Technical Report on
A Tool for Encrypted File Sharing

Course Code: MS 1003
Course Name: Secure Software Design and Programming

Submitted by
Lamisa Quaiyum Shamma
Debobrata Chakraborty

Supervised by
Dr. B M Mainul Hossain
Professor
Institute of Information Technology,
University of Dhaka

Date of Submission
4 July, 2023

Table of Contents

Contents

List of Figures

Chapter 1: Introduction

The rapid advancement of digital technology has transformed the way we store, access, and share information. With the increasing reliance on file sharing platforms, ensuring the security and privacy of transmitted data has become a paramount concern. Traditional file sharing methods often lack robust security measures, leaving sensitive information vulnerable to interception and unauthorized access. To address these challenges, the development of an encrypted file sharing system has emerged as a critical solution.

Encrypted file sharing refers to the secure exchange of files between users by encrypting the data before transmission. By using encryption key and an encryption algorithm, readable data is transformed into a format that is unreadable during encryption. The encrypted files can only be decrypted and accessed by authorized recipients with the corresponding decryption key. This ensures that sensitive information, such as personal data, financial records, intellectual property, or classified documents, remains secure during transit. Encrypted file sharing systems typically employ strong encryption algorithms, such as AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), or ECC (Elliptic Curve Cryptography), to protect the files. These algorithms use complex mathematical operations to scramble the data in a way that can only be reversed with the appropriate decryption key.

WebSocket is a communication protocol that enables full-duplex, real-time communication between a web browser and a server over a single, persistent connection. In the context of file sharing, WebSocket-based connections provide an efficient and reliable method for transmitting files. Unlike traditional HTTP-based connections that require multiple requests and responses for data exchange, WebSocket allows for simultaneous bidirectional communication. This enables fast and seamless file transfer by sending file chunks directly over the WebSocket connection.

This chapter describes the purpose, and definitions of the encrypted file sharing system tool.

1.1 Purpose

The purposes of this project are:

- **Security:** The project aims to provide a secure file sharing system by encrypting the files before they are transmitted over the network.

- **Real-time Communication:** The project enables real-time communication by using Socket.IO to establish a bi-directional communication channel between the server and clients. This allows multiple users to share and access encrypted files simultaneously, facilitating seamless collaboration and efficient file sharing.

- **User-Friendly Experience:** The project strives to provide a user-friendly experience by integrating modern web technologies such as HTML, CSS, and JavaScript.

1.2 Scope

The scope of the project is given below:

- The system is developed and tested on Ubuntu and node.js framework.

1.3 Assumptions

The assumptions of the project are:

- The underlying system is completely reliable.
- All the libraries are pre-installed.

1.4 Definitions

Socket.IO: A real-time library called Socket.IO enables bi-directional, low-latency communication between web clients and servers. On top of the WebSocket protocol, Socket.IO adds new features like broadcast support, automatic reconnections, and fallback to HTTP long polling.

AES: AES (Advanced Encryption Standard) is a widely used symmetric encryption algorithm that ensures secure file encryption. It employs a symmetric key to encrypt and decrypt data, making it efficient and fast for large file encryption. AES operates on fixed-size blocks of data and supports key lengths of 128, 192, or 256 bits, providing robust security. Its strength lies in its ability to resist various cryptographic attacks. AES encryption follows a substitution-permutation network structure, where multiple rounds of substitution, permutation, and mixing operations are applied to the data. With its strong encryption capabilities, AES is commonly used to protect sensitive data during storage or transmission.

Chapter 2: Implementation Details

This chapter describes the implementation details of the encrypted file sharing system.

This file sharing web app consists of a server-side component and a client side component. It is designed to allow seamless sharing of files from a sender(server) to a receiver(client). The implementation uses Node.js, Express, Socket.IO, Crypto web API and Ngrok to facilitate real time communication between the sender and the receiver.

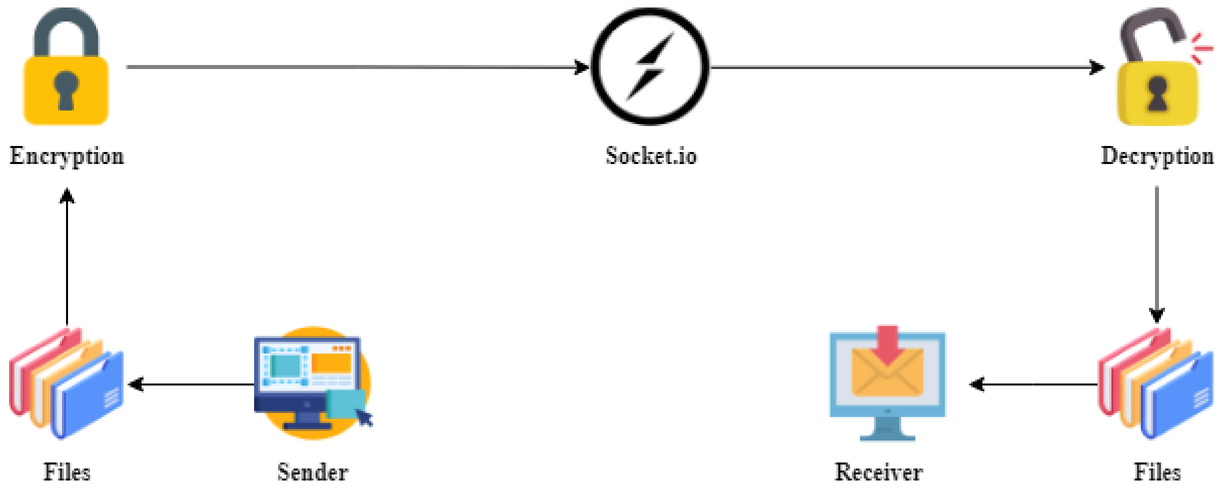


Figure 1: Overview of the encrypted file sharing tool

Sender(Server-side)

The sender side is responsible for initiating the file sharing process. It uses Node.js and Express to create a server that listens for incoming requests. After receiving a request from the client, the server establishes a socket connection using Socket.IO. The sender generates a unique room ID, which is basically a shared identifier for the sender and the receiver. When a receiver wants to receive files from the server, it joins the room using the room ID and a connection is established between them.

Receiver(Client-side)

The receiver side is responsible for receiving the shared files and handling them properly. Here , the user can access this receiver interface through a browser. This interface allows the user to enter a room by using the room ID provided by the sender and initiate the file sharing process. Once the room ID is entered, the client establishes a socket connection to the server. Upon successful connection, the receiver interface transitions to a file sharing interface. The receiver listens for the events emitted by the sender which indicates the progress of the files.

File sharing

When the sender selects a file to share, it undergoes a process that prepares it for transmission. The file is encrypted using AES-CBC encryption algorithm to ensure its security during transmission. The encrypted file is then divided into smaller chunks, which are sent to the receiver using the socket connection. The receiver listens for those file chunks and reconstructs the file on that end.This allows the receiver to progressively receive and display the file as it is being transmitted. After that the encrypted file is decrypted and downloaded automatically after that.

Encryption

Encryption plays a crucial role in ensuring the security and confidentiality of the shared files. The sender utilizes the crypto module in Node.js to encrypt the selected file using the AES-CBC encryption algorithm. During encryption, a random initialization vector (IV) is generated, which adds an additional layer of randomness to the encryption process. The file's contents are encrypted using a symmetric encryption key derived from a predefined secret key. The combination of the IV and the encrypted data is then sent to the receiver. This encryption process ensures that the file's contents are protected from unauthorized access during transmission.

Decryption

On the receiver's side, decryption is performed to recover the original file from the received encrypted chunks. The receiver utilizes the same AES-CBC encryption algorithm and the predefined secret key to decrypt the encrypted data. The receiver also receives the initialization vector (IV) used during encryption from the sender. By using the IV, the receiver can correctly decrypt the received chunks and reconstruct the original file. Once the decryption process is complete, the receiver has access to the original file, which can be downloaded or further processed as needed.

By combining the file sharing, encryption, and decryption processes, the web app provides a secure and efficient mechanism for transmitting files between the sender and receiver. The encryption step ensures the confidentiality of the shared files, while the decryption step allows the receiver to access the original file without compromising its security. This comprehensive approach ensures the integrity and privacy of the shared files throughout the file sharing process.

Ngrok

Ngrok is a cross-platform application that allows developers to easily expose a local development server to the Internet. It hosts a local web server on its own subdomain and makes a local development box available on the internet through tunneling.

Chapter 3: User Manual

A user guide is discussed in this section.. This manual will guide an user through the features and functionality of our secure file sharing platform.

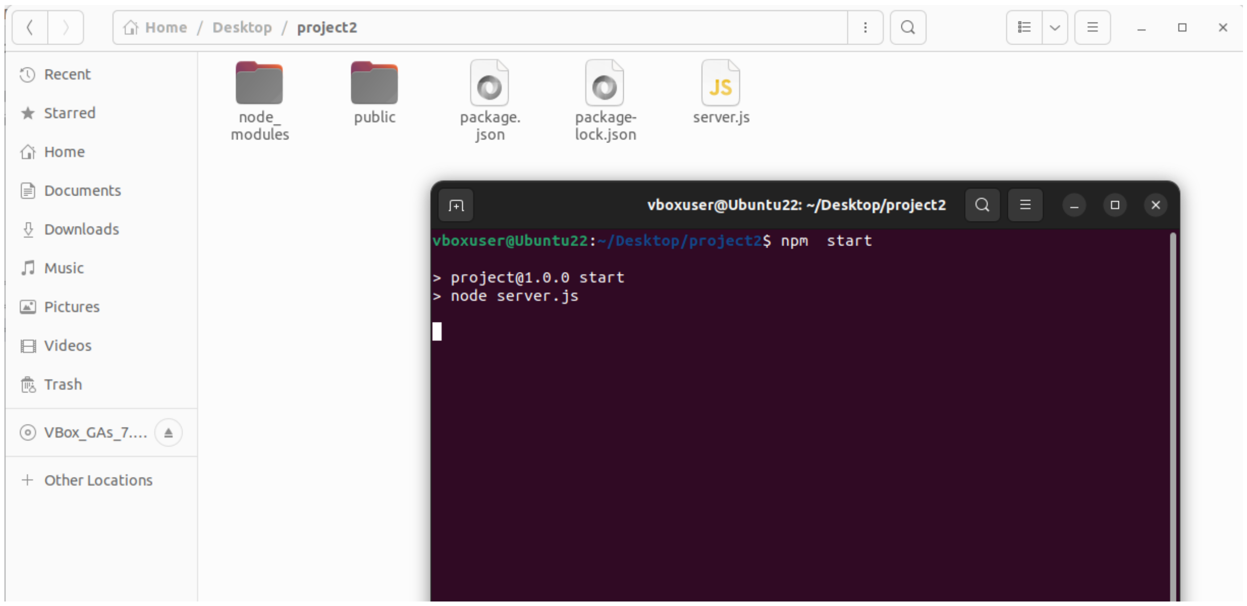
Our encrypted file sharing website provides a secure and convenient way to share files with others while ensuring the confidentiality and integrity of your data. By leveraging encryption techniques, we prioritize the protection of users files during transmission.

This user manual will cover the essential steps to room creating procedure, uploading and sharing files securely, accessing shared files.

The project's github link is provided below.
File sharing tool -
https://github.com/dckakon/Encrypted-File-Sharing-Tool/tree/version-0.0
File sharing tool (with encryption and decryption)-
https://github.com/dckakon/Encrypted-File-Sharing-Tool/tree/master

Getting Started with Node.js

1. Enter into the project folder first, then use terminal and type **"npm start"** command to start the project local server.



```
vboxuser@Ubuntu22:~/Desktop/project2$ npm  start

> project@1.0.0 start
> node server.js
```

Figure 2

2. This Node.js local server is running at port 5000. Now open any browser and navigate to **http://localhost:5000** to view the outcome.

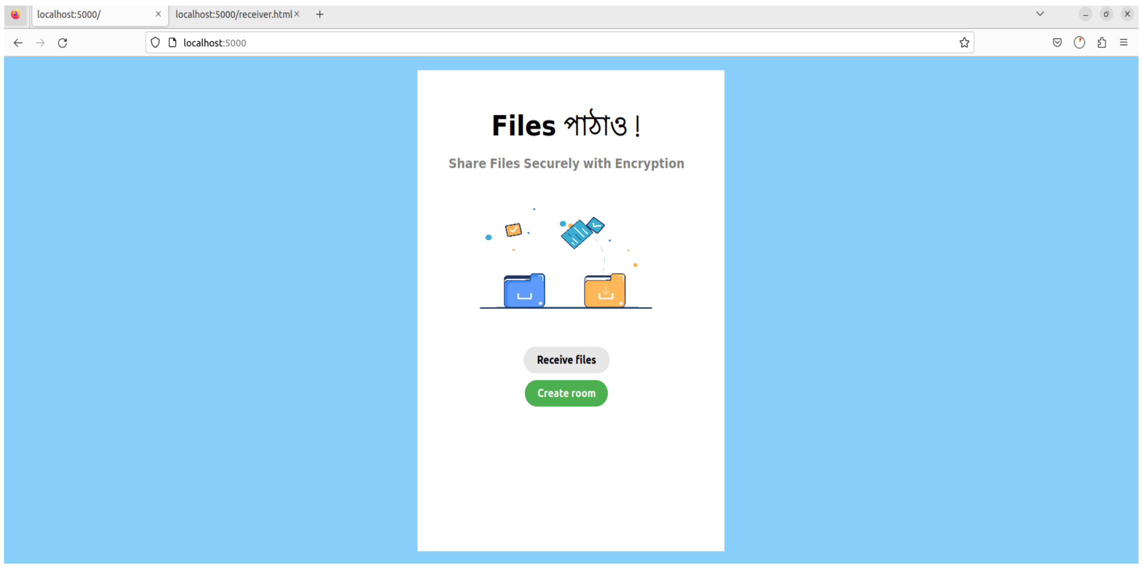
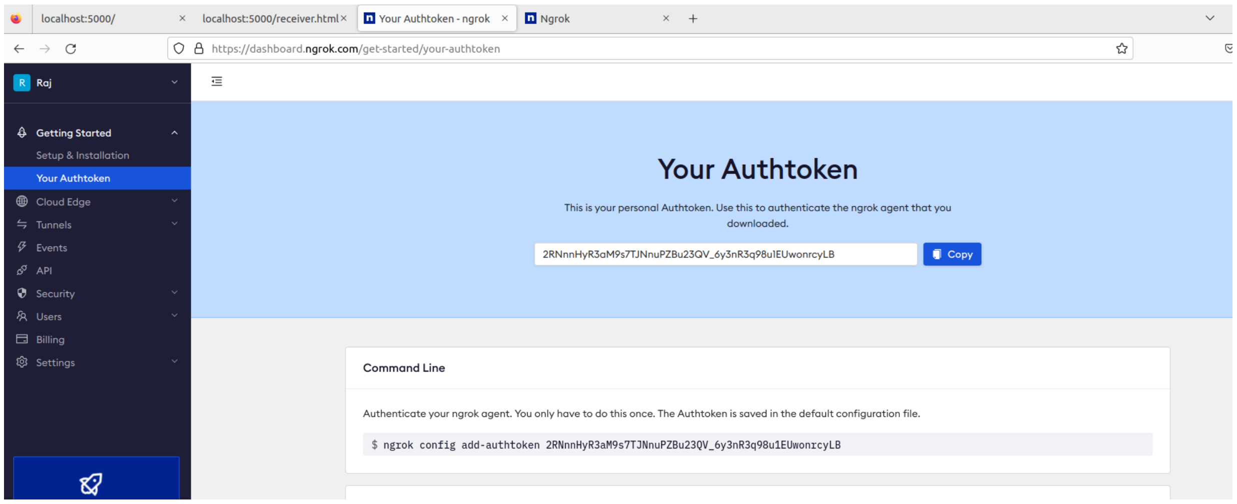


Figure 3

Getting Started with Ngrok - Exposing Local Server to the Internet

1. After starting the local server using the “npm start” command, Ngrok tool will connect the local web server to the internet. Use the following command to install Ngrok-
   sudo tar xvzf ~/Downloads/ngrok-v3-stable-linux-amd64.tgz -C /usr/local/bin
2. Now signup on the website - https://ngrok.com/
3. An authentication token will be generated after signing up on the Ngrok website; use that authentication token to activate the tool.



Your Authtoken

This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.

2RNnnHyR3aM9s7TJNnuPZBu23QV_6y3nR3q98u1EUwonrcyLB

Command Line

Authenticate your ngrok agent. You only have to do this once. The Authtoken is saved in the default configuration file.

```
$ ngrok config add-authtoken 2RNnnHyR3aM9s7TJNnuPZBu23QV_6y3nR3q98u1EUwonrcyLB
```

Figure 4

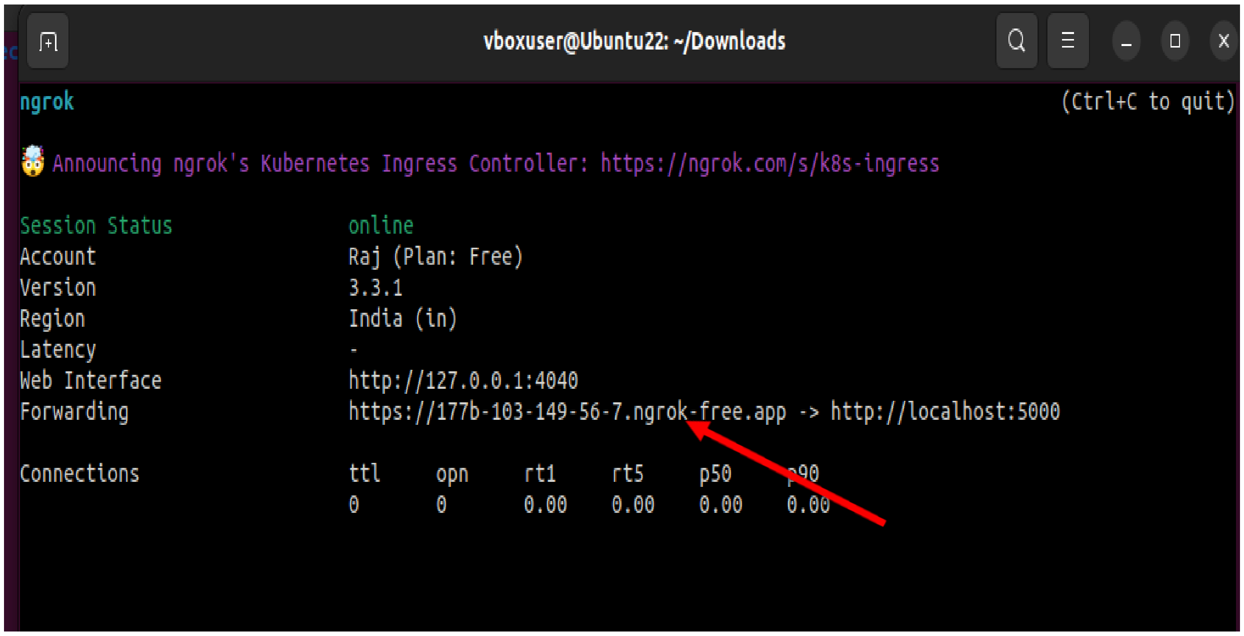
To add the token, run the following command line in the same location as where ngrok was downloaded -

ngrok config add-authtoken <token>

4. Use the following command to launch a tunnel on Ngrok, which will connect the local server to the internet as the server listening on port 5000.

   ngrok http 5000

5. After running the above command, a link connecting the local server to the internet can be found.



```
vboxuser@Ubuntu22: ~/Downloads

ngrok                                                                           (Ctrl+C to quit)

Announcing ngrok's Kubernetes Ingress Controller: https://ngrok.com/s/k8s-ingress

Session Status                online
Account                       Raj (Plan: Free)
Version                       3.3.1
Region                        India (in)
Latency                       -
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://177b-103-149-56-7.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Figure 5

6. Now, utilize this link to access the file sharing website. By sharing this link, anyone on the internet can gain access to the website.

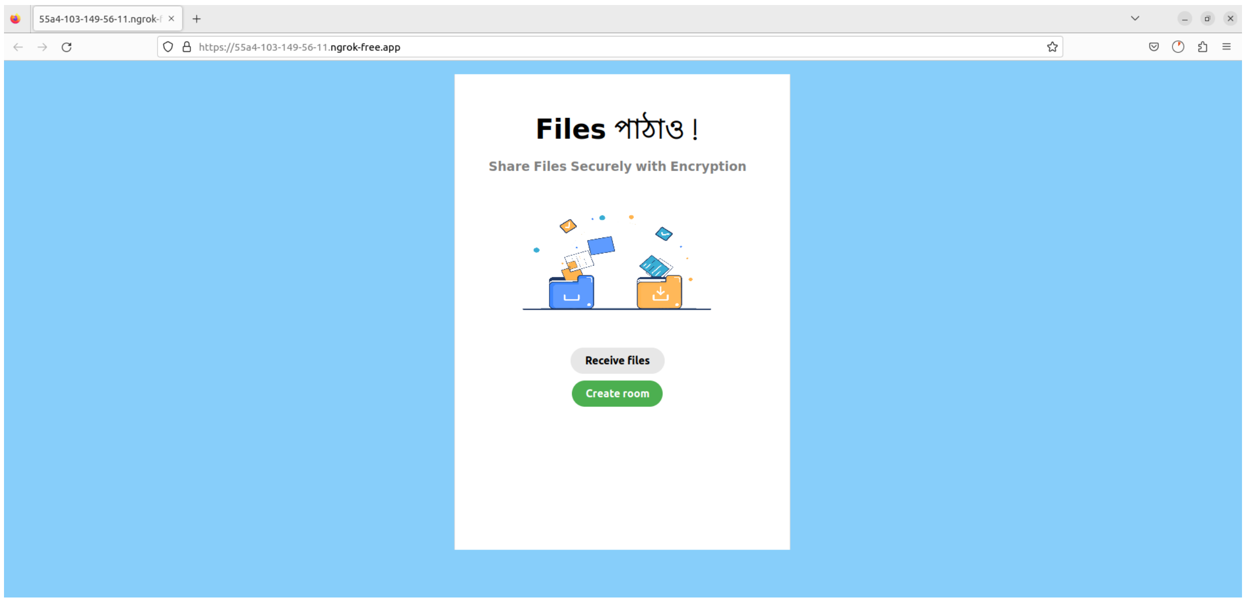

55a4-103-149-56-11.ngrok-
https://55a4-103-149-56-11.ngrok-free.app
Files পাঠাও !
Share Files Securely with Encryption
Receive files
Create room

Figure 6

Guide to Using and Navigating the Website

1. To send a file, the sender must first create a room by clicking the **"Create room"** button.

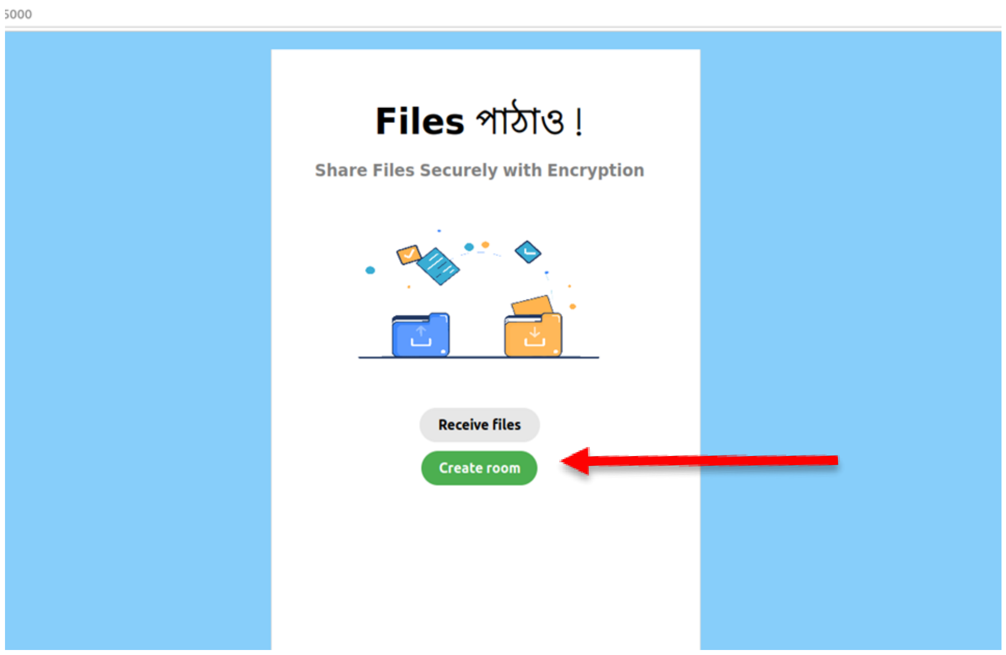


Figure 7

2. The sender user will generate a room id after clicking the "**Create room**" button and will need to send the room id to the receiver.

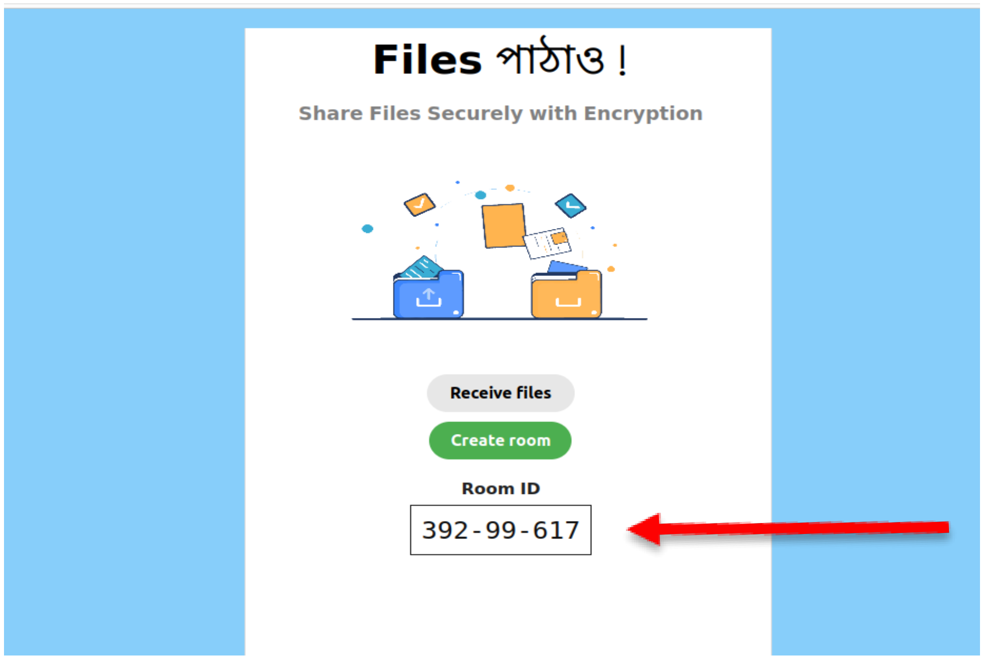


Figure 8

3. To proceed, the receiver must first click the **“Receive files”** button.

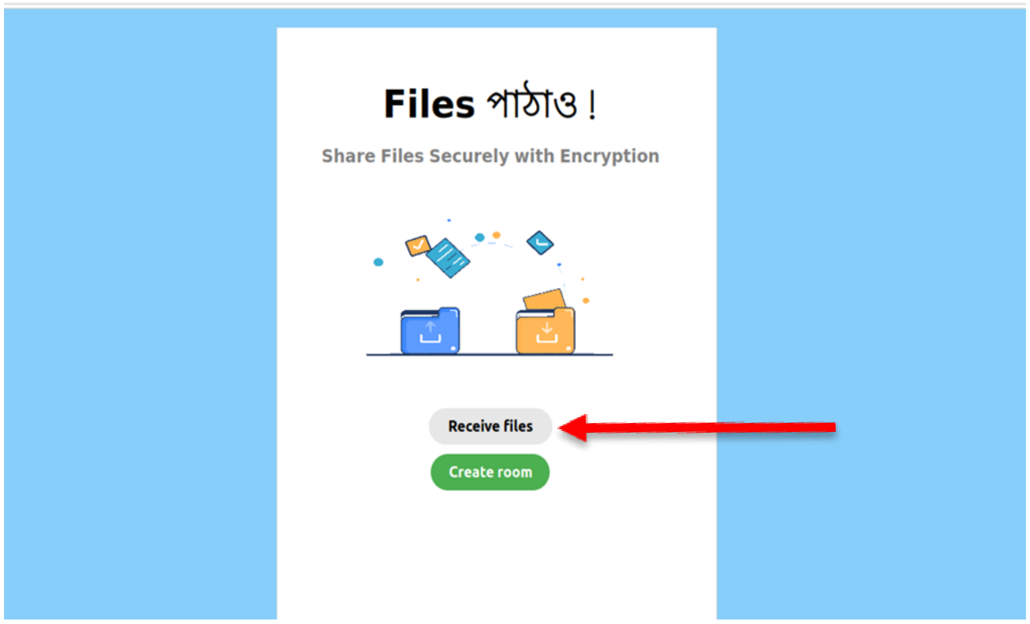


Figure 9

4. Following that, the receiver will arrive at this page and must enter the exact room id into the input box before clicking the **"Connect"** button to connect the communication socket between sender and receiver.

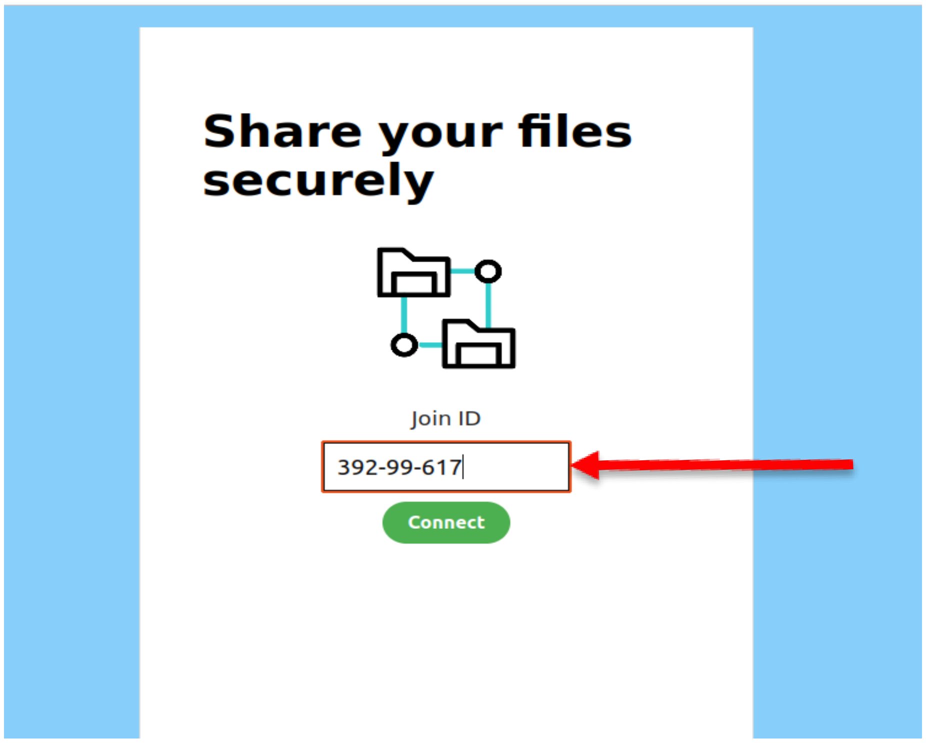
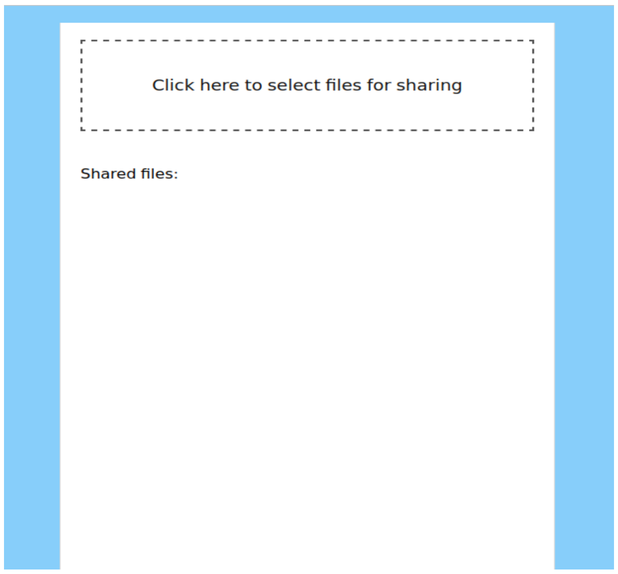
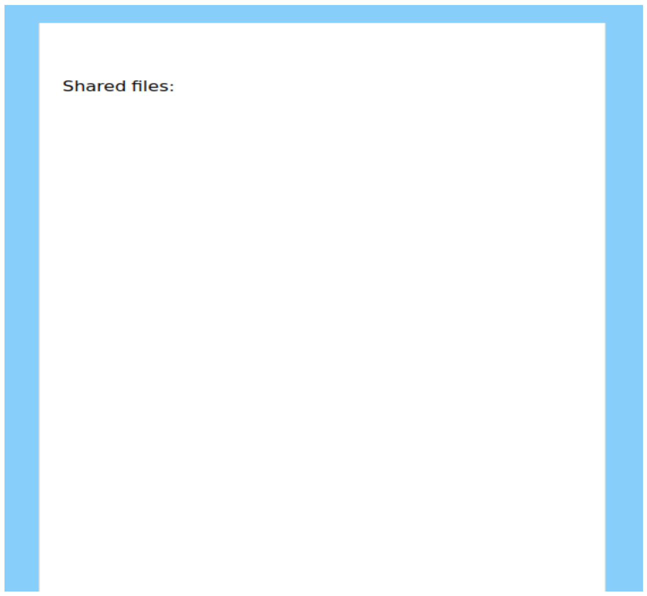


Figure 10

5. After clicking the connect button, both the sender and the receiver will land in this page, indicating that the room is now connected between two users.



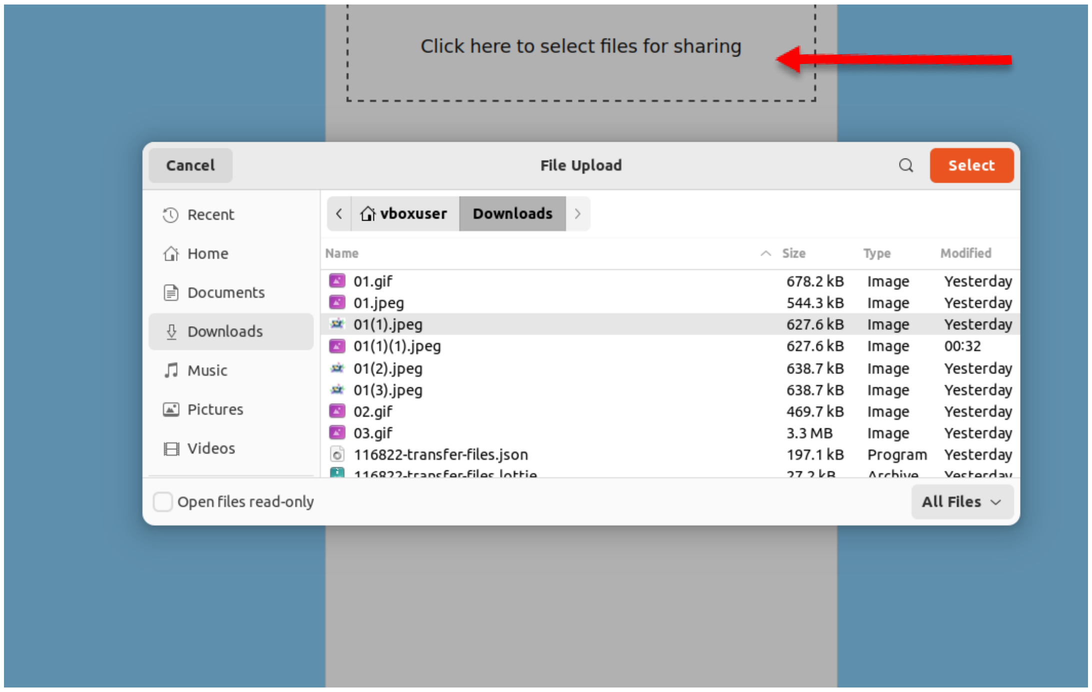
Sender window



Receiver window

Figure 11

6. The sender must then click on **"click here to select files for sharing"** to open file explorer and select a file.



Click here to select files for sharing

Cancel | File Upload | Select

vboxuser Downloads

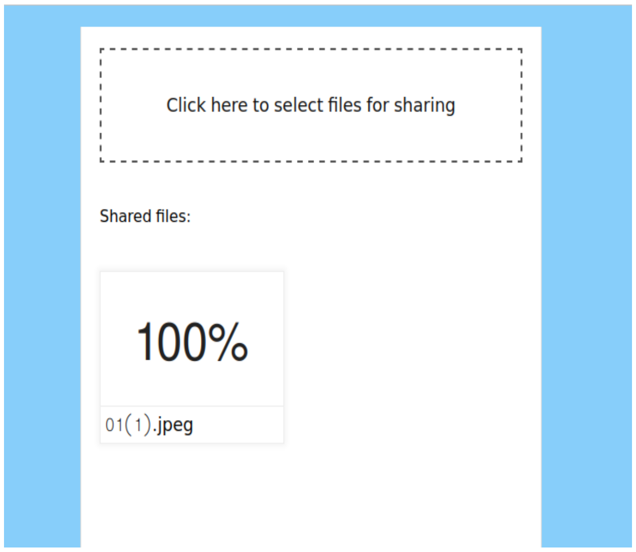
Recent, Home, Documents, Downloads, Music, Pictures, Videos

Name	Size	Type	Modified
01.gif	678.2 kB	Image	Yesterday
01.jpeg	544.3 kB	Image	Yesterday
01(1).jpeg	627.6 kB	Image	Yesterday
01(1)(1).jpeg	627.6 kB	Image	00:32
01(2).jpeg	638.7 kB	Image	Yesterday
01(3).jpeg	638.7 kB	Image	Yesterday
02.gif	469.7 kB	Image	Yesterday
03.gif	3.3 MB	Image	Yesterday
116822-transfer-files.json	197.1 kB	Program	Yesterday
116822-transfer-files.lottie	27.2 kB	Archive	Yesterday

Open files read-only | All Files

Figure 12

7. After selecting the file, the file will automatically send the files to the receiver end.

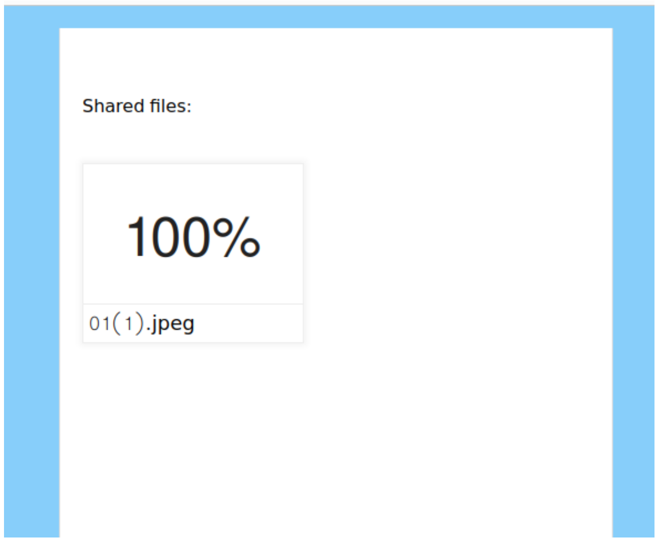


Click here to select files for sharing

Shared files:

100%

01(1).jpeg



Shared files:

100%

01(1).jpeg

Sender window

Receiver window

Figure 13

8. After fully transferring the file, 100% will be displayed, and the browser will automatically download the file.

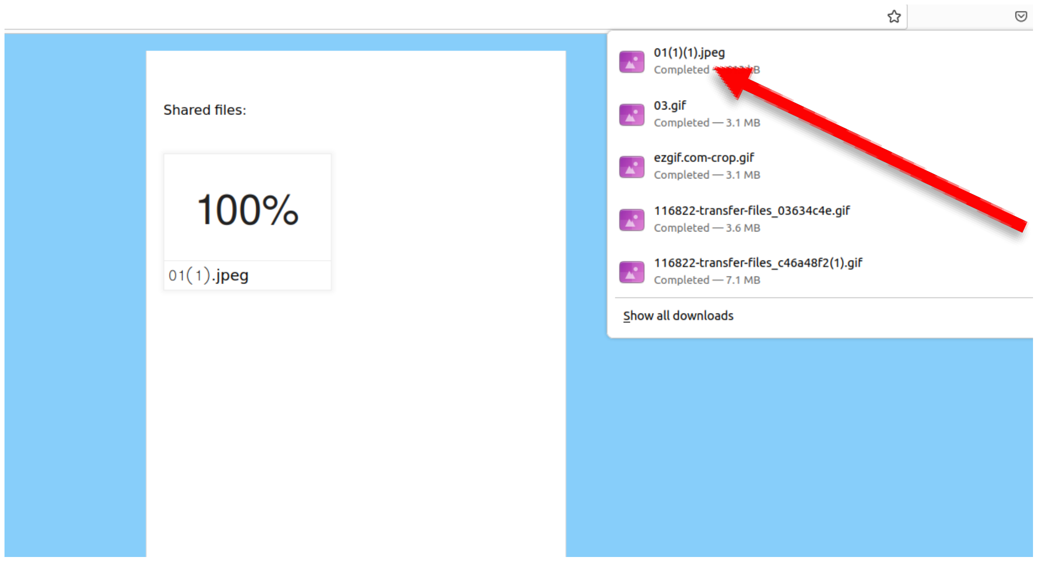


Figure 14

9. Similarly, the sender can send several files to the receiver through a single room. The sender or receiver can disconnect the room by refreshing the webpage.

Chapter 4: Discussion

In this project, an encrypted file sharing system employing the AES algorithm was developed. This project is reliable and allows for simple file transfers over different sockets while using encryption and decryption. The encryption and decryption mechanism is currently only functional for text-based files, such as.txt or.html files. The current version of this project can send and receive any type of file, but it is unable to fully decode image or pdf files.

References

1. https://ngrok.com/download
2. https://socket.io/docs/v4/
3. https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API
4. https://nodejs.org/api/crypto.html