DATA STEGANOGRAPHY

A Report submitted

by

Shivansh Gupta (170110010)
Nishnk Vishi (170110047)
Aditya Parashar (170110054)
Aditya Arun (170110055)

Under the Guidance of
Mayank Sah
Assistant Professor
Department of Computer Science

In partial fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING,
DIT UNIVERSITY, DEHRADUN

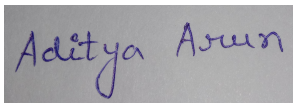
(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)

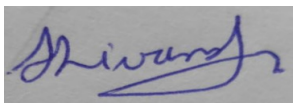
Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

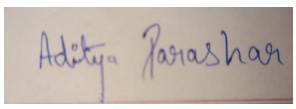
Declaration

We hereby certify that the work, which is being presented in the report/ project report, entitled **Data Steganography**, in partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology and submitted to the university is an authentic record of our own work carried out during the period August, 2020 to December, 2020 under the supervision of **Mr. Mayank Sah**, Department of Computer Science and Engineering, DIT University, Dehradun.

Date: 03/11/2020 **(Signature of Candidates)**







(Signature of Mentor)

Acknowledgement

First of all, with deep sense of gratitude We would like to thank our project supervisor Mr. Mayank Sah for providing us with his invaluable guidance and supervision at each step in carrying out this project throughout the period. It would have been impossible for me to progress in the project without the regular discussions with him and his worthy feedback. We would like to thank Dr.Vishal Bharti, HOD(CSE), DIT University, Dehradun for his encouragement and cooperation.

Aditya Arun
Aditya Parashar
Shivansh Gupta
Nishnk Vishi

Abstract

Steganography is the technique of concealing the fact that communication is taking place, by hiding information in other knowledge. we have a variety of carrier file formats to choose from, but choosing the images is the most viable option as they are flooded on the Internet. Information hiding in the image can be done in a variety of ways and each of them has their advantage and disadvantages. The requirement is specific to the applications and so is the steganography technique.

Some application may need total concealment of data other may require concealment of bulk data. This project hides the message within the image bytes using various algorithms like LSB and XOR approach, Neural Style transfer and Deep Steganography. The sender selects the cover image with the secret text or text/image file and hides it into the image with the bit replacement choice, it helps to generate the secure stego image. The stego image is then saved to the destination with the help of a private or public communication network. on the other side i.e. receiver. The receiver downloads the stego image and using the software retrieve the secret text hidden in the stego image.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose

The study of hiding messages and embedding it in a carrier/medium is called Steganography. Steganography is related to cryptography but both of them are different. Ancient Greeks used it to hide knowledge about troop movements by tattooing the details on someone's head and after that the person grows up their hair.
Steganography aims at hiding the messages and data. A cover file is always required in the steganography where a data is stored in the form of messages which will be protected by the algorithm that decides how to protect the data, and a key (optional) that will be used to randomize the placement of the data and perhaps even encrypt it.
Steganography is also known as covered writing is the science of hiding information "in plain sight".
The main difference between cryptography and steganography is that the goal of steganography is to completely perplex the existence of information rather than hiding its content. Currently, the most frequent usage of steganography is to hide one computer file inside of another computer file.

1.2 Objective

The objective of steganography is to conceive a secret data inside a cover-media in such a way that others cannot discern the presence of the hidden message. Technically

in simple words “steganography means hiding one piece of data within another”. The objective is to keep data safe from the third party so that they cannot decode or understand the message behind the image or any media .

1.3 Motivation

The opportunity to learn about a new area of computer science that is Data Security, which was not covered in lectures was appealing. We knew little bit of python, so to implement steganography through python was interesting. This project will give us a depth knowledge about how to secure data from various method , which will help us to explore various new research field.

1.4 Definition and Overview

Steganography is like the sibling of cryptography. While steganography is the technique to provide secrecy whereas cryptography is the technique to provide privacy. The method of stealthy transmitting is steganography. While taking the example of an image or a text file steganography is ta method that moves about obscuring the data in those files. The data is transmitted to the receiver without getting acknowledged by anyone by the sender over the communications channel. This procedure can be used by organizations to communicate their messages to the outside world without getting acknowledged by anyone. Here we have tried to perform the different approaches towards implementation of Steganography using any file (text or image).
Steganalysis is a fairly new section of data processing through which steganographic

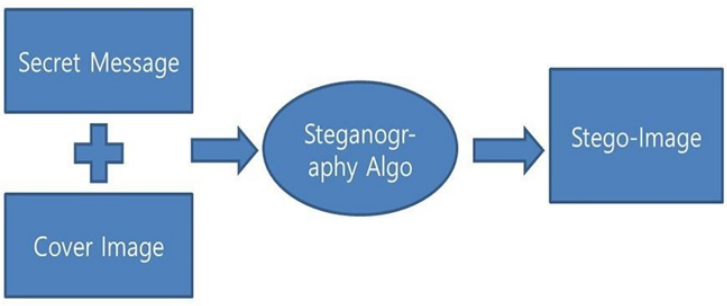


Figure 1.1: Process of Steganography

covers can be revealed, and if the possible messages can also be retrieved. This pro-

cess shares a lot of similarity with the cryptanalysis in cryptography. It is an ancient technique that has resurfaced itself in the modern world and is gaining popularity in the field of communication security. Now concealing the message is not the only prime concern but also to conceal the existence of the message.

- **Passive steganalysis:** Detect the absence or presence of a concealed message in an observed message identifier the type of embedding algorithm.
- **Active steganalysis:** Estimate/extract some properties of the message or the embedding algorithm.

For example, extract a (possibly approximate) version of the secret message from a stego message. We at first compare the algorithm with different sizes of images. We will be comparing, that which will take more time to execute. Through this we make our own algorithm.

Steganography is hiding of data inside some other data. Steganography is a data hiding method which can be used parallely with the cryptography to add an extra security layer. We can apply the technique of Steganography to video file or over an image file even over the audio file. Its usage in the image is so common as images are floated over the internet in abundant even though steganography is the written in the form of characters and the hash marking. Hence we apply this technique to protect the copying the data as an unauthorized viewing.

Chapter 2

Overall Description

2.1 Project Perspective

The perspective of the project is to make the algorithm, better than the one on which we are working. We will check or analyze the time required in different image sizes, with same code. Steganography is intended to provide secrecy. It will not only prevent the message being read by anyone also it will cover the existence of the message which frame the main objective of our project. The major perspective is to make the communication more safe and secret as steganography majorly works on secrecy.

2.2 Project Functions

The purpose of steganography is to convert communication to hide a message from an external party. Because of the art of secret writing its is different from cryptography, which is intended to make a message unreadable by a third party but does not hide the existence of the secret communication. This project will give the algorithm with less time complexity. In steganography there are many algorithm used for different media for hiding secret message and that message can only be decode by the second party .

2.3 Overview of our project

Created a Chat Application known as "Securedkhabri", whose aim is to provide a facility for chatting with different members by creating a room in the application without the security concern. The main aim of the chat application is that a user does not risk his/her privacy with other users(third party). It has some special features like Image steganography where user can hide another image or message in the cover image with the help of Neural style or Deep stag as an option for a user.

Using “digital image” which is also referred to “raster graphics”,which is a dot matrix data structure, represents a grid of pixels.This can help us to store in file of image with different type of formats.

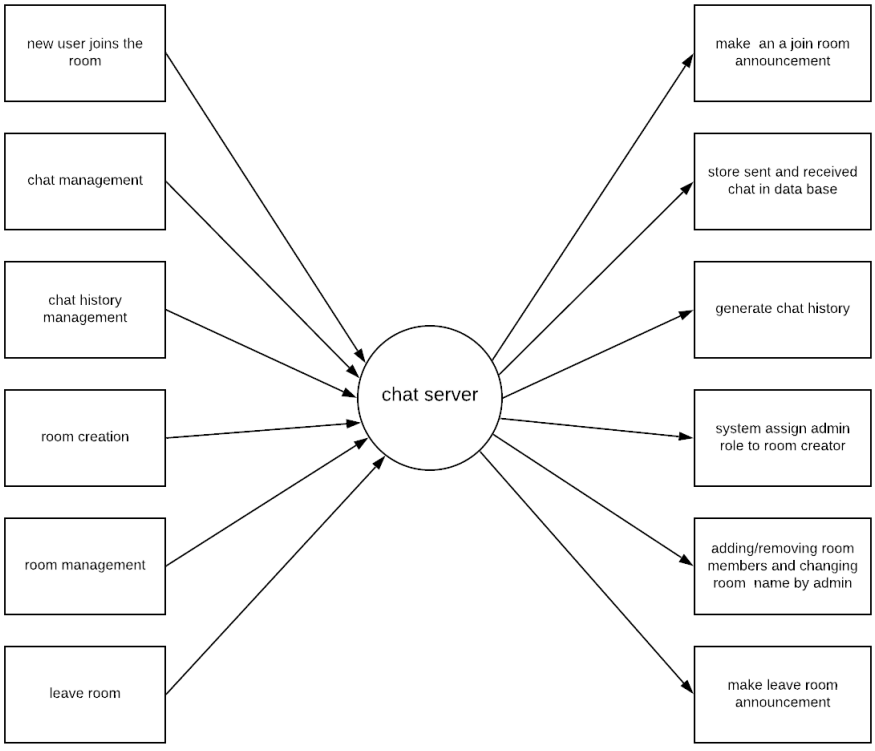


Figure 2.1: DFD of the chat app

2.4 Constraints and Assumptions

The assumption made in this project is that both the receiver and the sender have already shared some confidential data prior to the commencement of communication. Steganography in its real form signifies that no prior data is exchanged by two communication units. Steganography is limited by the same assumption that exists for encryption.

Information constraint should be put on the system otherwise significant amount of time will be consumed by the processes to take place. To solve the problem with the large data the feature to override the limit must be provided.

The following assumptions have been made:

- There can be only one admin in the chat group that is the creator of the group.

The following dependencies have been made:

- An assumption is made by the system design that there is absence any third party dependency and only standard host device and runtime are present.

Chapter 3

System Requirements

3.1 External Interface Requirement

Simple Mail transport Protocol(SMTP) will be used by our application for sending and receiving the texts and images after being steganographic. Database management (MongoDB) will be used for the user authentication, user database. Cloudflare protection will be used for firewall protection and DDoS attack. MongoDB database which we have choosen is cloud based.
There would be two secured external interfaces of our application one is database management system for storing the content in the app and the other is SMTP which is used for sending/receiving the content in the room to other users in the same room.

- Mail Transport Agents:
  The Sending/Receiving of content in our chat app is accomplished by using SMTP(Simple Mail Transfer Protocol)server which will work on port 25 is used. this library have support of POP3 and IMAP protocol. the configuration of this will be stored in non-relational database and will help user for sending/receiving of the content.

- Database:
  To add higher scalability, low latency and flexible schema it can process unstructured data, semi-structured and structured data. Complex relationships is reduced in this unlike RDBMS.

3.1.1 Hardware Interface

As their is not such a hardware requirement as app will be platform independent user only need a good internet connection and a web browser. only the specification given by browser will be sufficient and it will not directly communicate with hardware port.

3.1.2 Software Interface

- Internal Interfaces:
  Our Application is platform independent and work in cloud environment so the user can be a Linux/Mac/Windows it is not a big deal for us. User should have a good internet connection and web browser(Chrome/Firefox preferable).

- External Interfaces:
  A good internet connection is necessary to work on our application(atleast 25mbps) to interact in our application i.e. sending/receiving the content or to decode or encode the message in his/her private chat room. All the message will be save in the database(MongoDB) for security concern and storing the history of the user.

3.2 Functional Requirement

Base System Requirements:

- User can encode the message using a variety of options to hide the information within the carrier image
- User can decode the received message using the steganographic key to get the information.
- The app will provide the support to send the encoded steganographic images to other users if user authenticates and settings are enabled.
- The app will provide the support to receive and decode information if the option has been enabled if user authenticates and settings are enabled.
- The app will enable the user to choose between the algorithms to use and give user a choice for the level of security he/she wants.

3.3 Non-Functional Requirement

We tried our application should be easy to use that one who is new to our app can easily understand the UI and use. Our app is so interactive that any new/advance user will feel comfortable to use without any problem. Our app can be used by the user who is exploring the field of steganography that how it is used.

3.4 Hardware and Performance Requirement

In order to improve the performance take less hardware control. our application is fully work on cloud/internet.So a good internet is mandatory. Further improvement can be achieved by disabling the JavaScript in the web browser at a client side which will consume less internet.
The following performance characteristics were identified:

- Our scheme/ technique takes 1 minute to encode 150-200 words in the carrier image.
- Application launching time is dependent on the speed of the internet.
- Our scheme/technique takes 1 minute to decode the message (100-200) from the carrier image if correct key and image is used.

Chapter 4

System Features

4.1 Algorithms

4.1.1 LSB-XOR Algorithm

In this algorithm we initially read the secret file byte by byte i.e. one byte at a time. These individual chunk of bytes are then XOR-ed with the key which is entered by the user.The key is then rotated and the individual XOR-ed chunks are added together.We then traverse the cover image pixel by pixel and replace the last n bits(Least Significant Bits) of the specified spectrum(R,G,B or A) and replace it by the N bits of the XOR-ed data.So in this way we store the raw XOR-ed data in the specified channels of the consecutive pixel and since we are only altering the least significant bits of the pixel of the cover image so there is no tangible change in the appearance of the steganafied image.

While decoding the image we simply read last N bytes of the least significant bits of the specified spectrum of each pixel one by one. We then group these small individual chunks from the last N bits of the pixel into to a bigger chunk of data and again read it byte by byte and store it temporarily.Now this temporary byte is then again XOR-ed with the rotating key and the resultant bytes are appended together to obtain the original file.

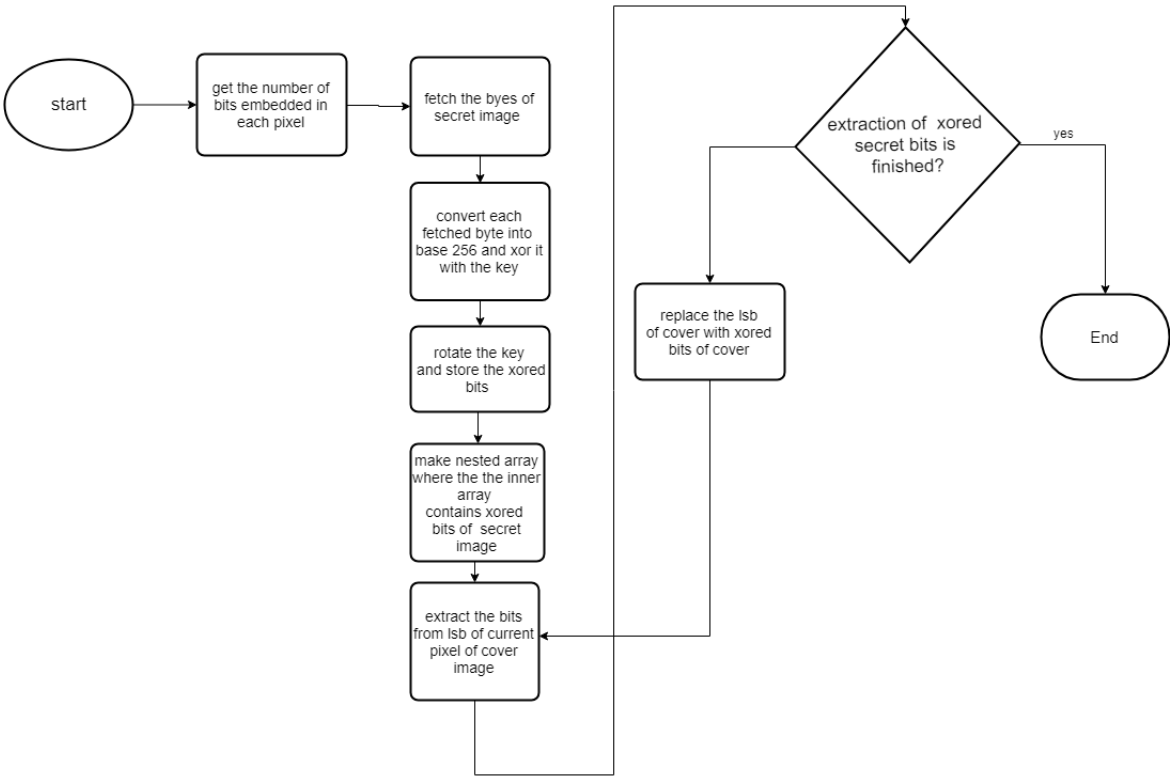


Figure 4.1: Flow Chart for Encoding

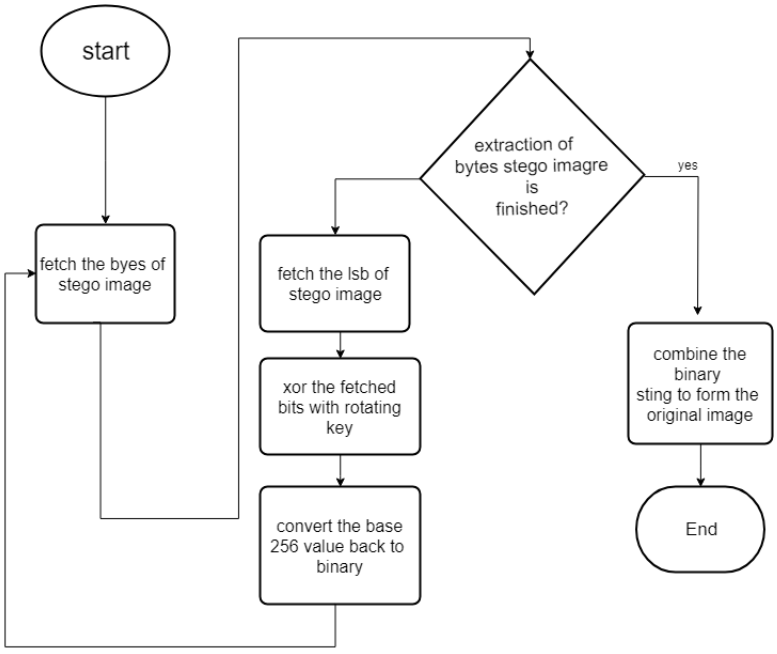


Figure 4.2: Flow Chart for Decoding

4.1.2 Neural Style Transfer

Neural style transfer is an optimization technique used to take two images—a content image and a style reference image and combining the content of one image with the style of another image using convolutional neural networks(CNN). Here VGG-19 network architecture is used. It is pretrained image classification network.

In this approach, the intermediate layers of the model are used to get the content and style representation of the image. Low-level feature like edges and texture are represented by first activation layers and high-level features-object parts like tyres or ears, eyes are represented by the last activation layers of the network.

The network understands the images as it is made for classification of images. This is done by taking the raw image as input pixels and then convert into a understanding of the features present in the image. CNN are able to capture generalize i.e. they can capture the invariances and define the features within classes. So it can be interpreted that in the intermediate layers of the model, we can describe the content and style of input images, as the model serves as a complex feature extractor for a given raw image and the output classification label.

Figure 4.3: Neural Style Transfer

Architecture and Implementation

The main intuition of implementing neural style transfer is same idea that is common to all deep learning algorithms. First, we define a loss function to specify according to the desired output, and then minimize this loss. Since, we want to conserve the "content" of the original image, while adopting the "style" of the reference image, we were able to mathematically define content and style. The an appropriate loss

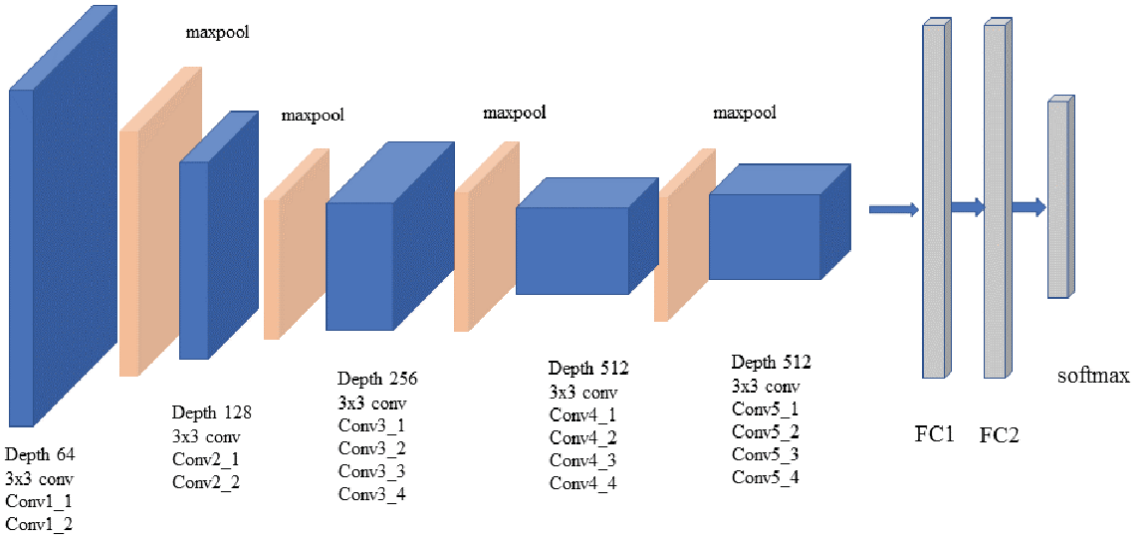


Figure 4.4: VGG-19 Architecture

function to minimize is given below:

$$loss = distance(style(reference\_image) - style(generated\_image)) + distance(content(original\_image) - content(generated\_image)) \quad (4.1)$$

where *distance* is a norm function, *content* is a function that takes an image and computes a representation of its "content", and *style* is a function that takes an image and computes a representation of its "style".

Minimizing this loss would cause $style(generated\_image)$ to be close to $style(reference\_image)$, while $content(generated\_image)$ would be close to $content(generated\_image)$, thus achieving style transfer as we defined it.

4.1.3 Steganography using Deep Learning(DeepSteg)

In this algorithm, we have used the power of deep learning to hide the images and implement steganography. The algorithm used here is inspired by the research paper entitled "Hiding Images in Plain Sight: Deep Steganography" published in 2017. The goal of the paper is to hide a full $N \times N \times RGB$ secret image in another $N \times N \times RGB$ secret image with minimal distortion.

We have inspired and implemented the same approach with some changes in architecture and parameters.

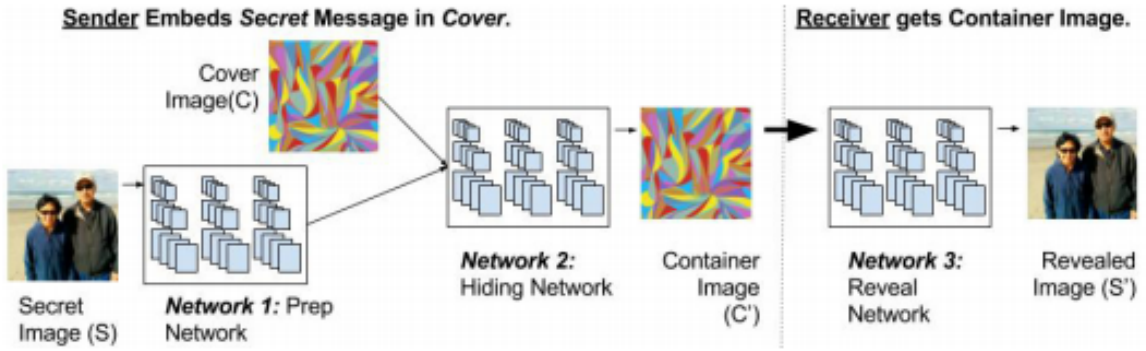


Figure 4.5: Network Architecture

Architecture and Error Propagation

In the approach, the model is inspired by the image compression through auto-encoders. The trained model learns to compress the information from the secret image into the least noticeable portions of the cover image.
The architecture comprises of three main parts:

- **Prep-network**

  The main purpose of this network is to prepare secret image of size $M \times M$ which is smaller than the size $N \times N$ of the cover image. This network increases the size of secret image to the cover image so that it can be distributed acroos the cover image.

- **Hiding Network**

  This Network takes the output of prep-network and and the cover image and creates the Container Image. It contains 5 convolution layer each with the kernel size of 3×3.

- **Reveal Network**

  Is the decoder. This Network is used by the receiver of the image. The decoder network removes the cover image from the container image to reveal the secret image.

The architecture is represented as in the figure4.7.

Since the network is inspired by auto-encoders, the two images are encoded such that the intermediate representation(container image) is appeared to be similar as possible to the cover image. The error function on which the model is trained is

```
Hide(
  (prepare): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): ReLU(inplace=True)
  )
  (hidding_1): Sequential(
    (0): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
  )
  (hidding_2): Sequential(
    (0): ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): ReLU()
    (5): Conv2d(16, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): Tanh()
  )
)
```

Figure 4.6: Hidden and Prep-Network Architecture

```
Reveal(
  (reveal): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): Conv2d(16, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (16): Tanh()
  )
)
```

Figure 4.7: Reveal Network Architecture

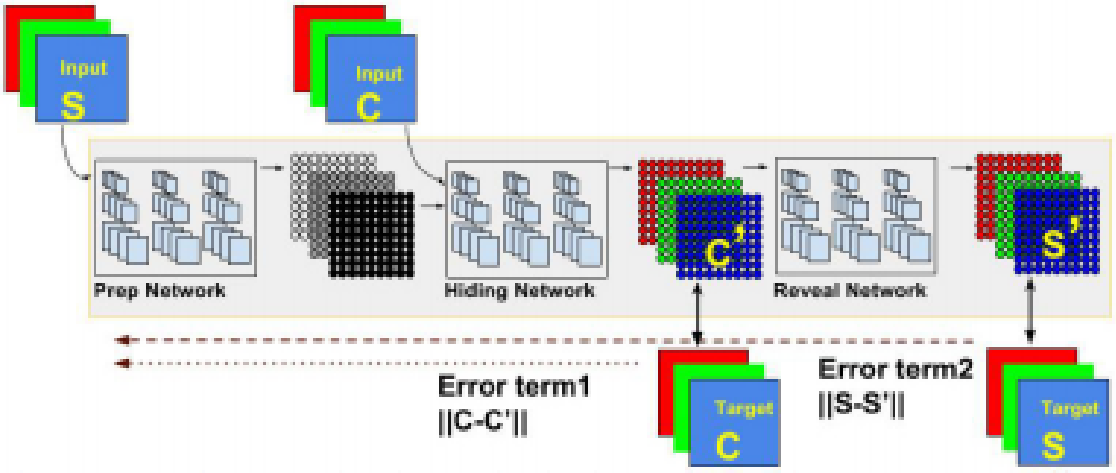


Figure 4.8: Error propagation in the architecture

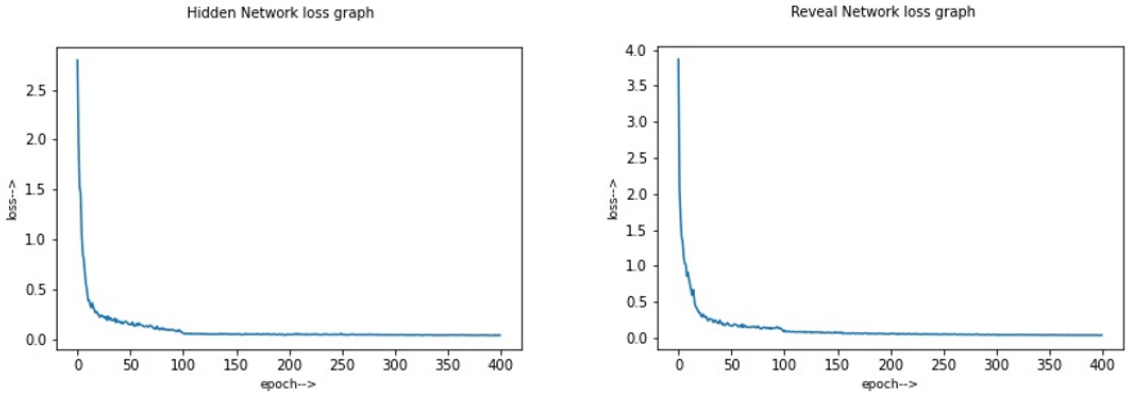


Figure 4.9: Loss Graphs for Hidden and Reveal Networks

given below. Here c and s are for cover and secret images respectively. β is used to determine how to weigh their reconstruction errors (initially according to the paper, it is set to 1.25)

$$\mathcal{L}(c, c', s, s') = \| c - c' \| + \beta \| s - s' \|$$

Note that the error term $\| c - c' \|$ only applies to the hidden network and the prep-network and on the other hand the error term $\beta \| s - s' \|$ is used for all the network for the reconstructing the image. This can be represented as in Figure 4.8. The loss graphs for both the hidden and reveal networks are shown in Figure 4.9.

Results and Evaluation

The model is trained for 400 epochs on Google Colab which provides high performance Tesla P4 GPU. The model took 7 hours to train. We track of performance of the model at certain epochs like at 20th, 50th, 100th etc epochs. The results are shown

in the Figure 4.10.

It can be observed that our model improved its performance over the training as the number of epochs increased.

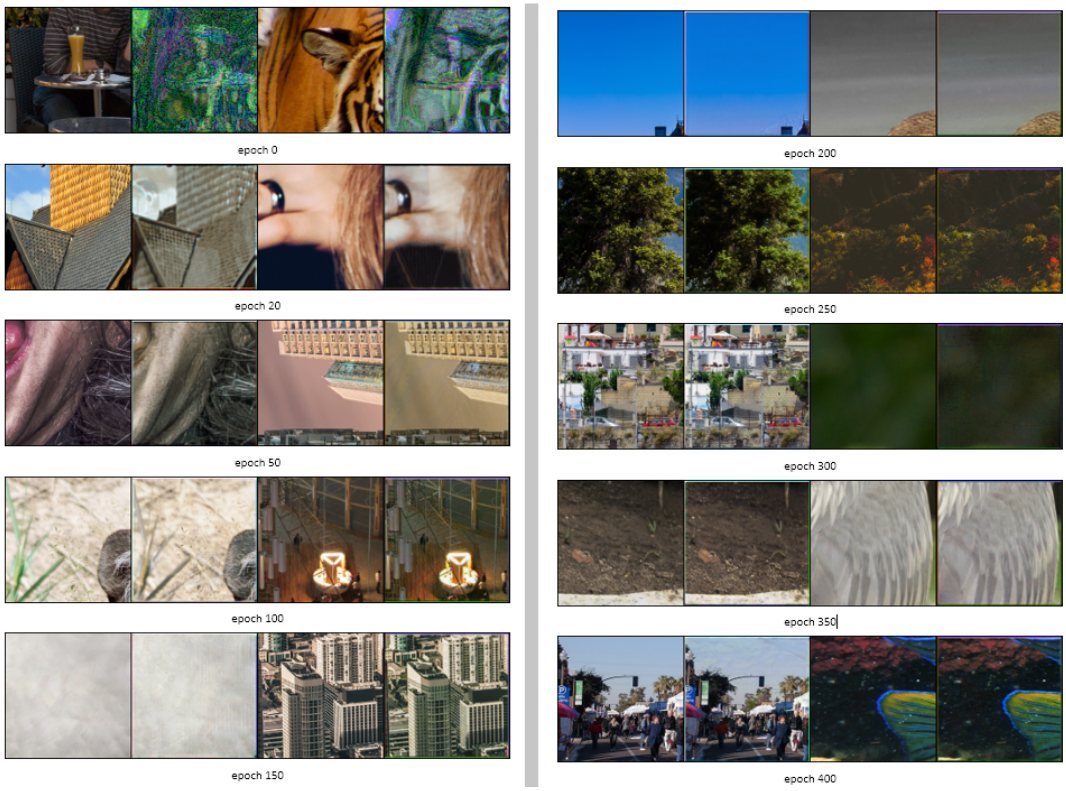


Figure 4.10: Model results from epochs(left to right): Cover Image, Stego Image, Secret Image, Reveal Image

4.2 Chat-App (Secured Khabhari)

We named our Chat Application as "Secured khabri", whose aim is to provide a facility for chatting with different members by creating a room in the application without the security concern. In other words our chat application primary concern is that a user does not risk his/her privacy with other users(third party). Which means no users other than sender or receiver can read the message. It has some special features like Image steganography where user can hide another image or message in the cover image with the help of Neural style or Deep stag technique as an option for a user according to his/her needs. A user can create a room and named their room as they want and can add or remover other members in the rooms and can chat or share their stuffs with them.

4.2.1 Saving and authenticating users in the database

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))

    message = ''
    if request.method == 'POST':
        username = request.form.get('username')
        password_input = request.form.get('password')
        user = get_user(username)

        if user and user.check_password(password_input):
            login_user(user)
            return redirect(url_for('home'))
        else:
            message = 'Failed to login!'
    return render_template('login.html', message=message)
```

Figure 4.11: Authenticate users

Saving and authenticating the users is really important task in our chat app which is done by the following functions. For saving the users we are using the functions save_user(username, email, password) in this we are accepting the unique username which will be used as their unique id in the database which will store their data as email and password (which is in encrypted format).

For getting the user information we are using the function get_user(username) which is returning the user data in the form of python dictionary which in turn will be used for authenticating the user and getting them using the app and get connected with their contacts.
Each user is allocated with unique id and they can be added in the room by using that id.

```
chat_db = client.get_database("ChatDB")
users_collection = chat_db.get_collection("users")
rooms_collection = chat_db.get_collection("rooms")
room_members_collection = chat_db.get_collection("room_members")
messages_collection = chat_db.get_collection("messages")


def save_user(username, email, password):
    password_hash = generate_password_hash(password)
    users_collection.insert_one({'_id': username, 'email': email, 'password': password_hash})


def get_user(username):
    user_data = users_collection.find_one({'_id': username})
    return User(user_data['_id'], user_data['email'], user_data['password']) if user_data else None
```

Figure 4.12: Save users

```
def save_user(username, email, password):
    password_hash = generate_password_hash(password)
    users_collection.insert_one({'_id': username, 'email': email, 'password': password_hash})


def get_user(username):
    user_data = users_collection.find_one({'_id': username})
    return User(user_data['_id'], user_data['email'], user_data['password']) if user_data else None
```

Figure 4.13: Retrieving users

4.2.2 Creating the chat history

```
def save_message(room_id, text, sender):
    messages_collection.insert_one({'room_id': room_id, 'text': text, 'sender': sender, 'created_at': datetime.now()})


MESSAGE_FETCH_LIMIT = 3


def get_messages(room_id, page=0):
    offset = page * MESSAGE_FETCH_LIMIT
    messages = list(
        messages_collection.find({'room_id': room_id}).sort('_id', DESCENDING).limit(MESSAGE_FETCH_LIMIT).skip(offset))
    for message in messages:
        message['created_at'] = message['created_at'].strftime("%d %b, %H:%M")
    return messages[::-1]
```

Figure 4.14: Save messages

As, we can see we are using mongoDB database, which is a live database. Here we are saving messages and creating the chat history in our database. We can see the function save_message(room_id, text, sender) by which we are able to save the messages from the different rooms without collision and the messages are safe in the cloud database. Hence implementing DbAAS (Database as a service). To retrieve the messages in the descending order of the time of chatting we are using the function get_messages(roomid, page) for the specific room using the room id's and then

retrieving the chat history and increasing the page count and retrieving the latest messages first.

4.2.3 Creating and editing the permanent rooms

```python
@app.route('/rooms/<room_id>/edit', methods=['GET', 'POST'])
@login_required
def edit_room(room_id):
    room = get_room(room_id)
    if room and is_room_admin(room_id, current_user.username):
        existing_room_members = [member['_id']['username'] for member in get_room_members(room_id)]
        room_members_str = ",".join(existing_room_members)
        message = ''
        if request.method == 'POST':
            room_name = request.form.get('room_name')
            room['name'] = room_name
            update_room(room_id, room_name)

            new_members = [username.strip() for username in request.form.get('members').split(',')]
            members_to_add = list(set(new_members) - set(existing_room_members))
            members_to_remove = list(set(existing_room_members) - set(new_members))
            if len(members_to_add):
                add_room_members(room_id, room_name, members_to_add, current_user.username)
            if len(members_to_remove):
                remove_room_members(room_id, members_to_remove)
            message = 'Room edited successfully'
            room_members_str = ",".join(new_members)
        return render_template('edit_room.html', room=room, room_members_str=room_members_str, message=message)
    else:
        return "Room not found", 404
```

Figure 4.15: Create rooms

So far, we have added the users in the database and they are easily connecting with other members but rooms and their data has to be saved permanently. So, for saving the rooms and creating them permanently we are using the functions save_room(room_name, created_by) which will store the data as room name and created by who will save the admin username. Only admin is allowed to perform changes in the room. To get the room details we are using the function get_room(room_id) which will return the room details retrieving the data using the room id.

```python
def update_room(room_id, room_name):
    rooms_collection.update_one({'_id': ObjectId(room_id)}, {'$set': {'name': room_name}})
    room_members_collection.update_many({'_id.room_id': ObjectId(room_id)}, {'$set': {'room_name': room_name}})

def add_room_member(room_id, room_name, username, added_by, is_room_admin=False):
    room_members_collection.insert_one(
        {'_id': {'room_id': ObjectId(room_id), 'username': username}, 'room_name': room_name, 'added_by': added_by,
         'added_at': datetime.now(), 'is_room_admin': is_room_admin})


def add_room_members(room_id, room_name, usernames, added_by):
    room_members_collection.insert_many(
        [{'_id': {'room_id': ObjectId(room_id), 'username': username}, 'room_name': room_name, 'added_by': added_by,
          'added_at': datetime.now(), 'is_room_admin': False} for username in usernames])
```

Figure 4.16: Update rooms

```python
def save_room(room_name, created_by):
    room_id = rooms_collection.insert_one(
        {'name': room_name, 'created_by': created_by, 'created_at': datetime.now()}).inserted_id
    add_room_member(room_id, room_name, created_by, created_by, is_room_admin=True)
    return room_id

def get_room(room_id):
    return rooms_collection.find_one({'_id': ObjectId(room_id)})
```

Figure 4.17: Save rooms

4.2.4 Getting info about the room members

In this we can see information about room members. This can be achieved by clicking the room link in the main menu. Request will be sent to the server appending the room id in the post format and it will redirect the user to the chat room where they can easily connect other members.

```
<h3>Hi {% if current_user.is_authenticated %}{{ current_user.username }}{% else %}Guest{% endif %}!</h3>

{% if current_user.is_authenticated %}
    <h3>My rooms</h3>
    <ul>
        {% for room in rooms %}
            <li><a href="/rooms/{{ room._id.room_id }}" id="lnk">{{ room.room_name }}</a></li>
        {% endfor %}
    </ul>
{% endif %}
```

Figure 4.18: Getting room Information

4.2.5 Addition and removal of users in the chat room

In this many can be added or removed by the admin. Admin can also edit the room name.

```python
def update_room(room_id, room_name):
    rooms_collection.update_one({'_id': ObjectId(room_id)}, {'$set': {'name': room_name}})
    room_members_collection.update_many({'_id.room_id': ObjectId(room_id)}, {'$set': {'room_name': room_name}})

def add_room_member(room_id, room_name, username, added_by, is_room_admin=False):
    room_members_collection.insert_one(
        {'_id': {'room_id': ObjectId(room_id), 'username': username}, 'room_name': room_name, 'added_by': added_by,
         'added_at': datetime.now(), 'is_room_admin': is_room_admin})


def add_room_members(room_id, room_name, usernames, added_by):
    room_members_collection.insert_many(
        [{'_id': {'room_id': ObjectId(room_id), 'username': username}, 'room_name': room_name, 'added_by': added_by,
          'added_at': datetime.now(), 'is_room_admin': False} for username in usernames])


def remove_room_members(room_id, usernames):
    room_members_collection.delete_many(
        {'_id': {'$in': [{'room_id': ObjectId(room_id), 'username': username} for username in usernames]}})
```

Figure 4.19: Add/ Remove members

4.2.6 Broadcasting images in the rooms

In this any user in the room can send the image in the chat room which will be broadcasted to other members in the room.

```python
@app.route('/save', methods=['GET', 'POST'])
def save():
    return render_template('upload.html')


@app.route('/upload', methods=['POST'])
def upload():
    APP_ROOT = os.path.dirname(os.path.abspath(__file__))
    target = os.path.join(APP_ROOT, 'static/')
    listd = os.listdir(target)
    if 'temp.jpg' in listd:
        os.remove(target+'temp.jpg')
    for file in request.files.getlist("file"):
        filename = file.filename
        filename.split('.')
        destination = '/'.join([target, 'temp.jpg'])
        file.save(destination)
    return render_template('upload.html')


@app.route('/remove', methods=['POST'])
def remove():
    APP_ROOT = os.path.dirname(os.path.abspath(__file__))
    target = os.path.join(APP_ROOT, 'static/')
    listd = os.listdir(target)
    if 'temp.jpg' in listd:
        os.remove(target+'temp.jpg')
    return render_template('upload.html')
```

Figure 4.20: Saving Image

```html
<div class="ib">
    <nav class="navbar navbar-expand-lg navbar-light bg-dark fixed-top">
        <label style="color: white;">image area</label>
    </nav>
<div id="Canvas" class="ibc_image">
<img src="{{url_for('static',filename='temp.jpg')}}" alt="Nothing to display" width="200px" height="80px">
</div>
<form action="/save" target="_blank">
<button type="submit" class="btn btn-success icb_btn">Upload/Remove</button>
</form>
```

Figure 4.21: Broadcasting Image

Chapter 5

Screenshots

5.1 Chat App

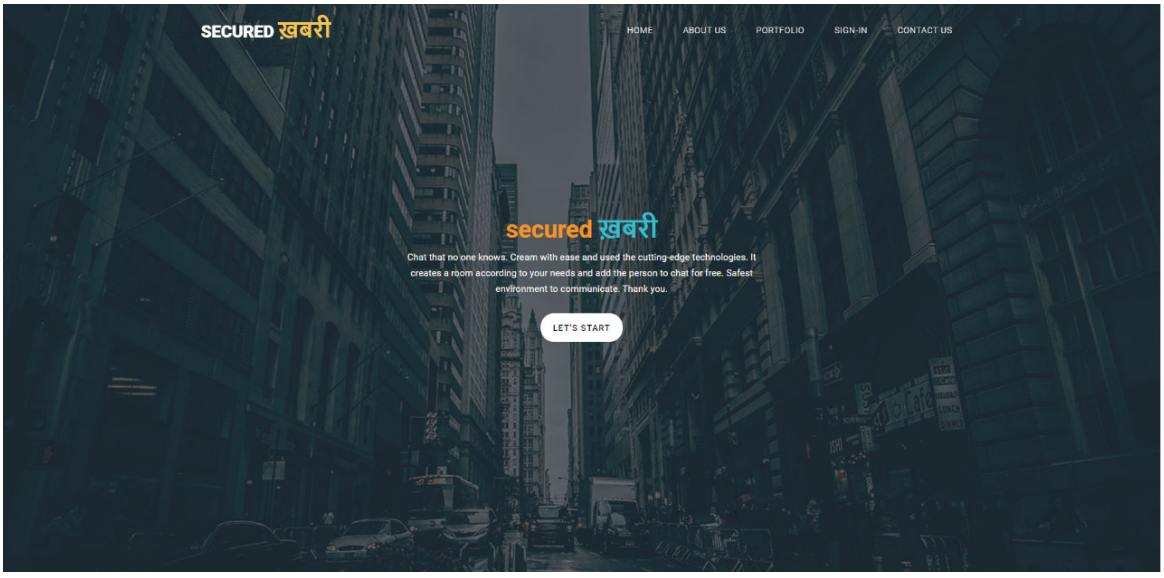


Figure 5.1: Home Page

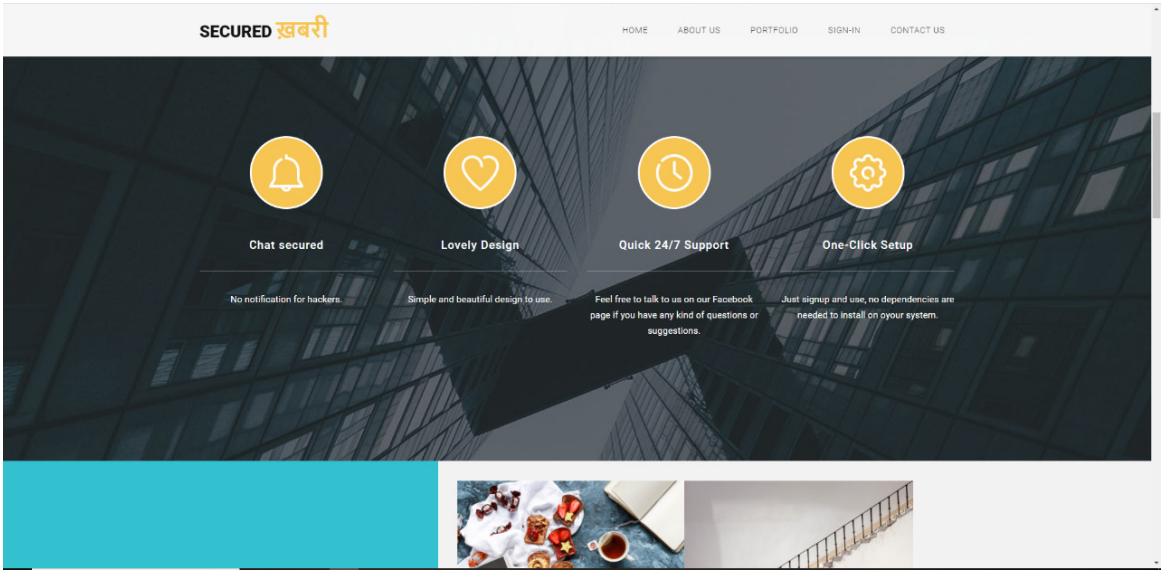


Figure 5.2: Services

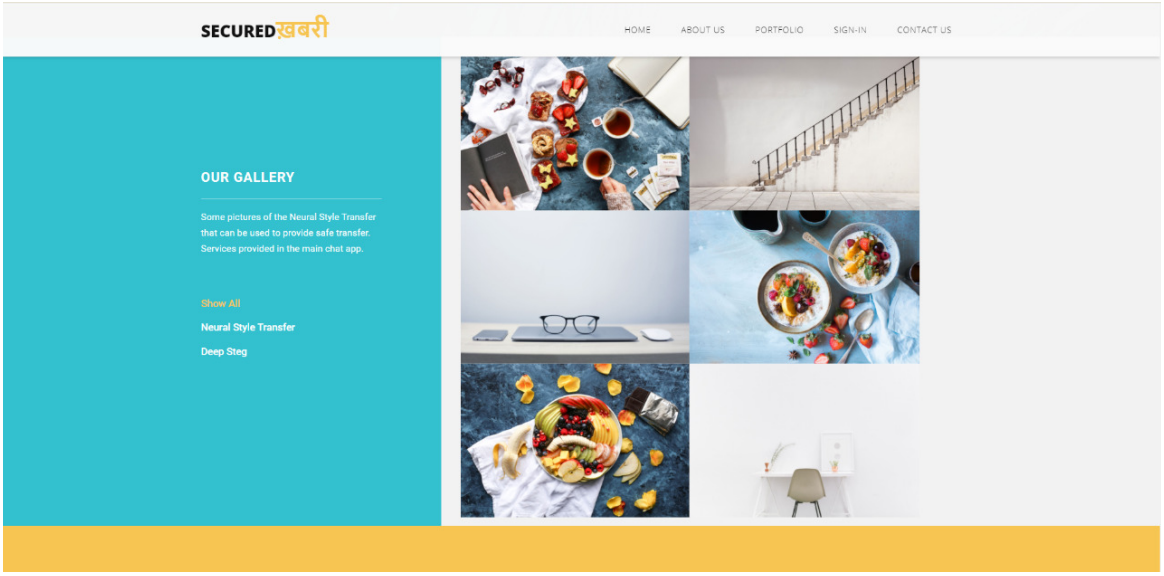


Figure 5.3: Gallery

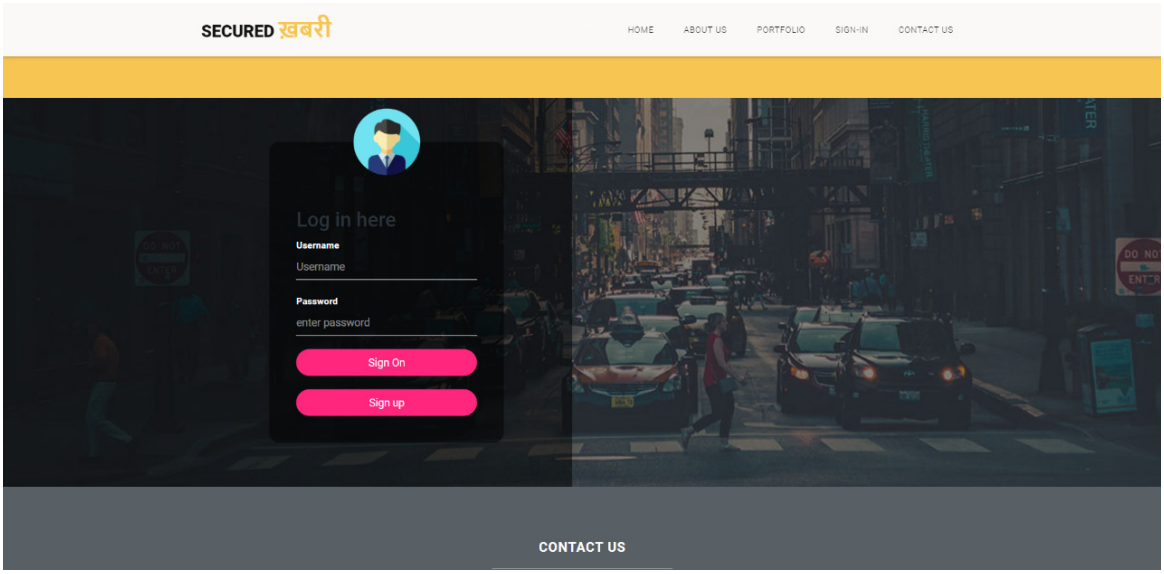


Figure 5.4: Sign-in/ Sign-up Section

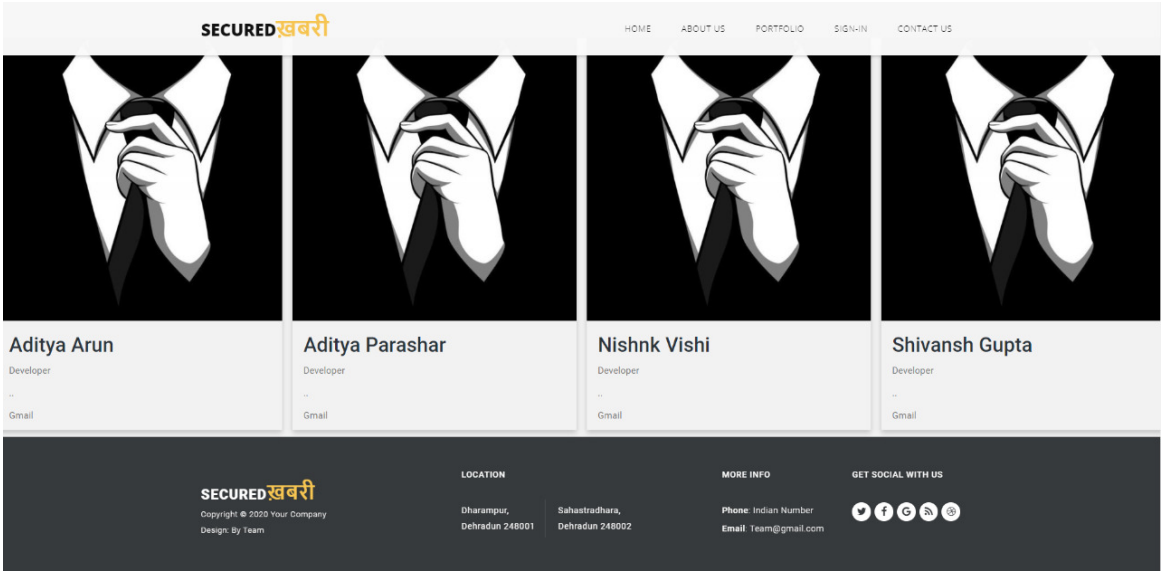


Figure 5.5: Contact Us Section

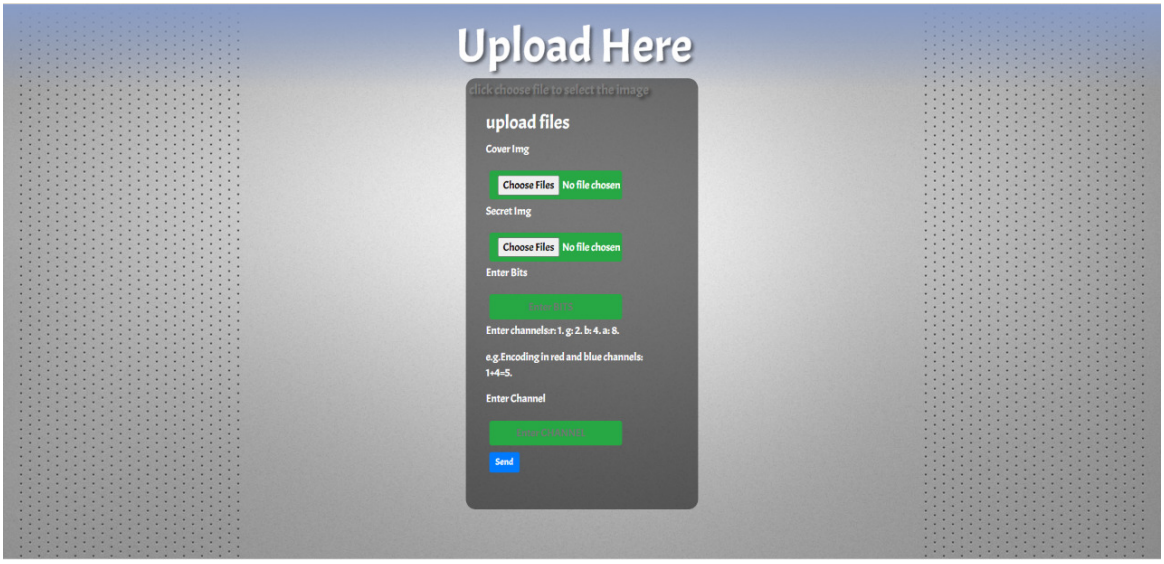


Figure 5.6: LSB-XOR Encryption Page

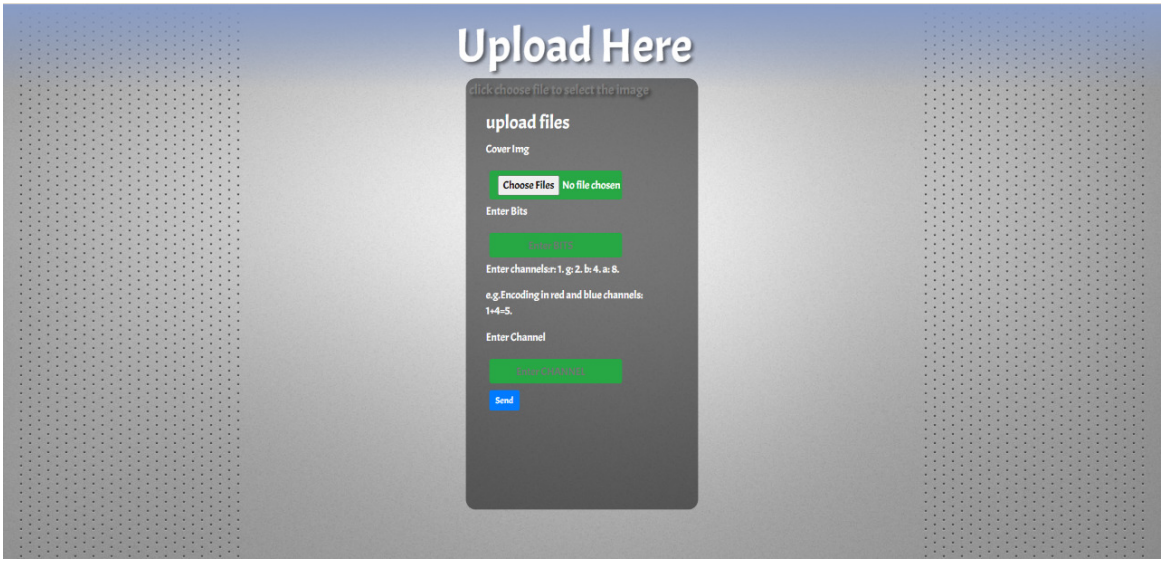


Figure 5.7: LSB-XOR Decryption Page

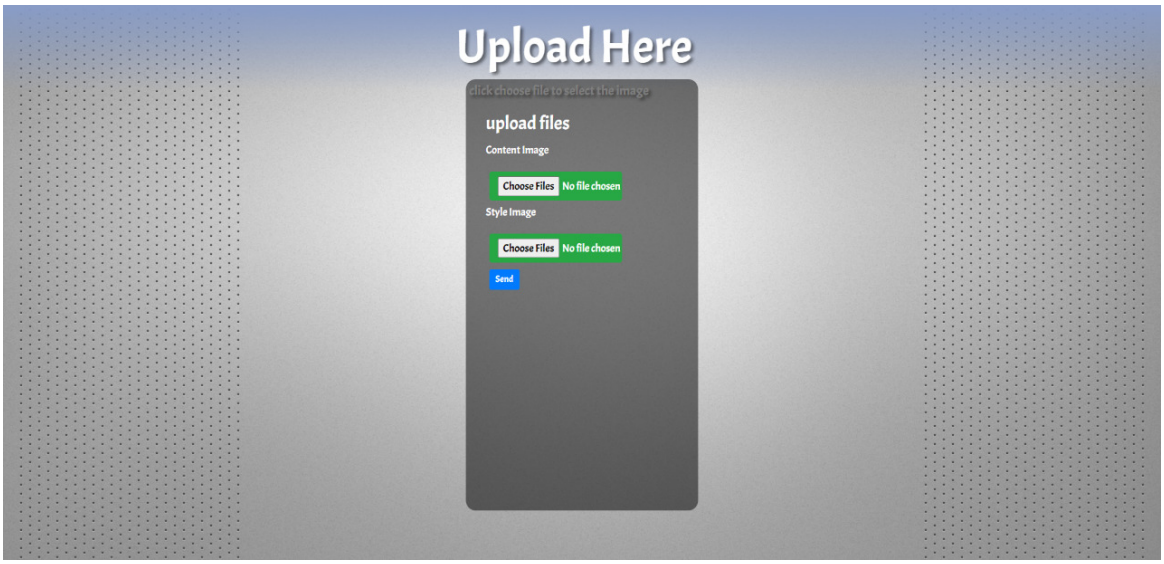


Figure 5.8: Page for Neural Style Transfer

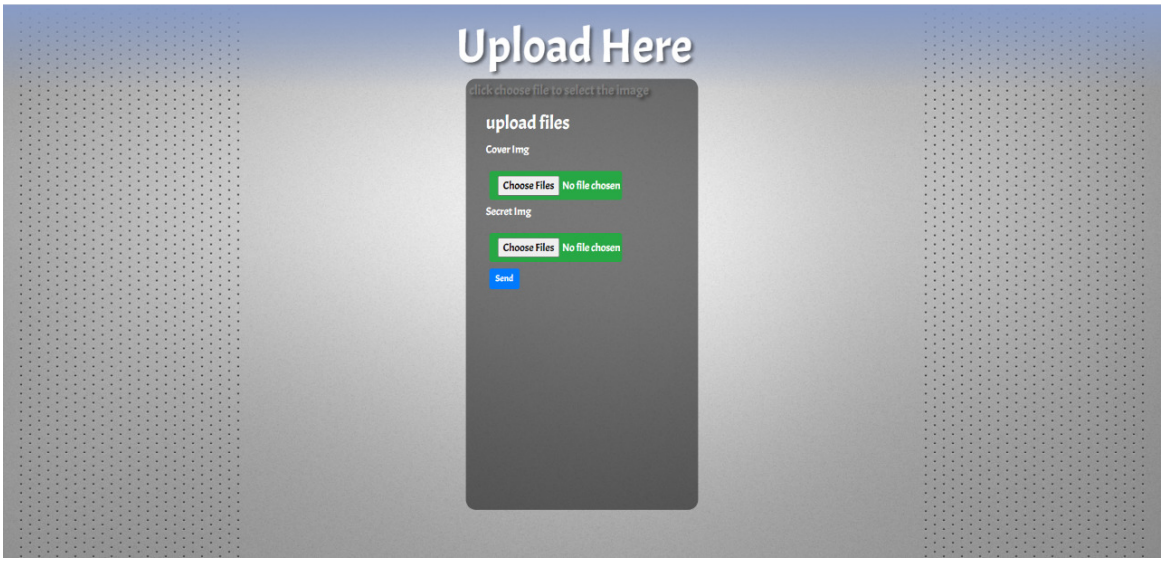


Figure 5.9: Deep Steg Encryption

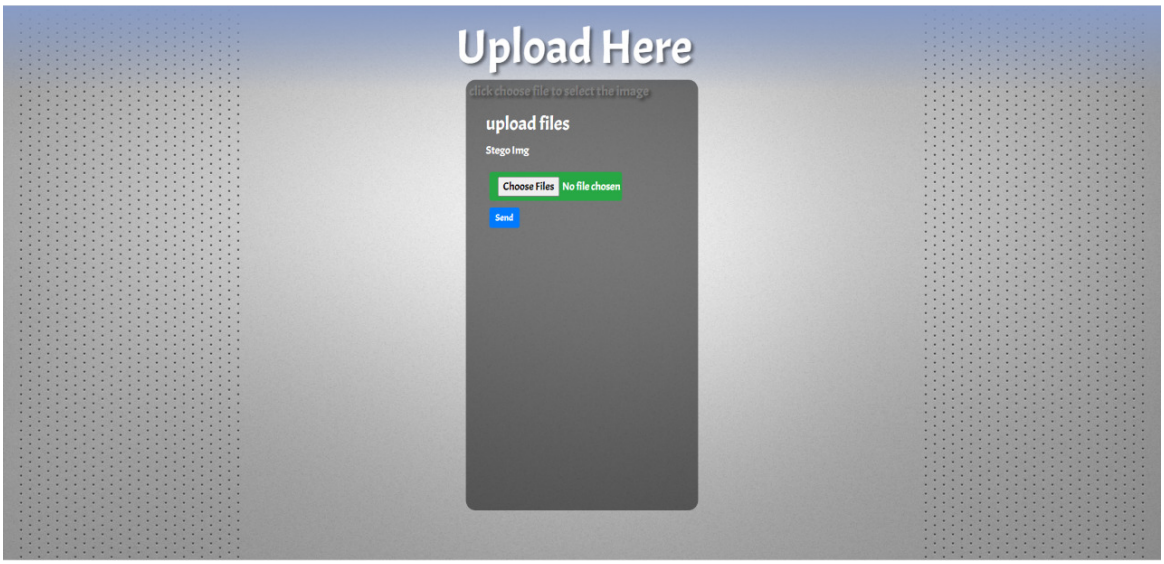


Figure 5.10: Deep Steg Decryption

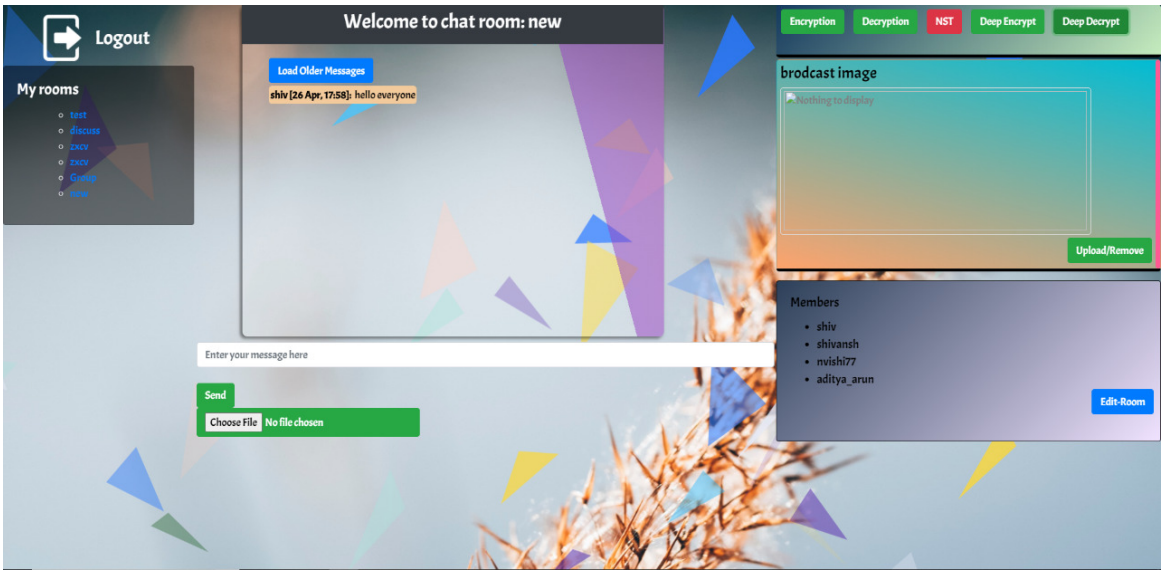


Figure 5.11: Chat room with Users

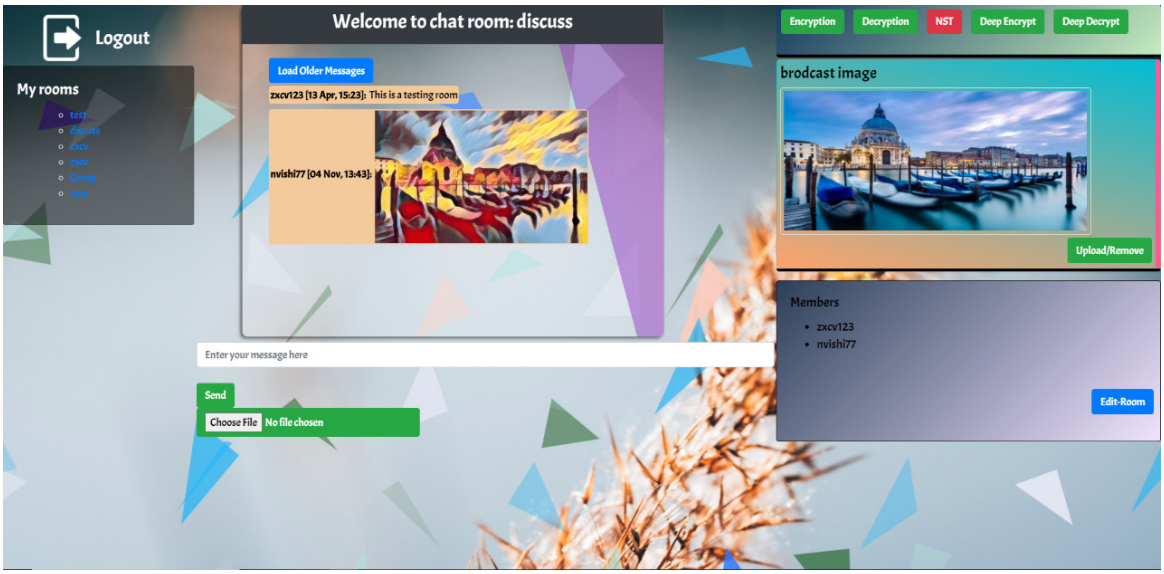


Figure 5.12: Send image in room

5.2 Database Management(MongoDB)

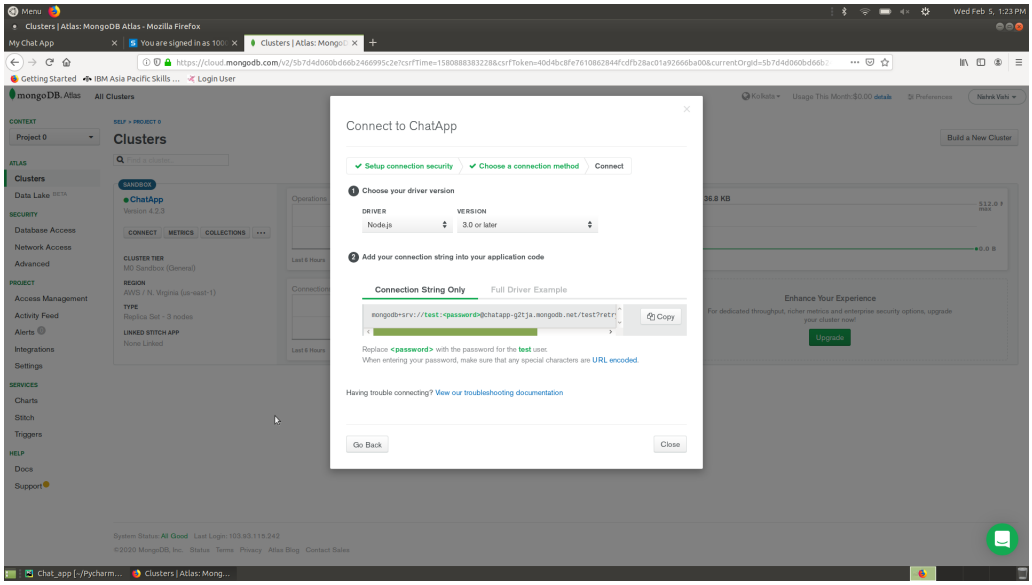


Figure 5.13: Connecting Mongo

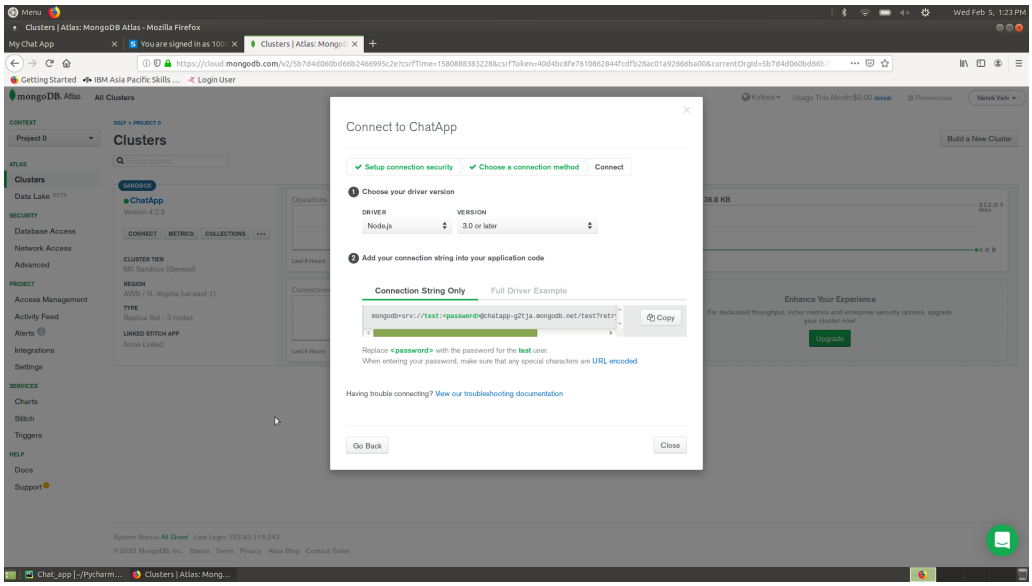


Figure 5.14: Connecting Mongo

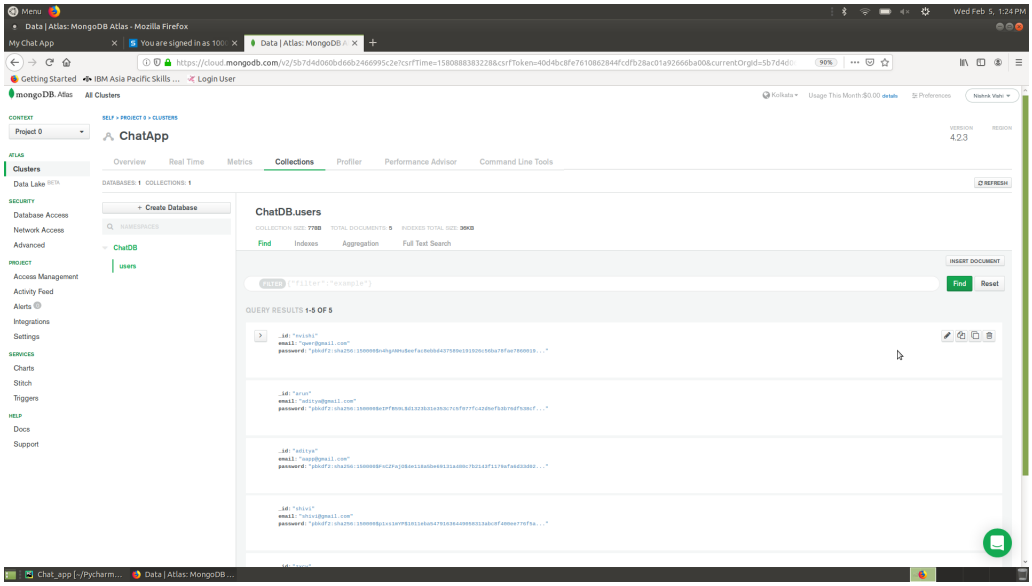


Figure 5.15: Mongo Collection

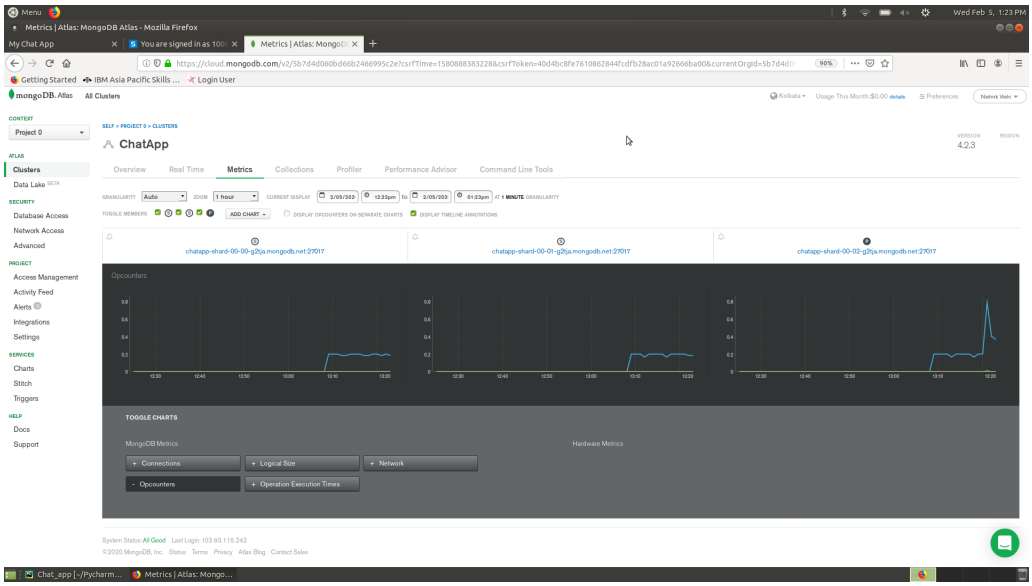


Figure 5.16: Analysing Database

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this Project, we have discussed the steganography technique to implement in Image-Based Steganography and also chat app is also implemented known as SecuredKhabri. In this we have first discussed about the algorithim for steganography and then we will integrate it in chat app like Whatsapp.

First we have discussed about the Steganogrphy algorthitms like combination of LSB and XOR techniques, use of Neural Style Transfer technique in steganography and Deep Steganography techniques. These combinations help us to make the traditional technique more secure. After implementing it in the form of scripts we have showed the results.Next we have developed a GUI in the form of chat app in which various users can communicate with each other in secure private rooms. They can broadcast or send images in the room or can chat in normal text. Communication channel in this room is End to End encrypted so, user have not to worry about privacy. As no other than them or person in the room can read the chat. If in worst scenario any third person other than room one see then he/she will can't understand as it is encrypted form.

As our app requires only require a web browser, it is platform independent. This app can be access from anywhere until the user has a Internet connection in the web browser. Various bugs have been figured out which will be fixed further.

6.2 Scope for Future Work

Our app is not limited to Steganography but can perform Steganography and Cryptography simultaneously which will make our communication channel more secure. It will help us to gain the trust of our consumers/clients. Further, we have to integrate our chat app with more Steganography algorithms and techniques to make the app more secure and give user a choice for the level of security he/she wants.

Also, we are planning to use Steganalysis in our app. This will make app more vast and also open the various research aspects in Encryption and Stegangraphy field. Also it is required to make the app's working more efficient and fast, which will not make heavy loads on server. These issues will be fulfilled further. We will try to deploy our app on cloud like AWS/Google Cloud/IBM cloud.this will help to reduce our hardware cost and other maintenance things.

In this project only image steganography is discussed. But our aim is to also cover audio and video steganography. This open more ways to research and go deep down in this field / topic.

Bibliography

[1] Jing, Yongcheng, et al. "Neural style transfer: A review." IEEE transactions on visualization and computer graphics (2019).

[2] [Online].Avaliable:https://www.researchgate.net/publication/258119723 _Steganography_Techniques_A_Review

[3] Shangping Zhong, Xin Fang, and Xiangwen Liao (2009) 'Steganalysis Against Equivalent Transformation Based Steganographic Algorithm for PDF Files', Proceedings of the 2009 International Symposium on Information Processing (ISIP'09) Huangshan, P. R. China

[4] Liu, Y., X. Sun, Y. Liu and C.T. Li (2008) 'MIMIC-PPT: Mimicking-based steganography for Microsoft power point document'. Inform. Technology

[5] Baluja, Shumeet (2017), 'Hiding Images in Plain Sight: Deep Steganography', Proceeding of Advances in Neural Information Processing Systems 30.

[6] M. Shaha and M. Pawar, "Transfer Learning for Image Classification," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2018

[7] Tan, Chuanqi, et al. "A survey on deep transfer learning." International conference on artificial neural networks. Springer, Cham, 2018.

[8] [Online].Avaliable:https://www.researchgate.net/publication/236334151_ABOUT _FEASIBLE_ USAGE_OF_THE_STEGANOGRAPHY_IN_PUBLIC _ADMINISTRATION

[9] [Online].Available:http://www.backbonesecurity.com/Latest_News/Entries/2010/4/29_Backbone _Security_Releases_New_Version_of_Popular_StegAlyzer_Tools.html