GEORGIA STATE UNIVERSITY
CSC 8224 – CRYPTOGRAPHY
SPRING 2022

PROJECT REPORT
Protecting Text on Steganography

Team Members

Gollangi Someswara Rao

Gaayathri Vaidhyanathan Pazhambalacode

Abstract

Steganography is a widely used approach in security aspect for transferring confidential information from one person to another. This not just encodes data but also makes sure that the hidden data is not visible to a third-party that is not authorized to have insight of it. To perform these, there are several open-source software available online. The issue with this is that, these open-source software are accessible by third-party as well. Therefore, our proposed project has a solution to this problem. It is an authentication site where steganographic encryption and steganographic decryption is performed. This site contains a two-factor authentication user for each registered user within the organization thereby making this site secure and only accessible to the users registered with it. Only user within the respective organization can register for it and every time the user login, the user goes through two-factor authentication to enhance security further. Both the sender and receiver must be registered users.

1. Introduction

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection. In other words, steganography hides a secret message inside an innocent-looking cover medium, drawing no attention to itself. The cover medium can be text, image, music, video, network packets. The content to be concealed through steganography -- called *hidden text* -- is often encrypted before being incorporated into the innocuous-seeming *cover text* file or data stream. If not encrypted, the hidden text is commonly processed in some way in order to increase the difficulty of detecting the secret content. This is then hidden using various steganographic algorithms within a cover medium such that it is not visible to naked eyes/third party people/attackers. The concealed cover medium (Stego Object) is sent to the receiver using some secure channel, who will decode this hidden data from the cover medium using the decoding algorithm and finally retrieve the hidden text.

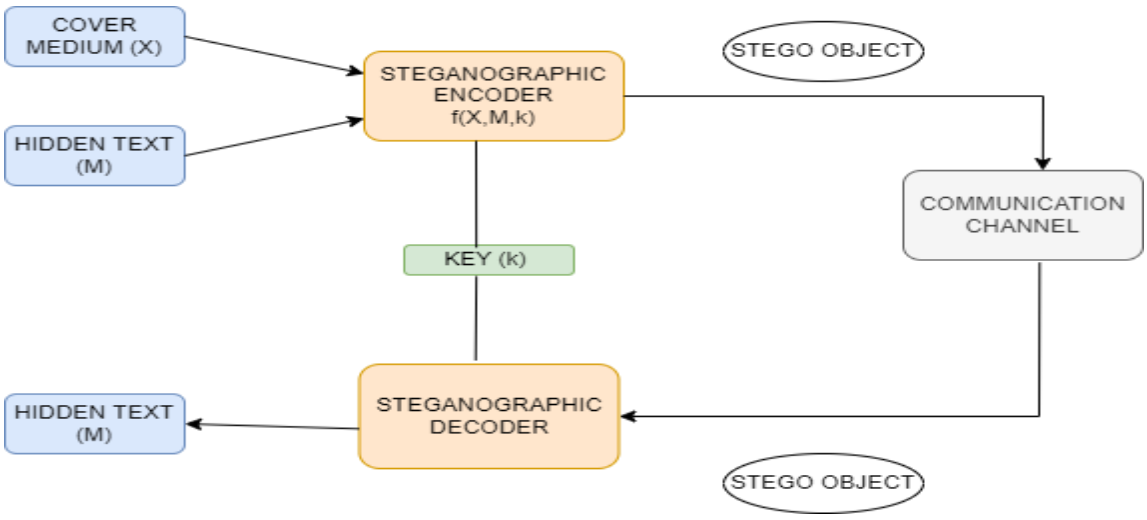

Fig 1: Steganography flow

2. Image Steganography

While there are many different uses of steganography, including embedding sensitive information into file types, one of the most common techniques is to embed a text into an image file. Hiding the data by taking the cover object as the image is known as image steganography. One technique is to hide data in bits that represent the same color pixels repeated in a row in an image file. By applying the encrypted data to this redundant data in some inconspicuous way, the result will be an image file that appears identical to the original image but that has "noise" patterns of regular, unencrypted data.

In digital steganography, images are widely used cover source because there are a huge number of bits present in the digital representation of an image. There are a lot of ways to hide information inside an image. Some common approaches include, Least Significant Bit Insertion, Masking and Filtering, Redundant Pattern Encoding, Encrypt and Scatter, and Coding and Cosine Transformation.

3. Related Work/Existing Approaches

Steganography software is used to perform a variety of functions in order to hide data, including encoding the data in order to prepare it to be hidden inside another file, keeping track of which bits of the cover text file contain hidden data, encrypting the data to be hidden and extracting hidden data by its intended recipient.

There is proprietary as well as open source and other free-to-use programs available for doing steganography. OpenStego is an open-source steganography program; other programs can be characterized by the types of data that can be hidden as well as what types of files that data can be hidden inside. Some online steganography software tools include Xiao Steganography, used to hide secret files in BMP images or WAV files; Image Steganography, a Javascript tool that hides images inside other image files; and Crypture, a command line tool that is used to perform steganography.

4. Problem Formulation

There are a few issues on steganography sites, one major issue of which is the security of the page. There usually is no security or authentication mechanism for encrypted files in these sites. So, if a third-party person or someone other than the sender and receiver gets access to the information, they can decode it easily.

To overcome this, we propose developing a login site for our application to address this issue with certain level of authentication techniques involved. This is something which only the registered users would be able to access then. This way only the sender and receiver may only read the message hidden in the steganography text or image without any fear of someone else being able to get a hold on the information.

5. Objectives/Design

The major objectives or functionalities to cover in this project are as follows. A desktop application has to be built for creating an authentication mechanism for steganography. There are three interfaces in this application: Registration, Login, Steganographic encryption and Steganographic decryption.

The first step a user/client has to take before connecting to the application is to register themselves into this application. This also requires them to enter a pin as well as a form of two-factor/multi-factor authentication along with their email id and password. After logging in, the user can create content, encode it, and share it with the recipient of the secret message. Similarly, at the recipient's end, the user has to login using multi-factor authentication, and decode the content from the file shared.

The main objectives of the proposed project include:

- Creating a more secured approach for the steganography.
- Creating and linking Two factor authentication.
- Having a smooth process for the user

6. Implementation

There are several modules within this implementation to build a secure portal to perform steganographic encryption and steganographic decryption procedures. The modules to name are, Register module, Login module, Steganographic Encryption module, and Steganographic Decryption module. Each of them functions as discussed below.

This entire application is built using Flask framework in python. It is a framework "is a code library that makes a developer's life easier when building reliable, scalable, and maintainable web applications" by providing reusable code or extensions for common operations.

One major advantage of using python is the vast amount of packages and libraries it contains to encrypt and decrypt data and before secure operations within the applications.

The flow of the application goes as follows:

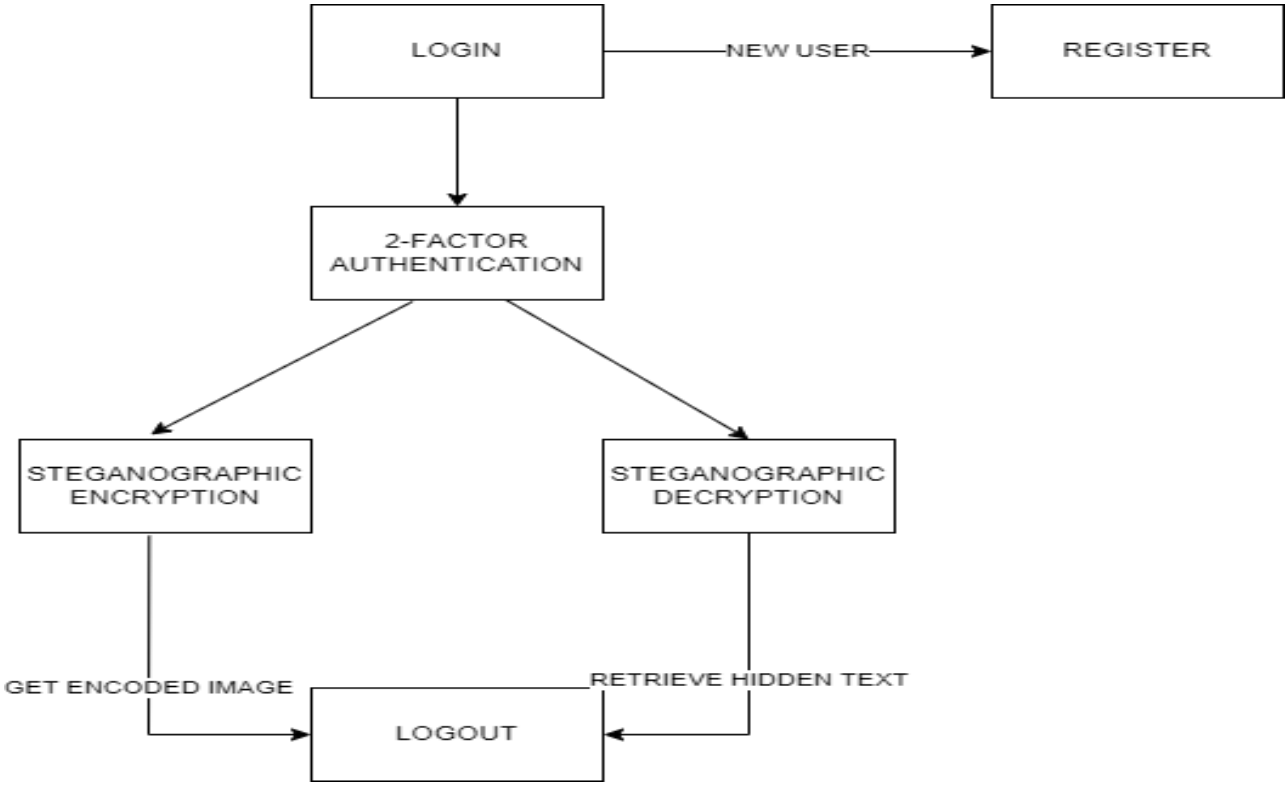

Fig 2: Flow Chart of the web application

6.1 Register Module

This is the primary module that a user/client has to sign up for, in order to access this web application and perform secure steganographic operations.

To make use of the application, the user needs to first create an account. They need to sign up to the application. For designing this, the user is by default set to the user role only if they are from the management the role is set to admin or PM based on which criteria they fall on. While signing up, it is mandatory for the user to have a organizational official email id (something similar to @gsu.edu in Georgia State University). This stands as a verification that the user belongs to the corresponding organization to avoid third-party/attackers to be able to register to the application thereby making it a bit more secure.

Along with this, the user has to set the password. During the sign-up process, encryption of the password takes place which along with the organizational email id acts as a medium for first-level of validation for authenticating the user at later stage. With completion and submission of this, these data are stored in the database as the user's credentials and details and marks the completion of creation of the account.

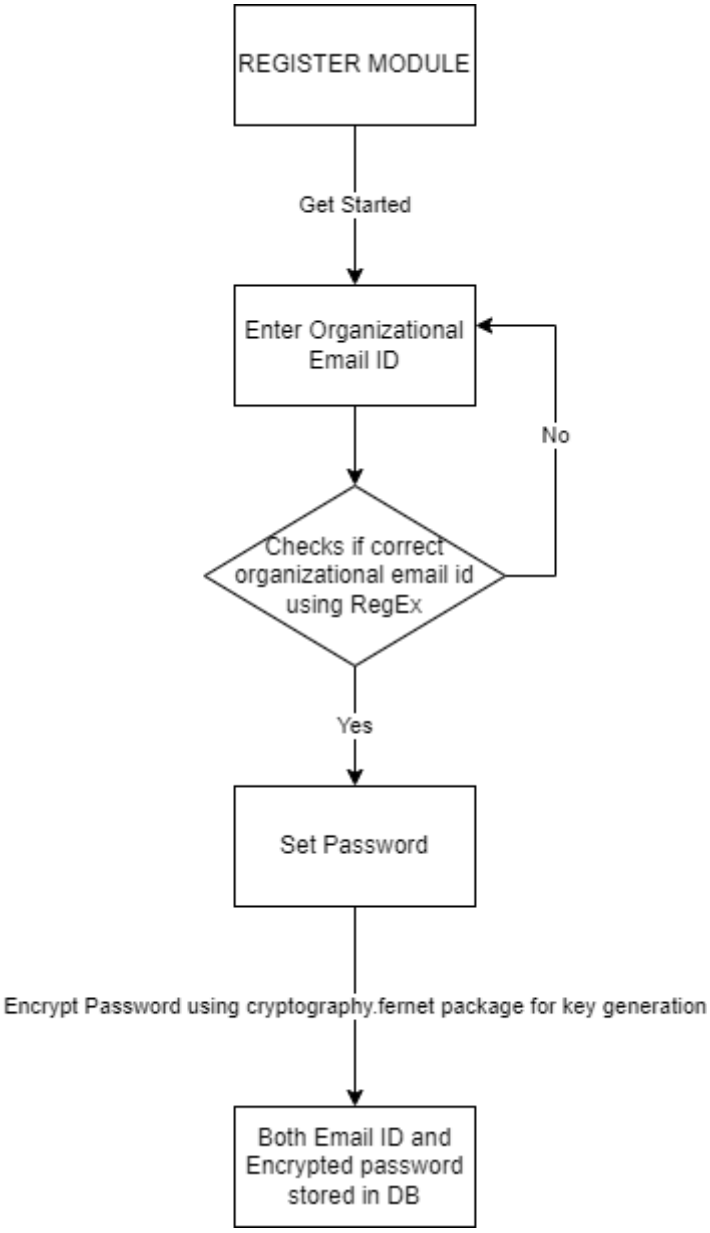

Fig 3: Register module flow/functionality

This way, the email id and encrypted password is stored within the database. Flask contains a library called flask_sqlalchamy that allows us to use our desired database as long as the correct URL to the location and credentials are provided. In our case, Postgresql is used. Below (fig 4) is the picture of the register page in the application.

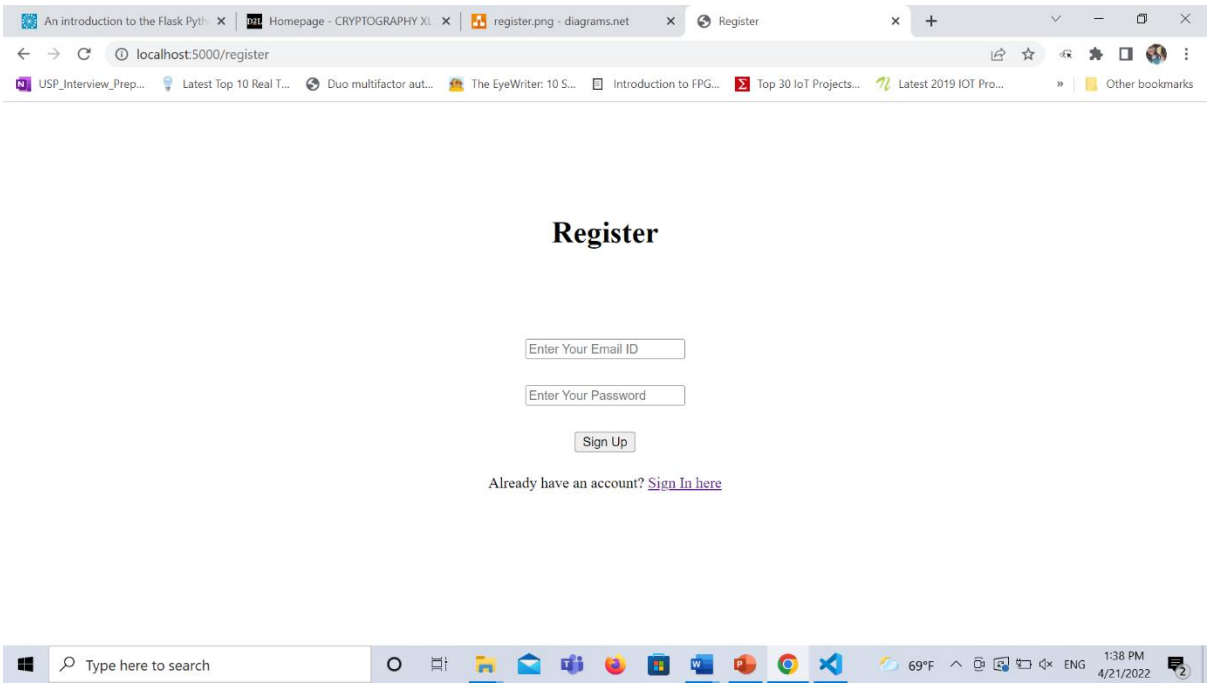

Fig 4: Register Page in the web application

6.2 Login Module

Once the account of the user is created, these credentials are stored in the database. After that, whenever the user logins to the application, using the above given credentials, the user is validated.

This authentication is performed using the organizational email id during the sign-up process. After this, the password entered by the user is cross-verified with the encrypted password stored in the database. Once this match, the user is taken for two-factor authentication. Here, a pin is sent to the organizational email entered which has a time factor of 10 minutes. Within this duration, this pin has to be entered in the page prompted by the application. Once this matches with the pin sent via email id, the user has access to the web application.

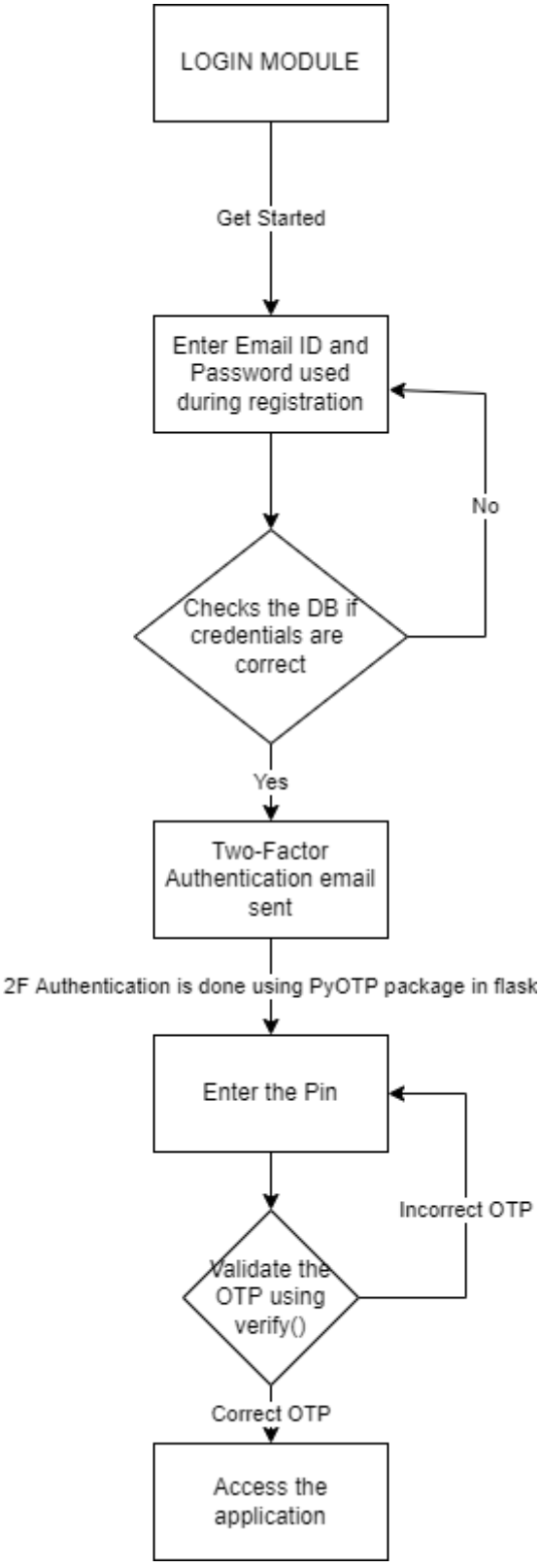

Fig 5: Login module flow/functionality

All of these functionalities are implemented using various libraries in python. To perform two-factor authentication, PyOTP package/library was used. This has the ability to generate random OTPs in hex format using the random_hex() method. This is then sent to the email provided while registering. Once the user enters this pin in the space prompted by the application, the verify(otp) method from PyOTP is used to validate the pin and thereby complete the two-factor authentication.

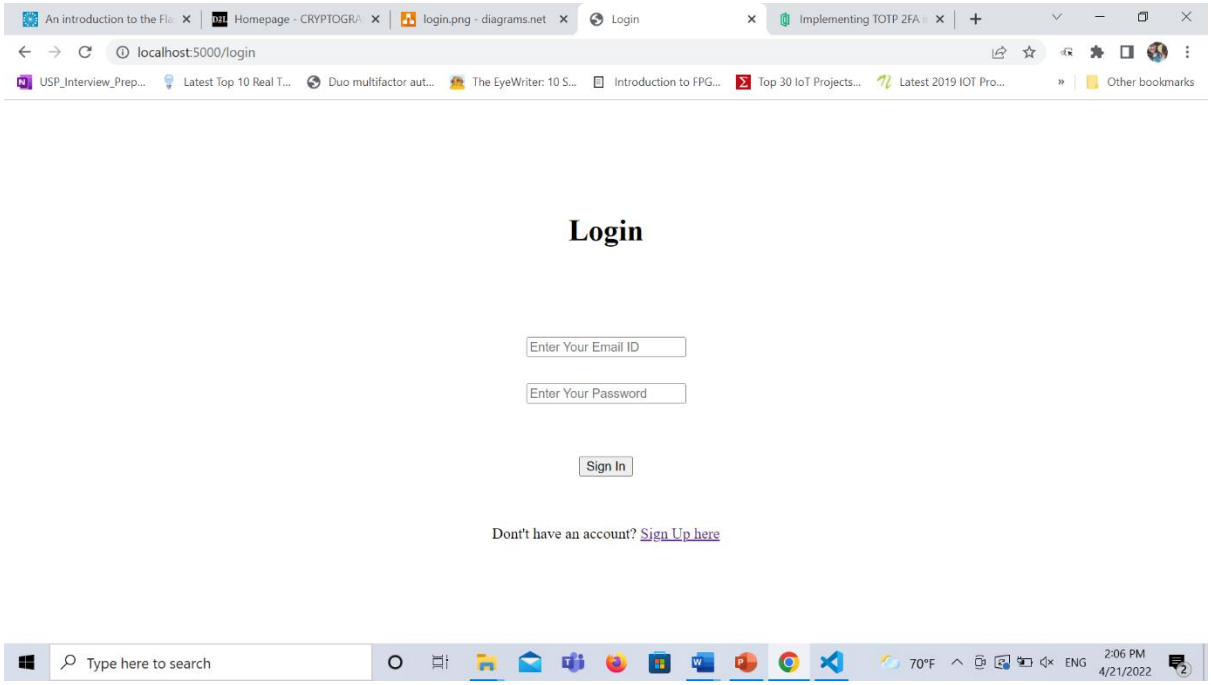

Fig 6: Login page in the web application

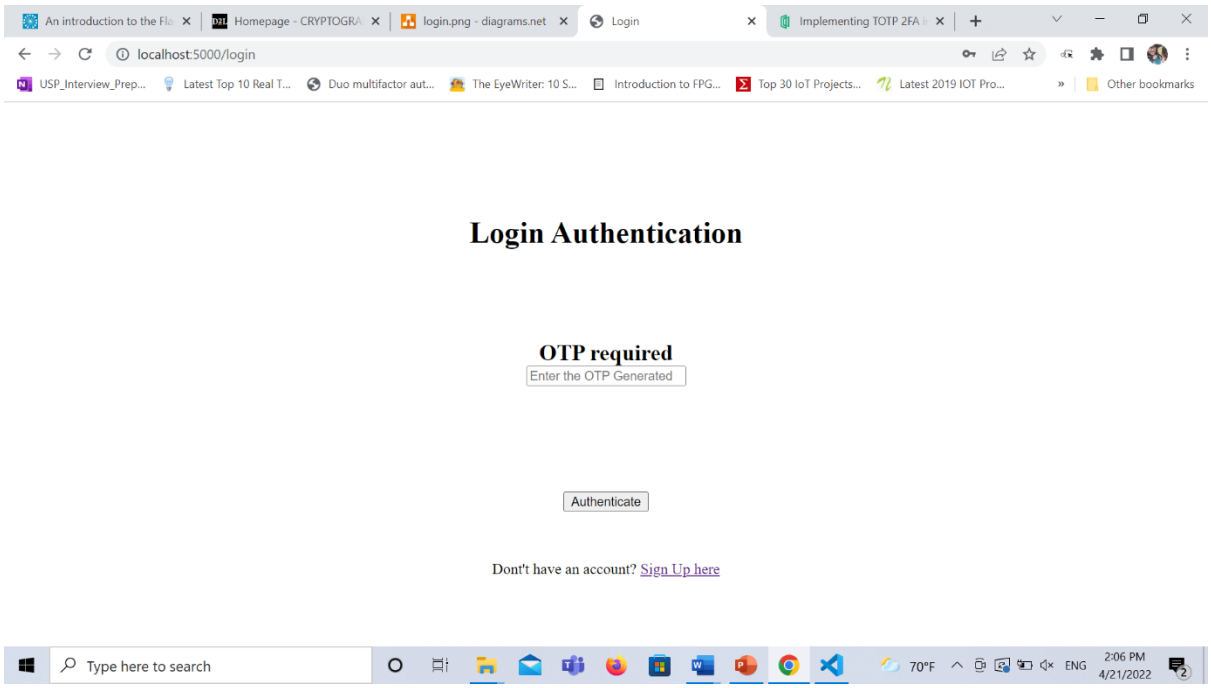

Fig 7: Two-factor authentication page in the web application

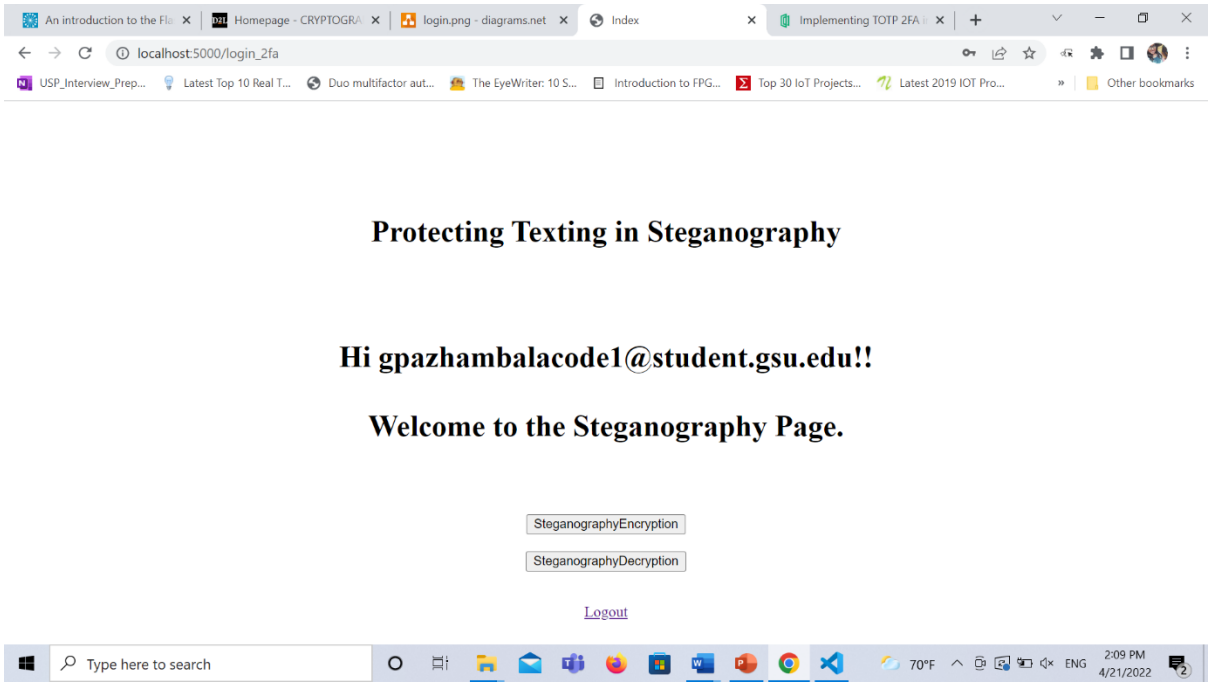

Fig 8: The Main page indicating the access granted to the user and displaying the steganographic operations available in the application

6.3 Steganographic Encryption Module

There are several steganographic algorithms available for image steganography. The algorithm implemented in this module is the Least Significant Bit (LSB) Steganography. In this type of steganography, the information hider embeds the secret information in the least significant bits of a media file. Since modifying the last bit of the pixel value doesn't result in a visually perceptible change to the picture, a person viewing the original and the steganographically modified images won't be able to tell the difference.

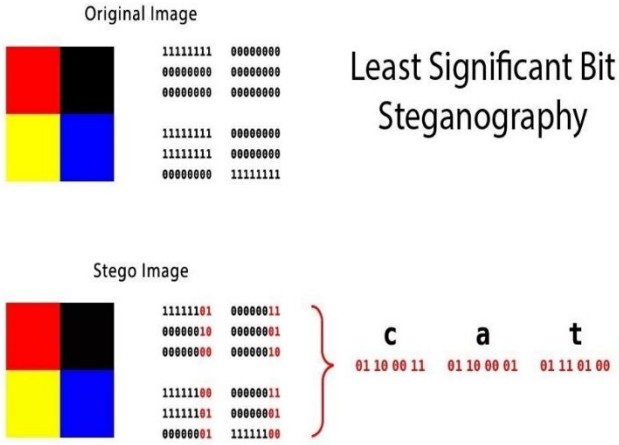

Fig 9: LSB Steganographic Approach

Here, when the user goes to the steganography encryption page, the application prompts the user to enter the text to be hidden within the file and to also upload an image file along with it. Once the upload button in clicked, the text is converted to binary format and replaces the LSB of the RGB pixels in the image. The operation is as follows.

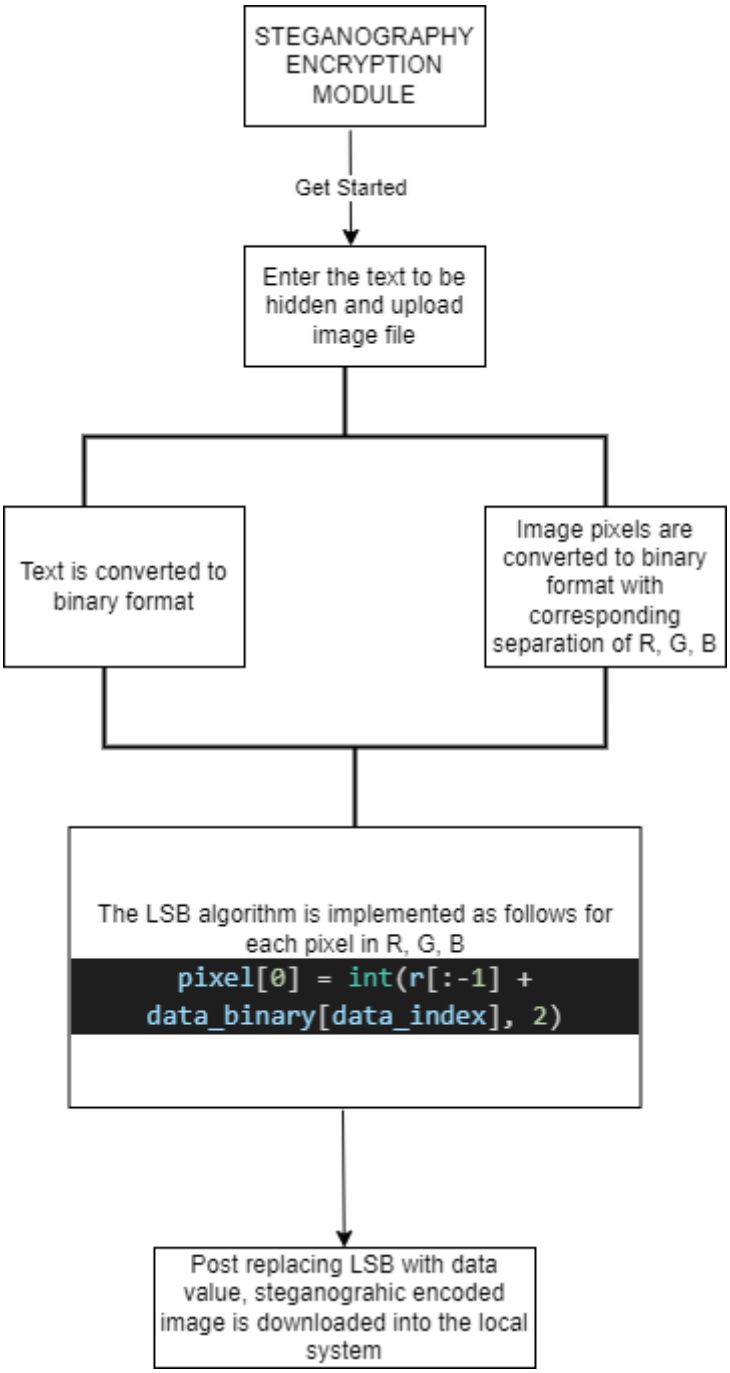

Fig 10: Steganographic encryption procedure

Since only the least significant bit in the pixel is altered, there is no major change in the image that can be caught to the visible eyes. This can be observed in Fig 12 where both the images before and after steganographic encryption can be observed. This way, even if the attacker happens to obtain the image, they will not be able to figure out whether it contains any hidden form of information within it.

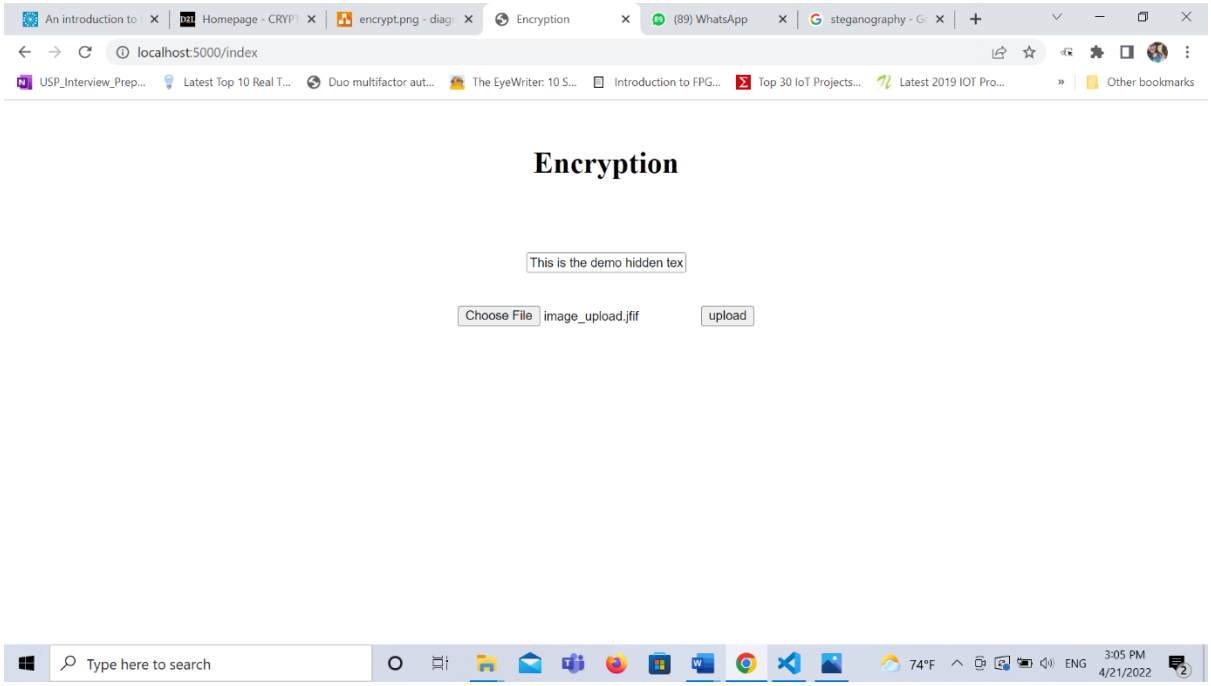

Fig 11: Steganographic encryption page in the web application

Fig 12a: Image before steganography

Fig 12b: Image after steganography

This way steganographic encryption is performed and the encoded image is stored in the local device with the name encoded_img.png. This can then be sent to the receiver using several available channels such as emailing, social media, etc.

6.4 Steganographic Decryption Module

The receiver who has the encoded image file now has to login to the same portal/application used for steganographic encryption. The decoding procedure/algorithm is exactly the reverse of the encoding operation. Here the LSB of each pixel in each R, G, B is extracted and converted binary to integer and thereby to character format to finally retrieve the hidden text. All these bits are combined and the hidden information is retrieved in the end which is displayed in the application.

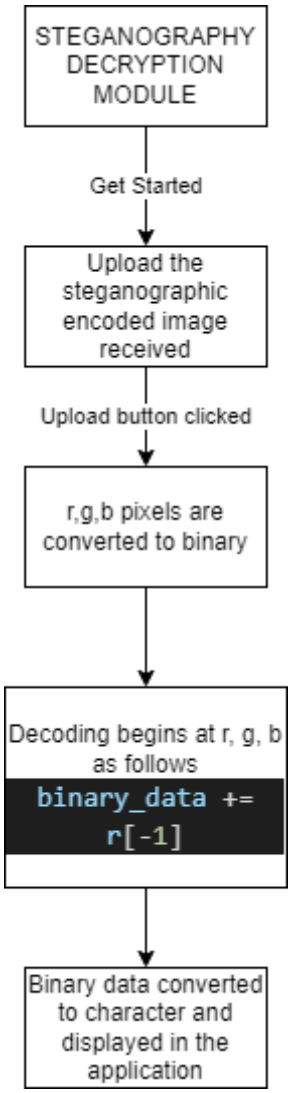

Fig 13: Steganorgaphic decryption procedure

This decoded text is displayed in the portal at the receiver's end.

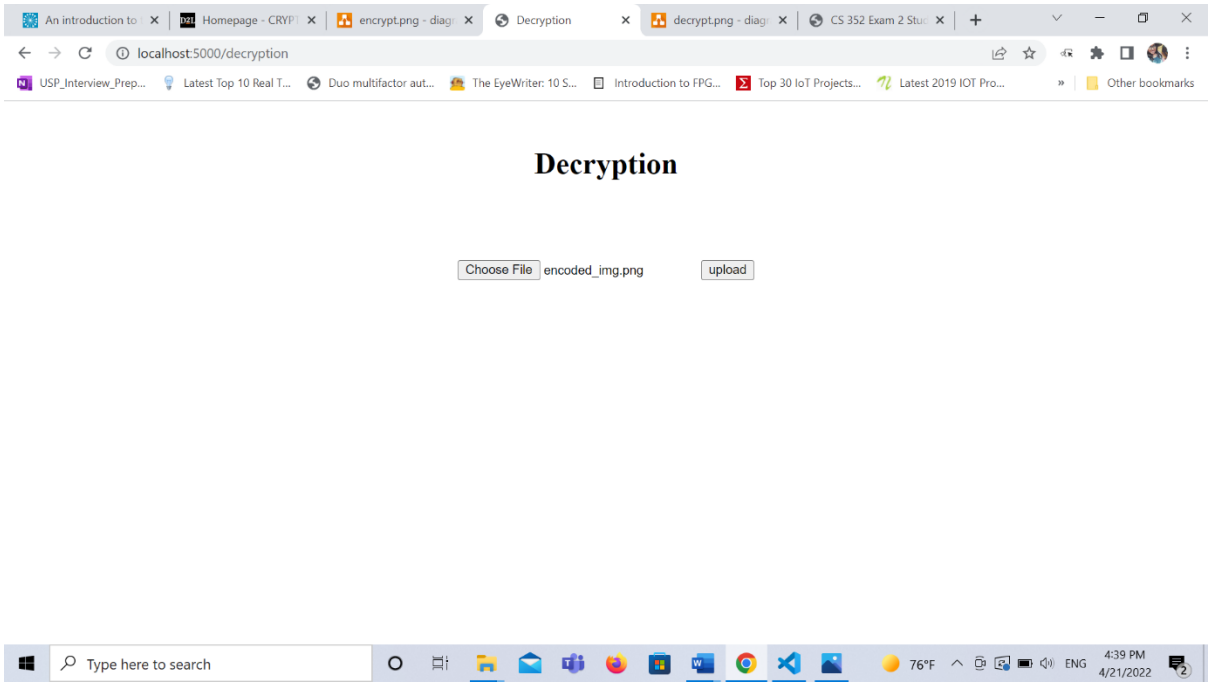

Fig 14: Steganographic decryption page in the application

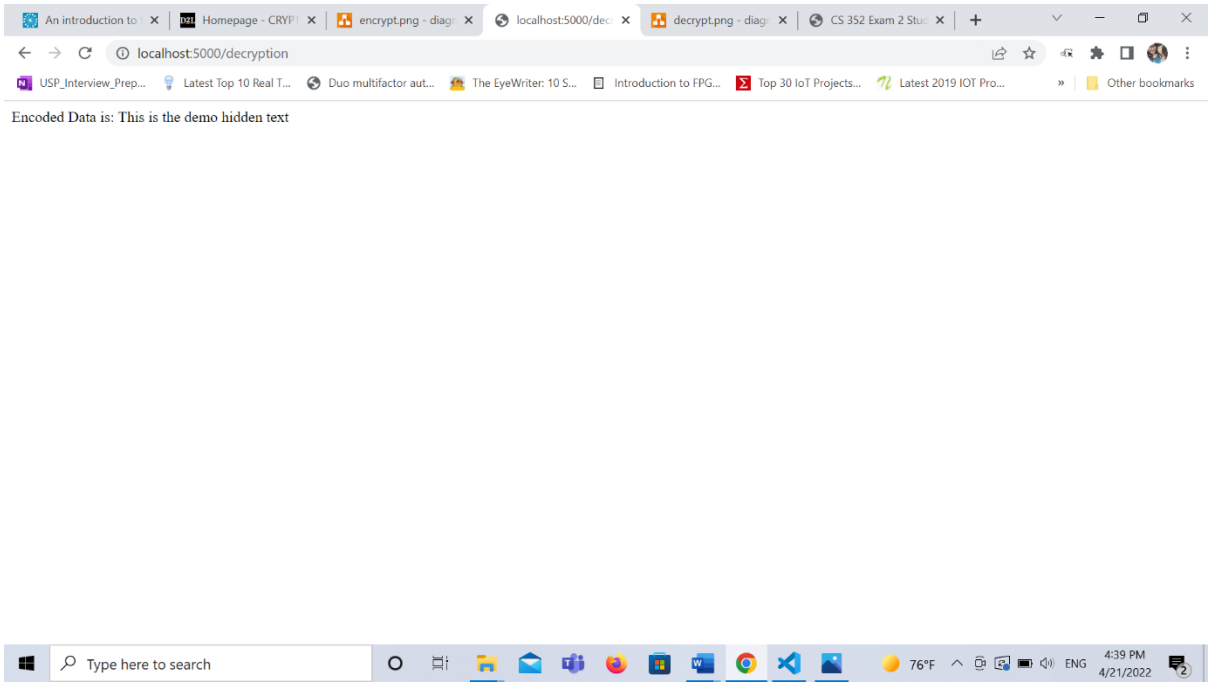

Fig 15: Hidden information/text is displayed to the receiver in the application

7. Conclusion

The project, by developing a secure strategy for the steganography website using Two-Factor authentication and a user portal environment, provides a simple and dependable environment for them to safely trade data without worry of it being disclosed.

This way, within an organization, confidential information can be delivered from one person to another is a secure way without a third-party person or attacker getting a hint of the information. Even if by any chance, the image is obtained, there is no way the attacker would be able to decode the message as they are not registered users who have access to the application.

8. References

[1] Agarwal, Monika. "Text steganographic approaches: a comparison." *arXiv preprint arXiv:1302.2718* (2013).

[2] Petitcolas, Fabien AP, Ross J. Anderson, and Markus G. Kuhn. "Information hiding-a survey." *Proceedings of the IEEE* 87.7 (1999): 1062-1078.

[3] Morkel, Tayana, Jan HP Eloff, and Martin S. Olivier. "An overview of image steganography." *ISSA*. Vol. 1. No. 2. 2005.

[4] https://www.geeksforgeeks.org/image-steganography-in-cryptography/

[5] Gupta, Shailender, Ankur Goyal, and Bharat Bhushan. "Information hiding using least significant bit steganography and cryptography." *International Journal of Modern Education and Computer Science* 4.6 (2012): 27.