
# Authentication Using Image and Steganography

TWO FACTOR AUTHENTICATION USING IMAGE AND STEGANOGRAPHY
| Graduation Project | 2024

## GRADUATION PROJECT

Cyber Security Department

Faculty of Information Technology and Computer Science

Yarmouk University

Irbid, Jordan


*Bachelor's degree Project*

*IT (Information Technology) Department / Cybersecurity program*

*Project ID:* ***YUIT-CYS-2024***

# Authentication Using Image and Steganography

Department of Information Technology/ Cybersecurity program

Faculty of Information Technology and Computer Sciences

Yarmouk University, Jordan

## *Contact Information*

*This project report is submitted to the Department of Information Technology/ Cybersecurity program at Yarmouk University in partial fulfillment of the requirements for the degree of Bachelor of Information Technology in Cybersecurity.*

***Author(s):***

*Aibak Awni Aljadayah*
*2019904038*
*Address: Irbid*
*E-mail: 2019904038@ses.yu.edu.jo*

*Khaled Mahmoud Hanandeh*
*2019804051*
*Address: Irbid*
*E-mail: 2019804051@ses.yu.edu.jo*

*Laith Khaled Sameh Yassin*
*2019804025*
*Address: Irbid*
*E-mail: 2019804025@ses.yu.edu.jo*

*Osama Mohammad Tafesh Alserhan*
*2019904027*
*Address: Mafraq*
*E-mail: 2019904027@ses.yu.edu.jo*

***University-supervisor(s):***
***SHATHA SHAKHATREH***
*Department Information Technology/ Cybersecurity program*

*Department of Information Technology* *Internet: http://yu.edu.jo*
*Faculty of Information Technology* *Phone: +962 2 72711111 Ext. 2634*
*and Computer Science Yarmouk University* *Fax: +962 2 7211111* *Jordan*

# Intellectual Property Right Declaration

This is to declare that the work under the supervision of Dr. Shatha Shakhatreh
Having title “Two Factor Authentication Using Image Steganography” carried out in partial fulfillment of the
requirements of Bachelor of Information Technology in Cybersecurity, is the sole property of the Yarmouk University and the respective supervisor and is protected under the intellectual property right laws and conventions. it can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, Etc., with the permission of the University and respective supervisor.

Date: 2024/01/

***Authors():***

*Name: Aibak Awni Aljadayah* *Signature*_____

*Name: Khaled Mahmoud Hanandeh* *Signature*_____

*Name: Laith Khaled Sameh Yassin* *Signature*_____

*Name: Osama Mohammad Tafesh Alserhan* *Signature*_____

***Supervisor(s):***

*Dr. Shatha Shakhatreh*

*Signature:* _____

# Anti-Plagiarism Declaration

This is to declare that the above publication produced under the supervision of Dr. Shatha Shakhatreh,

Having title “Two Factor Authentication Using Image Steganography” is the sole contribution of the Author(S) and no part here of has been reproduced illegally (Cut and Paste) which can be considered as plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: 2024/01/

**Author(s):**

Name: Aibak Awni Aljadayah Signature_____

Name: Khaled Mahmoud Hanandeh Signature_____

Name: Osama Mohammad Tafesh Alserhan Signature_____

Name: Laith Khaled Sameh Yassin Signature_____

# ABSTRACT

This project aims to develop a secure two-factor authentication system by integrating traditional methods with image steganography. Users provide images that undergo cryptographic transformations to embed secret codes, creating an encoded image used for authentication. This approach enhances security and offers a personalized user authentication experience.

By amalgamating steganography with conventional authentication methods, the project significantly bolsters the security of user authentication. Users are afforded the opportunity to upload a unique image of their choice, thereby personalizing their authentication process. This fusion of techniques not only enhances security but also provides a more user-centric approach to authentication.

**Keywords:** Two-Factor Authentication, Image Steganography, Cryptographic Transformations, User-Centric Authentication.

## Contents

# Chapter 1

## 1.1 Introduction

In the digital age, the security of user data is paramount. Traditional authentication methods, while effective to an extent, have shown vulnerabilities that can be exploited by malicious entities [1]. This project aims to develop a novel two-factor authentication system that leverages the principles of image steganography.

Two-factor authentication (2FA) is a security process in which users provide two different authentication factors to verify themselves [2]. This process is designed to better protect both the user's credentials and the resources the user can access. In this project, the two factors used for authentication are a password and a uniquely encoded image.

Image steganography is the art of hiding information within digital images. This technique can be used to embed secret data within a seemingly innocuous image file [3]. In the context of this project, steganography is used to hide hashed password data within user-provided images, creating a unique and secure authentication factor.

During registration, users must provide three images, a password, and an email address. The system then performs a series of cryptographic transformations on the provided data. The hash of the user's password is calculated and hidden within the first image. Subsequently, the hashes of the first and second images are calculated and hidden within the second and third images, respectively. A random key is generated and used to hide the hash of the second image within the third image. The final image, now containing the hashes of the password and the previous images, is given to the user to be used for login.

This innovative approach to two-factor authentication provides an additional layer of security. By requiring users to submit an image along with their password during login, the system ensures that even if a user's password is compromised, unauthorized access to the user's account is still prevented unless the attacker also has the uniquely encoded image.

The project represents a significant advancement in user authentication. By combining traditional password-based authentication with image steganography, the system offers a more secure and personalized user authentication experience. This project underscores the potential of image steganography in strengthening security protocols, paving the way for more secure and personalized user authentication systems in the future.

## 1.2 Problem Statement

The problem of this project aims to solve the vulnerability of single-factor, password-based authentication systems. By developing a two-factor authentication system that uses both a password and a uniquely encoded image. Accordingly, the project aims to significantly reduce the risk of security breaches and enhance the protection of user data. This paved the way for more secure and personalized user authentication systems in the future.

## 1.3 Motivation

The motivation behind this project is to enhance the security of user data and privacy. By integrating steganography with image-based authentication which can create a more secure and user-friendly authentication process.

## 1.4 Project Objectives

The primary objective is to design and implement a two-factor authentication system using images and steganography. The system should be able to:

- Allow users to register by inputting three images and generating a composite image.
- Authenticate users during login using the composite image, password, and email.
- Ensure the system is user-friendly and secure.

## 1.5 Limitations

Potential limitations could include:

- If the client stored the image inappropriately, then it could affect the login process.
- The cryptographic transformations and steganographic techniques used in this project could be computationally intensive, which might limit the system's scalability or performance on devices with lower computational power.

## 1.6 Document organization

The document is divided into 6 chapters as follows: the first one includes an introduction, the second one includes related works and background, the third one includes the workflow and design of the project, the fourth one includes implementation of the project, the fifth one includes the testing of the project and the last chapter includes the conclusion and future works.

# Chapter 2

## 2.1 RELATED WORK

**Digital image steganography using PVD (Pixel Value Differencing) and modulo operation.**

The research paper focuses on the issue of secure data delivery in digital communication, which has become on-demand because of the high number of network users. The approach has been proposed in the paper that uses pixel value differencing (PVD) and modulo operations (MO) as a solution. The image is divided into non-overlapping blocks consisting of three consecutive pixels, and the secret data is embedded in a block using two phases: pixel difference modulo operation (PDMO) phase, and average PVD (APVD) readjustment phase. The initial phase, the difference between two consecutive pixels of a block is discovered. This is accomplished by using an adaptive range table and modulo operation to embed the secret data. Data embedding using the PVD approach involves considering the average of the first two stego-pixels and the third pixel in the second phase. The paper concludes that the results of the proposed approach have been compared with existing approaches and found to be improved. The paper acknowledges the major contributions of the proposed approach, which include an increase in peak signal-to-noise ratio (PSNR), an increase in hiding capacity, and avoidance of the fall-off boundary problem (FOBP). However, the paper does not explicitly mention any drawbacks of the proposed method [4].

**A Secure Image Steganography Using LSB (Least Significant Bit) And Double XOR Operations**

Double XOR and LSB are combined in the paper on secure image steganography presented here. Replacement, by using this technique, concealed data can be inserted into photos while still maintaining high security and imperceptibility. It can be used by applications that need secure communication and information concealment. Protecting data by hiding text messages in cover images. The message can be encrypted using a secret key, double Xor operations, and an encrypted stream of bits using the LSB technique. The LSB method's ability can be predicted by the simplicity hide, which is a drawback [5].

**A Novel DWT and Correlation Based Audio Steganography**

Data hiding, also known as steganography, is the topic addressed by the paper 'A Novel DWT and Correlation Based Audio Steganography'. A method is proposed by the authors to conceal a text message in an audio file. The DWT and correlation are the foundations of this technique. Different frequency components are decomposed by DWT to form the audio signal, with the secret message embedded in them. The objective of this method is to hide the message's existence so that the viewer cannot detect it and thus cannot decrypt it.
Improvements to security and robustness against various attacks are offered by the proposed method. It demonstrates its versatility by functioning well in both uncompressed and compressed domains. It is possible for sophisticated steganography methods that aim to detect hidden information to detect it like any other steganographic technique. The embedding of the secret message may have some impact on the audio file's quality [6].

**Steganography based approach to image authentication**

Steganography is the basis of an image authentication method discussed in the paper. To guarantee integrity and authenticity, it focuses on integrating secret authentication data into photographs. Iterative steganography techniques are being investigated to detect unauthorized changes. The project intends to create a safe way for certifying digital images that can be used for forensic analysis, copyright protection, and validation of digital evidence. The challenge is to process images without losing or damaging any information. A neural network symmetric encryption hash functions can be used to combine cryptography and steganography. The proposed method's main weakness is that it relies on the confidentiality of a randomly generated Steganography key and the time required for embedding [7].

**The Process of Steganography image involves the following steps:**

1. **Selecting A Cover Image:** Choosing a cover image is the process by which the secret data will be hidden. A regular image file, such as a Jpeg or PNG file, is acceptable.
2. **Choosing the secret data:** Any form of digital information, such as text, images, audio, or even other files, can be classified as secret data. The size of the secret data should be considered carefully as it should not have any major impact on the cover image's size or visual quality.
3. **Encoding the secret data:** Encoding or embedding the secret data takes place within the cover image. The process typically involves adjusting the least significant bits (LSBs) of the pixel's values in the cover image. To conceal the information in the image, the LSBs are modified to represent bits of secret data.
4. **Generating a Stego image:** A modified version of the cover image with the hidden secret data is what the Stego image is. It is difficult to distinguish the hidden information without special Steganalysis techniques because the Stego image should look remarkably similar or identical to the original cover image visually.
5. **Extraction of the secret data:** The recipient uses a Steganography tool or software that is knowledgeable about the specific steganography technique used to retrieve the hidden information. By analyzing the Stego image and extracting the embedded secret data, the function allows the recipient to access the concealed information.

To conceal sensitive information or messages within an image file without affecting its appearance is the main objective of a steganography image. The changes made to the image can be difficult for the human eye to notice because they are often subtle and well-incorporated into its visual characteristics. This allows for the effective use of Steganography images in covert communication and secret data transmission.

## 2.2 BACKGROUND

When logging into online accounts or systems, two-factor authentication (2FA) adds an extra layer of protection to the authentication process. To increase account security, two separate pieces of evidence must be presented to verify the user's identity [8].

Typically, the Authentication Process involves entering a username or email Address along with a Password. Various security threats, such as brute-force attacks, phishing attempts, or password leaks, can make passwords vulnerable. The introduction of an extra factor in Two-Factor Authentication mitigates these risks by making it significantly harder for attackers to gain unauthorized access.

**Three main categories can be categorized by the two factors used in 2fa:**

1. **Something You Have**: Possessing a physical item or a unique token that is exclusive to the user is what this factor involves. You have the option to use either a smartphone, a hardware security key, or a smart card. To prove the user's identity, these devices provide cryptographic verification or OTP.
2. **Something You Are**: Biometric characteristics that are unique to every individual are included in this factor. Examples of these include fingerprint scans, facial recognition, iris scans, or voice recognition. The use of biometric authentication is on the rise, particularly in smartphones and other devices equipped with biometric sensors.
3. **Something You Know**: This factor refers to knowledge-based information that only the legitimate user should Possess. It is typically a password or a personal identification number (Pin). This factor is like the traditional authentication method, but it is complemented by an additional factor to strengthen security.

A higher level of security can be achieved by combining these factors in Two-Factor Authentication. Even If an Attacker gets one of these factors, then, he/she still needs to get the other one. This extra layer prevents unauthorized access attempts. The exact implementation of two factor authentication may depend on the service or platform, the common methods include:

1. **OTP by authenticator apps:** This method uses an android app that generates a different random number after a specific amount of time. In these cases, the user must enter the correct OTP and a correct password.
2. **Email Verification**: In this method the user gets an email with a link or a random number to use in the authentication process.
3. **API keys:** In case of using API, the user gets a unique and exceedingly long key for the API to make authorized request on the API to process some data.
4. **Biometric Authentication:** Using something you have in your body, that is unique in your genetics, will add another layer of security for authentication process.

Enhancing the security of online accounts and systems can be achieved by implementing Two-Factor Authentication. The project website's use of Two-Factor authentication decreases the risk of unauthorized access, making it more difficult for attackers to compromise accounts and prevent sensitive information from being compromised.

**Steganography:** It is the art of hiding data inside a multimedia or ordinary text to make some secret data hidden from the first view. In The Case of Steganography Image, In the case of steganography, it refers specifically to hiding information within digital images. [9].

- **Least significant bit (LSB) insertion:** By replacing the least significant bits of the pixel's values in the cover image with bits of the secret data, this technique is used. Due to the minimal contribution of the LSBs to the image's visual quality, this method can effectively conceal information without significant visual changes. [10].

- **Spread spectrum:** This technique spreads the secret data across multiple pixels in the cover image, utilizing techniques like frequency domain encoding or statistical analysis to ensure that the hidden data is evenly distributed and difficult to detect. [11].

- **Transform domain techniques**: These techniques involve embedding the secret data in the transformed domain of the image, such as the discrete cosine transform (DCT) coefficients in Jpeg images or the wavelet Coefficients in Wavelet-Transformed Images. [12].

**Encoded:** Converting something, like information, from one communication system to another. The purpose of this is to convert a message into code. A method for inserting a secret code into an image. [13].

**Decode:** To translate code into understandable language. To extract the code from the encoded image, its opposite is encoded. [14].

**Hashing:** This is a fundamental concept in computer science and cryptography that refers to the process of taking input data of any size and producing a fixed-size string of characters as output. A hash value or hash is commonly referred to as this output [15].

**The XOR (exclusive OR):** is a binary operation on Boolean operands. It takes two inputs and produces one output. The inputs and outputs may only take the values of TRUE or FALSE, which can also be considered as 1 or 0 respectively [16]**.**

Hash's primary goal is to create a unique and consistent representation of data. To be considered a good hash function, it must produce the same hash value for the same input data consistently and produce different hash values for different input data with a high probability. Furthermore, a slight change in the input data could lead to a Hash value that is significantly different.

**The key characteristics and uses of hashing:**

1. **Data integrity verification:** Data integrity is ensured by using hash functions extensively. By generating a hash value for a piece of data, such as a file or a message, one can later verify if the data has been modified during transmission or storage. To determine if any changes have occurred, it is possible to compare the hash value of the received data with the formerly computed hash value.
2. **Password Storage:** Storing passwords securely requires the use of hash functions. Instead of storing The storage of passwords in plaintext is a security risk if the database is compromised, so they are hashed and only the hashed values are stored. Upon attempting to log in, the user's password is hashed and compared to the stored hash value. If the hashes match, the password is considered valid.

3. **Digital Signatures:** Digital signature algorithms are dependent on hashing as a crucial component. In this scenario, the hash function creates a digest fixed in size for the message that must be signed. The signer's private key is used to encrypt this digest, creating a digital signature. By computing the hash of the received message and decrypting the signature using the public key of the signer, the recipient can verify the integrity of the message.
4. **Data Retrieval**: Hashing is utilized in data structures such as hash tables or hash maps to achieve efficient data retrieval. The use of hash functions allows for fast access to the corresponding values by mapping keys to specific slots or buckets. This enables fast searches, inserts, and deletions in large data sets.
5. **Unique Identifiers:** Unique identifiers or hash codes for data can sometimes be generated using hash functions. These identifiers quickly identify and compare data records without comparing the actual data itself. Databases, cache systems, and indexing algorithms use them extensively.

Hash functions commonly employed include Md5, Sha-1, Sha-256, and Sha-3. It is important to note that cryptographic hash functions such as Sha-256 are designed to be computationally infeasible to reverse-engineer or find collisions (different inputs producing the same hash). They are regarded as more secure for cryptographic applications as a result.

**Random Number:** A random number is a number that is chosen randomly from a set of numbers, just like the term suggests. Each number in a specified distribution has an equal chance of being randomly chosen. [17].

**Code:** A method of communication that utilizes signals or symbols (such as letters or numbers) to convey assigned and often confidential information. [18].

**Encryption:** The way information is transformed into a secret code that hides its real meaning. Cryptography is the science of encrypting and decrypting information. Plaintext is the term used for unencrypted data in computing, while encrypted data is referred to as Cipher Text [19].

**Decryption:** Decryption is the process of converting encrypted data into its original form. In general, it is a process that uses reverse encryption. Decryption requires a secret key or password to decrypt the data, so only authorized users can decrypt it. [20].

**Cryptography:** Encoding information in a secure manner is the science and practice of securing it.
Unauthorized individuals cannot read it. Techniques and algorithms are used to ensure confidentiality and integrity. Data authentication and non-repudiation. Ensuring the privacy and security of sensitive information in various fields, such as computer science, information technology, communications, and finance, is a vital role played by cryptography [21].

## 2.3 Environment

**The technology selection was based on their suitability for the project's requirements.**

- Programming Language: Python 3.11.7
- The primary programming language was chosen because of its extensive libraries and support for image processing and steganography techniques.
- Integrated Development Environment (Ide): Visual Studio Code.
- HTML, JavaScript, and CSS are all fundamental technologies used in web development. Html oversees determining the structure and content of web pages, while JavaScript handles interactions and dynamic behavior. CSS complements Html with its visual presentation styling and formatting. Their collaboration resulted in web applications that are engaging and user-friendly.
- Operating System: Windows 10
- The framework Django.

# Chapter 3

## 3.1 Introduction

This chapter delves into the design of the proposed two-factor authentication (2FA) system within a web application context. The chapter will commence by elucidating the design aspects of the registration and login processes, detailing the specific fields required from the user and their subsequent utilization within the system.
The chapter will further explore the intricate design of the cryptographic transformations and steganographic techniques employed in the system. Accompanied by flowcharts and diagrams, the chapter aims to provide a comprehensive understanding of the system's workflow, from user registration to successful authentication.

## 3.2 The Design

### 3.2.1 Registration process:

The registration process is the first interaction a user has with the system. It's designed to be secure and user-friendly. Here's how it works:
Upon accessing the registration page, the user is presented with a form that requires several inputs:

- Email Field: The user provides their email address.
- Password Field: The user inputs their chosen password.
- Confirm Password Field: The user re-enters their password for verification.
- First Image Upload Field: The user uploads the first image.
- Second Image Upload Field: The user uploads the second image.
- Third Image Upload Field: The user uploads the third image.

After the user has filled out all the fields and uploaded the images, they can submit the form. The system then performs a series of cryptographic transformations:

- The server checks if the password meets requirements which are: at least 8 characters, 1 lower case, 1 upper case and 1 special character.
- The server checks if the email is valid in the registration.
- The server calculates the hash of the user's password and hides it within the first image.
- The server calculates the hash of the first image (which now contains the password hash) and hides it within the second image.
- The server calculates the hash of the second image (which now contains the hash of the first image) and hides it within the third image.
- A random key is generated and hides it along with the hash of the second image within the third image.
- The final image, now containing the hashes of the password and the previous images and a unique key, is given to the user to be used for login. This completes the registration process.

This process ensures that each user has a unique image that is tied to their account, enhancing the security of the system by adding an additional layer of protection against unauthorized access. It ensures that even if a user's

password is compromised, an attacker would still need the unique image to gain access to the user's account. This makes the registration process more secure compared to traditional single-factor, password-based systems.

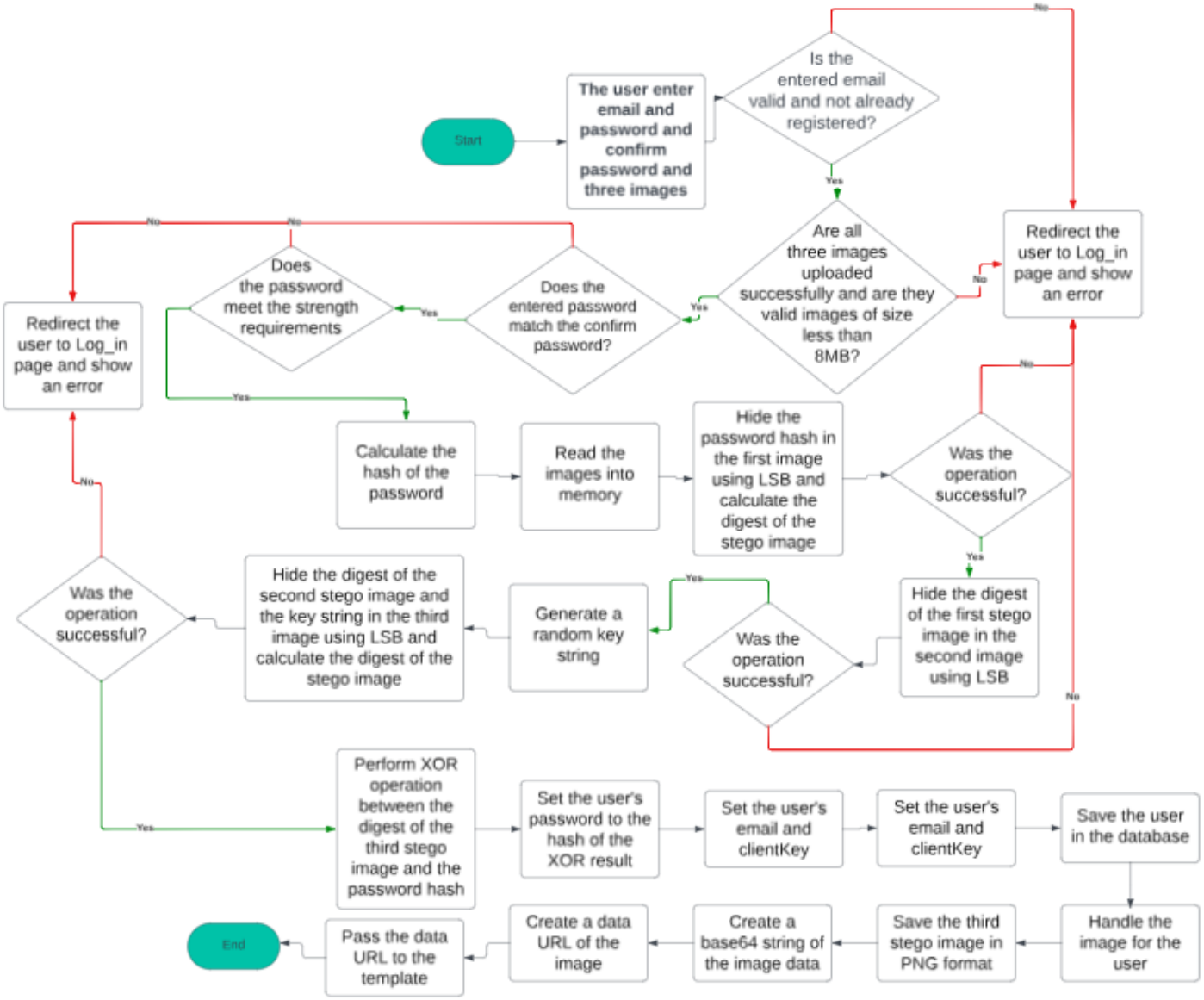


*Figure 3.2.1: Registration process.*

### 3.2.2 Login process:

The login process on the client-side is designed to be secure and user-friendly as shown in figure 3.2.2. Here's how it works:

Upon accessing the login page, the user is presented with a form that requires three inputs:

- Email Filed: The user enters his email.
- Password Field: The user enters the password that was set during the registration process.
- Image Upload Field: The user uploads the composite image that was downloaded after registration.

Upon the user entering their password and uploading the composite image, they can submit the form. After the user has submitted the form, the server verifies the existence of the email, then calculates the hash of the XOR result between the image and the hash of the password and extract the hidden key from the image if the hash and the key the same as saved in the database the user will log-in correctly, otherwise the user will not log-in.

This two-factor authentication process, which requires both a password and a unique image, enhances the security of the system by adding an additional layer of protection against unauthorized access. It ensures that even if a user's password is compromised, an attacker would still need the unique image to gain access to the user's account. This makes the login process more secure compared to traditional single-factor, password-based systems

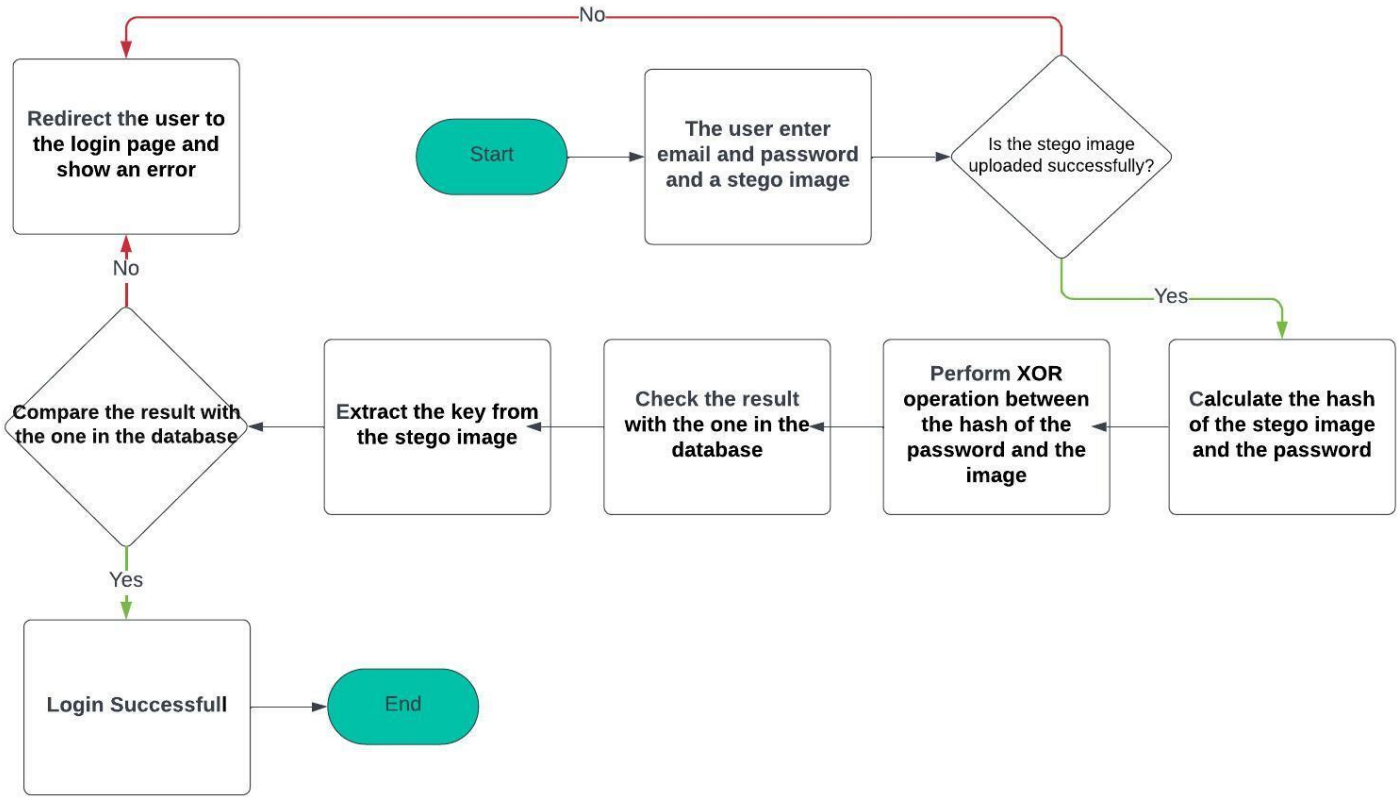


*Figure 3.2.2: Login process.*

### 3.2.3 User Interface Design:

The user interface design of the registration page is a crucial aspect of the system, as it is the first point of interaction for the user. The design is intuitive and user-friendly, ensuring a smooth user experience.

**The registration page** features a cosmic-themed background, creating an engaging visual experience for the user. The form is centrally placed, making it the focal point of the page.
The registration form consists of the following fields:

- **Email Field**: This is where users enter their email address. It's essential for communication and account recovery.
- **Password Field**: Users enter their chosen password here. The password is a crucial part of the two-factor authentication process.
- **Confirm Password** Field: Users re-enter their password here to ensure it has been entered correctly.
- **Image Upload Fields**: There are three separate fields for users to upload their images. These images will be used in the steganographic process.

Each image upload field comes with a "Choose File" button, allowing users to browse and select the images from their device. A text display next to the button shows whether a file has been chosen.
Finally, a "Register" button is provided at the bottom of the form. Once all the necessary information is filled out, users can click this button to submit the form and complete the registration process.
The design of the registration page is clean and straightforward, making it easy for users to understand what is required of them. The visually appealing cosmic-themed background adds a touch of creativity to the page, enhancing the overall user experience.

**The login page** like the registration page in the design and consist of the following fields:

- **Email Field**: This is where users enter their email address. It's essential for communication and account recovery.
- **Password Field**: Users enter their chosen password here. The password is a crucial part of the two-factor authentication process.
- **Image Upload Field**: Users enter their Stego image. This is the second factor of the authentication process.

# Chapter 4

## 4.1 Introduction

The chapter explains the technical implementation of the project for the registration, login, steganography algorithm and the other functions used to build the 2Fa authentication.

## 4.2 Auxiliary functions

The **keyString** function is used to create random 32 bytes using os library code encoded on base64 using base64 library; the function returns the key as string as shown in Figure 4.1.

```
# Generate a random 32-byte key and convert it to a string
def keyString():
    key = base64.b64encode(os.urandom(32)).decode('utf-8')
    return key
```

*Figure 4.1: Key generator.*

The **xor_hashes** function is used to calculate the XOR between two hashes and return the results in Ascii character as shown in Figure 4.2.

```
#Xor between 2 hashes
Comment Code
def xor_hashes(hash1, hash2):
    # Convert the hashes from hexadecimal to bytes
    hash1_bytes = bytes.fromhex(hash1)
    hash2_bytes = bytes.fromhex(hash2)

    # XOR the two hashes and return the result
    xor_result = bytes(a ^ b for a, b in zip(hash1_bytes, hash2_bytes))
    return binascii.hexlify(xor_result).decode()
```

*Figure 4.2: XOR between two hashes.*

The **hash_string** function is used to calculate the hash value for a specific string using the hashlib libary and return the hash as string as shown in Figure 4.3.

```
#Compute the digeest of a string
Comment Code
def hash_string(input_string):
    # Create a new sha256 hash object
    sha_signature = hashlib.sha256(input_string.encode()).hexdigest()
    return sha_signature
```

*Figure 4.3: Hash calculator for a string.*

The **image_digest** function is used to calculate the hash for a specific image using the hashlib library and return the hash of the image as string as shown in Figure 4.4.

```
#Calculate the digest of the imgae
Comment Code
def image_digest(image_data):
    # Read the image data
    data = image_data.tobytes()
    # Calculate the hash
    hash_object = hashlib.sha256(data)
    hex_dig = hash_object.hexdigest()
    return hex_dig
```

*Figure 4.4: Calculate the hash for an image.*

The **is_image** function is used to check if the uploaded file is an image or not using imghdr libary and return True or False as shown in Figure 4.5.

```
#Check if the file is image
Comment Code
def is_image(file):
    image_types = [ 'jpeg', 'png']
    return imghdr.what(file) in image_types

#Hiding data into img by LSB
```

*Figure 4.5: check if it is an image.*

This **LSB** function implements the Least Significant Bit (LSB) method of steganography. It takes two parameters: an image and a message as shown in Figure 4.6:

1. Message Conversion: The function first converts the message into binary format. This is done by iterating over each character in the message, converting it to its ASCII value using the ord() function, and then converting that value to an 8-bit binary string using the format () function.
2. Total Bits Calculation: The function then calculates the total number of bits that can be hidden in the image. This is done by multiplying the width and height of the image by the number of color components in a pixel (obtained using len(image.getpixel((0, 0)))).
3. Message Length Check: The function checks if the binary message is too long to be hidden in the image. If it is, the function raises a ValueError.
4. Output Image Creation: The function creates a new image to hold the output. This is done using the Image.new() function, which creates a new image with the same mode and size as the original image.
5. Pixel Modification: The function then iterates over the pixels in the image. For each pixel, it modifies the least significant bit of each color component to contain the binary message. This is done using bitwise operations: pixel[n] & ~1 clears the least significant bit of the color component, and | int(binary_message[message_index]) sets it to the current bit of the binary message.
6. Output Image Writing: The modified pixel is written to the output image using output_pixels[x, y] = tuple(pixel).

7. Output Image Return: Finally, the function returns the output image, which now contains the hidden message.

This function is a crucial part of the two-factor authentication system, as it allows the system to hide hashed data within user-provided images, enhancing the security of the system.

```
Comment Code
def LSB(image, msg):
    # Convert the message to binary
    binary_message = ''.join(format(ord(i), '08b') for i in msg)
    # Calculate the total number of bits that can be hidden in the image
    total_bits = image.width * image.height * len(image.getpixel((0, 0)))
    # Check if the binary_message is too long for the image
    if len(binary_message) > total_bits:
        raise ValueError("The message is too long to be hidden in the image.")
    # Create a new image to hold the output
    output_image = Image.new(image.mode, image.size)
    output_pixels = output_image.load()
    # Iterate over the pixels in the image
    message_index = 0
    for y in range(image.height):
        for x in range(image.width):

            # Get the current pixel
            pixel = list(image.getpixel((x, y)))

            # Modify the pixel to contain the message
            for n in range(len(pixel)):
                if message_index < len(binary_message):
                    # Change the least significant bit of each color component
                    pixel[n] = pixel[n] & ~1 | int(binary_message[message_index])
                    message_index += 1

            # Write the modified pixel to the output image
            output_pixels[x, y] = tuple(pixel)

    # Return the output image
    return output_image
```

*Figure 4.6: Hide the data inside an image*

The **extract_LSB** function, is used to extract the hidden message from an image. It takes one parameter: an image as shown in Figure 4.7:

1. Image Conversion: The function first converts the image into a NumPy array. This allows for efficient manipulation of the image data.
2. LSB Extraction: The function then extracts the least significant bit of each color component in the image. This is done using the np.bitwise_and() function, which performs a bitwise AND operation between the image data and 1 (which is 00000001 in binary). This operation effectively extracts the least significant bit of each color component.
3. Binary Message Conversion: The function converts the extracted bits into a binary string. This is done by flattening the array of bits into a 1D array using lsb_array.flatten(), converting each bit to a string using map (str, ...), and then joining these strings together into a single binary string.

4. Binary to Text Conversion: The function then converts the binary string into text. This is done by splitting the binary string into 8-bit chunks, converting each chunk into an integer using int(binary_message[i:i+8], 2), converting each integer into a character using chr(...), and then joining these characters together into a single string.
5. Hidden Message Extraction: The function finds the start and end of the hidden message within the text. This is done by finding the indices of the "#####" and "$$$$$" delimiters. If both delimiters are found, the function extracts the hidden message from between them. If not, the function returns a message stating that no hidden message was found in the image.
6. Output Message Return: Finally, the function returns the output message, which is the hidden message extracted from the image.

This function is a crucial part of the two-factor authentication system, as it allows the system to extract the hidden hashed data from the user-provided image during the login process, enhancing the security of the system

```python
def extract_LSB(image):
    # Convert the image to a numpy array
    image_array = np.array(image)

    # Extract the least significant bit of each color component
    lsb_array = np.bitwise_and(image_array, 1)

    # Convert the binary message to a string
    binary_message = ''.join(map(str, lsb_array.flatten()))

    # Convert the binary message to text
    output_message = ''.join(chr(int(binary_message[i:i+8], 2)) for i in range(0, len(binary_message), 8))

    # Find the start and end of the hidden message
    start = output_message.find("#####")
    end = output_message.find("$$$$$")

    # Extract the hidden message
    if start != -1 and end != -1:
        output_message = output_message[start+5:end]
    else:
        output_message = "No hidden message found in the image."

    # Return the output message
    return output_message
```

*Figure 4.7: Extract the hidden data from the image*

The **check_image_size** function is used to calculate the size of the image and return an integer value as shown in Figure 4.8.

```python
#Calculate the image of the size
Comment Code
def check_image_size(image):
    size = image.size
    size_in_MB = size / (1024 * 1024)  # Convert size to MB
    return size_in_MB
```

*Figure 4.8: Extract the hidden data from the image*

## 4.3 Registration

The **Register** function is the main function for the registration process. Firstly, the server checks if the request is Post or not as shown in Figure 4.9. If not, the server will reload the registration page, if it is a post method then will do the following:

```
def register(request):
    if request.method == 'POST':
```

*Figure 4.9: Check if it is a POST method*

1- Check if the imgae1, image2 and image3 tags in the request or not as shown in Figure 4.10. If it exists then go to the second step, otherwise the server will redirect the user to the registration page and show an error

```
    #Checking Image
    if 'image1' in request.FILES :
        image1 = request.FILES['image1']
    else:
        messages.info(request,"Please enter a correct imgae1")
        return redirect('register')

    if 'image2' in request.FILES :
        image2 = request.FILES['image2']
    else:
        messages.info(request,"Please enter a correct imgae2")
        return redirect('register')

    if 'image3' in request.FILES:
        image3 = request.FILES['image3']
    else:
        messages.info(request,"Please enter a correct imgae3")
        return redirect(request,'register')
```

*Figure 4.10: Check if the image tag inside the request*

2- Check if the file of image1, image2 and image3 is image or not using **is_image** function as shown in Figure 4.11. If it is an image then go to the third step, otherwise the server will redirect the user to the registration page and show error**.**

```
if not is_image(image1) or check_image_size(image1)>8:
    messages.info(request,"Please enter a correct imgae1")
    return redirect('register')

if not is_image(image2) or check_image_size(image2)>8:
    messages.info("Please enter a correct imgae2")
    return redirect(request,'register')

if not is_image(image3) or check_image_size(image3)>8:
    messages.info(request,"Please enter a correct imgae3")
    return redirect('register')
```

*Figure 4.11: Check if the file is an image or not*

3- Check if the password equals the confirmation as shown in 4.12. If it is then going to the 4th step, otherwise the server will redirect the user to the registration page and show an error.

```
if password != confirm:
    messages.info(request, 'Unmatch passwords')
    return redirect('register')
```

*Figure 4.12: Check if the password equals the confirm or not*

4- Check if the password has at least 8 characters, has integer, has string and has a special character as shown in Figure 4.13. If it meets the requirements then go to the 5th step, otherwise the server redirects the user to the registration page and shows an error.

```
cnt_strength_password_capital = True
cnt_strength_password_lower = True
cnt_strength_password_special = True
cnt_strength_password_number = True
cnt_strength_password_len = 0

for i in password:
    if i>='A' and i<='Z':
        cnt_strength_password_capital = False
    elif i>='a' and i<='z':
        cnt_strength_password_lower = False
    elif i in ['!','@','#','$','%','^','&','*','(',')','-','_','+','=']:
        cnt_strength_password_special = False
    elif i>='0' and i<='9':
        cnt_strength_password_number = False
    cnt_strength_password_len+=1

if cnt_strength_password_len<8 or ( cnt_strength_password_capital or  cnt_strength_password_lower or  cnt_strength_password_special
    messages.info(request,"Please pick a better password contains: 8 chars, lower, capital, special and a number")
    return redirect('register')
password_hash = hash_string(password)
```

*Figure 4.13: Check if the password met the requirements*

5- Read the images in memory using as shown in Figure 4.14.

```
    # Read the images into memory
    image1_data = Image.open(io.BytesIO(image1.read()))
    image2_data = Image.open(io.BytesIO(image2.read()))
    image3_data = Image.open(io.BytesIO(image3.read()))
```

*Figure 4.14: Read the images in memory*

6- Hide the hash of the password inside image1 using the LSB function as shown in Figure 4.15.

```
#Hiding by LSB R1 && calculating the digest of stego image
try:
    stego_image1_R1 = LSB(image1_data,password_hash)
    stego_digest1_R1 = image_digest(stego_image1_R1)
except:
    messages.info("Please pick a better image")
    return redirect("register")
```

*Figure 4.15: Read the images in memory*

7- Hide the hash of the first Stego image inside the second image as shown in Figure 4.16

```
#Hiding by LSB R2 && calculateing the digest and of stego2
try:
    stego_image2_R2 = LSB(image2_data,stego_digest1_R1)
except:
    messages.info("Please pick a better image")
    return redirect("register")
```

*Figure 4.16: Read the images in memory*

8- Create a key using **keyString** and then hide the key and the hash of the second Stego image inside the third image as shown in Figure 4.17.

```
#Hiding by LSB the stego_key && calculateing the digest and of stego3
try:
    keyHided = keyString()

    stego_image3_R3 = LSB(image3_data,image_digest(stego_image2_R2)+"#####"+keyHided+"$$$$$")
    stego_digest3_R3 = image_digest(stego_image3_R3)
except:
    messages.info("Please pick a better image")
    return redirect("register")
```

*Figure 4.17: Read the images in memory*

9- Calculate the Xor of the password hash and the third stego image and save the hash of the XOR result with salt in the database and save the key as shown in Figure 4.18.

```
#Saving the user in the database
xor=xor_hashes(stego_digest3_R3,password_hash)
user.set_password(hash_string(xor))
user.save()
user.email = email

user.clientKey = keyHided
user.save()
```

*Figure 4.18: Read the images in memory*

10- After producing the third Stego image the server handles the image to the user through directing the user to yourStego page as shown in Figure 4.19.

```
#Handle the image for the user
image_io = io.BytesIO()
stego_image3_R3.save(image_io, format='PNG')

# Create a base64 string of the image data
image_str = base64.b64encode(image_io.getvalue()).decode('utf-8')

# Create a data URL of the image
data_url = '' + image_str

# Pass the data URL to the template
return render(request, 'yourStego.html', {'image_data_url': data_url})
se:
```

*Figure 4.19: Handle the image to the user*

## 4.4 Login

The **log_in** function is the main function for the logging process. The server checks if the request is POST or not and checks if the user enters a valid email in a valid way. Then calculate the hash of the password and the stego image. Then calculate the hash of the XOR result of the XOR between the hash of the password and the hash of the Stego image and extract the hidden key. If the key and the hash match the ones save in the database then the user grant access to the hello page, otherwise, the server will redirect the user to the login page and show an error to the user as shown in figure 4.19

```
def log_in(request):
    if request.method == 'POST':
            password = str(request.POST['password'])
            #Checking email
            email = request.POST['email']
            for i in email:
                if (i<'A' or i>'Z') and (i<'a' or i>'z') and (i<'0' or i>'9') and i not in  ['_', '.','@']:
                    message (parameter) request: Any er a valide mail: only contain alphapets and numbers")
                    return
            if 'image1' in request.FILES:
                image1 = request.FILES['image1']
                if is_image(image1):
                    image1_data = Image.open(io.BytesIO(image1.read()))
                    password_hash = hash_string(password)
                    image1_digest = image_digest(image1_data)
                    xor_hash = hash_string(xor_hashes(image1_digest,password_hash))
                    user = authenticate(request, username=email, password=xor_hash)
                    yourKey = extract_LSB(image1_data)
                    if user is not None and yourKey == user.clientKey:
                        login(request,user)
                        return redirect('hello')
                    else:
                        messages.info(request,"Invalid emair or credentials")
                        return redirect('log_in')
                else:
                    messages.info("Please upload jpg or png image only")
                    return redirect('log_in')
            else:
                messages.info(request,'No image file uploaded')
                return redirect('log_in')
    return render(request, 'log_in.html')
```

*Figure 4.19: Login*

## 4.5 Index

The **index** function is used to redirect the user when request the Url_of_The_page to the index page as show in Figure 4.20

```
#The index page
Comment Code
def index(request):
    return render(request, "index.html")
```

*Figure 4.20: Index page*

## 4.6 Hello

The **hello** function is used to redirect the user when request Url_of_The_page/hello.html to the hello page as shown in Figure 4.21

```
#A hello page
Comment Code
@login_required
def hello(request):
    return render(request, 'hello.html')
```

*Figure 4.21: Hello page*

# Chapter 5

## 5.1 Introduction

The chapter explains the testing process for the web application. What happens when there is an error in the form or does not meet the requirement and check the security of the web application through checking what happened when the attacker tries to crack the password or try to find the image of the user online, so to show the tunning and the security of the application.

## 5.2 Testing

### 5.2.1 Enter invalid values in the fields

As a general rule, in designing applications is not trust the user, so the application should catch any mistakes and show an error for the user to enter the correct values. In the following table, there are different kinds of error and what is the error and what the expected message should be shown to the user. Each case has an image below.

| The kind of invalid value | The result |
|---|---|
| Already used Email | **Show Error: "Email already has been used!"** |
| The password doesn`t equal the confirm. | **Show Error:" Passwords do not match!"** |
| Invalid Images | **Show Error: "Please pick a better image"** |
| Invalid Email | **Show Error: "please enter a valid mail: only contain alphabets and numbers"** |
| The password is less than 8 characters | **Show Error:" Password must be at least 8 characters long!"** |
| The password doesn`t have a special character | **Show Error:" Password must contain at least one special character (e.g., !, @, #, $, %, ^, &, *)!"** |
| The password doesn`t have an Upper-case character | **Show Error:" Password must contain at least one uppercase character!"** |
| The password doesn`t have a Lower-case character | **Show Error:" Password must contain at least one lowercase character!"** |


*Figure 5.2.1: invalid values.*

## 5.3 Password cracking

While the hash has been saved is not related to a word rather than that it is related for different random inputs, that mean the hash should not be cracked by any well-known tool that use wordlists such as CrackStation, john. As shown in Figure 5.4 the hash can`t be found while is not related to normal character and words.

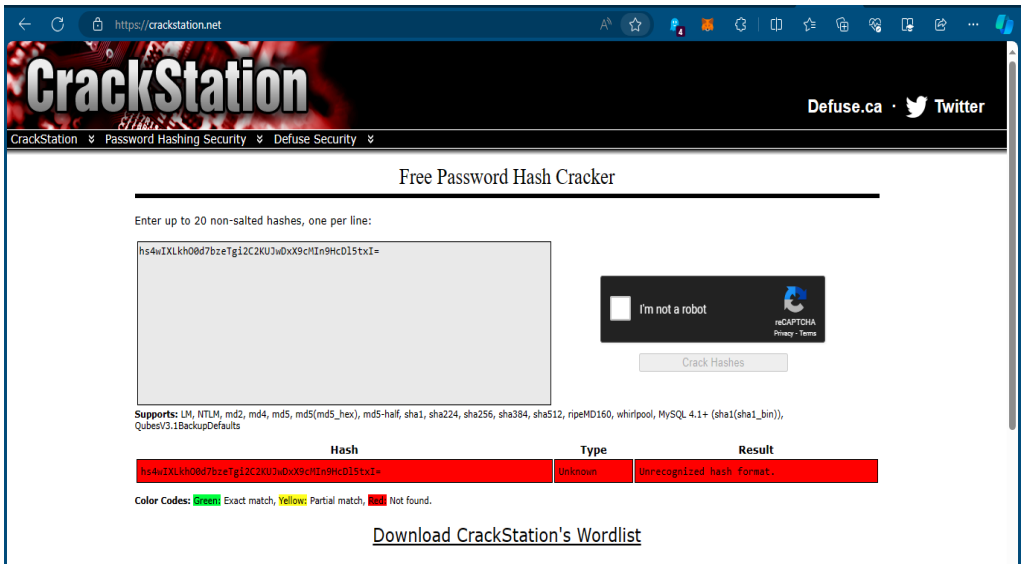


*Figure 5.3: The hash can`t be cracked*

## 5.4 Find a similar image

In case the attacker could find the Stego image online while the attacker saw the image of the user, then the attacker is not supposed to find the correct image while it has a unique key and hash value, so the attacker can`t get right image from any online page in normal cases as shown in Figure 5.4.1 and Figure 5.4.2. The hashes for the same image regarding the visual are not equal in hash because one of them has a unique hash and key and the quite the same while the PSNR equals 90.45443586507005 as shown in figure 5.4.3.

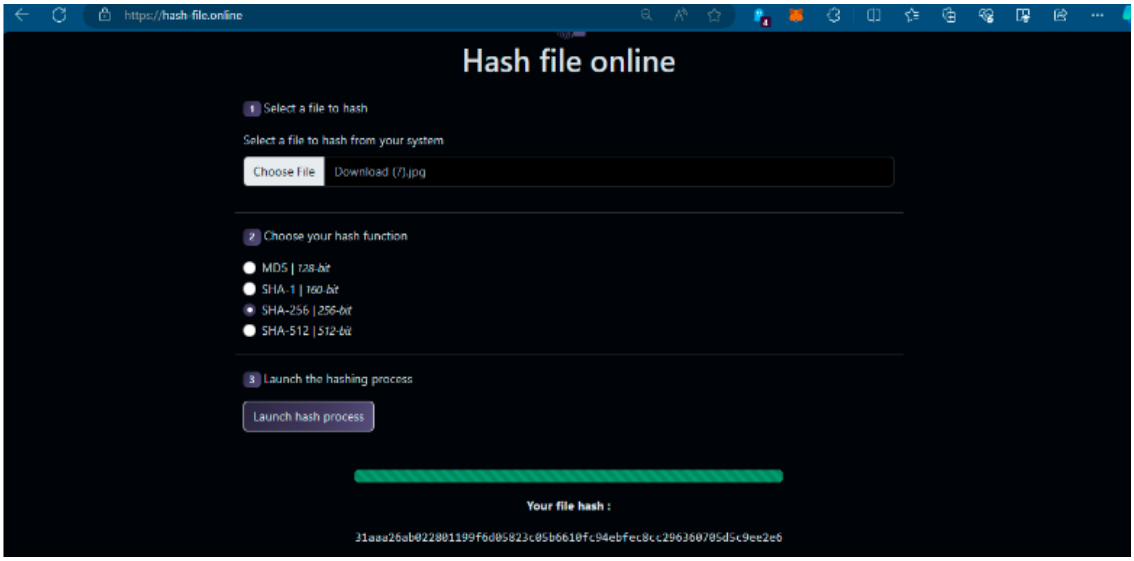


*Figure 5.4.1: Hash of the first image*

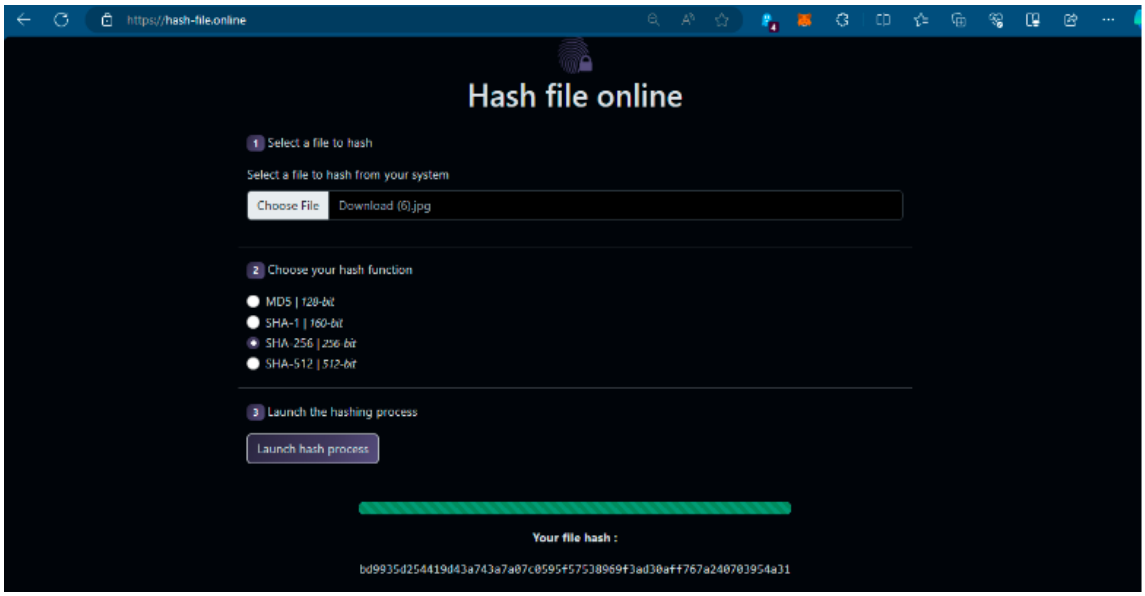


*Figure 5.4.2: Hash of the second image*

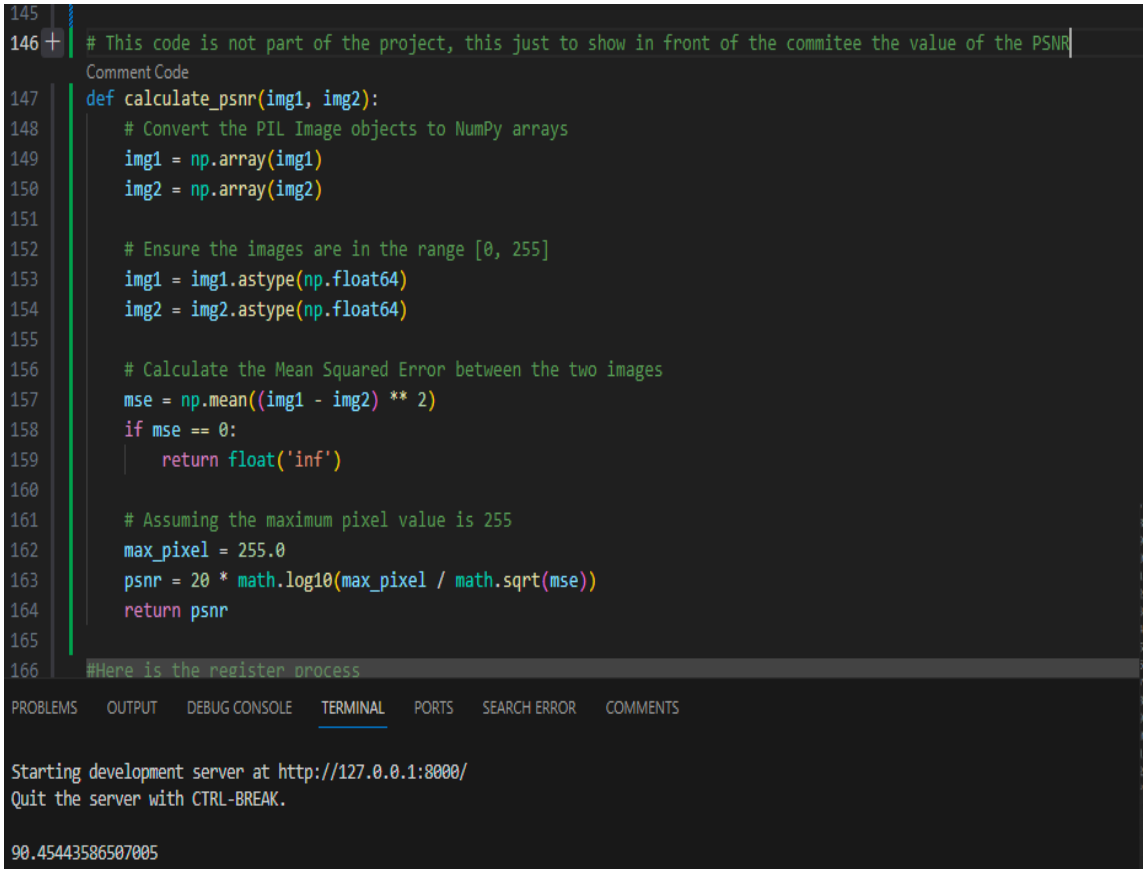


*Figure 5.4.3: PSNR difference between the original and the Stego image*

# Chapter 6

## Conclusion

In conclusion, the project has been a significant endeavor that has resulted in notable achievements and contributions. The aim of the project is to meet specific requirements and provide valuable solutions to target users. A robust and functional system was developed after successfully achieving several milestones during the implementation process.

The project achieved its primary objectives, including the implementation of key features. Making sure the system is reliable and meeting project deadlines. The system developed has had significant benefits for its intended users and has contributed to improving specific processes or solving specific problems.

There were challenges that came with the project. The implementation process faced challenges such as technical complexity and time management. Nonetheless, these challenges were provided.

There are opportunities to acquire valuable knowledge. Effective problem-solving, adaptability, and teamwork were gained through the project team's knowledge acquisition. Additionally, the project aided in improving technical abilities and enhancing an understanding of the technologies and frameworks employed. Even though the current implementation has been successful, there are still possibilities. The project has achieved the required security standards to ensure the protection of the site and the user data from theft, loss, or modification.

Some of the expected attacks are prevented such as SQL injection, CSRF, identity theft, password cracking.

In the future we could build an extension that works as a manager for images that are saved by the user for each website which makes him/her login automatically.

# REFERENCES


[1] Razaque, A., Amsaad, F., Khan, M. J., Hariri, S., Chen, S., Siting, C., & Ji, X. (2019). Survey: Cybersecurity vulnerabilities, attacks and solutions in the medical domain. *IEEE Access*, *7*, 168774-168797.

[2] Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., & Seamons, K. (2019). A usability study of five {two-factor} authentication methods. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)* (pp. 357-370).

[3] Maiti, C., Baksi, D., Zamider, I., Gorai, P., & Kisku, D. R. (2011). Data hiding in images using some efficient steganography techniques. In *Signal Processing, Image Processing and Pattern Recognition: International Conference, SIP 2011, Held as Part of the Future Generation Information Technology Conference FGIT 2011, in Conjunction with GDC 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings* (pp. 195-203). Springer Berlin Heidelberg.

[4] Sahu, A. K., & Swain, G. (2018). Digital image steganography using PVD and modulo operation. Internetworking Indonesia Journal, 10(2), 3-13.

[5] Ahmed, A., & Ahmed, A. (2020). A secure image steganography using LSB and double XOR operations. International Journal of Computer Science and Network Security, 20(5), 139-144.

[6] Nichal, A., Virkar, M. C., Kamble, M. P., Todkar, M. O., & Shirsat, M. A. A Novel DWT & Correlation Based Audio Steganography. International Journal on Recent and Innovation Trends in Computing and Communication, 3(3), 1637-1641.

[7] Forgáč, R., Očkay, M., & Javurek, M. (2021, October). Steganography Based Approach to Image Authentication. In 2021 Communication and Information Technologies (KIT) (pp. 1-6). IEEE.

[8] Schechter, S. E., Dhamija, R., Ozment, A., & Fischer, I. (2007, May). The emperor's new security indicators. In *2007 IEEE Symposium on Security and Privacy (SP'07)* (pp. 51-65). IEEE.

[9] "kaspersky," kaspersky, [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-steganography. [Accessed 8 6 2023].

[10] M. A. Aslam et al., "Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 32-38, doi: 10.1109/ICCIT52419.2022.9711628./

[11] D. Campana and P. Quinn, "Spread-spectrum communications," in IEEE Potentials, vol. 12, no. 2, pp. 13-16, April 1993, doi: 10.1109/45.283815.

[12] M. Rathod and J. Khanapuri, "A comparative study of transform domain methods for image resolution enhancement of satellite image," 2017 11th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 2017, pp. 287-291, doi: 10.1109/ISCO.2017.7856000.

[13] "merriam-webster," [Online]. Available: https://www.merriam-webster.com/dictionary/encode. [Accessed 8 6 2023].

[14] "merriam-webster," [Online]. Available: https://www.merriam-webster.com/dictionary/decode. [Accessed 8 6 2023].

[15] A. Zola, "techtarget," [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/hashing. [Accessed 8 6 2023].

[16] XOR ciphers model and the attack to it | Journal of Computer Virology and Hacking Techniques (springer.com)

[17] R. Awati, "techtarget," July 2022. [Online]. Available: https://www.techtarget.com/whatis/definition/random-numbers. [Accessed 8 6 2023].

[18] "merriam-webster," 1 jun 2023. [Online]. Available: https://www.merriam-webster.com/dictionary/code. [Accessed 8 6 2023].

[19] "techtarget," June 2022. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/encryption. [Accessed 8 6 2023].

[20] "Security Encyclopedia," [Online]. Available: https://www.hypr.com/securityencyclopedia/decryption#:~:text=Decryption%20is%20a%20process%20that,is%20being%20shared%20or%20transmitted.. [Accessed 8 6 2023].

[21] K. Richards, "techtarget," September 2021. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/cryptography. [Accessed 8 6 2023].