MINOR-2 PROJECT

SYNOPSIS
on
Secure Chat Application

Submitted By:

Name	Roll No	Branch
Aadeesh Jain	R2142220375	BTech CSF
Deepanshu Chowdhury	R2142220474	BTech CSF
Aman Anand	R2142220415	BTech CSF
Abhinav Saini	R2142220392	BTech CSF

Under the guidance of
Dr. Avishek Majumder

School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Dehradun 248007
Approved By

Project Guide **Cluster Head**

School of Computer Science'
University of Petroleum & Energy Studies, Dehradun
Synopsis Report (2025)

Index

Contents

Abstract

With the increasing demand for secure communication, encryption-based chat applications have become essential in protecting users' privacy. The Secure Chat Application is a real-time messaging system that ensures confidentiality, integrity, and authenticity by implementing AES for encryption, Diffie-Hellman for key exchange, and HMAC for message verification. This application is built using socket programming on localhost, allowing for multi-client communication while securing text messages. The project is designed to prevent eavesdropping, tampering, and unauthorized access, making it a reliable choice for secure communication.

Introduction

In today's increasingly interconnected digital world, messaging platforms have become indispensable tools for communication, revolutionizing the way we interact with others. Whether for personal conversations or professional exchanges, these platforms enable seamless and instant communication across vast distances. From casual chats between friends to critical business discussions, messaging services like WhatsApp, Signal, and Slack have become integral to our daily lives.

However, the very convenience that makes these platforms so popular also brings with it significant security and privacy concerns. The sensitive nature of the information shared—ranging from personal data to confidential business details—makes messaging platforms a prime target for cybercriminals, hackers, and unauthorized surveillance. As users exchange private messages, financial details, and business strategies over these channels, the potential risks of eavesdropping, hacking, and data breaches continue to rise.

Unauthorized access to chat messages can lead to severe consequences, including identity theft, financial loss, and the compromise of sensitive personal or corporate information. Moreover, the growing sophistication of cyber-attacks and surveillance technologies adds another layer of complexity to securing these platforms. Protecting the privacy and integrity of chat messages is no longer optional; it has become a critical issue in an era where data breaches and privacy violations are increasingly common.

Problem Statement

The widespread use of messaging platforms for exchanging sensitive information presents significant privacy and security risks, as these platforms are vulnerable to eavesdropping, hacking, and unauthorized access. The challenge lies in protecting the integrity and confidentiality of the messages exchanged in real time. Despite the availability of various security measures, many messaging platforms lack robust encryption mechanisms, leaving users' data exposed to potential breaches.

The Secure Chat Application aims to address these issues by:

- Implementing end-to-end encryption to protect messages from interception.
- Securing key exchange using Diffie-Hellman to prevent unauthorized access.
- Ensuring message integrity through HMAC verification.
- Providing secure authentication mechanisms to prevent unauthorized users from accessing the chat.

By developing a fully encrypted, localhost-based chat system, this project ensures private and secure real-time communication.

Literature Review

Several cryptographic techniques exist to secure digital communication. The most widely used encryption methods include AES (Advanced Encryption Standard) and Diffie-Hellman. While AES provides fast and secure symmetric encryption, Diffie-Hellman is commonly used for key exchange.

Key Research Areas:

1. **End-to-End Encryption:**

- AES (128-bit or 256-bit) is preferred due to its high speed and resistance to brute-force attacks.
- Diffie-Hellman is used for key exchange but is computationally intensive.

2. **Key Exchange Mechanisms:**

- Diffie-Hellman (DH) Key Exchange allows two parties to securely establish a shared secret over an insecure channel.
- Diffie Hellman Key Exchange ensures secure transmission of symmetric encryption keys.

3. **Message Integrity and Authentication:**

- HMAC (SHA-256) is used to verify that the message has not been altered.

- Password hashing techniques (e.g., bcrypt or SHA-256) are used to secure user authentication.

Existing secure chat applications like WhatsApp and Signal implement similar encryption techniques, but they rely on centralized servers. In contrast, this project provides a decentralized localhost-based implementation, offering insights into secure communication in a controlled environment**.**

Objectives

The primary objective of this project is to create a secure and private communication system that ensures message confidentiality, integrity, and authentication.

1. **Confidentiality, Integrity, and Availability (CIA):**
   - **Confidentiality:** Ensure that only authorized users can access sensitive information.
   - **Integrity:** Prevent unauthorized changes to messages.
   - **Availability:** Guarantee that services are available whenever needed.
2. **Authentication, Authorization, and Accounting (AAA):** Ensure secure access control by identifying, verifying, and tracking user actions within the platform.
3. **Secure Communication:** Protect all forms of digital communication, including text, file, video, and audio transmissions.

Specific Objectives:

- Develop a secure chat application with real-time encrypted messaging.
- Implement strong encryption using AES (for data confidentiality).
- Establish a secure key exchange mechanism (using Diffie-Hellman) to prevent unauthorized access.
- Ensure message integrity using HMAC (SHA-256) to detect tampering.
- Secure file transfers through AES encryption before sending.
- Authenticate users securely using password hashing (SHA-256/bcrypt).
- Support multiple clients using socket programming for real-time communication.

Methodology

The development of the Secure Chat Application follows a structured approach that includes system design, implementation, testing, and evaluation.

Implementation Steps:

Step 1: Server Starts

- The server runs on 127.0.0.1 (localhost) and listens on a specific port (e.g., 8080).
- It waits for client connections using socket programming.
- When a client connects, the server creates a new thread for that client.

Step 2: Client Connects to the Server

- The client connects to the server using 127.0.0.1:8080.
- The server assigns the client a unique ID and starts a dedicated thread.

Step 3: User Authentication

- The client logs in using a username and password.
- The password is hashed before being sent to the server.
- The server verifies the credentials from a user database.

Step 4: Key Exchange (Diffie-Hellman)

- After authentication, the client and server perform key exchange.
- This allows both parties to share a secret key securely.
- Once exchanged, this key is used for AES encryption.

Step 5: Secure Communication Begins

- The client types a message.
- The message is encrypted using AES before sending it to the server.
- The server receives the encrypted message and forwards it to the recipient.
- The recipient decrypts the message using the shared key.

Step 6: Message Integrity Check (HMAC)

- The sender generates an HMAC for the message.
- The receiver verifies the HMAC to ensure the message wasn’t tampered with.

Step 7: Connection Termination

- When a client disconnects, the server removes the client from the active list.
- The client's session ends, and all keys are discarded.

3. Technologies Used

- **Socket Programming** → For real-time communication.
- **AES Encryption** → For secure message confidentiality.
- **Diffie-Hellman** → For secure key exchange.
- **HMAC (SHA-256)** → For message integrity verification.
- **Password Hashing (SHA-256/Bcrypt)** → For user authentication.

4.Security Features

- **End-to-End Encryption:** Only sender & receiver can read messages.
- **Message Integrity:** HMAC ensures no message tampering.
- **Secure Authentication:** Hashed passwords prevent leaks.
- **Key Exchange Security:** Diffie-Hellman prevents MITM attacks.
- **Prevention Against Attacks:** Protection from replay attacks, MITM, and brute force.

System Requirements

Functional Requirements:

- **Messaging:** Real-time text message exchange.
- **File Transfer:** Secure transfer of files (documents, images, PDFs).
- **Video and Audio Transfer:** Send encrypted audio and video messages.
- **End-to-End Encryption:** AES encryption for all message types, ensuring confidentiality and message integrity.
- **User Authentication:** Multi-factor authentication to prevent unauthorized access.
- **Key Management:** Efficient symmetric key management for encryption and decryption processes.

Non-Functional Requirements:

- **Scalability:** The system should handle increasing numbers of users and messages.
- **Performance:** Fast message encryption and decryption, even for large files.
- **Reliability:** 99.99% uptime to ensure the platform is available to users at all times.

Software Requirements:

- **Programming Language:** C/Java
- **Libraries:** OpenSSL for cryptography, socket programming
- **Operating System:** Windows/Linux

Hardware Requirements:

- **Processor:** Intel i3 or higher
- **RAM:** Minimum 4GB
- **Storage:** At least 500MB free space

Application of the Project

- Secure Corporate Communication to protect sensitive data.
- Encrypted Government & Military Messaging for classified communication.
- Privacy-Focused Personal Messaging without data tracking.
- Safe File Sharing between trusted parties.
-

SWOT Analysis

Strengths:

- Strong end-to-end encryption prevents eavesdropping.
- Secure key exchange mechanism prevents MITM attacks.
- HMAC ensures message integrity and tamper detection.

Weaknesses:

- Runs only on localhost (not internet-based).

Opportunities:

- Can be extended for global network communication.
- Can be integrated into mobile applications for enhanced usability.

Threats:

- Potential brute-force attacks on authentication.
- Requires continuous security updates to prevent emerging threats.

PERT Chart

The PERT (Program Evaluation and Review Technique) Chart outlines key project phases:

- System Design & Planning
- Server & Client Setup
- Key Exchange Implementation
- AES Encryption & HMAC Integration
- File Transfer Security
- Testing & Debugging
- Final Deployment & Documentation

References

- Cryptography and Network Security by William Stallings.
- NIST (National Institute of Standards and Technology) guidelines on encryption.
- Research papers on AES, Diffie Hellman, and HMAC implementations.