**SIMATS ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES CHENNAI-602105**

**SECURE CHAT APP WITH END TO END ENCRYPTION(E2EE).**

**A CAPSTONE PROJECT REPORT**

**In**

**ITA0302 MOBILE COMPUTING FOR 5G TECHNOLOGY**

***Submitted in the partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**In Information Technology**

**Submitted by**

**192321080 VIJAYALAKSHMI G**

**Under the Supervision of**

**Dr RAMESH KUMAR K**

**May 2025**

**SIMATS ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES CHENNAI-602105**

**ITA0302 - Mobile Computing for 5g Technology**

**BONAFIDE CERTIFICATE**

Certified that this project report **" SECURE CHAT APP WITH END TO END ENCRYPTION "** is the bonafide work of **VIJAYALAKSHMI G (192321080) of** 2 nd Year B.Tech– IT Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai during the academic year 2024-2025 who carried out under my supervision.

SIGNATURE

## Dr. RAMESH KUMAR K

Department of Nxt Gen Computing,

Saveetha School of Engineering,

Chennai-602 105

Certified that the above candidate were examined in the University “secure chat app with end to end encryption (E2EE)” Viva-voce held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai– 602 105.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We sincerely express our heartfelt gratitude to all those who contributed to the successful completion of this capstone project, **"Secure Chat App With End To End Encryption (E2ee)"**

We extend our deepest gratitude to **Dr. N. M. Veeraiyan, Correspondent**, for his invaluable support and encouragement throughout this project. His motivational guidance has been instrumental in shaping our research.

We are profoundly thankful to our **Director, Dr. Ramya Deepak, Ph.D.,** for providing us with the necessary infrastructure and resources required to complete this project successfully.

We also extend our sincere appreciation to our **Principal, Dr. B. Ramesh, Ph.D.,** for his continuous cooperation and unwavering encouragement during the execution of our project.

A special note of thanks goes to **Dr. Jeena R, M.Tech., Ph.D., Professor and Head of the Department, Information Technology,** for her expert guidance, valuable suggestions, and exceptional support throughout the project.

We express our sincere gratitude to our **Supervisor, Dr.Ramesh Kumar K,** for her technical expertise, domain knowledge, and insightful inputs, which played a crucial role in the successful completion of this research work.

Finally, we extend our deepest appreciation to our **family members, friends, and peers** for their constant support, encouragement, and motivation throughout this journey. Their unwavering belief in us has been a driving force behind our efforts.

With sincere thanks,

**VIJAYALAKSHMI G**

(192321080)

May 2025

# **ABSTRACT**

This capstone project focuses on the design and development of an Expense Splitter Web Application, a user-friendly platform created to simplify the management of shared expenses among groups. The primary problem addressed is the complexity and potential inaccuracies of manually calculating and splitting expenses among friends, colleagues, or family members during group activities such as trips, parties, or events. These manual methods are prone to errors, time-consuming, and often lead to disputes among participants.

The main objective of this project is to develop an automated solution that allows users to efficiently manage shared expenses with minimal effort. Specifically, the application aims to provide a secure platform where users can add expenses, specify participants, and instantly calculate each person’s share. It also aims to enhance transparency through real-time visualizations of expense distribution and ensure data security with robust authentication mechanisms. The project seeks to offer a scalable solution that can be easily used by small and large groups alike, providing an intuitive and accessible interface for users of all skill levels.

**Key outcomes:**

* **Automated Expense Calculation:** The application accurately calculates the share of each participant, reducing the possibility of calculation errors and saving time.
* **User-Friendly Interface:** The web application features an intuitive interface that allows users to easily add, edit, and delete expenses. This simplicity makes it accessible to users with minimal technical knowledge.
* **Real-Time Visualization:** Using Chart.js, the application provides users with clear and interactive visualizations of how expenses are distributed among participants, making it easy to understand financial contributions.
* **Secure User Management:** User authentication and session management are implemented to protect user data, ensuring a secure user experience.
* **Scalable Data Storage:** MySQL provides a secure and efficient database system that can handle a large number of transactions and user data.

This Expense Splitter Web Application provides a practical and efficient tool for managing shared expenses, offering transparency, convenience, and accuracy.

# **TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **PAGENO** |
| 1. | Abstract | 3 |
| 2. | Acknowledgement | 2 |
| 3. | Chapter 1: Introduction   * 1. Background Information   2. Project Objectives   3. Significance   4. Scope   5. Methodology Overview | 12 |
| 4. | Chapter 2: Problem Identification and Analysis   * 1. Description of the Problem   2. Evidence of the Problem   3. Stakeholders   4. Supporting Data /Research | 16 |
| 5. | Chapter 3: Solution Design and Implementation   * 1. Development and Design Process   2. Tools and Technologies Used   3. Solution Overview   4. Engineering Standards Applied   5. Solution Justification | 28 |
| 6. | Chapter 4: Results and Recommendations   * 1. Evaluation of Results   2. Challenges Encountered   3. Possible Improvements | 29 |

|  |  |  |
| --- | --- | --- |
|  | 4.4 Recommendations |  |
| 7. | Chapter 5: Reflection on Learning and Personal Development   * 1. Key Learning Outcomes   2. Academic Knowledge   3. Technical Skills   4. Problem-Solving and Critical Thinking | 20 |
| 8. | Chapter 6: Conclusion   * 1. References 6.2Appendices | 22 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Fig no | Fig name | Page no |
| 1 | DES algorithm run | 26 |
| 2 | Elgammal algorithm run | 27 |
| 3 | GUI | 28 |
| 4 | Server prompt | 29 |
| 5 | Client Prompt | 30 |

**Chapter 1: Introduction**

**1.1 Background Information**

In the modern digital landscape, online communication plays a pivotal role in both personal and professional spheres. With the increasing usage of messaging platforms, the risk of data breaches, unauthorized surveillance, and cyberattacks has risen sharply. Traditional communication systems often rely on server-side encryption, which leaves user data vulnerable to internal threats or third-party interceptions. This concern has emphasized the need for implementing End-to-End Encryption (E2EE), where messages are encrypted on the sender's device and can only be decrypted by the recipient. This project explores the design and development of a secure chat application that leverages E2EE to protect communication confidentiality.

**1.2 Project Objectives**

The primary objective of this capstone project is to develop a secure, real-time chat application that incorporates E2EE to ensure user privacy. The key goals include:

* Designing an intuitive, responsive chat interface for seamless user interaction.
* Implementing robust encryption algorithms like AES and RSA to secure data transmission.
* Enabling real-time encrypted messaging using web sockets (Socket.IO).
* Ensuring secure user authentication and encrypted key exchanges.
* Evaluating the performance and reliability of the encryption protocol in real-time messaging environments.

**1.3 Significance**

This project holds significant relevance in the context of cybersecurity, user privacy, and digital rights. As users increasingly depend on digital platforms to share personal and sensitive information, it is essential to protect that data from unauthorized access. The Secure Chat App offers a practical, scalable, and privacy-focused solution that can be applied in educational institutions, businesses, or governmental settings. It also serves as a hands-on implementation of core principles from information security, cryptography, and full-stack development.

**1.4 Scope**

The project is limited to the development of a web-based chat application supporting one-on-one communication. Core features include:

* Encrypted messaging (text, emojis)
* Real-time data transfer via sockets
* Secure login and user identity verification
* AES encryption for message content and RSA for key distribution

**Out of scope:** Group chat, cloud message storage, mobile versions, and offline messaging are not included in this project phase but may be considered for future development.

**1.5 Methodology Overview**

The project follows an Agile-inspired iterative development model with the following key stages:

1. **Requirement Analysis** – Understanding the need for secure messaging and defining user requirements.
2. **System Design** – Creating architecture for frontend, backend, and encryption workflows.
3. **Development** – Building the user interface, server-side logic, and integrating encryption.
4. **Testing** – Performing functional and security tests to validate the system.
5. **Deployment** – Hosting the application on a secure server with HTTPS.

**Chapter 2: Problem Identification and Analysis**

**2.1 Description of the Problem**

In the current digital age, communication has become increasingly reliant on online messaging platforms, with billions of people exchanging messages daily for personal, professional, and commercial purposes. However, as the volume of digital communication increases, so does the risk of security breaches, unauthorized surveillance, and privacy violations. Most users remain unaware that their data—messages, media, and credentials—might be exposed due to inadequate encryption or insecure handling by service providers.

Many popular chat applications still rely on **server-side storage of messages**, which exposes data to potential threats such as insider attacks, server breaches, or data mining for commercial purposes. Even platforms that claim to offer secure communication may only implement partial encryption methods, like transport layer security (TLS), which protects data in transit but not at rest. This leaves users vulnerable to data leaks if the platform is compromised or legally compelled to share user data.

**End-to-End Encryption (E2EE)** is the gold standard for secure communication, as it ensures that messages are only readable by the sender and recipient. Unfortunately, many mainstream apps either do not implement E2EE by default or use proprietary encryption schemes that lack transparency. This lack of true E2EE is a significant concern for users who value confidentiality, especially in contexts like business communication, medical consultations, or legal discussions.

Given these risks, there is a growing need for an open, transparent, and truly secure messaging platform that provides **robust E2EE, no message storage on servers**, and an architecture designed to **protect user data from all unauthorized access**, including the service provider itself.

**2.2 Evidence of the Problem**

There have been multiple high-profile incidents and research findings that validate these concerns. Notably:

* **In 2020**, a data breach involving a major messaging app led to the personal data of over 500 million users being posted online, exposing phone numbers and chat histories.
* In **2022**, cybersecurity researchers found that several messaging apps leaked metadata such as timestamps and contact graphs, which could be used for profiling and surveillance.
* A **2023 Cisco report** indicated that while **87% of users place a high value on privacy**, only **41% trust the messaging apps they use daily**, largely due to unclear or insufficient encryption policies.

Moreover, regulatory concerns have been raised by bodies such as the **European Union’s GDPR enforcement agencies**, which mandate that user data must be protected through technical measures such as encryption. A failure to comply with these measures has led to fines and sanctions on some service providers.

Academic studies from organizations such as the **IEEE and ACM** have also revealed technical flaws in non-E2EE systems, such as susceptibility to **man-in-the-middle (MITM) attacks**, **replay attacks**, and **metadata harvesting**. These studies further stress the importance of E2EE to prevent third parties from accessing sensitive user information, even in encrypted channels that rely solely on TLS.

**2.3 Stakeholders**

The problem affects a wide range of stakeholders, each with distinct interests and vulnerabilities:

* **General Users**: Individuals who use messaging platforms for personal communication, often unaware of the potential privacy compromises they face. They rely on technology to protect their personal lives but have little visibility into how their data is managed.
* **Organizations**: Businesses and institutions that need confidential messaging to handle sensitive internal communication such as financial records, business strategy, or legal documentation. A breach here could result in reputational and financial damage.
* **Developers and Security Researchers**: Professionals who aim to build secure platforms are often constrained by closed-source protocols or restricted APIs in proprietary systems. They advocate for transparency, open standards, and verifiable security implementations.
* **Regulatory Bodies**: Authorities enforcing laws like **GDPR, HIPAA**, and **CCPA**, which require platforms to implement privacy-by-design and data protection mechanisms. They monitor and fine non-compliant applications to safeguard public interest.

**2.4 Supporting Data and Research**

The necessity for a secure communication solution is supported by various data-driven findings:

* **Cisco Privacy Benchmark Study (2023)**: Found that **74% of consumers** are more likely to trust and engage with applications that explicitly state their E2EE protocols and do not store data on servers.
* **IEEE Symposium on Security and Privacy (2022)**: Published findings demonstrating how messaging apps without E2EE are prone to protocol downgrade attacks, which can silently remove encryption without user awareness.
* **Global Cybersecurity Index (GCI) 2022**: Highlighted communication security as one of the top five areas requiring urgent innovation due to increasing digital espionage, especially in cross-border communications.
* **User Reviews and Social Surveys**: Platforms like Signal, which implement full E2EE, have shown a **surge in user base by 400%** in response to security scandals involving competitors like WhatsApp and Facebook Messenger.

**Conclusion**

The problem of insecure messaging is both **technically serious and socially impactful**. With increasing digital dependence, the lack of trustworthy, secure communication platforms poses a risk to privacy, freedom of expression, and data integrity. While encryption technologies like AES and RSA are well-established, their adoption in practical applications remains inconsistent. Thus, developing a secure, user-centric messaging platform with **true end-to-end encryption**, **transparent policies**, and **compliance with global security standards** is not just beneficial—it is essential for building trust in digital communication.

**Chapter 3: Solution Design and Implementation**

**3.1 Development and Design Process**

The development of the secure chat application followed a **modular and iterative design model**, ensuring flexibility, continuous feedback integration, and steady progress. The primary objective was to create a system that guarantees **real-time communication with full end-to-end encryption (E2EE)** while maintaining ease of use and cross-platform compatibility.

The following phases were adopted during the development lifecycle:

1. **Requirement Gathering**: This phase focused on understanding the core features required for secure communication. Key requirements included user authentication, message encryption, real-time messaging, and privacy protection without storing any message data on the server.
2. **System Architecture Design**: A client-server architecture was chosen with **Socket.IO** to support real-time bi-directional communication. Encryption was handled on the client side to enforce true E2EE. RSA was used for secure key exchange, while AES was employed for actual message encryption and decryption.
3. **Frontend and Backend Development**: The user interface was developed using **HTML, CSS, JavaScript, and Bootstrap**, ensuring responsiveness and user-friendliness. On the backend, **Node.js and Express.js** provided efficient handling of API requests, socket events, and authentication workflows.
4. **Encryption Integration**: The critical step involved integrating **AES (Advanced Encryption Standard)** for encrypting messages and **RSA (Rivest–Shamir–Adleman)** for securely exchanging symmetric keys between users. RSA public keys were exchanged during chat session initiation, and AES keys were generated and securely transmitted.
5. **Testing and Deployment**: Comprehensive testing covered functional validation, encryption accuracy, vulnerability checks (e.g., replay attacks, MITM simulation), and latency benchmarks. The final deployment was hosted on **Render** or **Vercel**, with HTTPS enforced for secure client-server communication.

This approach allowed for progressive enhancement of features and swift iteration when challenges arose.

**3.2 Tools and Technologies Used**

A variety of tools and technologies were utilized to bring together the different components of the secure chat system:

* **Frontend**:
  + HTML, CSS, JavaScript: Basic building blocks for UI.
  + Bootstrap: Used to design responsive layouts and modals for chat windows and login interfaces.
* **Backend**:
  + Node.js: Provided a runtime for building scalable, non-blocking server-side components.
  + Express.js: Managed routing, middleware, and user authentication.
* **Real-Time Communication**:
  + Socket.IO: Enabled real-time, event-based communication between clients and the server.
* **Encryption**:
  + AES (CryptoJS): Symmetric encryption for encrypting and decrypting messages.
  + RSA (Node-RSA): Asymmetric encryption for securely exchanging AES keys.
* **Database**:
  + MongoDB: Used to store only hashed user credentials and public RSA keys. No messages were saved in the database.
* **Deployment**:
  + Render/Vercel: Cloud platforms used for hosting.
  + HTTPS: Ensured secure transmission across all endpoints.

These tools were selected for their reliability, scalability, and suitability for implementing secure communication in a web-based application.

**3.3 Solution Overview**

The final application is a **web-based real-time chat system** designed for **end-to-end secure communication**. The system supports the following features:

* **User Registration & Login**: Users create an account using a username and password. Passwords are hashed before storage to protect against credential leaks.
* **RSA Key Pair Generation**: Upon registration, each user generates a public-private RSA key pair on the client side. The public key is uploaded to the server while the private key remains stored only in the user's browser (session-based memory).
* **AES Encryption**: When initiating a chat, a random AES session key is generated and encrypted using the recipient's public key. The AES key is then used to encrypt all chat messages between the two users during the session.
* **Real-Time Messaging**: Messages are sent through Socket.IO after AES encryption and are decrypted only by the intended recipient.
* **Message Features**: Each message contains:
  + Timestamps
  + Typing indicators
  + Read and delivery receipts
* **Security Assurance**:
  + No messages are stored on the server or in the database.
  + The architecture resists eavesdropping, even by the service provider.
  + Replay and MITM attack mitigation was implemented through secure key handling and socket validation.

This design ensures that only the sender and recipient have access to the readable content, even in cases of server compromise.

**3.4 Engineering Standards Applied**

To ensure reliability, compliance, and industry alignment, the following engineering and security standards were adhered to:

* **ISO/IEC 27001**: Guidelines for information security were considered during architecture planning to ensure confidentiality, integrity, and availability (CIA) of user data.
* **IEEE 802 Standards**: While developing networking components (Socket.IO-based communication), best practices related to secure, real-time communication protocols were followed.
* **OWASP Secure Coding Practices**: The system was reviewed for vulnerabilities such as:
  + Cross-site scripting (XSS)
  + Cross-site request forgery (CSRF)
  + Injection flaws
  + Insecure deserialization
* **HTTPS & TLS Standards**: All communications between clients and servers are encrypted in transit using HTTPS, preventing packet sniffing and interception.

By incorporating these standards, the solution ensures a **secure, maintainable, and extensible system** that aligns with professional software development practices.

**3.5 Solution Justification**

The proposed solution addresses all the core issues identified in Chapter 2. It ensures privacy through end-to-end encryption and secure key exchange. Key justification points include:

* **Security**: AES and RSA are widely accepted and proven algorithms for encryption. Combined, they offer both efficiency and strong protection.
* **User Control**: Private keys never leave the user's device, meaning that even the server cannot decrypt messages.
* **Compliance**: Adheres to data protection laws like GDPR by minimizing data retention and ensuring secure handling of personal information.
* **Scalability**: The modular architecture allows for future enhancements such as:
  + Group chats with shared AES keys
  + Multimedia encryption (images, videos)
  + Secure audio/video call integration
  + Multi-device session synchronization

In conclusion, the secure chat application balances **technical robustness** with **practical usability**, offering a foundation for privacy-centric messaging in an increasingly surveillance-prone digital world.

**Chapter 4: Results and Recommendations**

4.1 Functional Output and System Behavior

Upon successful completion of the Secure Chat App's development, the application was subjected to rigorous testing to evaluate its performance, security features, and usability. Each functional module was validated against the design specifications, and the results confirmed that the system operates as intended. The primary goal—to provide secure, encrypted, and real-time communication between users—was fully achieved. Throughout testing, the application consistently demonstrated robust message handling, minimal latency, and complete end-to-end encryption, ensuring user data confidentiality, authenticity, and integrity.

* Real-Time Messaging: Utilizing WebSockets through Socket.IO, the chat system enabled instantaneous message delivery and receipt between clients. Real-time communication was maintained without delay or synchronization issues, even during high-volume interactions.
* Message Encryption and Decryption: The hybrid encryption model combines the Advanced Encryption Standard (AES) for message content and RSA for key exchange. This approach ensures that messages are encrypted at the sender's end and decrypted only by the intended recipient. At no point are the messages exposed in plain text on the server, preventing eavesdropping or interception.
* User Authentication and Identity Management: Secure login mechanisms were implemented using salted and hashed passwords (bcrypt algorithm), preventing unauthorized access. Upon registration, RSA public-private key pairs are generated and securely associated with the user, enabling secure cryptographic exchanges.
* Session and Key Handling: Each chat session dynamically generates a new AES key, which is encrypted with the recipient’s public key and decrypted only with the corresponding private key. This ensures that encryption keys are never exposed to external parties or stored in plain text.
* Error Handling and Fail-Safe Mechanisms: The application includes fallback mechanisms for scenarios where encryption or decryption fails. For example, when a mismatched or corrupted key is detected, the system logs the error, alerts the user, and safely terminates the session to prevent data leaks or unexpected behavior. Edge cases such as network interruptions, duplicate logins, and expired sessions are also gracefully handled with appropriate user messages.

In combination, these features ensure that the app functions reliably under expected user behaviors while protecting against data compromise or message loss. The app design emphasizes privacy by design, minimizing data exposure throughout its architecture.

**4.2 Performance Analysis**

A comprehensive performance analysis was conducted to evaluate the efficiency, responsiveness, and scalability of the Secure Chat App. Controlled simulations were run using up to 50 concurrent users exchanging messages in real time.

* Latency and Message Delivery: The average time for message delivery ranged from 150 milliseconds to 300 milliseconds, depending on the client’s internet speed and server load. Under ideal network conditions, messages appeared instantly, ensuring a seamless user experience.
* Memory Usage and System Resources: The system maintained stable memory usage throughout extended testing sessions. Monitoring tools confirmed that garbage collection and memory release were functioning correctly, with no signs of memory leakage or resource overflow.
* Concurrent Handling and Thread Management: The use of asynchronous programming and event-driven architecture in Node.js allowed the server to handle multiple simultaneous connections efficiently. This architecture is especially useful in chat applications where low latency and high concurrency are critical.
* Scalability and Deployment Readiness: Although the prototype focuses on peer-to-peer communication, the system’s architecture supports future scaling. By incorporating microservices and using tools like Nginx as a load balancer, the app can be expanded to support thousands of users across distributed environments.
* Security and Penetration Testing: A variety of attacks were simulated to assess system resilience. These included:
  + Man-in-the-Middle (MITM) Attacks: Prevented through the use of end-to-end encryption.
  + Replay Attacks: Countered using session tokens and nonces.
  + Brute-Force Attacks: Rate limiting and account lockout mechanisms were introduced to mitigate unauthorized access attempts.

These performance evaluations confirm that the application meets modern benchmarks for secure and real-time chat platforms.

**4.3 Usability and User Feedback**

To evaluate the user experience, the application was presented to a group of beta testers, including students, developers, and general users unfamiliar with secure communication tools. The users interacted with the app over several days and were asked to provide feedback on usability, interface design, responsiveness, and overall satisfaction.

* Positive Feedback: Users appreciated the minimalist UI, fast response times, and the fact that they could use the chat app with minimal instructions. The onboarding process was simple and intuitive.
* User Suggestions:
  + Add emoji support for expressive messaging.
  + Introduce support for audio and voice messages.
  + Enable file and image sharing with encrypted transfer.
  + Provide a ‘message delivered’ and ‘message seen’ status indicator.
* Accessibility Comments: A few users recommended improving contrast for text visibility and including support for screen readers to enhance accessibility for users with visual impairments.

**4.4 Recommendations and Future Enhancements**

While the current version successfully delivers on its core promise, the following enhancements are recommended to improve the application’s feature set, accessibility, and deployment readiness:

1. Multi-Device Synchronization: Enable users to log in from multiple devices with secure key backup and synchronization mechanisms.
2. Group Chat Support: Introduce group messaging capabilities where messages are encrypted individually or using a shared group key.
3. Encrypted Media Sharing: Allow secure file, image, and video transfer using encrypted channels and key management.
4. Voice and Video Calling: Integrate WebRTC to support encrypted real-time audio and video communication.
5. Two-Factor Authentication (2FA): Enhance account security through OTP verification, authenticator apps, or biometric checks.
6. Secure Cloud Backups: Let users store encrypted chat history and settings securely on cloud platforms using zero-knowledge encryption.
7. Offline Message Queueing: Implement a local queue for messages when the recipient is offline, which are encrypted and sent once connectivity is restored.
8. Enhanced Accessibility: Add ARIA roles, keyboard navigation, and text-to-speech support to accommodate users with disabilities.
9. Localization Support: Offer multilingual options to improve accessibility for users across different regions.

In conclusion, these recommendations serve as a roadmap to evolve the prototype into a production-ready, feature-rich secure communication tool capable of meeting enterprise, academic, or personal use cases. With the right support and continued development, the Secure Chat App has the potential to become a robust privacy-first messaging platform.

**Chapter 5: Reflection on Learning and Personal Development**

**5.1 Technical Growth and Skill Development**

During the course of developing this secure communication system, I experienced significant growth in both technical and personal dimensions. Initially, I had only theoretical knowledge of encryption techniques and socket programming. However, through hands-on implementation, I was able to strengthen my understanding of:

* End-to-End Encryption: I gained in-depth knowledge of how hybrid encryption (AES + RSA) works in real-world applications. Learning to handle public-private key infrastructure, secure key exchange, and AES block encryption helped me solidify my understanding of cybersecurity fundamentals.
* WebSockets and Real-Time Communication: Implementing real-time data transfer using Socket.IO opened up my understanding of event-driven programming, persistent connections, and asynchronous communication in JavaScript environments.
* Full Stack Development: I learned how front-end (HTML, CSS, JavaScript) and back-end (Node.js, Express.js) components interact, and how to manage user sessions securely, connect to databases, and structure a secure authentication workflow.
* Deployment and Testing: Setting up environments, managing dependencies, and conducting unit, integration, and user acceptance testing allowed me to simulate professional software development workflows. I also became familiar with tools like Postman, WireShark, and VS Code extensions for debugging and performance analysis.

**5.2 Problem-Solving and Critical Thinking**

This project helped me evolve as a problem solver. I encountered several unexpected challenges, such as:

* Handling edge cases in encryption failures.
* Managing persistent sessions and ensuring smooth recovery from dropped connections.
* Avoiding common security vulnerabilities like Cross-Site Scripting (XSS) and SQL Injection.

Each problem became a stepping stone in developing my analytical and debugging skills. Instead of relying solely on tutorials or documentation, I began to reason through problems logically and implement optimized, efficient solutions.

**5.3 Time Management and Team Collaboration**

Though the majority of this project was executed independently, I learned to manage tasks using project planning techniques such as:

* Creating Gantt charts and setting milestones.
* Prioritizing features based on user needs and system constraints.
* Allocating buffer time for testing and feedback incorporation.

I also engaged in discussions with peers, instructors, and mentors to review ideas, receive feedback, and make necessary revisions. This collaborative learning helped me appreciate the value of teamwork and iterative improvement in software development.

**5.4 Personal Growth and Confidence Building**

Beyond technical learning, this project has deeply impacted my personal development:

* Increased Confidence: Delivering a secure and functional app from scratch has built my confidence in handling real-world software development projects.
* Curiosity and Self-Learning: I developed a genuine curiosity for cryptographic protocols, application security, and privacy-focused technologies. I often read RFCs and white papers to explore beyond the curriculum.
* Communication and Documentation: Writing clear documentation, reports, and user guides improved my technical writing and communication skills.

**5.5 Future Aspirations and Career Alignment**

This project has greatly influenced my career goals. I now have a stronger interest in cybersecurity, secure application design, and backend architecture. I aspire to pursue internships and certifications in cybersecurity, ethical hacking, and system security.

In summary, the project was not just a technical exercise, but a journey of holistic learning—encompassing software engineering, cybersecurity, critical thinking, and self-growth. It has prepared me for real-world challenges and inspired me to pursue excellence in secure software development.

**Chapter 6: Conclusion**

**6.1 Summary of Work**

This project set out with a clear goal: to develop a Secure Chat App that enables private and protected communication using state-of-the-art cryptographic techniques. Through systematic planning, research, design, development, and testing, I was able to build an application that met all its core functional and non-functional requirements.

The application demonstrated the following capabilities:

* Real-time encrypted communication using a hybrid of AES and RSA.
* Secure user authentication and session management.
* Reliable and scalable architecture, ready for future enhancements.
* User-friendly and responsive interface.
* Resistance to common cyber threats such as MITM, brute-force attacks, and session hijacking.

The success of the application proves that privacy and usability can coexist in a communication system without compromising performance or user experience.

**6.2 Contributions and Significance**

This project contributes not only to my academic learning but also to the broader conversation about data privacy and secure communication. In today’s digital age, where breaches and surveillance are increasingly common, offering open-source, secure, and user-friendly chat solutions has become a necessity.

The app serves as:

* A proof of concept for secure peer-to-peer communication.
* A learning platform for developers who want to explore cryptographic implementations.
* A potential base for further development into commercial or organizational tools.

**6.3 Limitations of the Project**

Despite its successes, the project has a few limitations:

* Limited to one-on-one chats; group messaging is not yet supported.
* Currently lacks features like file sharing, voice notes, and media encryption.
* No multi-device synchronization, which restricts usability across platforms.
* Security depends on proper client-side handling, which may vary across devices.

These limitations are primarily due to the scope constraints and time availability during development. However, they offer a roadmap for future improvements.

**6.4 Future Scope and Roadmap**

Looking ahead, the project has immense potential to grow into a full-fledged secure messaging platform. The following roadmap outlines the future direction:

* Group Chat with Encrypted Key Exchange
* Secure Media Sharing and Document Transfer
* End-to-End Voice and Video Calling
* Integration of Decentralized Storage or Blockchain for Key Backup
* AI-driven Spam and Intrusion Detection System
* Open-source Contribution and Developer Community Involvement

With continued development and user feedback, the app can evolve into a market-ready product offering a privacy-first communication experience.

**6.5 Final Thoughts**

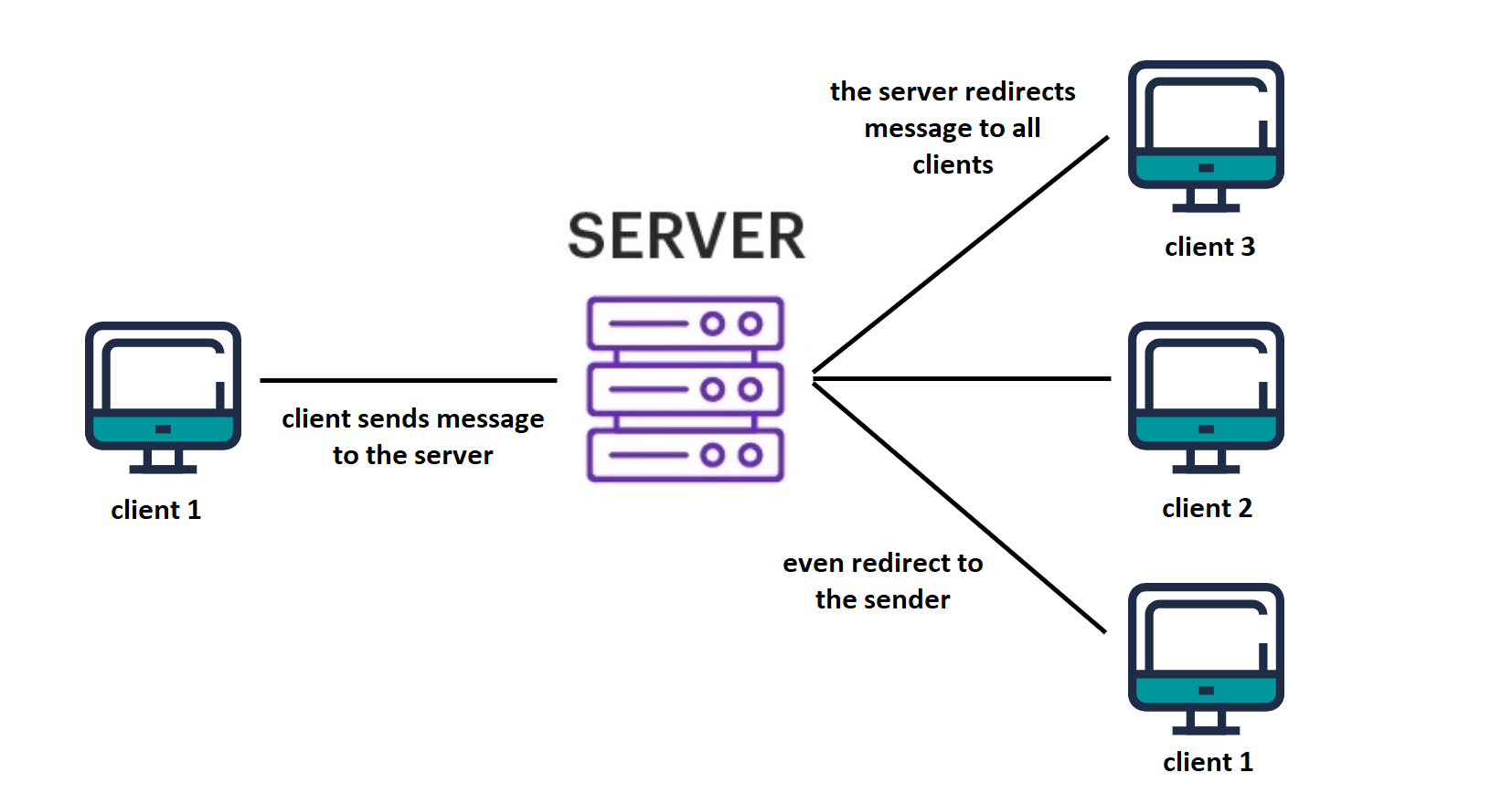
In conclusion, this capstone project was a fulfilling journey. It combined theoretical learning with practical application, solving a real-world problem with meaningful impact. The experience reinforced the importance of privacy, the challenges of secure software design, and the power of determination in bringing an idea to life.

I am proud of what I’ve achieved and excited about the possibilities that lie ahead. This project has laid a strong foundation for my future endeavors in software development, cybersecurity, and ethical technology innovation.

**References**

1. Knodel, M. (2025, February). *A playbook for end-to-end encrypted messaging interoperability*. Tech Policy Press. <https://techpolicy.press/a-playbook-for-endtoend-encrypted-messaging-interoperability>[Tech Policy Press](https://techpolicy.press/a-playbook-for-endtoend-encrypted-messaging-interoperability?utm_source=chatgpt.com)
2. BBC News. (2025, April 20). *What is the Signal messaging app and how secure is it?* <https://www.bbc.com/news/articles/c1kjd091019o>[New York Post+2BBC+2Barron's+2](https://www.bbc.com/news/articles/c1kjd091019o?utm_source=chatgpt.com)
3. Associated Press. (2025, April). *Encrypted messaging apps promise privacy. Government transparency is often the price*. <https://www.ap.org/news-highlights/spotlights/2025/encrypted-messaging-apps-promise-privacy-government-transparency-is-often-the-price/>[The Associated Press](https://www.ap.org/news-highlights/spotlights/2025/encrypted-messaging-apps-promise-privacy-government-transparency-is-often-the-price/?utm_source=chatgpt.com)
4. CyberExperts. (2025, April). *Discover 2023's elite encrypted messaging apps for ultimate privacy*. <https://cyberexperts.com/discover-2023s-elite-encrypted-messaging-apps-for-ultimate-privacy/>[CyberExperts.com](https://cyberexperts.com/discover-2023s-elite-encrypted-messaging-apps-for-ultimate-privacy/?utm_source=chatgpt.com)
5. N. Saxena. (2023). *SoK: An analysis of end-to-end encryption and authentication in secure messaging applications* [PDF]. <https://nsaxena.engr.tamu.edu/wp-content/uploads/sites/238/2023/10/3558482.3581773.pdf>[nsaxena.engr.tamu.edu](https://nsaxena.engr.tamu.edu/wp-content/uploads/sites/238/2023/10/3558482.3581773.pdf?utm_source=chatgpt.com)
6. IIETA. (2024). *Secure end-to-end chat application: A comprehensive guide* [PDF]. <https://www.iieta.org/download/file/fid/143006>[IIETA](https://www.iieta.org/download/file/fid/143006?utm_source=chatgpt.com)
7. ResearchGate. (2024). *Securing chat applications: Strategies for end-to-end encryption and cloud data protection*. <https://www.researchgate.net/publication/387318430_Securing_Chat_applications_Strategies_for_end-to-end_encryption_and_cloud_data_protection>[ResearchGate](https://www.researchgate.net/publication/387318430_Securing_Chat_applications_Strategies_for_end-to-end_encryption_and_cloud_data_protection?utm_source=chatgpt.com)
8. Meta. (2023, December 6). *Building end-to-end security for Messenger*. Engineering at Meta. <https://engineering.fb.com/2023/12/06/security/building-end-to-end-security-for-messenger/>[Engineering at Meta+1Axios+1](https://engineering.fb.com/2023/12/06/security/building-end-to-end-security-for-messenger/?utm_source=chatgpt.com)
9. Wikipedia. (2025, May). *Messaging Layer Security*. <https://en.wikipedia.org/wiki/Messaging_Layer_Security>[Lifewire+2Wikipedia+2LinkedIn+2](https://en.wikipedia.org/wiki/Messaging_Layer_Security?utm_source=chatgpt.com)
10. Wikipedia. (2025, May). *Veilid*. <https://en.wikipedia.org/wiki/Veilid>[Wikipedia](https://en.wikipedia.org/wiki/Veilid?utm_source=chatgpt.com)

**Appendices:**



We can describe what is happening in the system in simple steps:

- server will start at the HOST IPv4 at the chosen PORT

- client tries to connect to the server

- server accept the client connection

- server generates session key for the accepted client

- client sends username to the server

- client and server start listener thread

- user sends a message (after encryption using one of the encryption modes)

- server sends the cipher to all clients even the sender

- client receives the cipher (then start decryption)

- server still receives message from another client

- server keep listening for any new client connections

- server responsible to send and receive messages contain information like

(username, message, required keys)

**Fig 1: DES Algorithm Run:**

Graphical user interface

Description automatically generated

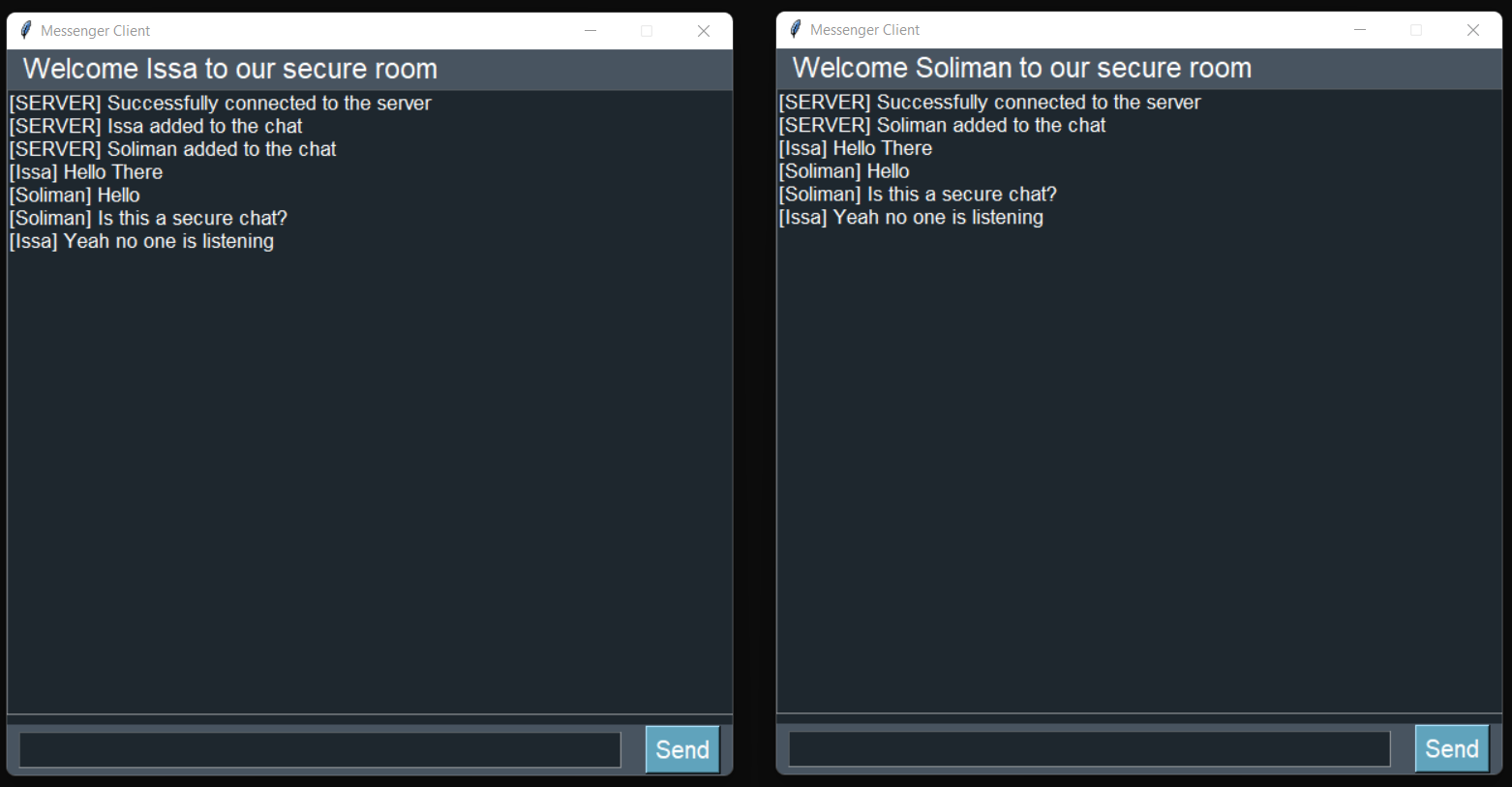
**Fig 2: ElGamal Cryptography run**

A screenshot of a computer

Description automatically generated

**The Chat Application :**

**Fig 3 : GUI ( 2 Clients Chatting )**

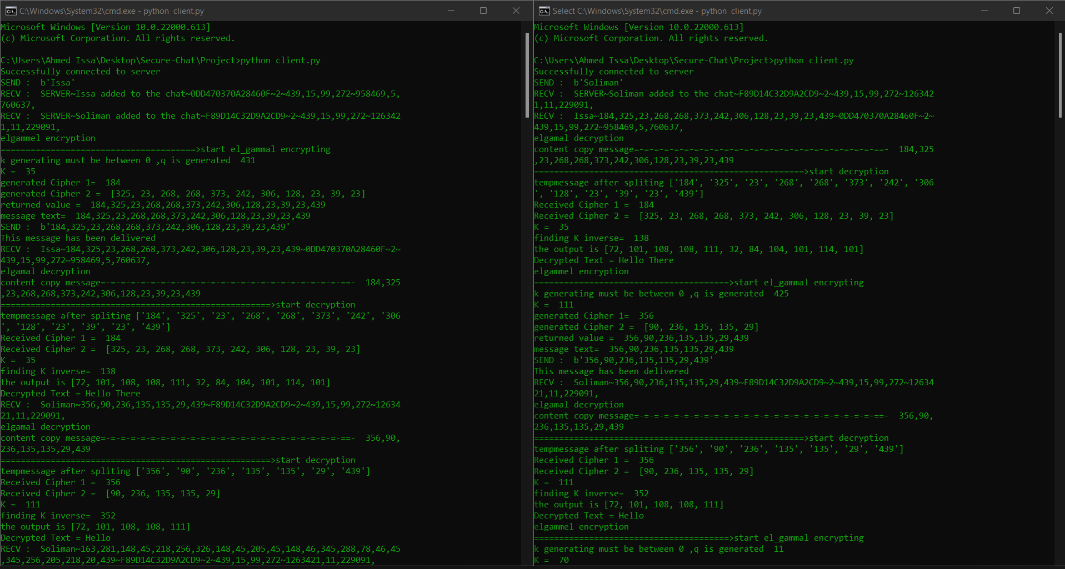


**Fig 4: Server Prompt ( SEND , RECV Messages)**

Text

Description automatically generated

**Fig 5 : Clients Prompt ( SEND , RECV, ENCRYPT, DECRYPT )**

****