GRAPHICAL PASSWORDS USING ZOOMING

By

SHIVANI PATEL

A thesis submitted to the

Graduate School - Camden

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Scientific Computing

Written under the guidance of

Dr. Jean-Camille Birget

and approved by

Dr. Jean-Camille Birget

Dr. Sunil Shende

Dr. Suneeta Ramaswami

Camden, New Jersey

May 2019

THESIS ABSTRACT

Graphical Passwords using zooming

By SHIVANI PATEL

Thesis Director:

Dr. Jean-Camille Birget

User authentication is a process of authenticating a person to enter the system, and with the growing need for digitizing the data, user authentication plays a vital role in any system. Traditional authentication techniques such as alphanumeric passwords can have many drawbacks since they are easy to crack and are not easy to remember. Many earlier graphical password authentication techniques are also not very easy to remember since it involves memorizing many click points. We are proposing a new technique that uses stored images or fractals as graphical password, where the user has to remember a click point at a particular zoom level. This creates a story for the user as the user zooms in or zooms out to reach the password point which ultimately makes it easy for the user to remember the password. We have also implemented a demo system for fractal as well as stored images for the demonstration purposes.

Contents

List of Figures

Chapter 1

Introduction

Data security and privacy are one of the major concerns in the modern world. For keeping the data secure, it is very important to allow only the authenticated users to access the system. The most common or traditional way to authenticate a user is by using alphanumeric passwords. Choosing an alphanumeric password that is easy to remember and at the same time hard enough for anyone else to guess is quite hard. So, users normally tend to choose passwords that are easy to remember [4] or in some cases they may choose to write their password, which might be accessible to an unauthorized user. User also tends to use the same password for different accounts [4] in order to avoid remembering many passwords which can be easy to break into. These practices can become a reason for major security attacks such as brute force attack, shoulder surfing attack, dictionary attack, etc can use these weaknesses for attacking the system.

Graphical password authentication system (Blonder [5]) is a type of authentication system that uses GUI (Graphical User Interface) for authenticating the user that is different from the traditional alphanumeric passwords which are hard to remember. Humans have a memory that can easily remember visual representation i.e. pictures can be remembered easily as compared to text.

Graphical passwords is an authentication system that uses GUI (Graphical User Interface) and provide a great alternative to traditional alphanumeric passwords.

People tend to remember an image better than a text or alphanumeric password. So it is easy to remember a graphical password than to memorize a text password as humans tend to remember pictures better than text and at the same time they would not have to write the password somewhere in order to remember the password. Also, it is difficult to guess a graphical password as the sample space for graphical passwords is really wide.

Various graphical password schemes have been proposed for possible alternative text-based schemes, inspired by the psychological fact that humans tend to remember pictures better than text. Also, the possible password space increases or it may exceed the range of text-based passwords if the number of images increases or we could say its somewhere near infinite which reduces the possibility of brute force attack but the possibility of shoulder surfing exists.

In this paper, we are discussing existing graphical password techniques and using zoom in functionality and the use of fractals as graphical passwords. Using fractals with zoom in functionality increases the space of passwords and also reduces the possibility of shoulder surfing problem improving both security and usability.

Chapter 2

Authentication techniques

There are different types of passwords widely used for user authentication. Following are some password authentication schemes based on password entity:

- Token based authentication

  Token-based authentication is some physical object which acts as an authentication proof. The person holding it is considered as the authenticated user or a legitimate user and is allowed to proceed. Bank/ ATM cards are the best example of this type of authentication.

- Biometric based authentication

  Biometric authentication is also based on a physical object but here, the object can be considered as some part of the user's body which can include user's fingerprint, voice, eyelids etc. The advantage of bio metric system is that user is not required to physically carry an object and also it doesn't require the user to make any efforts to carry or store the object in a safe place.

- Knowledge based authentication

  In this type, users are expected to remember some information which they supply at the time of authentication. Only if it matches with the stored information, entry to the system is permitted otherwise it is declined. In this case, the user is expected to remember all the details.

- Behavioural-biometric authentication

  Behavioral biometric authentication focuses on analyzing the behavior of users and using that as a mode of authentication. The user is expected to repeat the same actions, in order to be authenticated. Some of the examples of these type of systems are signature authentication and keystroke dynamics.

- Location based authentication

  The location of the user can be tracked using various GPRS systems and based on that location, the user can be authenticated if the user is present at the expected location at that point of time.

- Hybrid authentication

  Hybrid authentication is based on the fact that combining multiple authentication systems in a proper way can help reduce the potential attacks on the system. Hence, multiple systems are combined into one in order to make it resilient and less prone to attacks.

2.1 Types of graphical passwords

2.1.1 Recall based graphical password system

In this system, the user is expected to recall the password set by the user during the registration phase. This system is further divided into two categories:

2.1.2 Pure recall based technique

In this technique, the user is expected to reproduce the password without any hints provided by the system. For example DAS, Grid Selection, Passdoodle, etc. We will also discuss the fractal-based password system which falls under this category.

- Draw A Secret (DAS)

  In this method, the user is expected to draw the stroke defined by the user earlier. The interface consists of a grid and the stroke should be a sequence of each cell, separated by a pen up event. The stroke is defined by a sequence of coordinates of each cell and the sequence is saved in the following way

  (2,2),(3,2),(3,3),(2,3),(2,2),(2,1),(5,6)

  The user is said to be authenticated if the user is successful in reproducing the stroke [10]. The drawback of this method is that according to the survey conducted by Goldberg, most of the users forgot their stroke order and also most of the users chose a vulnerable password that is prone to a graphical dictionary attack [8].

- Passdoodle

  Passdoodle method consists of handwritten designs or text, usually drawn on a touch-sensitive screen [22]. It has a wide range of passwords than text-based passwords. The system cannot merely authenticate a user without setting a threshold of likeliness or similarity which prevents authenticating a random user. Though the passwords are easy to remember and the method seems fascinating, the drawback of this method is that it requires a lot of computation and a lot of training samples to authenticate the user and hence the speed and accuracy [16] always remains a priority.

- Syukri Algorithm

  The Syukri algorithm is an algorithm that focuses on achieving authentication by the user using a mouse to draw their signature [20]. The user is supposed to draw the signature during the registration and when the user tries to log in, the input signature is verified with the signature drawn during the registration phase (Note : The signatures are stored and verified in normalized form) [14].

- Grid Selection

  Thorpe and Van Oorschot [21] studied that draw a secret password space depends a lot on the stroke count. The password range decreases for fixed length passwords and few strokes. According to Thorpe and Oorschot [21], the security can be increased by using a larger selection grid which significantly increases the range of passwords.

  The user is expected to select an N x N drawing grid from a very large grid and then the user can zoom in to that area as per the traditional DAS.

2.1.3 Cued recall based technique

Using the cued recall based technique, the user is expected to recall the password selected by the user during the registration phase by getting a hint from the system. Let us go through some techniques based on this:

- Blonder

  Blonder method was named after Greg E. Blonder [5], who introduced this method in 1996. In order to be authenticated using this method, the user is expected to click on the points in the same order the user clicked earlier during the registration phase [14]. This method is immune to brute force attack since it has millions of regions that can be set as password i.e. it has a large sample space. This method seems secure but at the same time, in order to have a secure password using this system, the password has to be long enough which can be hard to remember as the length increases. Also the images have to be simple, artificial or predefined and any real life image could not be used [24].

- PassPoint

  Passpoint [25] method has a chosen image that can be any natural image or painting that has high resolution so that there can be a large range of click points that can be set as a password. During the authentication phase, the user is expected to click on the click points selected during the registration phase in the same order as chosen during the registration phase. Each click point has a defined boundary and if the user clicks within the defined boundary of that click point in the same sequence, then the user is successfully logged into the system.

2.1.4 Recognition based graphical password system

Recognition based graphical password techniques are based on the user selecting pictures, icons or symbols from a large set which usually stored in the database or generated using a program. The user is expected to remember their selection and recognize their choice in sequence during the authentication phase in order to log in successfully. Studies show that ninety percent of the people didn't face difficulty using this technique but the drawbacks related to longer log-in times as compared to the traditional alphanumeric passwords makes this technique less feasible for regular use.

Some examples of these systems are as follows:

- To overcome the storage problems, a new scheme was proposed by Dhamija and Perrig [7] in which random program generated images are displayed to the user. The user is expected to select images from those images and if the user successfully selects all the images selected earlier, then the user is authenticated.

- Several recognition techniques such as picture recognition, object recognition, and pseudoword recognition were introduced by Weinshell and Kirkpatrick [23] in which there is a large database of images containing around 20 thousand images. From this large set of images, the user is expected to recognize around 100-200 images to be authenticated. Experiments show that the users were able to remember almost 90 percent of the images in a time span of one to three months. Since the time span of one to three months is very long to learn a password, this method is less feasible for practical applications.

- A new technique was introduced by Sobrado and Birget [18] that is shoulder-surfing resistant by suggesting to use a display of 1000 objects and in which the pass-objects that were selected by the user earlier are in the form of the convex hull. For authentication, the user is expected to click within the convex hull formed by the pass-objects. The drawback of using this technique is that the display is very crowded and to avoid that problem if fewer objects are used, then the password range will reduce gradually and an attacker might be able to log in just by clicking randomly or by rotating. This technique is also extremely slow and hard to remember which leads to a reduction in the use of this technique.

- A shoulder-surfing resistant technique proposed by Man et al [12] has various pass objects and a unique code assigned to each object. To create a password using this technique, the user has to select pass objects and at the time of authentication, each pass has multiple pass objects selected by user and the rest are random objects, to select the correct object, the user is expected to write the code corresponding to the pass object and its relative location. This technique is shoulder surfing resistant since, even if the video is made, the password is not known as no mouse clicks are involved. The only drawback this method has is remembering the code for each variant which is hard to memorize.

- To create a password using Passface Technique (Real User Corporation [17]), the user is supposed to choose four images from the database where the database consists of images of human faces. Once the user chooses four human faces than during the authentication phase, a grid consisting of one face chosen by the user during password creation and some random faces from the database are presented to the user. This grid is presented to the user several times and if the user is successfully able to identify the correct human face every time, then the user is logged into the system. An assumption that people can remember faces easily led to the successful execution of this technique that has two-thirds of the login success rate of alphanumeric passwords [19].

Chapter 3

Graphical passwords using zooming

In the earlier sections, we discussed various graphical password techniques and their advantages over alphanumeric passwords and at the same time highlighted the shortcomings of the graphical password techniques introduced earlier in order to propose a technique to overcome those shortcomings.

We are proposing a scheme in order to overcome the shortcomings discussed in prior methods. The proposed system is focused on introducing more information upon zooming the image that increases the password space size and the password can be selected only on the correct zoom level which in turn reduces the possibility of brute force attack and also reduces the effect of shoulder surfing since determining zoom level could be tricky.

In order to avoid various attacks on graphical passwords, as an enhancement of cued recall based graphical password scheme such as Blonder [5] (as discussed in Chapter 2), we are proposing a system based on the concept of real zooming i.e. new information is introduced on zooming in the image ensuring that the password can be selected only on zooming in right amount in the correct area. The password we are referring here can be any object or point which acts as password only at a particular zoom level.

Unlike selecting and remembering multiple points on a single image (like in blonder scheme), the proposed algorithm focuses on remembering how to get to the password i.e. the password is just a single click point in the image, which is not visible in the image presented at first. The user is expected to zoom in correctly and find the password or click point at a particular zoom level and click the correct point within the radius of $\approx 1.8mm$ in order to log in to the system.

3.1 Zooming

The most common definition of zooming is termed as magnification or enlargement of the image by interpolation or refinement of the image pixels [6]. Here, the term zooming is referred as enlarging or focusing a part of an image in such a way that it introduces new information upon zooming in, like the way it is in **GIS** (Geographic Information Systems). In our prototype, we will be storing image data locally and using the image path to retrieve the images when the user zooms in, in a way that new information is introduced to the user as the image is zoomed more and more.

We are proposing two systems and the zooming works differently in both the systems. For adding more clarity on how the zooming works, let us go through both the systems :

- **Zooming stored images :**

  The user can zoom into these images by using one of the following methods:

  - Use zoom control buttons on the screen i.e. there are (+) and (-) buttons on the screen that can be used to zoom in and zoom out the image. On clicking (+) button, the image will be zoomed 2^z times (where z represents the zoom level which is any real number and it can be either positive or negative). The image is zoomed considering the center of the previous image as the center of the zoomed image containing more information about the image. On click of (-) button, the image will be zoomed 2^{-z} times the image at that moment reducing the information about the image.

  - Double-clicking anywhere on the image will zoom the image considering the clicked point as the center of the zoomed image. The image only zooms in on double clicking and the zoomed image will be 2^z times the previous image containing more information about the details of the zoomed part.
  - Moving the cursor anywhere on the image and scrolling on the image will zoom in or zoom out the image just like using zoom control buttons but the image will be centered around the cursor position. Scrolling down will zoom in the image whereas scrolling up will zoom out the image.

- **Zooming fractal images :**

  There is only one way to zoom in to the fractal images and i.e. by clicking (single click) on the image. On clicking the fractal image, that click point will be considered as the center and the fractal will be recalculated and zoomed. The new or zoomed in the image will be 2^z times (where z represents the zoom level which is any real number and it can be either positive or negative.) the previous image centered around the cursor point. In order to zoom out the fractal, the user can use $CTRL + click$ and the fractal will be zoomed 2^{-z} times where z is the zoom level and it can be any real number, positive or negative.

  Note: The reason there is only one way to zoom into fractal images is because fractals are self-similar images and using zoom control buttons and scrolling provides less control on zooming, since on using zoom control buttons, the zooming is performed considering image center as the center and for choosing the right password the user should have more control on zooming as one wrong zoom point can confuse the user. Also, using scroll can sometimes zoom the image continuously and provide less control on zoom level. Since fractals are self-similar images and if there is no proper control on how the

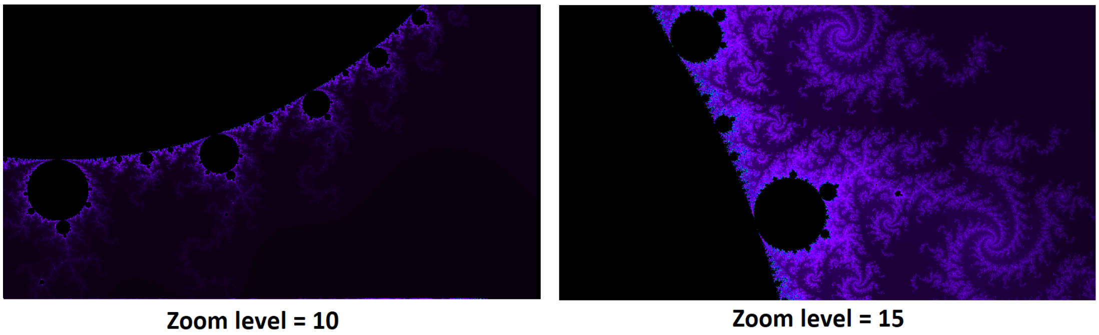

Figure 3.1: Fractal zoom stages

fractal is zoomed then, it will lead to loss of track on where the user has zoomed.

3.2 Password creation or change

Password creation refers to the algorithm for creating a password and storing it. We are proposing two different systems that use different types of images for creating a graphical password:

We will learn more about the generation and storage of these images in the further sections.

The password creation works similarly for both the systems except for the zooming of the images, so let's go through the password creation algorithm for both the systems.

While creating or changing the password, it is assumed that the user is already logged into the system and is in the password creation module of the respective password system and the following steps need to be performed to successfully create/ change the password:

- **SYSTEM** : The create/ change password module will contain a "Submit Password" button to submit the selected password, a "Clear" button that clears all the selection and displays the initial image, a "Go Back" button to go to the main page and an "Exit" button to exit the whole system.
- **USER** : User can use the various zoom control options for the respective system as mentioned above in section 3.1. (Note: The user is expected to zoom in at least four times before selecting the password.)
- **SYSTEM** : Ensure that the image is zoomed at least four times before selecting the password.
- **USER** : Once the user zooms into the image (at least four times) and reaches a point where the user wants to set the password. The user is expected to click on the point that the user wants to set as password and click on the button "Submit Password".

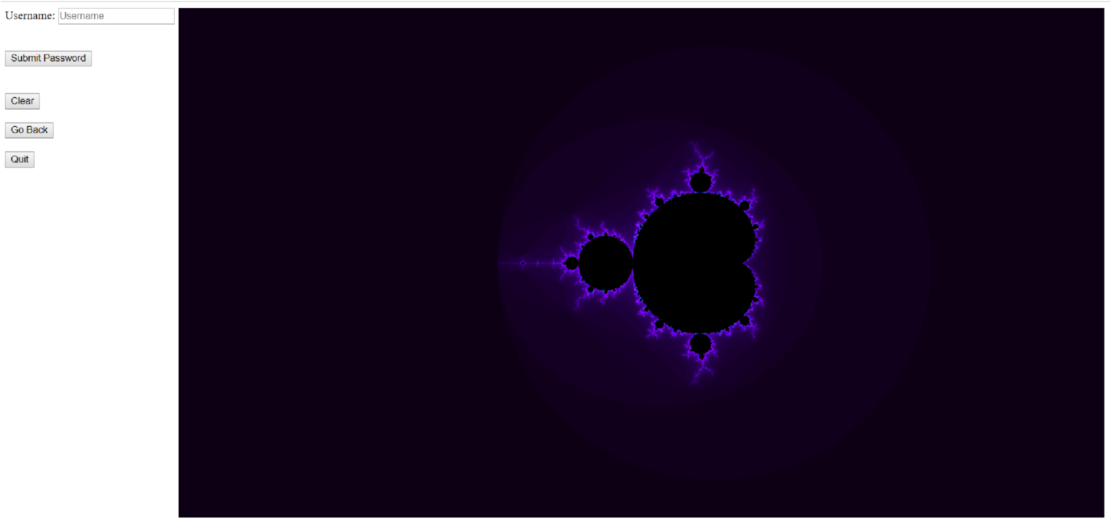

Figure 3.2: Fractal create/ change password module

- **SYSTEM** : Make sure all conditions are satisfied before creating or changing the password. In case the user fails to satisfy all the above conditions then the user will see an alert to make the necessary changes to satisfy all the conditions for password creation/ change.
- **USER** : Make necessary changes if system notifies, in order to satisfy all the above conditions.
- **SYSTEM** : Once the user satisfies the above conditions for creating/ changing the password and clicks on “Submit Password” button, the user will be notified that password creation/ change was successful. The selected/ updated username, zoom level and the coordinates of the selected password point will be stored locally as the password to the system.

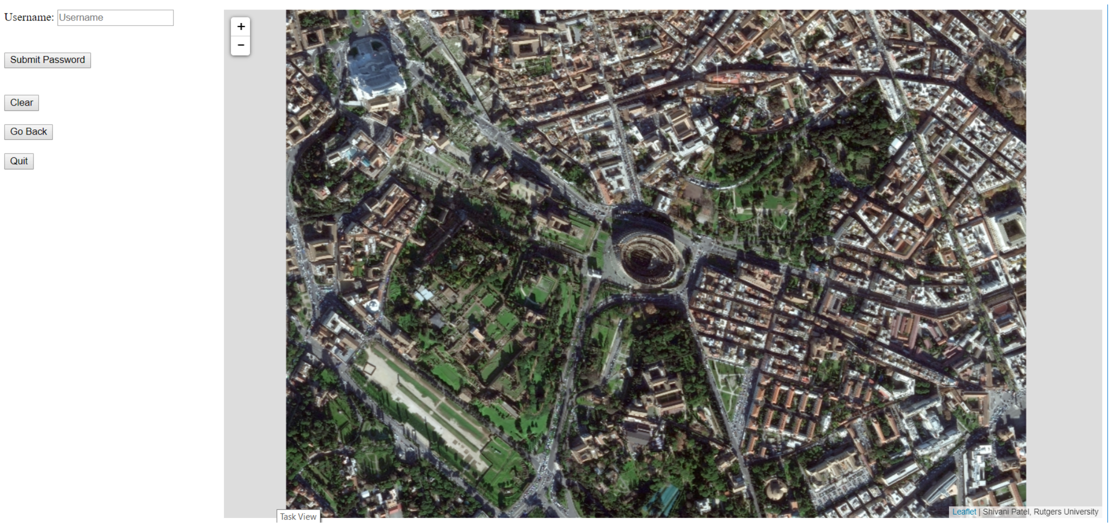

Figure 3.3: Stored images create/ change password module (Satellite image [2])

3.3 Log in to the system

The graphical password interface for login module of the proposed system consists of the zoomable image and buttons to zoom in or zoom out the image. The login module for both the systems are similar so to understand the login phase for both the systems, let us go through the login algorithm for both the systems.

During the login phase, it is assumed that the user has already created a password for the password system user is trying to log into and is already in the login module of the respective password system and the following steps need to be performed to successfully login:

- **USER** : The login module will contain a “Clear” button that clears all the selection and displays the initial image, a “Go Back” button to go back to the main page and an “Exit” button to exit from the whole system.
- **USER** : User can use the various zoom control options for the respective system as mentioned above in section 3.1.
- **USER** : User is expected to reach the password point selected during create/ change password phase and click on the point.
- **SYSTEM** : If the clicked point lies within a radius of 7px from the original click point then log the user into the system or if it doesn’t then nothing should happen and if the number of clicks exceeds a count of 20 then display “Login failed” message.

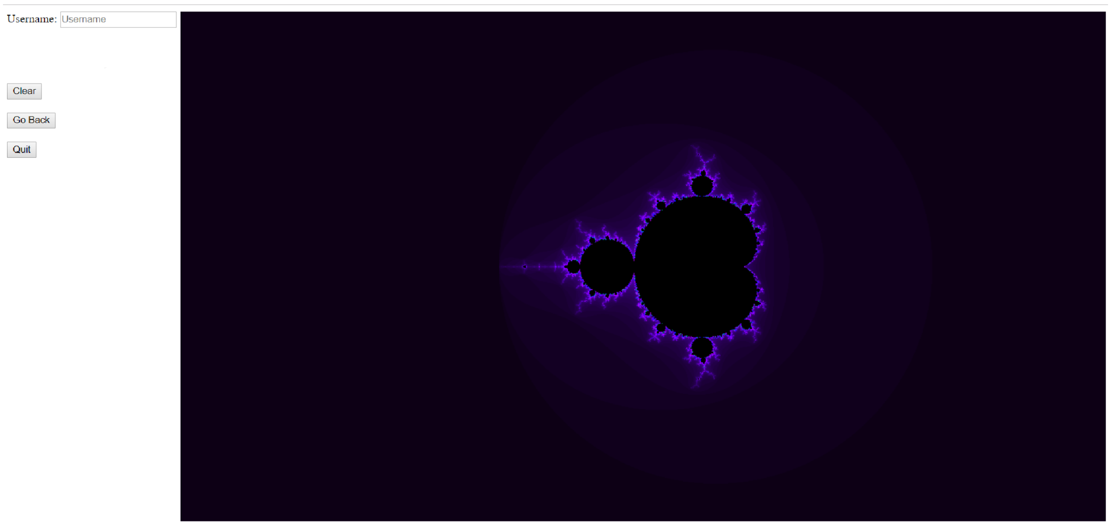

Figure 3.4: Fractal login module

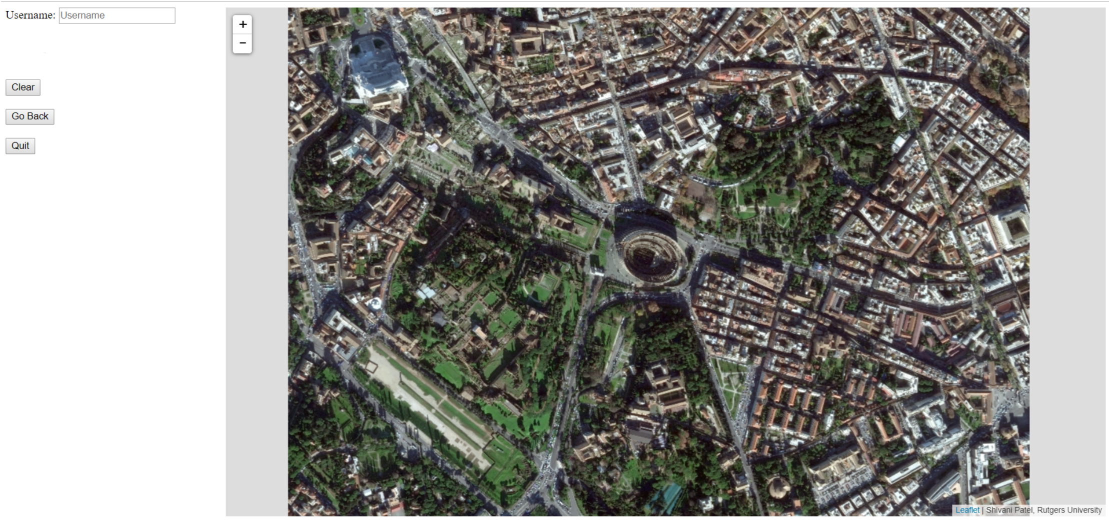

Figure 3.5: Stored images login module (Satellite image [2])

3.4 Zoomable images

In this project, we are using two types of images that the user can zoom into and set their password in order to log into the system.

3.4.1 Stored images with high resolution

Image resolution holds a direct relationship with the number of details image consists of, higher resolution images can hold more details as opposed to low-resolution images which can hold a limited number of details. We are using the concept of real zooming, which means that when the image is zoomed in, it will hold more details about the area that was zoomed.

Any image can be stored in two ways :

- Storing the images in a database using BLOB datatype. Storing large images in a database as BLOB by compressing the images can be effective when frequent database backup is required.
- Using this method, the actual image files are stored on the cloud or some other type of memory and to access the image, it is required that the image is accessed using the image path.

The proposed system requires that the images are high resolution and the details are visible to the user for more precise detection of click points. For the images to be high resolution, it is required that the images are not compressed and also it is not a very good practice to store very high-resolution images in the database, so the images will be stored in memory as files and the image path will be used to access the image.

All the image files that belong to a particular zoom level are grouped and stored in one folder and are accessed using the path of the image.

3.4.2 Fractal images (Mandelbrot set)

A fractal can be any shape that exhibits the property of self-similarity [11]. It can be an object in the real world like a fern leaf or a snowflake, or it can be a geometric shape computed on a complex plane such as Julia set, Mandelbrot set, etc. Fractal images are computed in real-time using an iterative equation or a set of equations. Here we are computing the Mandelbrot set which is calculated using a recursive equation and assigning a value to each pixel of the image [11]. Fractal images exhibit self-similar or self-repeating patterns, and after each iteration or after each zoom the image within that zoomed area is recalculated since the image is recalculated after each zoom.

The Mandelbrot set fractal exhibits the property of self-similarity and has a non-integer dimension, i.e. fractals are generated using complex equations that contains a real number and a complex number, which means that the fractal has dimensions that are not whole numbers [9]. To understand it better let us say, a line in a normal plane that is said to be one-dimension, but when that line is represented on a fractal curve, it is said to have a dimension between one and two because of the complex number or non-integer dimension present in fractals.

Any point (C) is chosen on a complex plane, and a complex number is represented in the form of C = a + bi where a is the horizontal axis, and b is the vertical axis which is imaginary since i is $\sqrt{-1}$ which is not a real number [9].

The Mandelbrot set is calculated by recursively using the below expression for all the pixels :

$$
\begin{gathered}
Z_0 = 0 \\
Z_n = Z_{(n-1)}^2 + C
\end{gathered}
$$

For the first iteration in the above example, we can obtain a result for C by replacing zero in place of Z_0 and assigning the result to Z_1 and the resulting complex number is assigned to Z_2 and iterating the same steps over and over for n number of times. So, the value of Z_0 will be 0, then the value of Z_1 will be C, and the value of Z_2 will be C^2+C and it keeps on iterating until escape condition is reached.

When the initially chosen point C is iterated repeatedly then the point either goes closer and closer to the origin or it goes away from the origin. If the point is going closer and closer to the center then it is considered to be in the Mandelbrot set and assigned a black color, on the other hand, if the point goes away from the origin then it is assigned a color depending on the speed at which the point “escapes” from the origin. If the point is far from the center of the Mandelbrot set, then it reaches the escape condition immediately, and if it is close to the center but not within the set then it will escape slowly on the other hand if it lies within the set, it will never escape [26].

3.5 Implementation details

We are proposing two different graphical password systems that use different types of zoomable images, one of them is stored and the other type is calculated in real time. We will discuss more related to its implementation and storage structure in further topics:

3.5.1 Zoomable images

3.5.1.1 Tiling structure

The basic idea behind tiling images is the regular recursive subdivision of images in equal-sized tiles. During each iteration, each tile is divided into four equal sized tiles and this is repeated until fixed number of iterations are performed.

Each tile is of standard size based on its original image and it is made sure that no two tiles overlap each other when considered from the top view of the image. Java TileUp library [15] was used to create these tiles that reads the image as input and scales the image according to its input size and tile size.

The image is read from the input folder and for each zoom level a separate subfolder is created and all the images within a particular zoom level are stored in that subfolder corresponding to that zoom level. So while fetching the images, all the images within a subfolder are fetched and displayed from the subfolder that belongs to that particular zoom level.

The generated map tiles will follow the naming convention according to the following:

- All the map tiles will follow a general naming convention of “map_tile_x_y.png” where map_tile is common across all the tiles and x and y are variables that represent the row and column of the tile respectively. The row and column number are determined by considering the image as a grid and the map tiles

> as the part of the grid where row count begins from the upper-left corner and the column count begins from left to right and increases accordingly. So, once the row ends the calculation starts from the next row. The same pattern is followed for each zoom level and hence the final representation of the storage structure is "zoom-level/map_tile_x_y.png" where zoom-level is the subfolder for that particular zoom level and x and y represent the row and column number for that zoom level respectively [13].

Once all the sub-folders for the tiles on different zoom levels are generated and each folder has the tiles of a particular zoom level, we use leaflet library [1] in JavaScript which is an open-source JavaScript library that helps to access these tiles and display them in map form. Leaflet is a open source library that is used to build web mapping applications. The leaflet library accesses the image tiles at each layer and creates tile layers that act as a map and in our case, it acts as a zoomable image which is used for generating a password. On zooming the image, the tiles from that particular zoom level are displayed in front of the user.

3.5.2 Calculating zoomable images

3.5.2.1 Calculating fractal image

We are calculating the Mandelbrot set by using the escape time algorithm [26] that uses iterative and repetitive calculation for each point or pixel (x, y) on the canvas plot. For each of the coordinates on the plot the initial value of the first iteration is the value of that coordinate and after each iteration, the value of the previous iteration is considered as the input value for the next iteration. These iterations are continued until some escape condition is reached. During each iteration, it is checked that if the values have reached the escape condition.

Now, let us understand what is escape condition, we are checking the following two escape conditions:

- The number of iterations are less than the maximum number of iterations set.
- If the sum of the squares of the real and imaginary parts exceeds 4, the point is said to have reached escape condition.

A fractal image is rendered on a complex plane that is considered to be divided into a certain number of pixels and a color is assigned to each pixel based on when the pixel reaches the escape condition. To understand how it works, let us consider a point c as the center that is iterated through 0 under P_c and while we iterate the point, during each iteration we check if the modulus is less than 2. If the modulus exceeds 2 then we know that the point is not inside the Mandelbrot set and based on the number of iterations required for the point to reach the escape condition, the pixel is assigned a color and if the point doesn't reach the escape condition before reaching the maximum number of iterations defined then the point is said to be within the Mandelbrot set or very close to the set and is assigned black color [9].

The pseudo code of the algorithm looks as follows:

```
For all the pixels (x, y)
{
  (x0, y0) : scaled x and y coordinate
  maxiterations = 1000
  iteration = 0
  while ((x*x + y*y <= 4) && iteration < maxiterations )
  {
        double xt = x*x - y*y + x0;
        (used for complex number representation)
        double yt = 2*x*y + y0;
        x = xt;
        y = yt;
        iteration++;
  }
  if (iterations == maxiterations)
  {
    color = {r : 0, g : 0; b : 0} (color the pixel black)
  }
  else
  {
    color = palette[iteration] (color the pixel based
    on the number of iterations required to to escape)
  }
}
```

For the values that belong to the Mandelbrot set [26], the escape will never occur and hence all those pixel values that never reach the escape condition are set to black color. All the other values that don’t belong to the set will reach the escape condition before the maximum number of iterations are performed. As the maximum number of iterations increase, the rendered image is more detailed but it is directly proportional to the time needed to render the final image. As the number of iterations increase, the time required to render the fractal image also increases.

The color palette is generated in such a way that it has brighter colors initially and it gets darker as the value increase. The color of a pixel is determined by the iteration number at which the pixel reaches the escape condition, so the pixels that reach escape condition earlier are assigned a brighter color and the pixels that fail to reach the escape condition within defined iteration limit are assigned black color. This is basically a representation of the number of iterations required for a pixel before it reached escape condition. We are using a color palette with a shade of blue and set the RGB offset of the colors so that when the escape condition occurs, that color is assigned to the pixel.

The pseudo code to generate the palette looks as below:

```
rpalette = 10;
gpalette = 0;
bpalette = 14;
for (i from 0 to 256)
{
  palette[i] = { r: rpalette, g: gpalette, b: bpalette };
  if (i < 64) {
    bpalette = bpalette + 7;
    rpalette = rpalette + 3;
  } else if (i < 128) {
    rpalette = rpalette - 5;
  } else if (i < 192) {
    bpalette = bpalette - 5;
    gpalette = gpalette + 3;
  }
  i = i + 1;
}
```

Now, let us see how to zoom the fractal. For zooming a fractal image, the zoom factor is set to 2 as we want the zoom factor to be 2^z where z is the zoom level, so every zoom factor is twice the previous zoom factor. We are using a variable *zoom* that is of type boolean and if *zoom* is true then the fractal coordinates are multiplied to zoom factor in order to zoom in and if the variable *zoom* is false then it is divided in order to zoom out the fractal. The scaled x and y coordinate are sent to iterate and generate the zoomed in or zoomed out fractal.

The pseudocode is described below :

```
// Zoom in
zoom *= 2;
dx = 2 * (x + (-width/2) + dx);
dy = 2 * (y + (-height/2) + dy);

// Zoom out
zoom /= 2;
dx = (x + (-width/2) + dx) / 2;
dy = (y + (-height/2) + dy) / 2;
```

3.6 Storing graphical passwords

In the prototype implementation of both the systems which are just a demo single user system, we are storing the username, the x and y coordinates and the zoom level of the password using window localstorage which allows storing key/value pairs in a web browser. The values update when the user changes the password and are not deleted upon closing the browser and are available without any expiration date.

As an enhancement in order to store the passwords securely, we are proposing a technique to store the graphical passwords securely in hashed form. Since the proposed systems stores the coordinates of the password and the coordinates to log into the system cannot be the same every time the user tries to log into the system as we cannot expect the user to click on the same point every time. The proposed technique deals with overcoming this problem by storing the password using the steps described below :

- While creating the password, the x and y coordinates and the zoom level at that point are appended and hashed to store as the password, i.e. (x, y, z) where x is the x coordinate, y is the y coordinate and z denotes the zoom level at that point. So, H(x, y, z) is stored as the password to log into the system.
- When the user tries to log into the system, the user is expected to click within a radius of 7px (approximately 1.8mm) in order to log in to the system. So, when a user clicks, all the combinations of H(x, y, z) within the radius of 7px (approximately 1.8mm) are compared with the stored hashed value.

  If the value matches then the user is logged into the system otherwise the user is expected to try again.

3.7 Security analysis

The proposed system has an original image of approximately 4000 pixels that can be zoomed in 20 times, so the password space comes to :

$$4000 \times 2^{20} \approx 2^{32}$$

Since the proposed technique uses only one click point at a particular zoom level hence the password space is 2^{32} which is 4 bytes and that is vulnerable to brute force attack. In order to make the password stronger, the length of the password can be increased by using one of the following techniques :

- Increasing the number of click points for the password i.e. increase the password length.
- Increasing the zoom factor from two times to four times or eight times which in turn will increase the space to $4^{20} = 2^{40}$ or $8^{20} = 2^{60}$ which is large enough to resist brute force attack.

Chapter 4

Conclusions and future work

In this paper, we have introduced graphical password techniques that use fractal images or high-resolution zoomable images along with other pass phases like zooming and image click points to login to the system. The main objective of introducing these techniques was to allow the user to remember the password easily by creating a story as to how to reach the password and at the same time, the login time is acceptable. The demo systems are not secure since the stored password is not hashed, so we aim to add security in the password system by storing it in hashed form.

Bibliography

[1] Leaflet javascript library documentation. https://leafletjs.com/index.html.

[2] Worldview-2. http://www.effigis.com/wp-content/uploads/2015/02/DigitalGlobe_WorldView2_50cm_8bit_Pansharpened_RGB_DRA_Rome_Italy_2009DEC10_8bits_sub_r_1.jpg.

[3] Vladimir Agafonkin 2010-2018, CloudMade 2010-2011, and now openSource. Leaflet javascript library. https://github.com/Leaflet/Leaflet. commit 61b818fcd456794926fa1b7cc89b5241d7c865a1.

[4] A. Adams and M.A. Sasse. Users are not the enemy. *Communications of the ACM (CACM) - Vol 42*, pages 40–46, 1999.

[5] Greg E. Blonder. Graphical password. *U.S. Patent No. 5559961*, 1996.

[6] S. Chadda, N. Kaur, and R. Thakur. Zooming techniques for digital images : A survey. *International journal of computer science and technology - Vol 3*, pages 519–523, April 2012.

[7] R. Dhamija and A. Perrig. Deja vu: A user study using images for authentication. *Proceedings of the 9th conference on USENIX Security Symposium - Vol 9*, pages 45–48, 2000.

[8] Paul Dunphy and Jeff Yan. Do background images improve draw a secret graphical passwords? *Proceedings of the 14th ACM conference on Computer and communications security. Alexandria, Virginia, USA. ACM.*, pages 36–47, 2007.

[9] Bastian Fredriksson. An introduction to the Mandelbrot set. https://www.kth.se/social/files/5504b42ff276543e4aa5f5a1/An_introduction_to_the_Mandelbrot_Set.pdf, January 2015.

[10] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin. The design and analysis of graphical passwords. *Proceedings of the Eighth USENIX Security Symposium. August 23-26 1999. USENIX Association*, pages 1–14, 1999.

[11] R. Jovanovic, M. Tuba, and D. Simian. A New Visualization Algorithm for the Mandelbrot Set. *Proceedings of the 10th WSEAS International Conference on mathematics and computers in biology and chemistry*, pages 162–166, 2009.

[12] S. Man, D. Hong, and M. Mathews. A shoulder-surfing resistant graphical password scheme. *Proceedings of the International Conference on Security and Management. Las Vegas, NV*, pages 105–111, 2003.

[13] Oliver Marriot. Using leaflet.js with non-geographic imagery. "http://omarriott.com/aux/leaflet-js-non-geographical-imagery/".

[14] M. Masrom, F. Towhidi, and A. H. Lashkari. Pure and cued recall-based graphical user authentication. *2009 International Conference on Application of Information and Communication Technologies*, pages 1–6, Oct 2009.

[15] metraTec GmbH. Java tileup. https://github.com/metratec/javatileup, 2016. commit 5f96ba5e2f070d92c6483fdd85f3065e9d413cbf.

[16] Karen R. On user involvement in production of images used in visual authentication. *Visual Languages and Computing - Vol 20*, pages 1–15, 2009.

[17] RealUser. Passfaces: Two factor authentication for the enterprise. http://www.realuser.com/index.htm.

[18] L. Sobrado and J. C. Birget. Graphical passwords. *The Rutgers Scholar, An Electronic Bulletin for Undergraduate Research - Vol 4*, 2002.

[19] X. Suo, Y. Zhu, and G. S. Owen. Graphical passwords : A survey. *21st Annual Computer Security Applications Conference (ACSAC), Tucson, AZ*, pages 463–472, 2005.

[20] A. F. Syukri, E. Okamoto, and M. Mambo. A user identification system using signature written with mouse. *Third Australasian Conference on Information Security and Privacy (ACISP): Springer-Verlag Lecture Notes in Computer Science (1438)*, pages 403–441, 1998.

[21] J. Thorpe and P. C. v. Oorschot. Towards secure design choices for implementing graphical passwords. *20th Annual Computer Security Applications Conference, Tucson, Arizona*, pages 50–60, 2004.

[22] Christopher Varenhorst, Max Van Kleek, and Larry Rudolph. Passdoodles: a lightweight authentication method. *Massachusetts Institute of Technology, Research Science Institute*, July 2004.

[23] D. Weinshall and S. Kirkpatrick. Passwords you'll never forget, but can't recall. *Conference on Human Factors in Computing Systems (CHI). Vienna, Austria: ACM*, pages 1399–1402, 2004.

[24] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Authentication using graphical passwords: effects of tolerance and image choice. *Proceedings of the Symposium On Usable Privacy and Security (SOUPS'05)*, pages 1–12, 2005.

[25] Susan Wiedenbeck, Jim Waters, Jean-Camille Birget, Alex Brodskiy, and Nasir Memon. Passpoints: Design and longitudinal evaluation of a graphical password system. *International Journal of Human-Computer Studies - Vol 63*, pages 102–127, 2005.

[26] Wikipedia contributors. Mandelbrot set — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Mandelbrot_set&oldid=894081641.