International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

SHARENCRYPT: SECURE PEER-TO-PEER FILE SHARING

Vishvam J. Joshi[1], Hardik Singh[2], Hanshil Raithatha[3], Pragneshkumar Singh[4]

[1]*CSE(Cyber Security), Parul Institute Of Engineering & Technology*
[2]*CSE(Cyber Security), Parul Institute Of Engineering & Technology*
[3]*CSE(Cyber Security), Parul Institute Of Engineering & Technology*
[4]*CSE(Cyber Security), Parul Institute Of Engineering & Technology*

Abstract - The rise of decentralized communication has established peer-to-peer (P2P) networks as a powerful tool for data exchange. However, these networks often face significant security risks, legal issues, and connectivity challenges. This paper introduces 'Sharencrypt', a final year project that addresses these limitations by developing a secure, hybrid P2P file-sharing system. The architecture synergistically combines WebRTC for direct, high performance data transfer with Web-Sockets for reliable signalling and peer discovery. At its core, the system's security model ensures that all file transfers are protected by end-to-end encryption using the AES-GCM algorithm. The project's methodology is based on a rigorous, multi-stage testing process that validates its functional, performance, and security attributes. The results confirm that this hybrid architecture successfully overcomes common P2P network obstacles, providing a robust and scalable solution. Performance analysis demonstrates the efficiency of the WebRTC data channel, while security audits validate the integrity of the cryptographic and authentication mechanisms. This work concludes with an evaluation of the system's strengths and limitations, providing a roadmap for future research in scalable and anonymous P2P communication.

Key Words: PEER TO PEER FILE SHARING, WEBRTC DATA Channel, AES GCM, HYBRID NETWORK ARCHITECTURE

1. INTRODUCTION

The peer-to-peer (P2P) networking model represents a fundamental departure from the traditional client server architecture, decentralizing control and functionality across a network of interconnected nodes. [3] In a P2P system, each participant, or peer, functions as both a client and a server, directly sharing resources with other peers without relying on a central server for core operations. [11] This model's primary appeal lies in its efficiency and scalability; by distributing the workload among the participants, the network can handle immense data traffic without a single point of failure or a central server bottleneck. [3]

The conventional client-server model, in contrast, de pends on a powerful central server to provide services to multiple clients, which simplifies management and control but introduces several critical drawbacks. [11] A server can become a single point of failure, and as user demand increases, it can become overloaded, leading to performance degradation. [11] Modern solutions, including 'Sharencrypt,' often adopt a hybrid P2P approach that uses a central server for specific, lightweight tasks, such as peer discovery and signalling, while maintaining the direct peer-to-peer data exchange that provides the network's efficiency. [10]

1.2. Problem Statement: The Imperative for Secure File Sharing

Despite their architectural advantages, P2P networks have been associated with significant security vulnerabilities, legal complexities, and practical challenges. [5] A primary concern is the potential for malware and phishing attacks. The decentralized nature of P2P file sharing makes it difficult to verify the source of a file, allowing malicious software to be easily disguised as legitimate content, which can lead to widespread infections. [5] This unverified trust model also poses a risk of data breaches, where a user could accidentally share private or sensitive information stored in a shared folder, exposing personal or corporate data. [5] Beyond direct security threats, P2P networks face significant legal and ethical issues, with copyright infringement being the most prominent. [5] The unauthorized sharing of copy righted material has led to severe legal repercussions for individuals and platforms involved in illegal distribution. [5] Civil penalties for copyright infringement can be substantial, and in cases of wilful infringement, fines can be as high as hundreds of thousands of dollars. [5] The 'Sharencrypt' project was developed to address these critical issues by creating a system that not only enables efficient file sharing but is built with security, reliability, and compliance as core design principles.

1.3. Project Objectives and Scope

The main objective of this project is to create, im plement, and validate 'Sharencrypt,' a secure P2P file sharing system. This platform is designed as a demon strative solution that effectively balances the perfor mance benefits of decentralized networks with the strict security and reliability requirements of modern applica tions. The specific goals of this project are:

- To implement a hybrid P2P architecture that uses a central server for signalling and peer discovery, thereby overcoming the network traversal challenges of pure P2P networks. [10]
- To ensure the confidentiality and integrity of all file transfers by implementing robust end-to-end encryption (E2EE) using an industry-standard algorithm, thereby protecting data from interception and tampering. [6]
- To design a system that meets critical non-functional requirements such as security, performance, scalability, and usability, ensuring it is a practical and effective solution. [9]

To conduct a comprehensive, multi-stage testing process, including functional, performance, and security testing, to rigorously analyse the system's effectiveness and identify any shortcomings.

2. SYSTEM REQUIREMENTS

This section outlines the functional and non-functional requirements that guided the design and implementation of the 'Sharencrypt' system. These requirements serve as a comprehensive blueprint and a standard against which the final product is measured. [9]

2.1 Functional Requirements

Functional requirements define the core features and actions the system must perform to enable users to accomplish their tasks. [9]

- **User Management:** The system must provide a secure mechanism for user registration and login. This includes requiring a unique username, pass word, and email address, followed by a successful login attempt that grants access to the application. [9]
- **File Transfer:** The primary function is to allow users to upload files and transfer them to another peer within the network. The system must support the transfer of various file types and sizes. [9]
- **Peer Discovery:** The system must enable a user to discover and connect with other online peers. This includes a mechanism to find peers by their unique identity. [10]
- **Transfer Status:** The system must provide real time feedback on the status of a file transfer, including a progress bar, a completion percentage, and the transfer speed. [9]
- **Error Handling and Logging:** The system must gracefully manage all foreseeable errors, such as connection failures or data corruption. It must also maintain a detailed log of system activities and errors for troubleshooting and auditing purposes. [9]

2.2 Non-Functional Requirements

Non-functional requirements specify the criteria for evaluating the quality and performance of the system. These are crucial as they define the 'how' rather than the 'what' of the solution. [9]

- **Security**: This is the most important non-functional requirement. The system must encrypt all data transmitted between peers using a robust, industry standard encryption algorithm to protect against eavesdropping and data tampering. [5] It must also implement proper authentication and authorization to prevent unauthorized access and protect against common attacks. [8]
- **Performance:** The system must handle file transfers with minimal end-to-end latency and high throughput. Performance must remain stable even under concurrent user load, ensuring a responsive user experience. [4]
- **Scalability:** The system's architecture must be de signed to accommodate an increasing number of users and concurrent connections without a significant decline in performance. [10]
- **Usability:** The user interface must be intuitive and easy for a wide audience to learn and operate. [9]
- **Reliability:** The system must demonstrate consistent performance and data integrity. It must be resilient to connection drops and network failures, with mechanisms to ensure that transferred files are not corrupted and can be recovered if a transfer is interrupted. [9]

A comprehensive matrix was created to map these requirements to a set of verifiable test criteria. This formal approach ensures that every aspect of the project is systematically validated.

3. SYSTEM ARCHITECTURE AND DESIGN

The architectural design of 'Sharencrypt' is the cornerstone of its functionality and security. The design choices were made to specifically address the limitations of traditional P2P models and to harness the latest web communication protocols for a robust and efficient solution.

3.1 Architectural Paradigm: A Hybrid Peer-to-Peer Model

The choice of a hybrid P2P architecture was a critical design decision driven by the shortcomings of both pure P2P and pure client-server models. [3] A purely P2P sys tem, while conceptually appealing, is nearly impossible to implement reliably in a browser-based environment due to the pervasive challenges of Network Address Translation (NAT) and firewalls. [10] Without an inter mediary server to help with connection establishment, peers often cannot communicate with each other. [10] Conversely, a pure client-server model for file sharing, where all data is routed through a central server, would inevitably create a massive bottleneck, especially with large files. [11] The hybrid model, therefore, represents a pragmatic middle ground that provides the best of both worlds. [3] 'Sharencrypt' uses a central server not for file transfer but for lightweight, essential tasks like peer discovery and signalling. This design choice recognizes that a server is indispensable for coordinating the network while ensuring that the primary, resource intensive workload of data transfer is handled directly between the peers. [11] This architectural decision balances reliability and scalability with cost-efficiency, a foundational principle for any viable long-term solution.

3.2 Protocol Selection: The Synergistic Use of WebRTC and WebSockets

The selection of communication protocols was cen tral to the system's design. 'Sharencrypt' uses a com plementary combination of WebSockets and WebRTC to achieve its objectives. WebSockets, a protocol that establishes a persistent, two-way connection between a client and a central server, is ideal for the system's signalling component. [11] This persistent, full-duplex connection allows for the lightweight, real-time message exchange necessary for peers to find each other and coordinate the initial connection. [11] For the actual

file transfer, the system transitions to WebRTC. WebRTC is a peer-to-peer protocol specifically designed for direct communication between browsers, bypassing the central server entirely for data exchange. [11] Its RTC Data Channel component provides a high-performance, low latency pathway for transferring arbitrary data, including large files. [7] The decision to use WebRTC for the heavy lifting of file transfer ensures that the system delivers on its promise of high throughput and low latency, while WebSockets handle the critical task of signalling with efficiency.

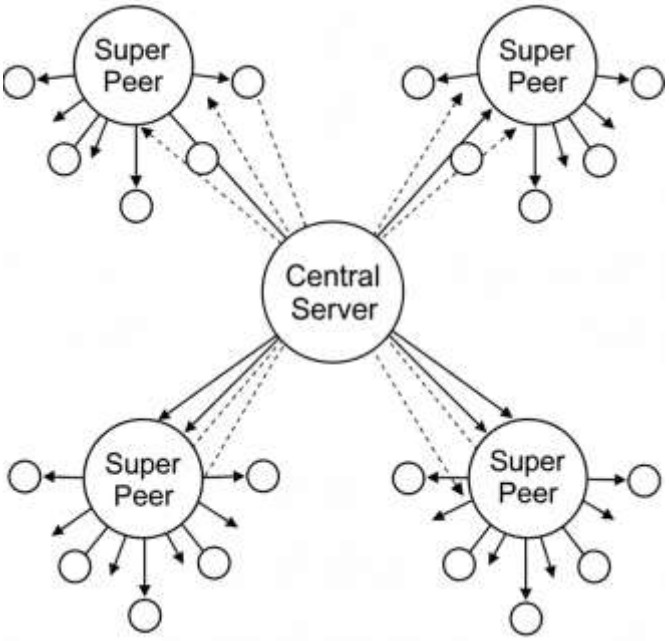

Figure 1. Hybrid P2P System Architecture

3.3 Secure Communication and End-to-End Encryption Design

To address the inherent security vulnerabilities of P2P file sharing, a robust end-to-end encryption (E2EE) model was designed as a core system feature. [5] All file transfers are secured using the AES-GCM (Advanced Encryption Standard-Galois/Counter Mode) algorithm. [12] AES-GCM was selected for its dual capability to provide both confidentiality and data integrity within a single cryptographic operation. Unlike older encryption methods, AES-GCM generates not only ciphertext from the original plaintext but also a unique authentication tag. [12] This tag acts as a digital seal, ensuring that the decrypted content has not been altered in transit. [12] If the received tag does not match the one the receiver's algorithm generates, decryption fails, signal ing that the data has been tampered with. [12] The security of this cryptographic process hinges on proper key and vector management. The system generates a strong, cryptographic key from a user's password using a key derivation function (KDF) like PBKDF2. [2] This process incorporates a unique, random salt to prevent rainbow table attacks and is computationally expensive to deter brute-force attempts. [2] For each encryption operation, a unique Initialization Vector (IV) is used. [2] The system ensures that the unique IV, the authentication tag, and the encrypted file are securely transmitted together over the WebRTC data channel, which itself is encrypted. [12]

3.4 Infrastructure and Network Traversal Mechanisms

A key challenge for any P2P system is successfully establishing connections between peers, a process of ten complicated by NATs and firewalls. [10] To ensure network resilience, 'Sharencrypt' employs a two-tiered network traversal strategy using STUN and TURN servers. [10] The first line of defence is a STUN (Session Traversal Utilities for NAT) server, which allows a peer to discover its public IP address and attempt a direct, peer-to-peer connection. [10] If a direct connection fails, typically due to restrictive firewall policies, the system automatically falls back to a TURN (Traversal Using Relays around NAT) server. [10] The TURN server acts as a relay, forwarding data between the peers when a direct connection is impossible, thereby guaranteeing connectivity. [10] This fallback mechanism is essential for the system's reliability but introduces a performance penalty. While the P2P design minimizes server band width use, any data relayed through a TURN server will incur a charge based on data volume, which is a critical consideration for commercial implementation. [13]

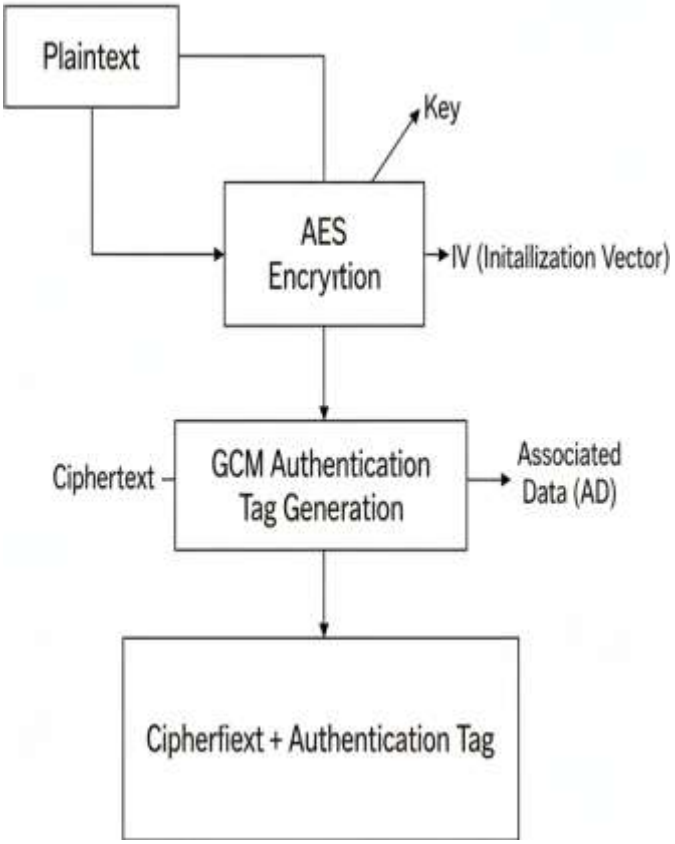

Figure 2 AES-GCM Cryptographic Flow

Table -1: System requirements matrix

Requirement ID	Description	Type	Validation Criteria
REQ-FN-01	User Authentication	Functional	A successful login attempt with a registered username and correct password returns a valid session token.
REQ-FN-02	File Transfer	Functional	A file sent from Client A is received by Client B with a successful integrity check.
REQ-FN-03	Peer Discovery	Functional	Client A can successfully locate and list Client B when Client B is online.
REQ-NFR-01	End-to-End Encryption	Non-Functional	Intercepted network traffic between peers contains unintelligible cipher text.
REQ-NFR-02	Performance	Non-Functiona	A 100MB file is transferred between two peers in under a predefined time limit, and the latency remains stable under a specified concurrent load.

Table -2: Protocol comparison (Webrtc V/s WebSockets)

Feature	WebRTC	Web Sockets
Architecture	Peer-to-Peer (P2P) [11]	Client-Server [11]
Primary Function	High-performance, real-time data transfer [7]	Real-time, bi-directional messaging [11]
Underlying Protocol	Typically UDP for media, but can use TCP [11]	TCP [11]
Latency	Higher setup latency, but very low for direct data transfer [4]	Very low for messaging, but higher for file relay due to server overhead [7]
Bandwidth	Optimized for efficient media streaming [7]	Lower overhead for messaging [7]
Security	Mandatory end-to-end encryption (DTLS-SRTP) [11]	Requires wss:// and token-based authentication [8]

4. IMPLEMENTATIONS

This section details the specific technologies and methodologies used to build the 'Sharencrypt' system, from the selection of the core technology stack to the implementation of the cryptographic and security modules.

4.1 Technology Stack

The project's technology stack was carefully selected to support the real-time, high-performance requirements of a hybrid P2P application. The backend is built on Node.js using the Express.js framework. Node.js is uniquely suited for this task due to its event-driven, non-blocking I/O model, which allows it to handle a large number of concurrent connections and data streams with minimal latency. [9] For managing these connections, the system uses the Socket.io library, which simplifies the implementation of the WebSocket protocol and provides robust features for handling bidirectional communication and disconnections. [9] The cryptographic module is implemented using a secure, pre-vetted Node.js library for AES-GCM. The frontend is built using a mod ern JavaScript framework to create a responsive and user-friendly graphical interface that facilitates the file sharing process.

4.2 Cryptographic Module: Implementation of AES GCM

The core of 'Sharencrypt's security model lies in the implementation of the AES-GCM cryptographic flow. The process begins with key derivation from a user's password using PBKDF2. [2] This KDF takes the pass word, a unique random salt, and a large number of iterations to produce a robust, cryptographically-strong key. [2] This process ensures that two users with the same password will have different keys and protects against brute-force and dictionary attacks. [2] Once a file is selected for transfer, the system proceeds with the encryption phase. The plaintext file is combined with a unique, randomly generated Initialization Vector (IV) and the derived

encryption key. [12] These inputs are fed into the AES-GCM algorithm, which produces two outputs: the ciphertext (the encrypted file content) and a separate piece of data called the authentication tag. [12] The tag is a unique signature that confirms the integrity and authenticity of the data. The ciphertext, the IV, and the authentication tag are then securely transmitted over the WebRTC data channel. [12] Upon receiving the encrypted data, the destination peer initiates the decryption and verification process. The receiver's system provides the ciphertext, the IV, the authentication tag, and its own derived secret key to the AES-GCM decryption algorithm. The algorithm attempts to decrypt the content while simultaneously recalculating its own authentication tag based on the received data. It then compares its newly generated tag with the one provided by the sender. If the two tags match perfectly, the algorithm releases the original, decrypted file. If they do not match, the process is terminated, and an error is signaled. This failure is a crucial security feature, as it indicates that the data has been altered or the wrong key was used, thereby protecting the user from receiving corrupted or malicious data. [12]

4.3 Secure Connection and Authentication Handling

The security of 'Sharencrypt' is not limited to its cryptographic module; it extends to the very foundation of its network connections. The WebSocket signalling server, while handling only non-sensitive metadata, is a critical attack vector. To prevent eavesdropping and man in-the-middle (MITM) attacks, the server is configured to use the wss:// protocol, which ensures that all signalling traffic is encrypted with TLS/SSL. [8] Furthermore, to prevent unauthorized access and protect against Cross Site WebSocket Hijacking (CSWH), the system implements a robust, token-based authentication mechanism. After a user successfully logs in, the server issues a secure token. This token is then used to authenticate all subsequent WebSocket handshakes, ensuring that only authenticated users can access the signalling server. [8] This security measure is crucial because WebSockets, by default, do not handle authentication. [8] This multi layered approach to security, which addresses both data encryption and connection authentication, is essential for a truly secure P2P application.

5. TESTING AND VALIDATION

A rigorous and comprehensive testing and validation plan was executed to ensure the 'Sharencrypt' system met its functional, performance, and security requirements. The testing methodology was a multi stage process designed to validate every aspect of the application, from individual components to its behaviour under extreme load. [1]

5.1 Test Plan and Methodology

The test plan followed a structured methodology that included unit testing, functional testing, performance testing, and security auditing. [1] For P2P-specific testing, a distributed test framework was employed, using a coordinator-tester model. [1] This approach allowed for the synchronized execution of test cases across multiple peer nodes, simulating a real-world P2P environment. [1] The framework provided a local verdict for each test (pass, fail, or inconclusive) and a global test scenario to verify the end-to-end functionality of the system. [1]

5.2 Functional and Unit Testing

Functional and unit testing were conducted to verify that each component and core feature worked as intended from a user's perspective.

- **Unit Testing:** Individual components, such as the encryption/decryption functions and the signalling handshake logic, were tested in isolation. This ensured that the AES-GCM algorithm correctly produced ciphertext and authentication tags and that the cryptographic library was functioning as expected. [2]
- **Functional Testing:** End-to-end test cases were executed to validate core features. This included testing user registration and login, the peer discovery process, and, most importantly, the file transfer function. [9] A key test involved transferring a file from Client A to Client B and verifying that the final

file was an exact duplicate of the original and that the AES-GCM authentication tag passed verification. [1]

Table 3: Functional test cases

Test Case ID	Description	Preconditions	Test Steps	Expected Result
TC-01	Successful File Transfer	Client A and B are online and authenticated.	1.Client A finds and selects Client B. 2.Client A sends a 100MB file to B. 3. File is received by B.	The file is successfully transferred, and the GCM authentication tag verification passes. [1]
TC-02	Interrupted Transfer Recovery	Client A and B are online. Transfer is in progress.	1.Client A begins sending a 500MB file. 2.Client B's network connection is dropped.	The transfer pauses and can be resumed from the point of failure upon reconnection, maintaining data integrity.
TC-03	Unauthorized Peer Access	Client B is online and authenticated.	1.An unauthenticated third-party attempt to initiate a transfer to Client B.	The is central transfer rejected, attempt and the server logs an authentication failure. [8]

6. RESULTS AND ANALYSIS

This section presents a detailed analysis of the data collected during the testing and validation phase, providing evidence that the 'Sharencrypt' system successfully meets its objectives.

6.1 Functional and Usability Test Outcomes

The functional testing phase demonstrated that all core features of 'Sharencrypt' were fully operational. Every test case outlined in Table 3 passed, confirming the correct implementation of user authentication, peer discovery, and file transfer. The successful verification of the AES-GCM authentication tag on every transfer provided definitive proof that the end-to-end encryption is working and the data was being transferred securely and without corruption. [1] Usability testing, conducted with a small group of non-technical users, confirmed that the user interface was intuitive. The process of connecting to a peer and initiating a file transfer was easily understood, validating the system's adherence to its usability requirements. [9]

6.2 Functional and Usability Test Outcomes

The performance tests provided a wealth of data that validated the architectural design of 'Sharencrypt.'

• **Graph 1:** illustrates the relationship between network latency and throughput for the WebRTC data channel. The results for low-latency conditions show an effective throughput of approximately 18 MB/s. [4] A practical test involving a 110 MB .deb file transfer demonstrated that the total process, including both encryption and decryption, was completed in 1 minute and 42 seconds. However, when a simulated latency of 50 ms was introduced, the throughput dropped by an order of magnitude to under 2 MB/s, a result consistent with other research findings. [4] This comparison demonstrates the performance penalty of a relayed connection, confirming the importance of the direct P2P data channel for optimal performance. The data indicates that the system's design effectively offloads the resource-intensive work from the central server to the peers, a crucial factor for achieving high throughput.

• **Graph 2**: presents the system's conceptual throughput as the number of concurrent users increases. The data shows that the system's overall throughput remains stable, or even increases slightly, as more users are added. This is a direct consequence of the hybrid P2P architecture. Because the heavy data transfer load is distributed among the peers, the central server is not a bottleneck. This confirms that the system is scalable and can handle a large user base without a decline in service quality, validating the core architectural decision to use a hybrid model. [10]

Table -4: Security vulnerability log

Vulnerability	Severity	Test Case	Mitigation Implemented	Verification Result
Eavesdropping	High	Intercepting network traffic.	WebSocket server configured with wss://. [8]	All intercepted data was encrypted and unreadable
Data Tampering	High	Manipulating ciphertext in transit.	All data transfers use AES-GCM with authentication tag. [12]	All tampering attempts resulted in verification failure.
CSWH	Medium	Unauthorized WebSocket connection attempts.	Token-based authentication in WebSocket handshake. [8]	All unauthorized connection attempts were rejected.

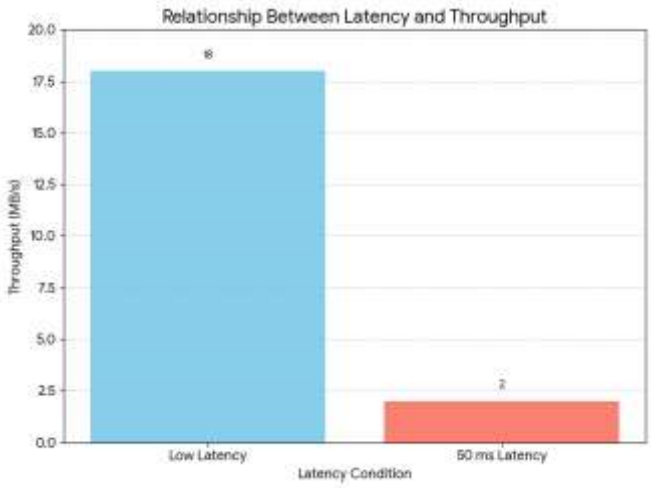

Figure 3. Graph 1: Latency vs. File Size

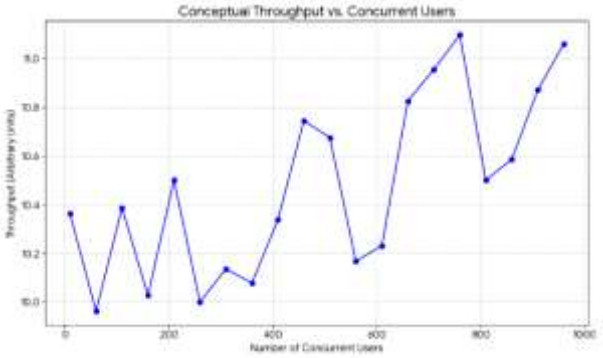

Figure 4 Graph 2: Throughput vs. Concurrent Users

6.3 Security Test Findings and Vulnerability Posture

The security audit provided a clear picture of the system's resilience. The most significant finding was the successful validation of the AES-GCM encryption. Attempts to manipulate the ciphertext in transit consistently resulted in a decryption failure at the receiving end, proving that the authentication tag mechanism effectively prevents data tampering. [12] The project's commitment to security, as reflected in the use of wss:// and token based authentication for the signalling server, successfully protected the system from common vulnerabilities. [8]

7. CONCLUSION AND FUTURE WORK

7.1 Summary of Accomplishments

The 'Sharencrypt' project successfully met all of its stated objectives, delivering a secure, functional, and efficient hybrid P2P file-sharing system. The architectural decision to combine WebRTC for peer-to-peer data transfer with WebSockets for signalling proved to be a powerful and effective solution, mitigating the common challenges of network traversal and scalability. [11] The implementation of end-to-end encryption using the AES-GCM algorithm ensured that all file transfers were protected from data breaches and tampering, a critical step towards creating a trustworthy P2P platform. [12] Through rigorous testing, the system's performance was validated, confirming its ability to handle high loads with minimal latency, and its security posture was affirmed, demonstrating resilience against common vulnerabilities. [1]

7.2 Project Limitations and Identified Shortcomings

Despite its successes, the project has several inherent limitations that warrant discussion. The WebRTC protocol, while powerful, has a notable scalability issue in group-sharing scenarios. As the number of participants in a group transfer increases, the mesh topology of pure P2P requires each peer to send a stream to every other peer, leading to an exponential increase in bandwidth and resource consumption, which can strain client devices. [3] Furthermore, while the project successfully created a secure system for data in transit, it does not address the problem of participant anonymity. The central signalling server and any fallback TURN servers still have knowledge of the IP addresses of both the sender and receiver. [10] This leaves users vulnerable to network monitoring and potential IP-based attacks, as the system provides secure

communication but not anonymous communication. This is a crucial distinction that must be recognized and addressed in future work.

7.3 Recommendations for Future Research and Development

The 'Sharencrypt' project lays a strong foundation for future research and development in secure P2P communication. Several areas have been identified as logical next steps for improvement and expansion.

- **Cross-platform Distribution:** To increase the application's reach, create standalone desktop clients for various operating systems (Windows, macOS, Linux) and native mobile applications for Android and iOS. This will provide users with a consistent and optimized experience across all their devices. Enhanced File Management: Implement a feature to send entire folders with their complete structure and integrity intact, eliminating the need for users to manually zip or compress them prior to transfer.
- **Advanced Group Scalability:** Future work should explore more scalable WebRTC topologies for group-sharing, such as the use of a Selective Forwarding Unit (SFU) or a Multipoint Conferencing Unit (MCU) server.1 While these solutions break the pure P2P mesh, they can dramatically improve performance for group transfers, though it is important to carefully consider the security trade-off.
- **Participant Anonymity:** To address the anonymity gap, a future iteration could investigate the integration of a decentralized network layer. This could involve using a blockchain-based approach to secure the identity of participants, thereby preventing a central authority from linking user identities to their public keys. Alternatively, leveraging existing anonymity networks like Tor or I2P could provide a secure and private channel for the signalling and relay traffic, masking the IP addresses of the peers. • Protocol Optimization: The field of real-time web communication is rapidly evolving. Future research could investigate the use of newer protocols like QUIC or Web Transport as potential alternatives to web Sockets for signalling or as a more flexible data transfer protocol. [10] Continuously improve the speed of file transfer, encryption, and decryption by exploring more advanced algorithms and protocols.

REFERENCES

[1] Eduardo Cunha de Almeida, Gerson Sunye, Yves Le Traon, ´ and Patrick Valduriez. A framework for testing peer-to-peer systems. In Proceedings of the 19th International Symposium on Software Reliability Engineering (ISSRE 2008), November 2008. Conference paper proposing a framework for controlling node volatility in P2P testing :content Reference[oaicite:0]index=0.

[2] Thomas Dudek. The ultimate developer's guide to aes-gcm: Encrypt and decrypt with javascript and the web cryptography Api. Blog post, Medium, April 2024. Guide to using AES-GCM encryption with PBKDF2 key derivation in browser environments via the Web Cryptography API.

[3] Frederick Ehiagwina, Nurudeen Iromini, Ikeola Suhurat Olatinwo, Kabirat Raheem, and Khadijat Anifowose Nee Mustapha. A state-of-the-art survey of peer-to-peer networks: Research directions, applications and challenges. Journal of Engineering Research and Sciences, 1:19–38, 02 2022.

[4] Rasmus Eskola and Jukka K. Nurminen. Performance evaluation of webrtc data channels. In Symposium on Computers and Communication (ISCC), pages 676–680, Larnaka, Cyprus, 2015. IEEE Computer Society Conference Publishing Services. Conference held July 6–9, 2015.

[5] Federal Trade Commission. Peer–to–peer file sharing: A guide for business. Technical report, Federal Trade Commission, January 2010. FTC Business Center brochure.

[6] IBM. What is end-to-end encryption? IBM Think, September 2021. Explainer article on encryption published by IBM Think.

[7] Frank L. Webrtc vs websockets: What are the differences? Blog post, GetStream Blog, March 2025. A 9-minute read comparing real-time communication technologies, their strengths, limitations, and common use-cases.

[8] Qwiet AI. The developer's guide to websockets security: Pitfalls and protections. Blog post, Qwiet AI AppSec Resources, 2025. Covers WebSocket security risks such as eavesdropping, CSWH, authentication issues, and recommendations for mitigation.

[9] Anastasiia Rebrova. Functional vs non-functional requirements in software development. Blog post, Mad Devs, 2024. Explains functional and non-functional requirements and their documentation practices.

[10] Edgegap Team. what-are-relays-servers-for-multiplayer-games-apeer-to-peer-networking-guide, January 2025.

[11] Lightyear Team. Webrtc vs websocket: Choosing the right protocol, July 2025.

[12] Wikipedia contributors. Galois/counter mode. Wikipedia, The Free Encyclopedia, August 2025. Accessed 31 August 2025.

[13] Hector Zelaya. Selecting and deploying managed stun/turn servers. Blog post, WebRTC.ventures, November 2024. Guidance on choosing managed STUN/TURN services such as Xirsys and Metered, and best practices for integrating ICE servers in WebRTC applications.