

Software Engineering Department  
ORT Braude College



**WhisperSend**

Capstone Project Phase B

24-1-D-21

GITHUB: <https://github.com/jessicamirov/WhisperSend.git>

**Supervisors:**

Alexander Keselman

**Authors:**

Jessica Amirov: 318553583

Yarden Gabay: 318222171

**Table of Content**

[1. Abstract 4](#_Toc177229311)

[2. Introduction 5](#_Toc177229312)

[2.1 Motivation 5](#_Toc177229313)

[3. Background & Related Work 5](#_Toc177229314)

[3.1 Peer to peer- 5](#_Toc177229315)

[3.2 Symmetric Encryption- 7](#_Toc177229316)

[3.3 Asymmetric Encryption- 8](#_Toc177229317)

[3.3.1 Elliptic curve cryptography - 9](#_Toc177229318)

[3.3.2 Elliptic curve key Generation - 9](#_Toc177229319)

[3.4 API - 9](#_Toc177229320)

[3.4.1 peers API - 10](#_Toc177229321)

[3.4.1.1 PeerJS - 10](#_Toc177229322)

[3.4.2 WebRTC - 10](#_Toc177229323)

[3.5 Node.js - 11](#_Toc177229324)

[3.6 NPM – 11](#_Toc177229325)

[3.6.1 Elliptic Package – 11](#_Toc177229326)

[3.6.2 Ethers Package – 12](#_Toc177229327)

[4. Expected Achievements 12](#_Toc177229328)

[4.1 Success Criteria 12](#_Toc177229329)

[4.2 Unique Features 13](#_Toc177229330)

[5. Engineering Process 13](#_Toc177229331)

[5.1 Process 13](#_Toc177229332)

[5.1.1 Tools and client interaction 13](#_Toc177229333)

[5.1.2 Networking and Servers 14](#_Toc177229334)

[5.1.1.1 PeerServer 14](#_Toc177229335)

[5.1.3 Encryption 15](#_Toc177229336)

[5.1.3.1 Generate Public and Private Keys 15](#_Toc177229337)

[5.1.3.2 Encryption and Decryption implementation 15](#_Toc177229338)

[5.1.4 Client-side 16](#_Toc177229339)

[5.1.4.2 Tailwind CSS 16](#_Toc177229340)

[5.2 Product 16](#_Toc177229341)

[5.2.1 client interface 16](#_Toc177229342)

[5.2.2 home page 16](#_Toc177229343)

[5.2.3 Chat Page 18](#_Toc177229344)

[5.2.4 Mnemonic Phrase 20](#_Toc177229345)

[5.2.5 Self Encryption 20](#_Toc177229346)

[5.2.6 Self Decryption 20](#_Toc177229347)

[5.2.7 Model and Diagrams 22](#_Toc177229348)

[6. Verification and Evaluation 23](#_Toc177229349)

[6.1 Testing plan 23](#_Toc177229350)

[7. Challenges and Solutions 24](#_Toc177229351)

[8. Conclusion 25](#_Toc177229352)

[9. References 26](#_Toc177229353)

# 1. Abstract

In recent years, the internet has become an indispensable tool for daily activities like shopping, banking, social media and sharing sensitive information. However, many websites rely on a single central server to manage data flow, which poses security risks and exposes information to potential threats.

Our project aims to address this concern by developing a user-friendly web application for secure file transfer. To ensure privacy and security, we conducted extensive research to identify the most effective methods for protecting sensitive information.

Through our research, we discovered that combining peer-to-peer communication with asymmetric encryption provides the highest level of security. P2P technology allows users to communicate directly without relying on a central server, thus minimizing the risk of data exposure. Additionally, asymmetric encryption enhances security by using public and private keys to encrypt and decrypt information, ensuring that only authorized recipients can access the data.

While there are various encryption and file transfer solutions available, our platform stands out due to its accessibility to users of all levels, from individuals to large organizations. We have selected efficient and economical technologies like preact and node.js for development, along with Tailwind design packages to facilitate interface development. This strategic approach allows us to prioritize security while ensuring a user-friendly experience for all our platform's users.

Our website offers two essential services: direct encryption of user files stored on their computer and initiating secure chat conversations with other users on the network, enabling the sharing of important information in the safest way possible

At any point in the process, users can request the creation of new encryption keys and These keys are never stored in any database.

Additionally, our commitment to privacy extends to the automatic deletion of encryption keys upon leaving the site.

Our goal is to provide a secure and accessible solution for file transfer, emphasizing user privacy and ease of use.

# 2. Introduction

In the past, sensitive information used to be stored and transmitted through physical means such as sealed envelopes and couriers. However, with the rise of the Internet at the end of the 20th century, data began to be transferred between users through digital channels, leading to intense change in communication and file transfer methods.

The transformation of the network into a central tool for transferring sensitive information makes it vulnerable and exposed to exploitation and hacking by malicious parties. Therefore, there is a distinct need for secure and private communication to transfer files between two parties over the Internet.

The methods used to communicate and share information in the network often require the involvement of an intermediary. As a result, there is an increased risk of potential loss and theft of sensitive information from users.

Hence the clear need to provide solutions that provide the highest level of privacy and minimal reliance on a third party.

## ****2.1 Motivation****

Our solution involves the development of an accessible website for encrypting files using an asymmetric method. This ensures the secure and confidential transfer of files to the destination while preserving data integrity.

Additionally, our solution utilizes peer-to-peer communication. A peer-to-peer network is primarily employed in file-sharing networks, where each peer acts as both a client and a server. Each peer has the capability to initiate or terminate connections and provide or request services. This decentralized network operates without reliance on any central entity.

Our website enables various users to transfer files in encrypted form, guaranteeing the highest level of security throughout the process. The stakeholders of our website are private users, businesses, and various organizations. The solution will help private users transfer sensitive medical, financial, or legal documents in a secure manner.

Businesses and organizations are potential customers due to the need to protect trade secrets and obey information security regulations.

# 3. Background & Related Work

## 3.1 Peer to peer-

"Peer to peer networks first made their first appearance in the late 70's early 80's with USENET and FidoNet. At that time there was a strong need for decentralization in network architecture" [6].

"P2P technology implements a distributed system, in which each node acts as both a client and a server" [3]. In P2P, the peers can connect directly, and there is no need to use centralized servers. In this autonomy, each peer has the same capabilities. As a result, none of the entities has complete control over the entire network. Peers depend on each other and help each other to share and receive information.

“Every node participating in the network acts as a server that can upload, download, and share files with other nodes. The nodes use their hard drives instead of a central server to store this data. As these capabilities to transmit, receive and store files lie with each node, the P2P network is more secure, fast, and efficient.” [4]

"Conceptually, P2P computing is an alternative to the centralized and client-server models of computing, where there is typically a single or small cluster of servers and many clients" [8]. As P2P networks grow, efficiency increases, contrasting with client-server systems, where efficiency decreases with more users.

To communicate with P2P technology, the peers must use the same or a compatible program, it allows to share information and resources with each other.

The peers in the network are represented by nodes. A relationship between peers is represented by an edge between the nodes, indicating the ability of the two to communicate. To optimize search times when establishing communication, it is common to use the Distributed Hash Table (DHT) method.

Peers mapped to a specific region (geographically close, same functionality); they will receive a key that belongs to the key range of that region.

When a new node is added to the network the neighbors share their range with it. When a node leaves the network, his neighbors take over his share.

This process helps ensure the resilience and efficiency of the system in managing distributed resources.

P2P technology enables file sharing, data transfer and communication between computers directly.

As with any computer system, a P2P system emphasizes the needs of its users, and can meet several goals:

1. Cost reduction - in centralized systems, one central server bears the burden. In P2P the burden is divided equally between the peers.
2. Resource aggregation and improved performance - Each node contributes its own resources such as its storage space, computing power, and bandwidth capacity. Therefore, it is possible to accumulate resources of all the nodes in the system and thus complex computational operations can be performed with improved efficiency.
3. Autonomy and privacy - in the P2P system, the peers communicate directly and autonomously, minimizing reliance on a central server. The users prefer to perform actions locally than to transmit information through the server, thus being able to maintain their anonymity on the network.
4. Dynamism and scalability - Peers can join and leave without causing data loss or system failure. Therefore, the network knows how to deal with a massive increase in the number of its participants without harming the performance or reliability. "As the number of clients increases so does the capability of the network. While in the client/server model more users would decrease the efficiency of the Network" [6].

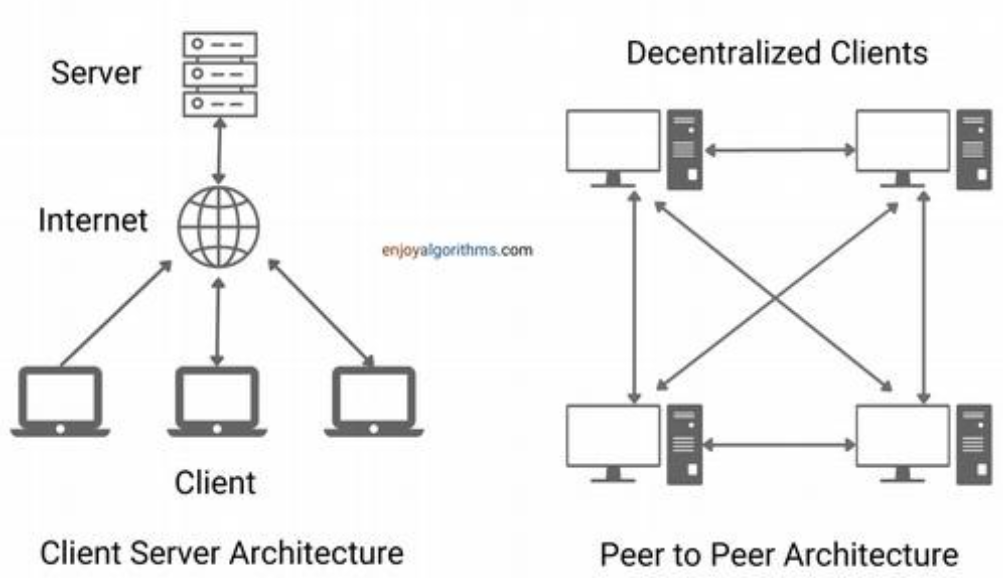


Figure 1: p2p network vs client-server network

"Anonymous approaches are designed with the following three goals: to protect the identity of provider, to protect the identity of requester and to protect the contents of transferred data between them" [10]. The direct communication between the peers allows to transfer encrypted data in Anonymous way. P2P networks use security tools like encryption, authentication, and access controls to let only the right users access shared resources. Without relying on one main server, P2P makes it harder for hackers to attack or gain access without permission.

The P2P network architecture is a fundamental component of blockchain technology, which enables the transfer of cryptocurrencies without the need for an intermediary.

In the blockchain network, each user keeps an updated record of their transactions and digital assets. This contrasts with making transactions through a bank that functions as a central entity that keeps all the data and charges a fee for each transaction. Using a P2P system with blockchain technology allows you to reduce costs significantly and maintain maximum anonymity.

In summary, peer-to-peer technology enables decentralized networks, offering efficiency, strength, and user autonomy. It revolutionizes file sharing, communication, and blockchain transactions, providing cost reduction, resource aggregation, privacy, and scalability.

## 3.2 Symmetric Encryption-

In Symmetric Encryption the same key is utilized for both encrypting plaintext into ciphertext and decrypting ciphertext back into plaintext.

With symmetric algorithms, the decryption key can be calculated from the encryption key. The encryption algorithm uses the secret key to create the ciphertext, which is incomprehensible and can only be read with a decryption key. The sender and receiver must agree on a key before they can communicate secretly. The key must be kept secret from others.

Symmetric encryption offers efficiency and simplicity by utilizing a single key. “Many organizations use symmetric encryption because it is relatively inexpensive” [11].

Additionally, symmetric encryption is often faster than asymmetric encryption, making it suitable for large-scale data transmission and real-time communication systems. However, the challenge in this encryption is in distributing the key securely to the parties involved, as any compromise of the key could potentially compromise the security of the entire system.

Furthermore, there is vulnerability to brute-force attacks, where an attacker systematically tries every possible key to decrypt the message. With sufficient computing power, such attacks can undermine the security of the encrypted data. Symmetric encryption lacks methods for verifying the identity of the sender, unlike asymmetric encryption which employs digital signatures. This absence of authentication means that encrypted messages can be intercepted, altered, and re-encrypted by unauthorized individuals. Without reliable ways to confirm the sender's identity, there is a risk that the transmitted information may be compromised, posing significant challenges in ensuring secure communication channels.

In conclusion, symmetric encryption stands as a foundational technique in information security yet confronts challenges in key distribution and vulnerability.

## 3.3 Asymmetric Encryption-

In asymmetric encryption algorithms, different keys are used for encryption and decryption. Each user in the network needs to create two keys: a private key and a public key.

"Public keys are publicly distributed and used to encrypt, while private keys are kept confidentially by the user, and they are used for decryption." [2]

For each public key there is only one matched private key. The public key is exposed and visible to everyone and it is used for encryption, while the private key is secret and only accessible to its owner, which will decrypt the messages he receives with the matched private key. "In this kind of Encryption Anyone can encrypt a message, but only parties in possession of the private key can decrypt messages." [1]

Thus, when user A wishes to send an encrypted message to user B, he will use B's public key in order to encrypt the message intended for user B. since there is only one matched key to the public key which is the private secret key of User B, only him can decrypt the message and get the original massage .thus the users guarantee secure communication, Only the user for whom the message is intended will be able to decrypt it with his secret key.

The key generation process is fundamental to achieving the required security. Public and private keys are intricately linked mathematically, but deriving the private key from the public key is impossible. This ensures that although the public key is disclosed, the security of the private key remains unaffected.

Furthermore, Digital signatures are crucial in asymmetric encryption, ensuring authentication, integrity, and non-repudiation in digital communication. Created with the private key and verified using the matching public key, they serve as vital security measure.

In addition, ensuring the authenticity of publicly distributed keys is crucial in preventing "man-in-the-middle" attacks. Techniques such as digital certificates and public key infrastructures (PKI) are used for secure key distribution, allowing users to verify the legitimacy of the public key.

Asymmetric encryption, though slower than symmetric encryption, offers robust security for key distribution. It is used in secure email, online banking, digital signatures, and secure network access, prioritizing security in vital applications.

### 3.3.1 Elliptic curve cryptography -

Elliptic curve cryptography (ECC) is a popular encryption algorithm used to protect sensitive information. It is based on elliptic curves, which are mathematical structures with unique properties that allow the creation of strong encryption systems.

ECC is a public-key cryptosystem in which every user possesses a public and a private key. The vital component of any cryptosystem involving elliptic curves is the elliptic group. All public-key cryptosystems have some underlying mathematical operation. Both parties agree upon certain publicly known data items, such as the values of 'a' and 'b' in the elliptic curve equation and the prime number 'p'. The elliptic group is computed from the elliptic curve equation, along with a base point 'B' selected from the elliptic group.

The primary advantage of the ECC encryption algorithm lies in its robust resistance to decryption, even against huge computational power. If a hacker were to obtain the public key and attempt to calculate the private key, the computational effort required would span many years. Additionally, ECC demonstrates efficiency, it is simple to implement and demands fewer computational resources compared to alternative encryption algorithms.

### 3.3.2 Elliptic curve key Generation -

We employ the Elliptic Curve Diffie-Hellman algorithm to generate the shared key. Each user computes their public/private key pair. The public key is derived from the combination of the private key and the base point, while the private key is a selected prime integer. Each side send each other their public keys.

Both take the product of their private key and the other user’s public key. The resulting public key is a point on the elliptic curve.

## 3.4 API -

"An Application Programming Interface (API) is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software." [2]

An API is a method through which companies or internet services expose and share the data they hold with third-party entities in an organized and controlled manner.

It is a type of software interface that provides access to information from an external service, typically through code, which programmers can use to integrate into their own applications.

### 3.4.1 peers API -

A peer API provides developers tools to create direct communication between multiple devices or nodes without relying on a centralized server. It offers functionalities such as establishing connections, exchanging data, managing identities, and handling events related to peer-to-peer communication. Peer APIs are essential for building decentralized systems.

### 3.4.1.1 PeerJS -

PeerJS is a JavaScript library that simplifies the implementation of peer-to-peer communication in web applications like WebRTC.

PeerJS facilitates peer-to-peer communication for data, video, and audio calls. Each Peer object receives a unique, randomly generated ID upon creation.

To establish a connection with another peer, you will need to obtain their unique peer ID, which you can then use to call the destination peer.

PeerJS includes a signaling server implementation known as PeerServer, that helps peers find each other and establish the initial communication.

In contrast to data connections, incoming calls require answering. Otherwise, no connection will be established.

### 3.4.2 WebRTC -

WebRTC (Web Real-Time Communication) is an open-source project that offers web browsers and mobile applications with real-time communication (RTC) through application programming interfaces (APIs). WebRTC allows browsers to stream files directly to one another, eliminating the need for server-side file hosting, enabling each peer to function as an edge server by utilizing the client's bandwidth to upload media to other connected peers. Major companies like Apple, Google, Microsoft, and Mozilla support this open-source project.  
"Peer connections is the part of the WebRTC specifications that deals with connecting two applications on different computers to communicate using a peer-to-peer protocol." [[7](https://webrtc.org/getting-started/peer-connections)]

Developers use these APIs to create peer-to-peer communications between internet web browsers for audio, video, or text-based content, without worrying about compatibility and support. WebRTC enables real-time communication simply by opening a webpage.  
The peer-to-peer connectivity is managed by the RTCPeerConnection interface. That interface represents a WebRTC connection which serves as the main hub for creating and controlling connections between similar applications, facilitating communication between local and remote peers.  
It provides methods to connect to a remote peer, maintain and monitor the connection, and close the connection once it is no longer needed. Additionally, the WebRTC specification includes APIs for interacting with ICE (Internet Connectivity Establishment) Servers, essential for establishing network connections.

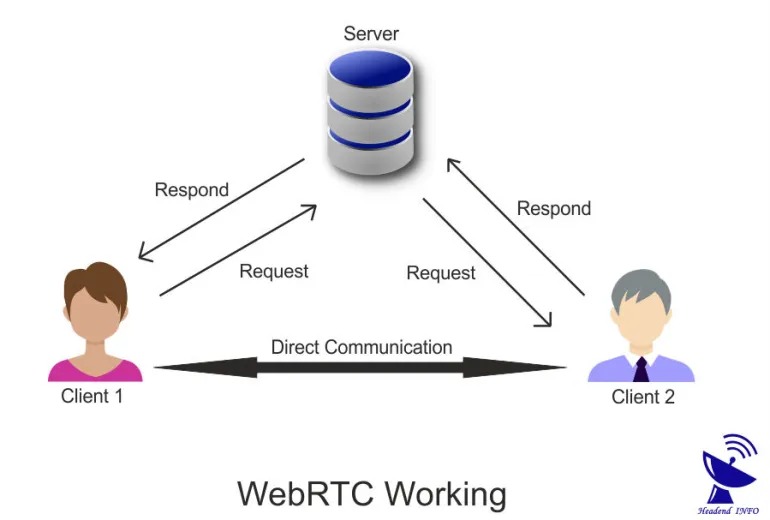


Figure 2: How WebRTC server works

"WebRTC allows browsers to stream files directly to one another, reducing or entirelyremoving the need for server-side file hosting." [14]

## 3.5 Node.js -

Node.js is a runtime environment that allows JavaScript programs to be written and executed outside of the browser. It is an open-source project maintained by a community of programmers.

Node.js enables developers to use JavaScript for various tasks, such as writing command-line tools, server-side scripting, and generating dynamic web page content before sending it to the user's browser. This environment is ideal for building real-time web applications that require a lot of interactivity and fast response times, such as chat applications, social media platforms, and collaboration tools. It is also well-suited for building streaming applications that require real-time processing of large amounts of data, such as video streaming services, music streaming services, and online gaming platforms.

Additionally, Node.js has an event-driven architecture capable of asynchronous I/O operations. It is not a programming language or a framework, but a runtime environment that offers numerous benefits to developers worldwide.

## 3.6 NPM –

NPM (Node Package Manager) is the default package manager for the JavaScript runtime environment, Node.js. NPM helps developers easily find and use pre-written packages, making project development faster.

With NPM, developers can quickly install, update, and remove packages using simple commands, as well as share their own packages with the NPM community. It is a handy tool that simplifies building software with Node.js.

### 3.6.1 Elliptic Package –

The Elliptic package, available on NPM, is a tool for implementing elliptic-curve cryptography in JavaScript. It allows developers to perform secure cryptographic tasks such as signing, verifying, and generating keys efficiently using elliptic curves. Elliptic is known for its fast performance compared to other JavaScript cryptography tools. It is a valuable resource for developers looking to incorporate strong and efficient cryptography into their JavaScript projects.

### 3.6.2 Ethers Package –

"The ethers.js library in NPM aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem."[5]

The Ethers package is a TypeScript-based library for Ethereum, a digital platform for transactions and applications. It ensures the security of private keys and supports various wallet formats. Ethers enables interaction with the Ethereum network to access data and perform transactions. It supports Ethereum Name Service (ENS) names, providing human-readable addresses like domain names on the internet, simplifying Ethereum transactions.

"It is often used to create decentralized applications (dApps), wallets (such as MetaMask and Tally), and other tools and simple scripts that require reading and writing to the blockchain." [5]

# 4. Expected Achievements

We aim to expand the range of secure file transfer methods available over the Internet. Our goal is to develop an accessible and user-friendly website suitable for both individual and organizational use. Through our website, users can securely transfer files in encrypted form without exposing them to third parties, all while maintaining complete anonymity. Our platform allows the transfer of sensitive and confidential files through peer-to-peer communication. This communication is initiated by a personal invitation to a chat meeting between the two users who want to share files.

By using asymmetric encryption algorithms like RSA or Elliptic Curve Cryptography, we ensure that only the intended recipient possesses the decryption key, thereby preventing unauthorized access to files. Furthermore, our platform prioritizes user anonymity and data privacy. Each user is assigned a unique ID, ensuring the confidentiality of their personal information throughout the file transfer process.

Our platform is versatile and can handle various file formats and sizes, including sensitive documents, multimedia files, and large datasets. Our goal is to provide a reliable solution for secure file transfer, enabling the sharing of sensitive information without compromising privacy or security. With our user-friendly interface, strong encryption protocols, and commitment to anonymity, we aim to redefine the standards of secure file sharing in the digital age.

## 4.1 Success Criteria

-Accessible and user-friendly website for individual and organizational use.

-User details remain confidential and are not revealed.

-Encrypted messages cannot be deciphered by hackers.

-Minimal reliance on intermediary servers.

-High level of user satisfaction with platform security measures.

-Efficient and quick file transfer process.

-Positive feedback from users regarding ease of use and accessibility.

-Compliance with relevant privacy and security regulations.

## 4.2 Unique Features

- Free accessibility.

- Secure transfer of files through encrypted communication.

- Peer-to-peer communication for direct file sharing.

- Implementation of asymmetric encryption algorithms for enhanced security.

- User anonymity and privacy protection through unique user IDs.

- Support for various file formats and sizes, including sensitive documents.

# 5. Engineering Process

## **5.1 Process**

In Part A of our project, we conducted thorough research into encryption and network security to determine the most effective communication methods for ensuring user privacy. After evaluating various solutions, we selected the most suitable option for our project's requirements. As we progressed to more advanced stages, we focused on detailed planning and engineering development for the site, addressing anticipated challenges and meeting the project goals.

In Part B of our project, At the beginning of the development process, we reviewed important libraries needed for building the website, such as PeerJS, Ethers, and Elliptic. In our project, we needed the user's PeerID to be their public encryption key, which meant we had to modify PeerJS and combine it with a cryptography library to generate the keys. We researched the key generation process by going through the documentation of these libraries. During the development, we also looked at existing platforms, like popular chat services and file-sharing websites, to make sure our system followed common standards.

Initially, we used encryption keys from the Elliptic library, but their size and format didn’t match what PeerJS required for public keys. To solve this, we switched to the Ethers library, where we created a wallet that gave us public, private, and mnemonic keys, which we used throughout the project.

## **5.1.1 Tools and** clie**nt interaction**

In our project we involved our friends and family during the development process by asking them to test the website and try out its features. Based on their feedback, we identified areas where we needed to improve user notifications and guidance. This helped us make the system more user-friendly, especially for people who were using the platform for the first time. By adding additional prompts and feedback messages, we ensured that even new users could easily navigate the system.

For efficient development, we used the following tools:

**PeerJS:** Used for implementing peer-to-peer communication between users.

**Ethers:** Used for generating encryption keys and utilizing digital wallets to ensure privacy and confidentiality in communication.

**Elliptic**: A library used early in the project for generating encryption keys based on elliptic curves.

**NPM**: The package manager used to add various libraries to the project, such as PeerJS and Ethers.

**Preact:** Used for building the UI efficiently and quickly, with a focus on better performance for smaller projects.

**Tailwind CSS**: Used for creating a visually accessible interface with a modern and simple appearance.

**GitHub:** Used for code collaboration and tracking, enabling simultaneous work, version control, and efficient development. It allowed us to revert to previous points and maintain a detailed code history.

### 5.1.2 Networking and Servers

#### In our project, we aim to ensure decentralized communication, minimizing dependence on intermediaries. To achieve this, we examined various communication methods within the network, utilizing academic articles and online research. We compared these methods based on several parameters, including security, scalability, latency, and ease of implementation.

### 5.1.1.1 PeerServer

After a comprehensive review of network servers, we chose to use the PeerServer, which is part of the PeerJS library. We utilize this only for the initial communication and for managing the IDs of peers within the direct network we've established. Here are the steps we will follow:

**1. Initialization:**

Each peer initializes their PeerJS client library within their application and configures it to connect to the PeerServer at the application layer.

**2. Peer Registration:**

In Peer Registration, the PeerJS library typically assigns a unique ID to each peer upon connection to the server. However, in our implementation, the user's ID is the public key created in the first step of the key generation function. The PeerServer then verifies the validity and availability of these IDs. Additionally, the PeerServer maintains a list of registered peers along with their connection details.

**3. Server signaling:**

Server signaling involves the exchange of session data, including the session's start and end times, network details like IP address and port, as well as preferences and settings. Peers then reach an agreement on the session parameters. The PeerServer queries its registry to identify suitable peers based on the request criteria.

Once signaling is complete and the connection is established, the peers communicate directly with each other, bypassing the PeerServer for data transmission. Data is exchanged directly between the peers over the established connection, enabling real-time communication or data transfer.

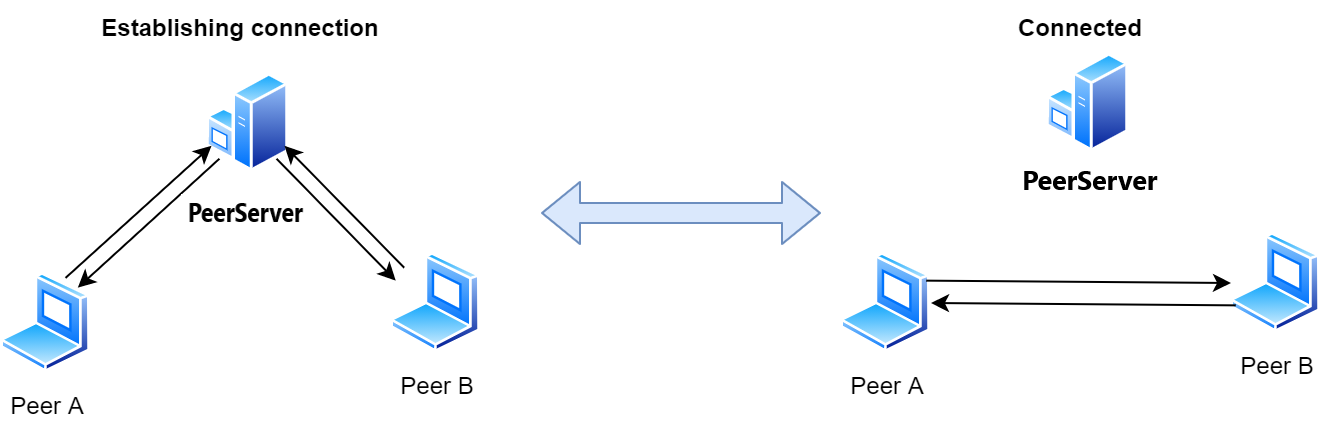


Figure 3: How p2p server works

The PeerServer continues to monitor the connections between peers, including peer presence detection, disconnection handling, and reconnecting. This ensures efficient and reliable peer-to-peer communication within the PeerJS interface.

### 5.1.3 Encryption

In our project, prioritizing the utmost security of file transfer within our chat system, we meticulously compared two encryption methods: symmetric and asymmetric. After thorough analysis, we chose asymmetric encryption. Despite its higher resource consumption and costs, its significantly enhanced security made it the ideal choice for our project's objectives. We designed our encryption protocol with meticulous care: the public key for message encryption is derived from each user's network ID, ensuring accessibility for encryption, while the private key remains confidential, granting sole decryption capability to the intended recipient. This approach not only fortifies the privacy of transferred files but also aligns seamlessly with our project's overarching goal of robust data security.

### 5.1.3.1 Generate Public and Private Keys

We utilize functions from the Ethers library to generate both a public**,** private keyand mnemonic. Initially, the ethers library handles the key generation, including the use of the secp256k1 Elliptic Curve.

### 5.1.3.2 Encryption and Decryption implementation

After we generate the public-private key pairs for both the sender and the recipient using elliptic curve cryptography, we will use functions from the TweetNaCl.js library for the encryption and decryption process. To encrypt data, we will use the nacl.box function with parameters including the sender's private key, the receiver's public key, and a nonce. A nonce, a "number used once," is a randomly generated value used only once during the encryption process. This ensures that each encrypted message appears unique, even when the same data is sent repeatedly.

For decryption, we will use the nacl.box.open function. This function allows the recipient to decrypt the message using their private key, ensuring that only the intended recipient who owns the correct private key can access and decipher the original content.

## 5.1.4 Client-side

Client-side technologies play an important role in shaping the user experience of web applications. The technologies we chose can significantly impact performance and user satisfaction. In our project, we selected the best server-side technologies that would guarantee optimal performance, scalability, and maintainability.

### 5.1.4.2 Tailwind CSS

In our project, we prioritize ensuring a comfortable and intuitive user experience, ensuring that all users can navigate the site without any prior knowledge. To achieve this goal, we chose to use Tailwind CSS. This framework offers pre-prepared design classes that facilitate the creation of responsive and visually appealing interfaces with minimal complexity. Using these classes will allow us to design quickly, without writing long and complicated CSS code. In this way, we will develop a website that is pleasing to the eye and enjoyable for the user, and we will also save a lot of development time.

# **5.2 Product**

## 5.2.1 client interface

In designing our website, we prioritized easy navigation, appealing visuals, and seamless adaptability across all devices to ensure a smooth user experience. The platform offers a modern, user-centered design for secure encryption while maintaining a clean and friendly interface.

Our site is fully optimized for smaller screens, displaying only the most relevant information to avoid clutter. However, users can always access additional details by clicking a designated button, ensuring an effortless and intuitive experience.

Upon entering the site, one-time encryption keys are automatically generated for each user, enabling secure P2P communication, as well as the encryption and decryption of files.

## 5.2.2 home page

On our updated homepage, visitors are greeted with a brief overview of the website and its functionalities. To make the user experience smoother, we have added a Frequently Asked Questions (FAQ) section, addressing common inquiries from users who have tested the site, making it easier to understand the usage process.

Additionally, we have included user testimonials to showcase feedback from those who have used our services. Visitors can also reach out to us directly through the contact form, which allows them to send us inquiries via email.

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

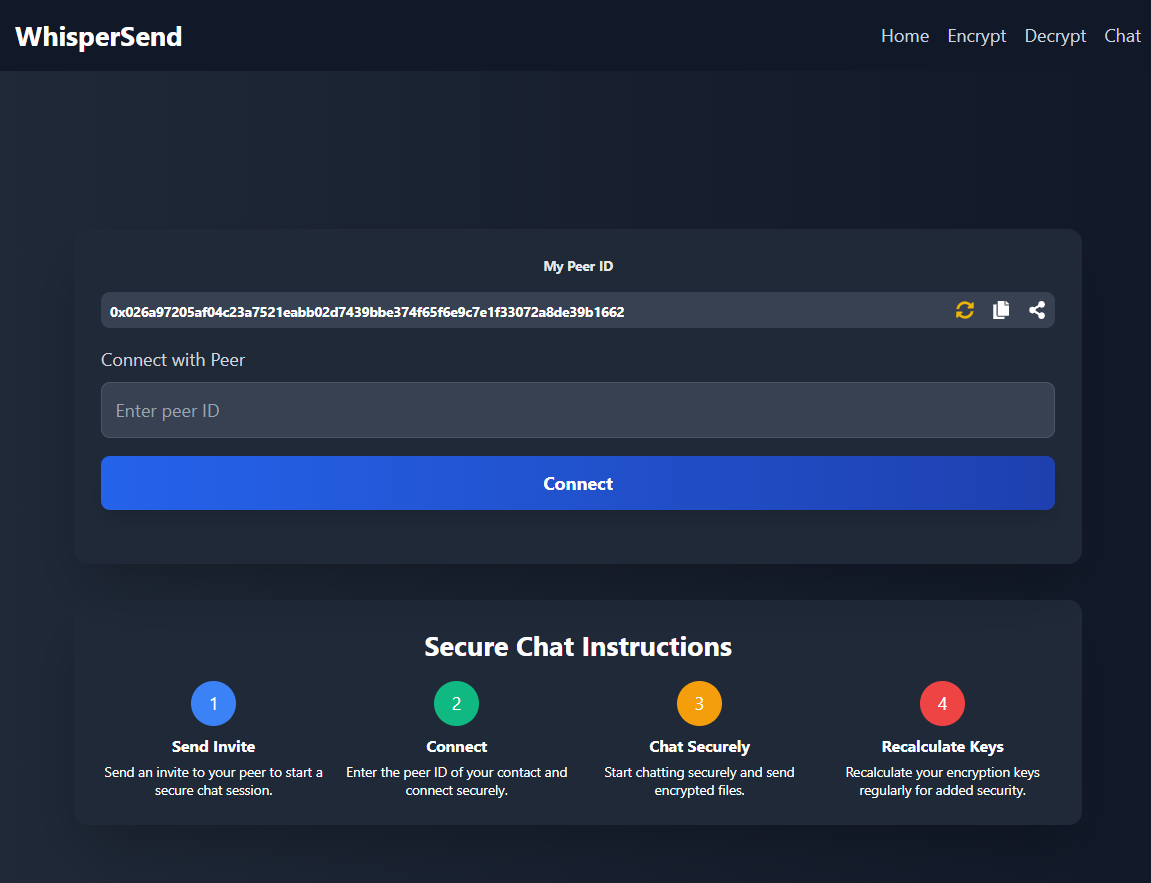
## 

## 5.2.3 Chat Page

When a user enters the chat page, they must input the PeerID of the person they wish to communicate with and wait for that user's approval. At any time, the user has the option to cancel the connection request or terminate the Peer connection.

There is the option to recalculate the encryption keys by a dedicated icon in the chat window. The user can send his connection information through a variety of social networks or through email.

Similarly, the recipient can choose to decline the connection request. If an invalid PeerID is entered, the connection will not be established. Once the correct PeerID is provided and the request is accepted, both users will be redirected to the chat page.



On the chat page, users can send regular or encrypted files, receive decrypted or regular files, and securely send messages and emojis.

At any time, users can end the conversation by clicking the disconnect button. This will notify the other participant that the user has disconnected. The remaining user can choose to stay in the chat to view the previous conversation or disconnect as well.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

## 5.2.4 Mnemonic Phrase

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטיThe mnemonic modal displays a sequence of words that represent the user's keys for the current session on the website. This page will appear only once during browsing and specifically on the encrypt, decrypt, and chat pages. Users must save their mnemonic to ensure they can decrypt files in future sessions that were encrypted with their previous keys. The user is required to confirm that they have saved the mnemonic. Otherwise, the notification will remain visible.

## 5.2.5 Self Encryption

On the encryption page, the user is required to upload a file they wish to encrypt. The file will be encrypted using the mnemonic stored for their current session on the site. If the user wishes to decrypt the file later, they must enter the same key that was used during the encryption process.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

## 

## 5.2.6 Self Decryption

On the decryption page, users can upload an encrypted file and choose from two options for decryption:

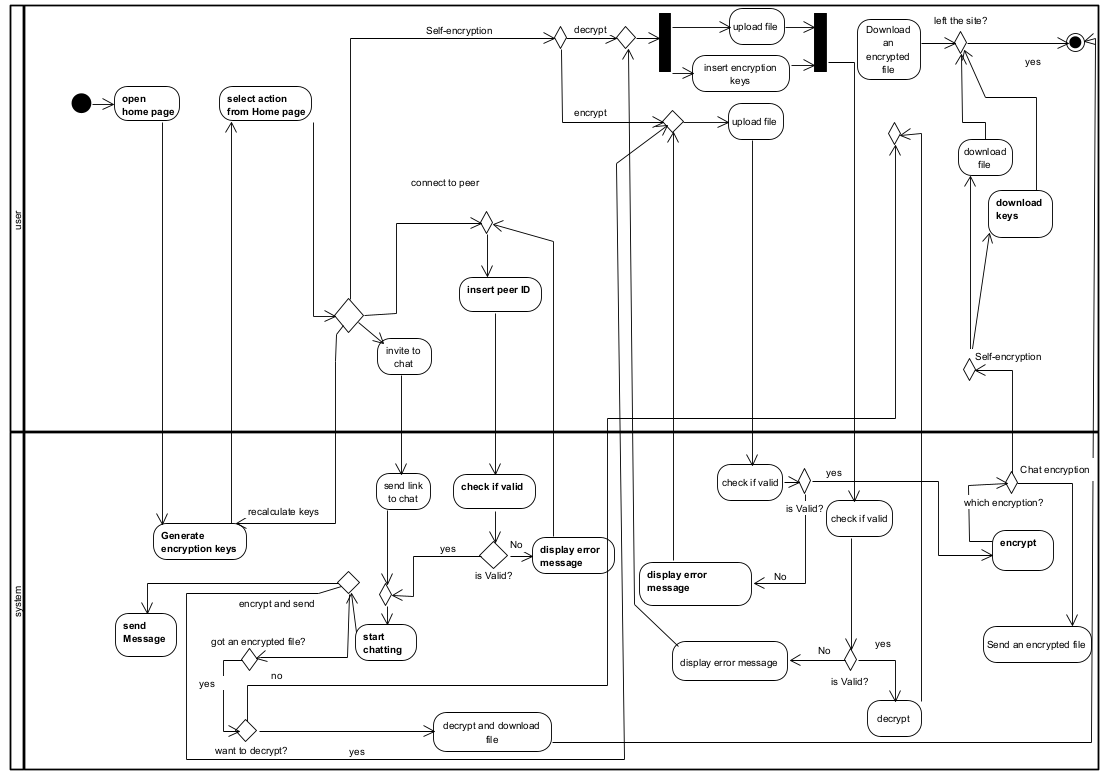
**Input an old mnemonic**: The user can use a mnemonic saved from a previous session to decrypt older encrypted files.

**Use the current session's mnemonic**: If no key is entered, the file will be decrypted using the mnemonic from the current session.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

## 5.2.7 Model and Diagrams

Activity diagram:

# 6. Verification and Evaluation

## 6.1 Testing plan

| **Test ID** | **Function Under Test** | **Procedure** | **Outcome Expected** | **Pass/Fail** |
| --- | --- | --- | --- | --- |
| 1 | Secure File Transfer | User attempts to transfer an encrypted file to another peer. | The file is transferred securely without errors. | **Pass** |
| 2 | Encryption Process | A file is encrypted using the provided public key. | The file is encrypted correctly and only decryptable by the intended recipient's private key. | **Pass** |
| 3 | Decryption Process | An encrypted file is decrypted using the recipient's private key. | The file is decrypted correctly, and contents match the original. | **Pass** |
| 4 | Peer Connection | Establish a direct peer-to-peer connection between two users. | The connection is established without reliance on intermediate servers. | **Pass** |
| 5 | Key Generation | Generate a pair of public/private keys upon user registration. | The keys are generated correctly and securely stored.  The keys are unique. | **Pass** |
| 6 | Recalculate Keys | A user requests to regenerate their public/private key pair. | New encryption keys are generated, and the peer ID is updated on the home page. | **Pass** |
| 7 | Real-Time Messaging | Send a message through the peer-to-peer network. | Message is sent and received in real-time without delays. | **Pass** |
| 8 | System Load Handling | Simulate high traffic by multiple users accessing the system simultaneously. | System remains stable and responsive under load. | **Pass** |
| 9 | Continuous Operation | Test system uptime over an extended period without interruptions. | System operates continuously without downtime or performance degradation. | **Pass** |
| 10 | Error Handling | Test response to unsupported file types or corrupted data inputs. | System displays appropriate error messages and does not crash. | **Pass** |
| 11 | Multi-Device Support | Access the system simultaneously from different types of devices. | Consistent user experience across all devices without functionality loss. | **Pass** |
| 12 | Access control | Attempt to access files without proper authorization permissions. | System enforces access control policies based on user roles and permissions, restricting access to authorized users only. | **Pass** |
| 13 | Send the Same File in Chat | |  | | --- | | User attempts to send the same file multiple times in a chat. |  |  | | --- | |  | | |  | | --- | | The same file can be sent again without errors. | | **Pass** |
| 14 | Send Encrypted/Regular File | |  | | --- | | User sends an encrypted or regular file in the chat. |  |  | | --- | |  | | |  | | --- | | Both encrypted and regular files are successfully sent and delivered correctly to the recipient. |  |  | | --- | |  | | **Pass** |
| 15 | |  | | --- | | Disconnect Notification |  |  | | --- | |  | | |  | | --- | | A user clicks the **disconnect** button during a chat session. |  |  | | --- | |  | | |  | | --- | | A notification is displayed to the remaining user, indicating that the other user has disconnected. |  |  | | --- | |  | | **Pass** |
| 16 | |  | | --- | | Mnemonic Confirmation Display |  |  | | --- | |  | | |  | | --- | | User has not confirmed saving their mnemonic. |  |  | | --- | |  | | |  | | --- | | The mnemonic notification remains visible until the user confirms saving it, as it is a requirement for usage. |  |  | | --- | |  | | **Pass** |

# 7. Challenges and Solutions

During the development of the project, we encountered several significant technical and engineering challenges, both analytical and technological. These challenges primarily concerned decentralized communication, secure encryption mechanisms, key management, and performance optimization. Here is a description of the challenges we meet and the solutions we implemented to solve them:

1) Ensuring decentralized peer-to-peer communication without over-reliance on central servers.

Solution: Implemented a PeerServer from the PeerJS library, used solely for initial communication. After peers are connected, they communicate directly, minimizing central server dependency. This architecture also handles peer presence detection and reconnections, ensuring reliable communication.

2) Simulating Chat Features

In our project, we wanted to create a chat system that works like those found on social media platforms. This includes being able to join and leave conversations, cancel chats, and notify users when someone exits the chat.

Solution: We explored the PeerJS library in detail and used its features to manage how participants connect and disconnect from chats. PeerJS helped us handle real-time communication between users and provided the tools to notify participants when someone leaves the conversation.

3) Ensuring that large and sensitive files can be securely transferred between users without compromising privacy.

Solution: Files are encrypted before being sent, using the nacl.box function from the TweetNaCl.js library. A nonce is generated to ensure each encryption is unique. Decryption is only possible using the correct private key, ensuring privacy.

4) Using window confirm for User Notifications

At first, we used window confirm for all user confirmation messages because it was easy to implement. However, we found that it didn’t always work well in different browsers. Also, if the site’s window was minimized or a different tab was open, the confirmation message didn’t appear.

Solution: We replaced window confirm with a custom modal window. This modal adjusts based on the message and works reliably across browsers. We created a general component that defines the modal, allowing us to display different messages depending on the situation. This improved both functionality and user experience.

# 8. Conclusion

**Did We Achieve the Project Goals?**

We successfully achieved the goals set for the project.

**Secure File Transfer Platform**: Our main goal was to create a platform that allows secure file transfers using encryption. We achieved this by implementing asymmetric encryption and a Peer-to-Peercommunication system. This ensures that data is transferred directly and securely between users, without relying on a central server.

**User Privacy**: We ensured privacy by generating unique encryption keys for each session, which are automatically deleted when the user leaves the site. These keys are not stored on the server, protecting users' data.

**Performance and User Experience**: By using Preact and Tailwind CSS, we were able to develop a fast and user-friendly platform that works well on different devices, improving both performance and ease of use.

**Supporting Diverse Needs**: The system supports a wide variety of file types and sizes, including sensitive documents, making it suitable for both individual users and organizations looking to securely transfer important information.

The project successfully achieved its goals by creating a secure, efficient, and easy-to-use platform for file transfers that prioritizes user privacy.

**Decision-Making Considerations**

-We wanted to ensure users received clear instructions on how to use the platform without affecting the site's performance or design. To achieve this, we added concise explanations within each section of the platform, making them easily accessible while maintaining a clean and efficient user experience.

-We had to decide how to securely transfer the public key and establish a connection with another peer while minimizing the exposure of user details. To achieve this, we designed a system where only the necessary information is shared during the connection process, ensuring that no sensitive data is revealed.

**Did We Work the Right Way? What Would We Change?**

At the start of the project, we worked on the encryption and decryption pages first, and only later focused on the chat feature. Looking back, this wasn’t the best decision. The chat page turned out to be the most complicated and difficult part, which we didn’t realize when we started. We thought it made sense to first build the encryption functionality and user key generation as the foundation. However, connecting peers and starting the chat communication was much harder than we expected.

# 9. References

*[1] An ISO 3297: 2007 Certified Organization. A review on asymmetric cryptography – RSA and ElGamal algorithm.*

*[2] Berlove, O. Public – Private Key Pairs & How they work. (G. Laroche, Reviewer).*

*[3] CoachButterfly3935. (2024, January 31). Peer to Peer (P2P) Architecture [PDF]. College Sidekick. Retrieved from https://www.collegesidekick.com/study-docs/2680116*

*[4] Digital Assets- CFTE. What is a Peer-to-Peer Network in Blockchain? Retrieved from https://blog.cfte.education/what-is-p2p-network-blockchain/*

*[5] Ethers Documentation. Ethers documentation. Retrieved from https://docs.ethers.org/v6/*

*[6] GE-IJER. (2013, December). Peer to peer computing. GE International Journal of Engineering Research, ISSN: 2321-1717.*

*[7] Google for Developers. WebRTC. Retrieved from https://webrtc.org/?hl=he*

*[8] Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., & Xu, Z. (2003). Peer-to-peer computing. HP Laboratories. HPL-2002-57 (R.1)*

*[9] NewSoftwares.net. Dark side of symmetric key encryption. Retrieved from https://www.newsoftwares.net/blog/dark-side-of-symmetric-key-encryption/*

*[10] Saboori, E., & Mohammadi, S. (2012). Anonymous communication in peer-to-peer networks for providing more privacy and security. International Journal of Modeling and Optimization, 2(3).*

*[11] TechSlang. What is symmetric encryption? Retrieved from https://www.techslang.com/definition/what-is-symmetric-encryption/*

*[12] Wallach, D. S. (2003). A survey of peer-to-peer security issues. Rice University, Houston, TX. Retrieved from https://www.cs.rice.edu/~dwallach/pub/tokyo-p2p2002.pdf*

*[13] Wikipedia contributors. (2023a). API. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/API*

*[14] Wikipedia contributors. WebRTC. In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/WebRTC*