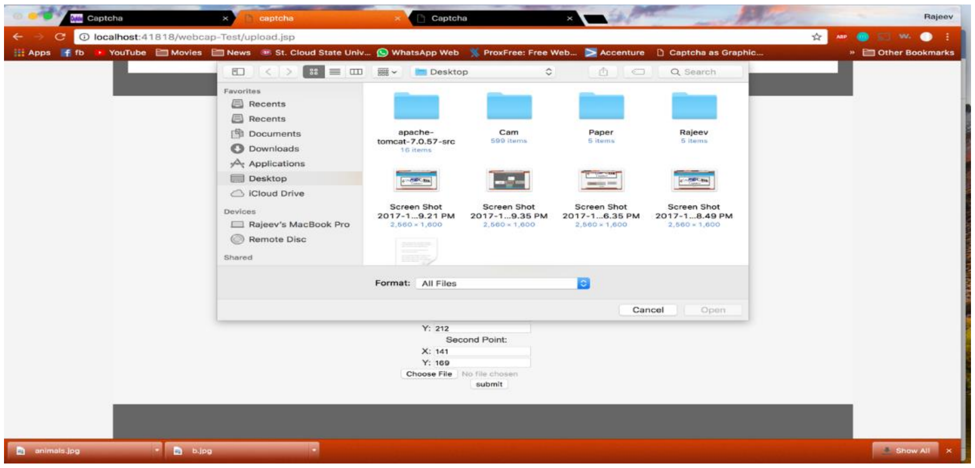

localhost:41818/webcap-Test/upload.jsp

Click on the image to set the coordinates.

First Point:
X: 62
Y: 212
Second Point:
X: 141
Y: 169

Choose File Untitled clipping.textClipping

Figure 31. Upload Page

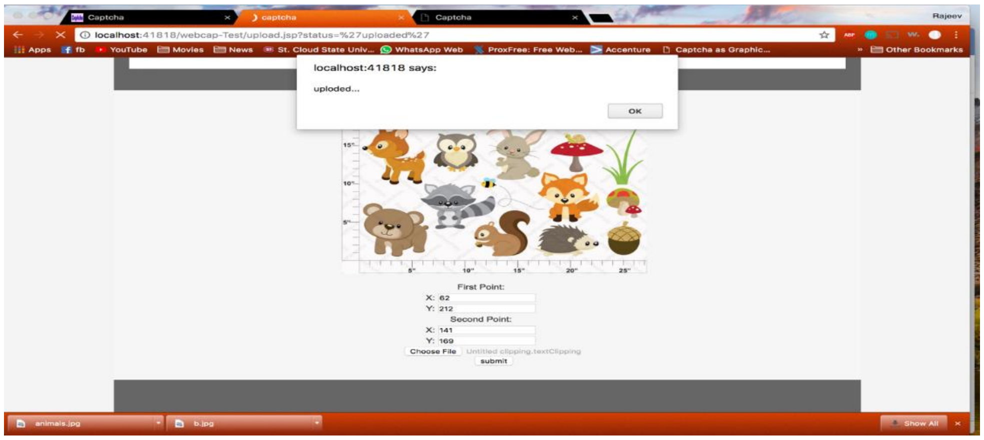

localhost:41818/webcap-Test/upload.jsp

Y: 212
Second Point:
X: 141
Y: 169

Choose File No file chosen

Format: All Files

Figure 32. Attaching a File to the Upload Page


localhost:41818/webcap-Test/upload.jsp?status=%27uploaded%27

localhost:41818 says:

uploded...

First Point:
X: 62
Y: 212
Second Point:
X: 141
Y: 169

Figure 33. Successful Upload Page

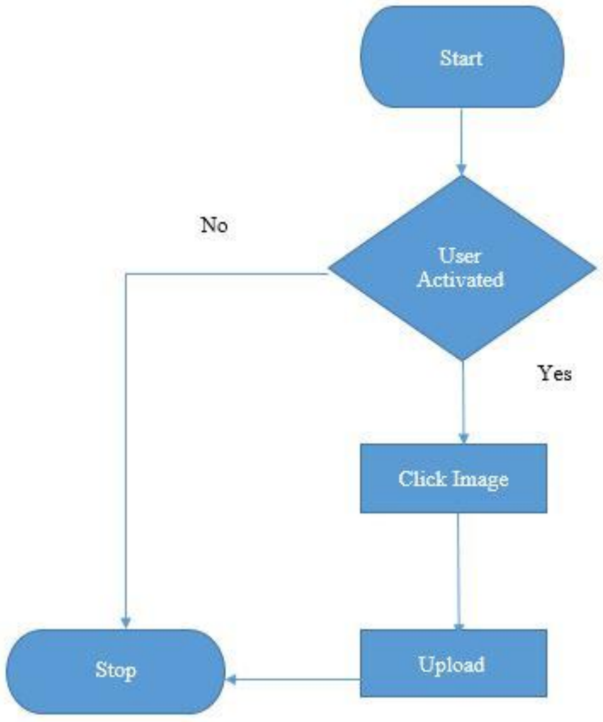
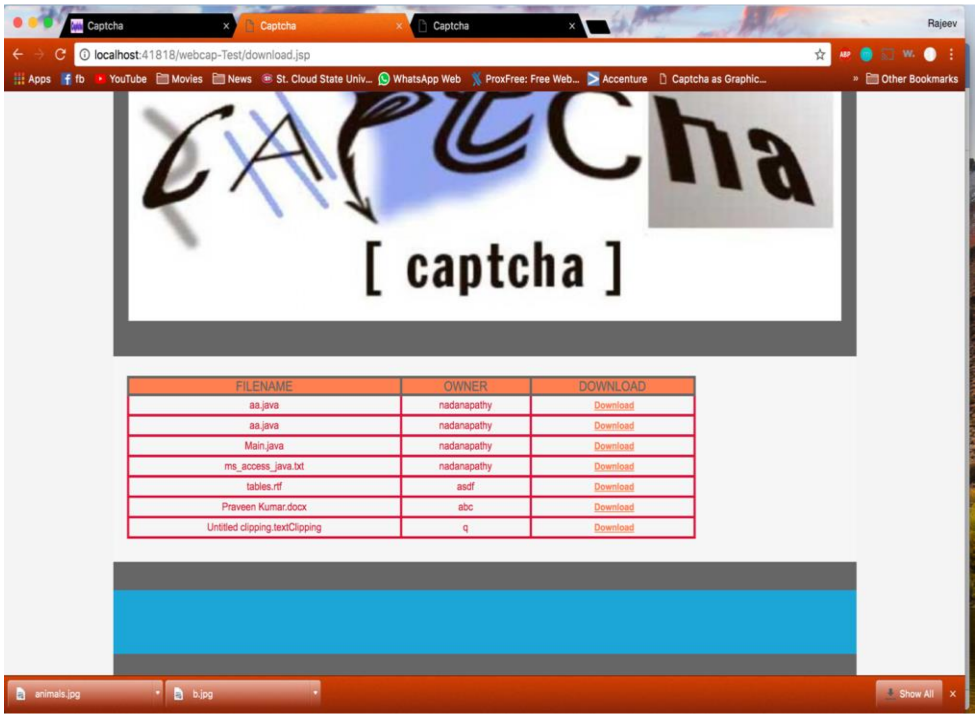


Figure 34. Upload Flowchart

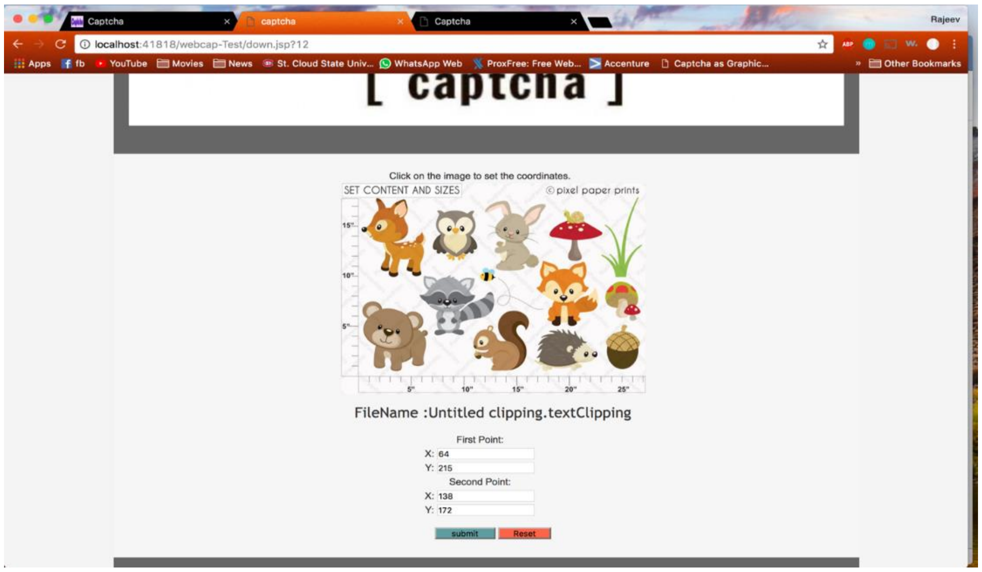
Download

If the user previously uploaded any file, this page will show the name of the file along with a download link. After clicking on the download link on the respective filename, the user will be shown the same image at the time of uploading the file. The user then needs to click on the same coordinates, with some tolerance allowed, to download the file. Matching coordinates will result in the downloading of the file, otherwise a popup with the error message: “Invalid Coordinates” will appear.



FILENAME	OWNER	DOWNLOAD
aa.java	nadanapathy	Download
aa.java	nadanapathy	Download
Main.java	nadanapathy	Download
ms_access_java.txt	nadanapathy	Download
tables.rtf	asdf	Download
Praveen Kumar.docx	abc	Download
Untitled clipping.textClipping	q	Download

Figure 35. Download Page

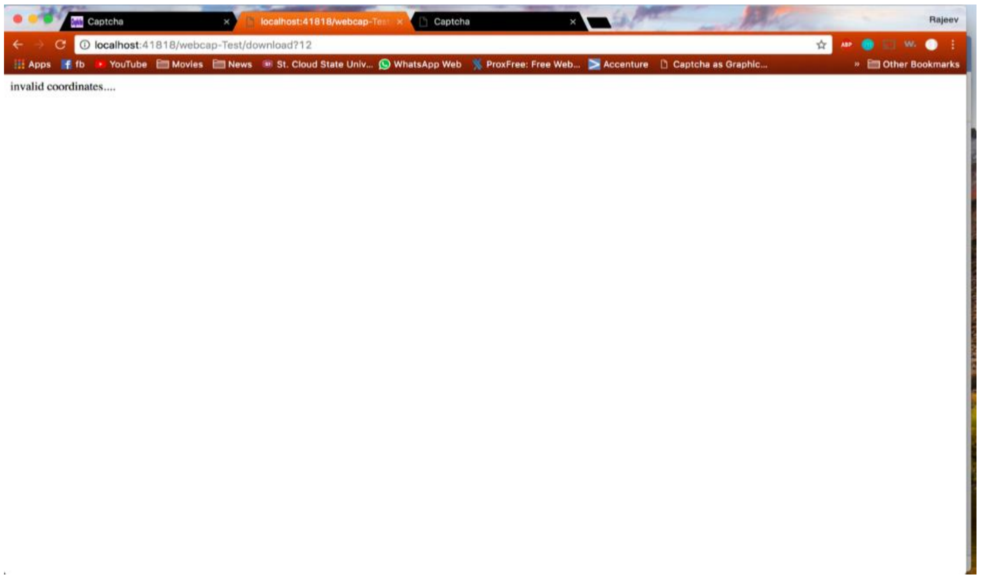


Click on the image to set the coordinates.

FileName :Untitled clipping.textClipping

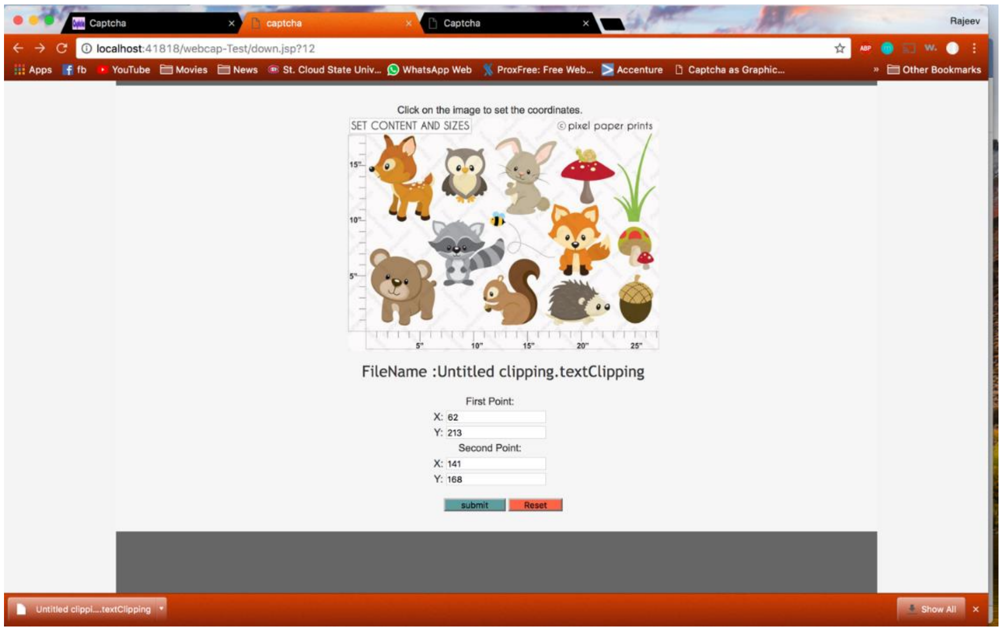
First Point:
X: 64
Y: 215
Second Point:
X: 138
Y: 172

Figure 36. Coordinates Download Page



invalid coordinates....

Figure 37. Invalid Coordinates Page



Click on the image to set the coordinates.

FileName :Untitled clipping.textClipping

First Point:
X: 62
Y: 213
Second Point:
X: 141
Y: 168

submit Reset

Figure 38. Successful Download Page

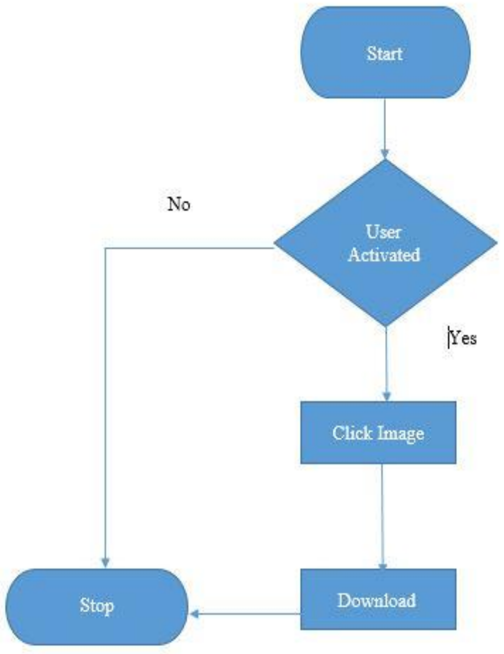


Figure 39. Download Image

ClickText

This page is implemented from recognition-based CaRP, like the registration page but instead of Captcha images, a 4x4 discrete image with alphanumeric symbols is used. The user then selects a password by clicking on the letters or numbers from the image. A random image will be generated each time and is stored in the database along with the user details. This helps to retrieve the same image at the time of login. There is an option to select from an image or you can enter the password. This is implemented by using a Discretized Centralization Algorithm. Discretization is a method which is used in the click-based passwords. So that approximately correct entries can be accepted by the system (Chiasson, Srinivasan, Biddle, & van Oorschot, 2008). After generating the password, it uses the SHA-1 algorithm to make it encrypted by padding data on both sides of the password.

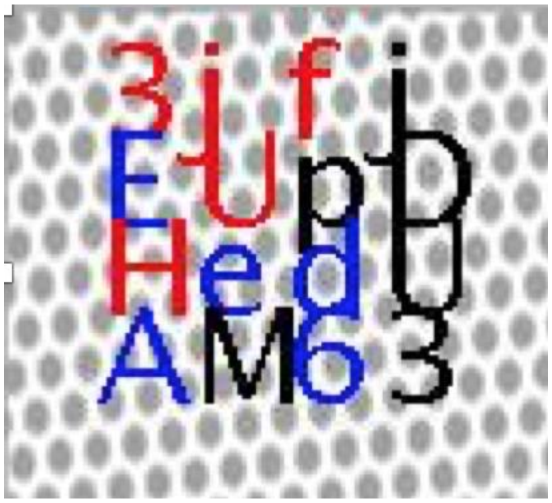
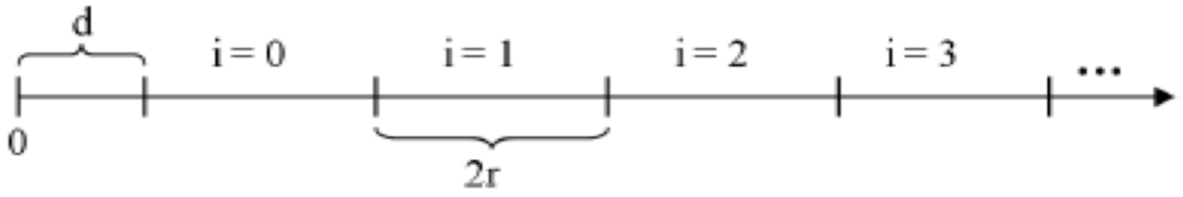


Figure 40. ClickText Image

Let us consider the password ‘3Ud6’, so that the user clicks on the image with letters in the correct sequence. The discretized centralization algorithm is as follows:



i = [(x-r)/2r], d=(x-r)mod2r, i' =[(x'-d)/2r]

where i=index, d=displacement, i'=index reverse and r=tolerance

From Figure 35, if the coordinates of 'U' are (30,20) and tolerance r=5 then

i=(x-r)/2r = (30-5)/10=25/10=2

d=(x-r) mod 2r = (30-5) mod 10 = 25 mod 10 = 5.

Suppose the user clicks on (34,27) then

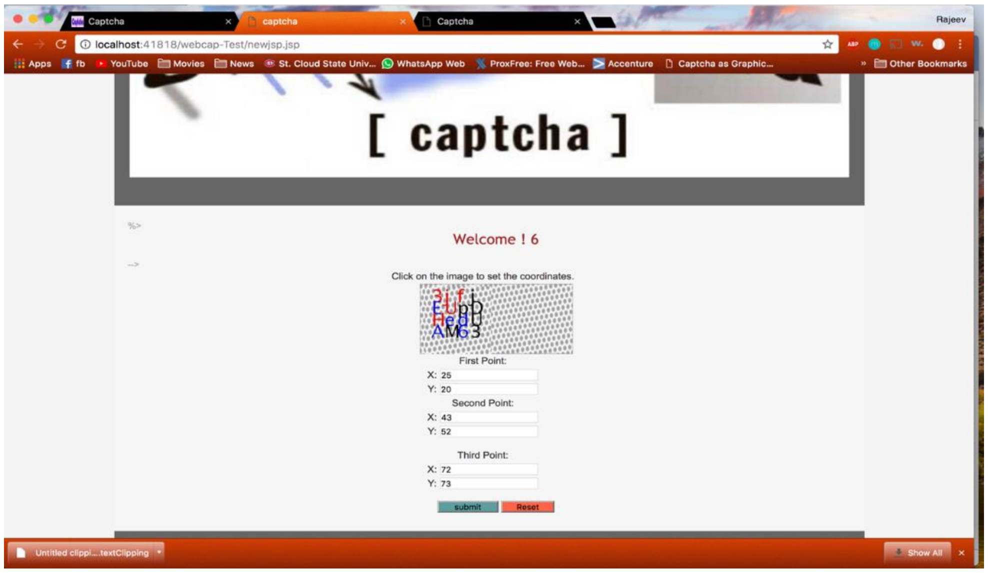
i' = (x'-d)/2r = (34-5)/10 = 2 from this

i = i' which is acceptable.

Suppose the user clicks on (24,27) then

i' = (x'-d)/2r = (24-5)/10 = 1 from this i not equal to i' so this is rejected.

With Centered Discretization, the rate of false accepts and false rejects is zero by definition since centered-tolerance implies that the system will only accept click-points that are within *r* from the original point (Chiasson et al., 2008).



Welcome ! 6

Click on the image to set the coordinates.

First Point:
X: 25
Y: 20
Second Point:
X: 43
Y: 52
Third Point:
X: 72
Y: 73

Figure 41. ClickText Page for Registration

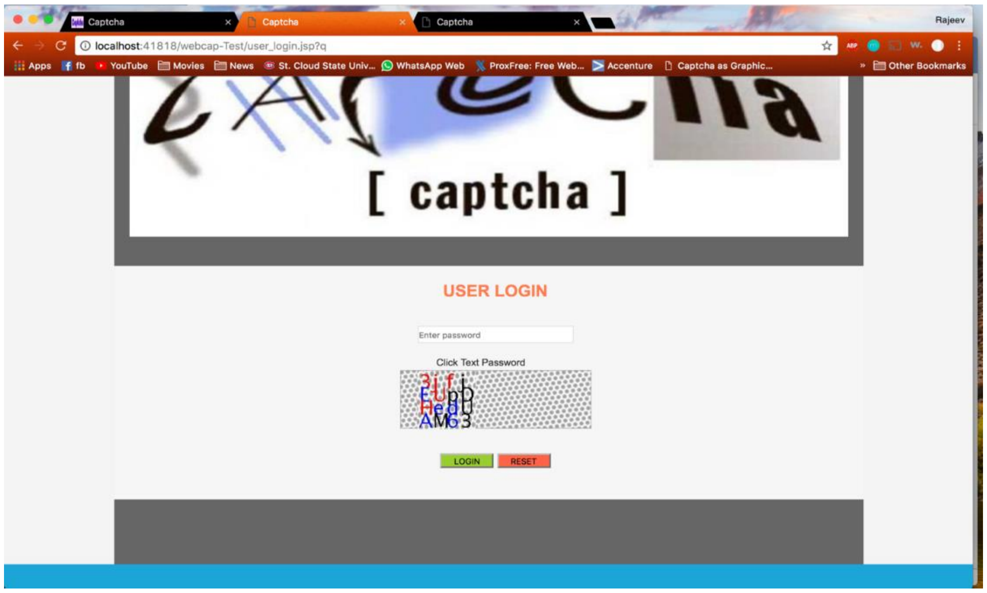


Figure 42. ClickText Page User Login

Admin

The admin module is a special feature to keep track of all the user activities and, if in any case something went wrong, this module helps to get the user back to active. Also, you can keep track of all uploaded files and timestamps. With this module, the admin can activate or deactivate any user. There is a total of three pages for admin:

1. Login page used to validate admin credentials.
2. User Activities page is used to monitor user details like activating, deactivating and updating phone numbers or addresses.
3. Upload files page is used to keep track of all the uploaded files along with a date stamp and the user who uploaded it.

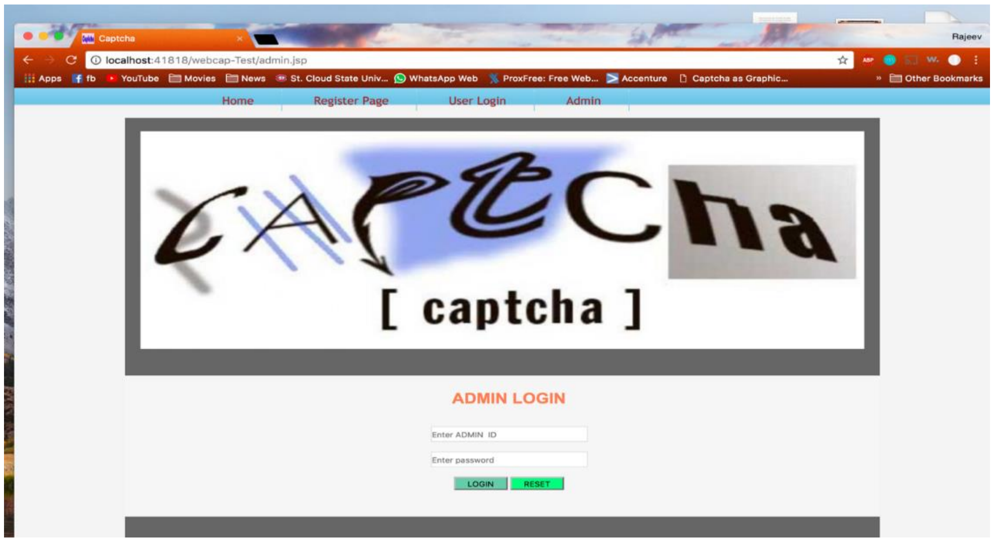
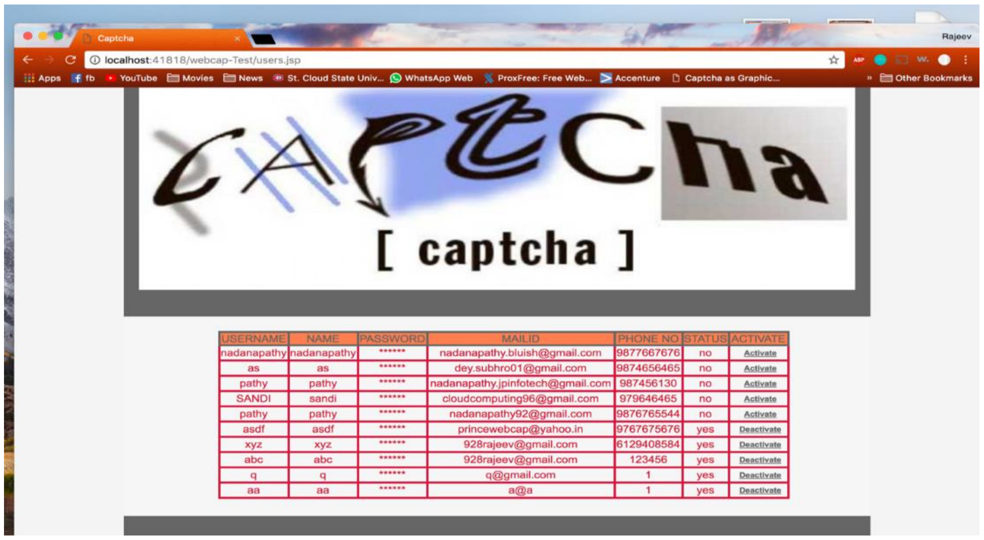
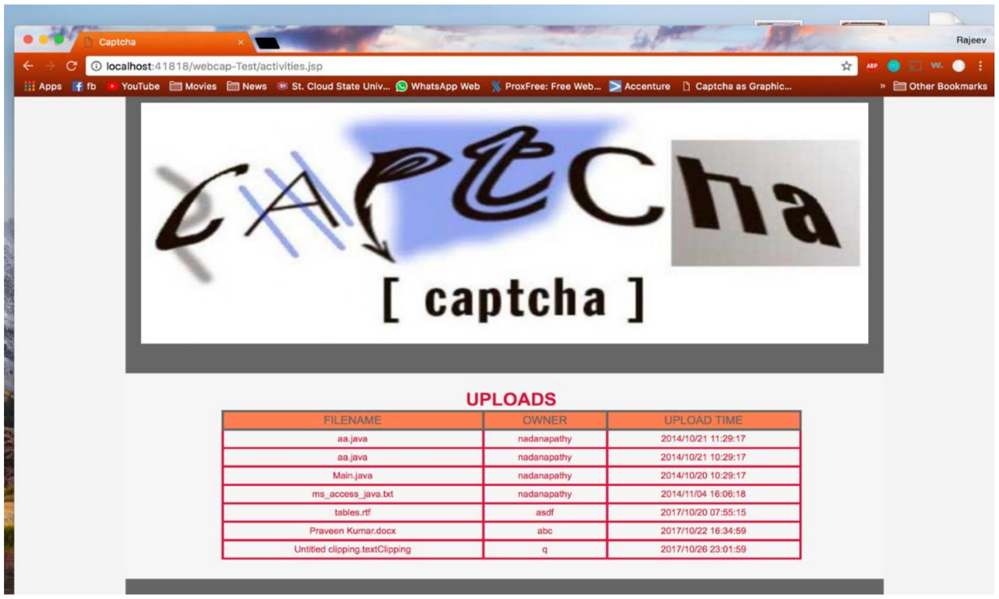


Figure 43. Admin Login



USERNAME	NAME	PASSWORD	MAILID	PHONE NO	STATUS	ACTIVATE
nadanapathy	nadanapathy	******	nadanapathy.bluish@gmail.com	9877667676	no	Activate
as	as	******	dey.subhro01@gmail.com	9874656465	no	Activate
pathy	pathy	******	nadanapathy.jpinfotech@gmail.com	987456130	no	Activate
SANDI	sandi	******	cloudcomputing96@gmail.com	979646465	no	Activate
pathy	pathy	******	nadanapathy92@gmail.com	9876765544	no	Activate
asdf	asdf	******	princewebcap@yahoo.in	9767675676	yes	Deactivate
xyz	xyz	******	928rajeev@gmail.com	6129408584	yes	Deactivate
abc	abc	******	928rajeev@gmail.com	123456	yes	Deactivate
q	q	******	q@gmail.com	1	yes	Deactivate
aa	aa	******	a@a	1	yes	Deactivate

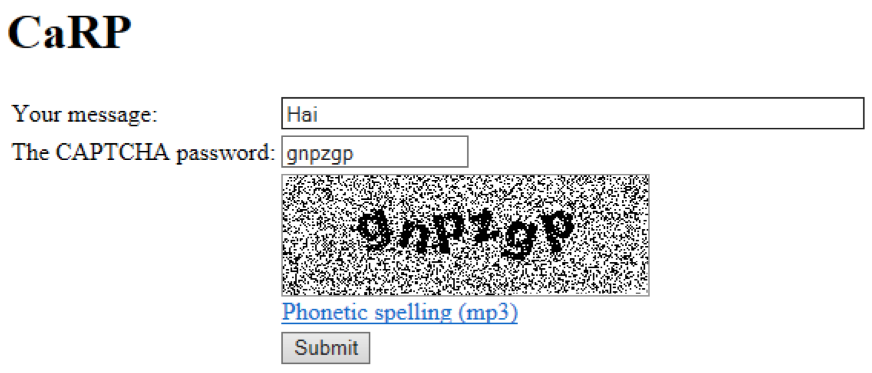
Figure 44. User Activities



UPLOADS

FILENAME	OWNER	UPLOAD TIME
aa.java	nadanapathy	2014/10/21 11:29:17
aa.java	nadanapathy	2014/10/21 10:29:17
Main.java	nadanapathy	2014/10/20 10:29:17
ms_access_java.txt	nadanapathy	2014/11/04 16:06:18
tables.rtf	asdf	2017/10/20 07:55:15
Praveen Kumar.docx	abc	2017/10/22 16:34:59
Untitled clipping.textClipping	q	2017/10/26 23:01:59

Figure 45. Upload Activities


CaRP

Your message: Hai

The CAPTCHA password: gnpzgp

Figure 46. CaRP Image

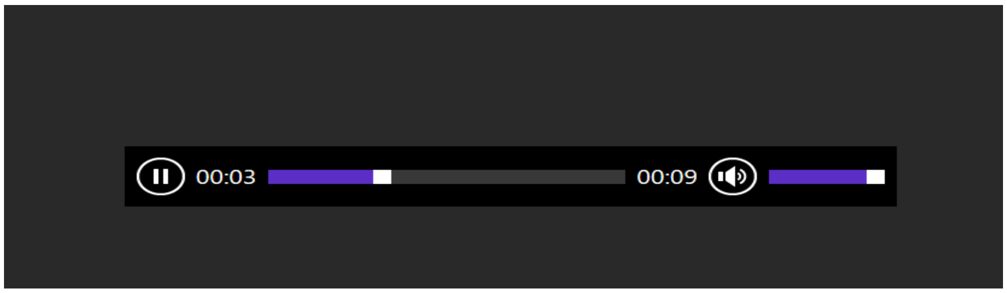
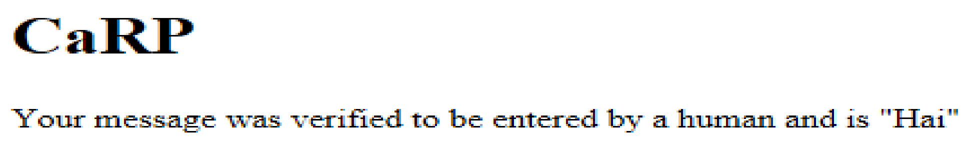


Figure 47. CaRP Audio


CaRP

Your message was verified to be entered by a human and is "Hai"

Figure 48. CaRP Successful Submission

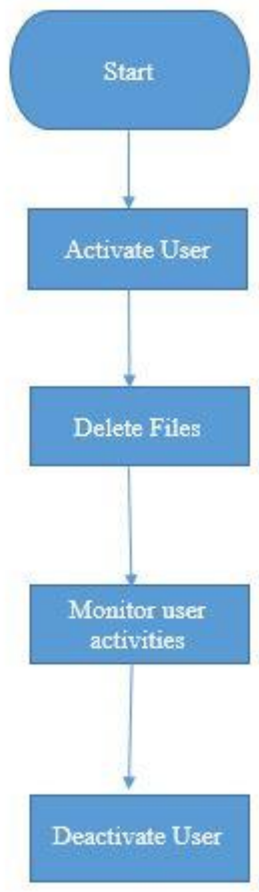


Figure 49. Data Flow

Chapter VIII: Conclusion

The purpose of this paper is to provide maximum security to ensure the protection of sensitive information. CaRP is a combination of graphical and text passwords and provides security towards costly human-based attacks. Most people have leaned towards ClickText and Animal Grid rather than PassPoints or a mix of content and graphical passwords because they are easier to use and memorize the password. The ease of CaRP is by utilizing images of various levels of complexity based on login history and the machine used (and how frequently) it can be easier to remember. With the combination of both a password and Captcha, it helps to secure and solve the hard AI problems. ClickText provides a greater amount of security from hackers than Captcha. As this generates a new challenge dynamically, each time it is used it becomes more difficult for hackers to compromise. This then provides a new challenge to Artificial Intelligence. ClickText provides security against online guessing and human guessing attacks. As one challenge is broken a new and more difficult challenge is generated.

In addition to CaRP, SHA-1 not only adds more security to passwords but also checks for data integrity and checks against data corruption. SHA-1 counter attacks online guessing attacks which include brute force attacks. As many graphical passwords are vulnerable to online guessing attacks, with this approach hotspots on CaRP images are no longer an issue. If CaRP is combined with dual-view technology, it can then counter relay attacks and shoulder-surfing attacks. The higher the correlation of user-clicked points between different login attempts is, the less effective the protection the dual-view technology would provide to thwart shoulder- surfing attacks. CaRP can also help to counter spam emails. There is also potential future work which needs to be done on CaRP for further refinements.

References

Alsaleh, M., Mannan, M. & van Oorschot P. C. M. (2012). Revisiting defenses against large-scale online password guessing attacks. *IEEE Transactions on Dependable and Secure Computing*, *9*(1), 128-141.

Biddle, R., Chiasson, S., & van Oorschot, P. C. (2012). Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR)*, *44*(4), 2012.

Bonneau, J. (2012). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pp. 538-552.

Buvanesvari, R., & Prasath, V. (2013). A new security mechanism for graphical password authentication using combo captcha in video technology. *International Journal of Science and Research, 4*(1).

Chellapilla, K., Larson, K., Simard, P., & Czerwinski, M. (2005). Designing human-friendly human interaction proofs. In *Proceedings of the SIGCHI Conference on Human Factors in Computer Systems*, pp. 711-720.

Chiasson, S., Forget, A., Biddle, R., and van Oorschot, P. C. (2008). Influencing users towards better passwords: Persuasive cued click-points. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers*, pp. 131-130.

Chiasson, S., Srinivasan, J. Biddle, R., & van Oorschot, P. C. (2008). Centered discretization with application to graphical passwords. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*. Retrieved from https://www.usenix.org/legacy/event/upsec/tech/full_papers/chiasson/chiasson_html/index.html

Chiasson, S., van Oorschot, P. C., & Biddle, R. (2007). Graphical password authentication using cued click points. In *Proceedings of 2007 12th ESORICS Conference on Research in Computer Security*, pp. 359-374.

Dailey, M., & Namprempre, C. (2004). *A text graphics character CAPTCHA for password authentication*. Retrieved from http://ieeexplore.ieee.org/abstract/document/1414527/

Data Encryption Standard. (n.d.). Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Data_Encryption_Standar

Davis, D., Monrose, F., & Reiter, M. (2004). On user choice in graphical password schemes. In *Proceedings of USENIX Security Symposium*.

Dhamija, R., & Perrig, A. (2003). Déjà Vu: A user study using images for authentication. In *Proceedings of the 9th USENIS Security Symposium*.

Dierks, T., & Allen, C. (2003). *The TLS protocol, Version 1.0*. Retrieved from http://www.ietf.org/rfc/rfc2246.txt

Diffie-Hellman key exchange. (n.d.). *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Dunphy, P., & Yan, J. (2007). Do background images improve “draw a secret” graphical passwords. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 36-47.

Elson, J., Douceur, J. R., Howell, J., & Saul, J. (2007). Assirra: A CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 366-374.

Gao, H., Liu, X., Wang, S., & Dai, R. (2009). A new graphical password scheme against spyware by using CAPTCHA. In *Proceedings of the 5th Symposium on Usable Privacy and Security*.

Ghorpade, J., Mukane, S. Patil, D., Poal, C., Prasad, R. (2014). Novel method for graphical passwords using CAPTCHA. *International Journal of Soft Computing and Engineering, 4*(5), 2231-2307.

Gyorffy, J. C., Tappenden, A. F., & Miller, J. (2011). Token-based graphical password authentication. *International Journal of Information Security, 10*(6), p. 321.

HP TippingPoint DVLabs. (2010). *Top cyber security risks report*. Retrieved from http://dvlabs.tippingpoint.com/toprisks2010

Jermyn, I., Mayer, A., Monrose, F., Reiter, M., & Rubin, A. (1999). The design and analysis of graphical passwords. In *Proceedings of the 8th USENIX Security Symposium, 1999* (pp. 1-15).

Jo, J., Kim, Y., & Lee. S. (2014). *Mindmetrics: Identifying users without their login IDs.* Paper presented at IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA.

Kim, S., Cao, X., Zhang, H., & Tan, D. (2012). Enabling concurrent dual views on common LCD screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2175-2184.

Li, S., Shah, S. A. H., Khan, M. A. U., Khayam, S. A., Sadeghi, A-R., & Schmitz, R. (2010). Breaking e-banking CPTCHAs. In *Proceedings of ACSAC, 2010* (pp. 1-10).

Lin, R., Huang, S.-&., Bell, G. B., & Lee, Y.-K. (2011). A new CAPTCHA interface design for mobile devices. In *Proceedings of the AUIC 12th Australasian User Interface Conference*, pp. 3-8.

Modern Cryptography. (n.d.). *Khanacademy*. Retrieved from https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/diffie-hellman-key-exchange-part-2

Motoyama, M., Levchenko, K., Kanich, C., McCoy, D., Voelker, G. M., & Savage, S. (2012). Re: CAPTCHAs—Understanding CAPTCHA-solving services in an economic context. In *Proceedings of 2010 USENIX Security Symposium*.

Murugavalli, S., Jainulabudeen, S., Senthil Kumar, G., and Anuradha, D. (2016). Enhancing security against AI problems in user authentication. International Journal of Advanced Computer Research, 6(24).

Pinkas, B., & Sander. T. (2002). Securing passwords against dictionary attacks. In *Proceedings of ACM Conference on Computer and Communications Security, 2002* (pp. 161-170).

Rashmi, B. J., & Maheshwarappa, B. (2003). Improved Security Using Captcha as Graphical Password. *International Journal of Advanced Research in Computer and Communication Engineering, 4*(5), 352-354

Real User Corporation. (2012). *The science behind passfaces*. Retrieved from http://www.realuser.com/published/ScienceBehindPassfaces.pdf

Rittenhouse, R. G., Chaudry, J. A., & Lee, M. (2013). Security in graphical authentication. *International Journal of Security and Its Applications*, *7*(3), p. 347.

Tao, H., & Adams, C. (2008). Go: A proposal to improve the usability of graphical passwords. *International Journal of Network Security*, *7*(20), 273.

Thomas, S. (2000). *SSL and TLS essentials: Securing the web*. New York, NY: John Wiley & Sons.

Transport layer security. (n.d.). *Wikipedia*. Retrieved from http://en.wikipedia.org/wiki/Transport_Layer_Security

van Oorschot, P. C., & Stubblebine, S. (2006). On countering online dictionary attacks with login histories and humans-in-the-loop. *ACM Transactions on Information and System Security*, *9*(3), 235-258.

Von Ahm, L., Blum, M., Hopper, N. J., & Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In *Proceedings of EUROCRYPT 2003,* pp. 294-311.

Weinshall, D. (2006). Cognitive authentication schemes safe against spyware. In *Proceedings of 2006 IEEE Symposium on Security Privacy*, pp. 295-300.

Wiedenbeck, S., Waters, J., Birget, J. C., Brodskiy, A., & Memon, N. (2005). PassPoints: Design and longitudinal evaluation of graphical password system. *International Journal of Human-Computer Interaction*, *63*, 102-107.

Wolfram MathWorld. (n.d.). Diffie-Hellman protocol. Retrieved from http://mathworld.wolfram.com/Diffie-HellmanProtocol.html

Wolverton, T. (2002). *Hackers attack eBay accounts* [Online]. Retrieved from http://www.zdnet.co.uk/news/networking/2002/03/26/hackers-attack-ebay-accounts-2107350/

Yan, J., Dunphy, P. (2007). Do background images improve "draw a secret" graphical passwords.

Zhu, B. B., Yan, J., Bao, G., Yang, M., & Xu, N. (2014). Captcha as graphical passwords—A new security primitive based on hard AI problems. *IEEE Transactions on Information Forensics and Security, 9*(6), 891-904.

Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., . . . Cai, K. (2010). Attacks and design of image recognition CAPTCHAs. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 187-200.