A SYNOPSIS ON

Secure Spam Protected Messaging Platform

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Harshit Tewari 2261255

Tanmay Shah 2261569

Harshita Jalal 2261257

Shivam Rawat 2261527

Under the Guidance of

Mr. Anubhav Bewerwal

Assistant Professor

Project Team ID: 112

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **"Secure Spam Protected Messaging Platform"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Mr. Anubhav *Bewerwal*, *Assistant Professor***, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Harshit Tewari	**2261255**
Tanmay Shah	**2261569**
Harshita Jalal	**2261257**
Shivam Rawat	**2261527**

The above mentioned students shall be working under the supervision of the undersigned on the **"Secure Spam Protected Messaging Platform"**

Signature
Supervisor

Signature
Head of the Department

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members: **Signature with Date**

1.

2.

Table of Contents

Chapter 1

Introduction and Problem Statement

1. Introduction

In the modern era of digital communication, instant messaging applications have become an essential part of personal and professional interactions. However, most traditional messaging platforms rely on centralized servers and internet connectivity, raising concerns regarding privacy, security, and network dependency. In environments such as corporate offices, educational institutions, or research facilities, where secure and controlled communication is required, dependency on the internet can be a major limitation.

To address this issue, we propose a Command-Line Interface (CLI)-based Peer-to-Peer (P2P) Chat Application that allows users to communicate securely over a Local Area Network (LAN) without requiring an internet connection. Unlike conventional messaging applications, this system eliminates the need for third-party servers, ensuring that all communication remains private within the network.

The application leverages socket programming to establish direct peer-to-peer connections and uses multithreading to handle multiple conversations simultaneously. To ensure data security, it integrates Data Encryption Standard (DES) encryption, protecting messages from unauthorized access during transmission. This encryption mechanism ensures that only the intended recipient can decrypt and read the messages, making the system suitable for confidential communication.

Since the project is CLI-based, it is lightweight, fast, and easy to deploy on any system without the need for complex graphical user interfaces. This design choice also makes the application efficient in terms of resource consumption, allowing it to run on low-end devices without performance issues.

By enabling secure, real-time messaging within a closed network, this project provides a cost-effective and reliable alternative for organizations, educational institutions, and businesses seeking a secure internal communication system without relying on internet-based services.

2. Problem Statement

Traditional messaging platforms often rely on centralized servers, which pose risks such as privacy concerns, server downtime, and data breaches. Additionally, existing chat applications may lack encryption, making user conversations susceptible to eavesdropping and cyberattacks.

The key challenges this project aims to address are:

1. Direct Peer-to-Peer Communication – Eliminating the need for a central server and enabling direct connections between users.
2. Secure Message Transmission – Implementing end-to-end encryption to prevent unauthorized access to messages.

3. Efficient Multi-Peer Communication – Managing multiple connections using multithreading, ensuring seamless message exchange.
4. LAN and Global Connectivity – Allowing communication within a local network while also supporting remote connections via port forwarding.
5. Spam Detection – Filtering out potentially harmful or unwanted messages to improve user experience.

By addressing these issues, the project provides a secure, efficient, and decentralized chat system that can be used for various applications, including private communications, enterprise networks, and secure information exchange.

Chapter 2

Background/ Literature Survey

The need for **secure peer-to-peer (P2P) communication** has grown significantly with the increasing risks of cyber threats, data breaches, and unauthorized surveillance. Traditional messaging applications often rely on centralized servers, where messages pass through third-party infrastructures, making them vulnerable to attacks and privacy violations. In contrast, P2P communication eliminates central intermediaries, allowing direct messaging between users. However, without proper encryption, messages transmitted over a network are susceptible to interception and decryption by attackers.

Encryption in P2P communication has evolved over time, with different algorithms being used to protect data integrity and confidentiality. One of the earliest and widely recognized encryption standards is **Data Encryption Standard (DES)**, developed by IBM and standardized by NIST. DES is a symmetric-key encryption algorithm that operates on 64-bit blocks of data and uses a 56-bit key for encryption and decryption. Despite being succeeded by more advanced algorithms like AES, DES remains a fundamental cryptographic technique for understanding block cipher operations.

In the context of P2P chat applications, implementing DES encryption ensures that messages remain confidential while being transmitted over a local area network (LAN) or the internet. The integration of **socket programming** enables direct connections between peers, while **multithreading** allows simultaneous handling of multiple conversations. Compared to traditional encrypted messaging applications that depend on centralized servers, this system provides a **decentralized and self-managed** approach, offering better control over message security.

Several studies and implementations have demonstrated the efficiency of symmetric encryption in real-time messaging. While DES has known vulnerabilities against brute-force attacks, it still serves as an essential building block for secure communication. This project explores the practical application of DES encryption in a P2P chat system, showcasing how cryptographic techniques can enhance network security and private messaging.

Chapter 3

Objectives

The primary objective of this project is to develop a secure peer-to-peer (P2P) chat application that enables direct communication between users within a local network (LAN) without requiring an internet connection. This makes it particularly useful for campuses, organizations, or corporate environments, where internal communication is essential, and data privacy is a priority. The key objectives of this project are as follows:

1. **Offline Communication within a Local Network** – The system allows peers to connect and exchange encrypted messages within a LAN, making it ideal for universities, workplaces, or military bases where internet-independent communication is needed.
2. **Secure Message Transmission using DES Encryption** – To protect the confidentiality of conversations, the application implements Data Encryption Standard (DES), ensuring that messages remain encrypted while being transmitted over the network.
3. **Decentralized Peer-to-Peer Architecture** – Unlike traditional messaging platforms that rely on a centralized server, this chat application follows a P2P model, where users communicate directly without intermediaries, reducing the risks of external surveillance or data breaches.
4. **Multi-User Support with Efficient Threading** – The system allows multiple peers to communicate simultaneously using socket programming and multithreading, ensuring smooth real-time message exchange.
5. **Spam Detection and Message Filtering** – To enhance usability and prevent unnecessary or malicious messages, the application integrates a basic spam detection **mechanism** that filters common spam phrases before sending messages.
6. **Portability and Easy Deployment** – The chat system is lightweight and can be deployed across various devices within an organization, including desktops, laptops, making it a versatile solution for internal communication.
7. **Potential for Future Enhancements** – The system provides a foundation for further development, such as integration with stronger encryption (AES), file sharing, voice communication, or authentication mechanisms for added security.

This project ultimately serves as a fast, secure, and efficient communication tool for closed networks, offering privacy and ease of use without relying on external servers or an internet connection.

Chapter 4

Hardware and Software Requirements

1. Hardware Requirements

S1. No	Name of the Hardware	Specification
1	Processor	Minimum Dual-Core(Intel/AMD)
2	RAM	At least 2GB
3	Storage	At least 100MB free space
4	Network	LAN/Wi-Fi Router for peer connectivity
5	Devices	PC, Laptops, or Mobile Phone(for testing)

2. Software Requirements

Sl. No	Name of the Software	Specification
1	Operating System	Windows
2	IDE	PyCharm(or any Python-supported IDE)
3	Programming Language	Python (version 3.6 or later)
4	Required Libraries	socket, threading, re

Chapter 5

Possible Approach/ Algorithm

1 .Chat Module

This module is responsible for managing peer-to-peer communication using sockets. Each peer in the network acts as both a server and a client, allowing multiple users to connect and exchange messages within a local network without requiring an internet connection.

To facilitate simultaneous communication, the chat module uses multithreading extensively. The program creates a separate thread for each peer connection, ensuring that messages can be sent and received in parallel. When a new peer joins the chat, a new thread is spawned to handle that connection without affecting the ongoing conversations.

- Listening for Incoming Connections:
    - Each peer starts a server socket that listens for incoming connections from other peers.
    - When a new connection request is received, the server accepts it and spawns a new thread to handle that peer's communication.
- Connecting to Other Peers:
    - Peers can manually connect to others by entering their IP address and port number.
    - Once connected, the peer stores the connection and starts a dedicated thread to handle incoming messages from that specific peer.
- Sending and Receiving Messages Simultaneously:
    - The program uses separate threads for sending and receiving messages, ensuring that users do not have to wait for a message to be received before they can send another one.
    - This non-blocking architecture allows multiple peers to communicate efficiently in real time.
- Ensuring Thread Safety:
    - Since multiple threads access shared resources like the list of connected peers, thread synchronization is implemented using threading.Lock().
    - This prevents issues like simultaneous modifications leading to unexpected behavior.

By handling multiple peers through multithreading and socket programming, the chat module ensures smooth, real-time communication in a decentralized manner.

2. Encryption Module

To ensure that messages remain secure and private, the encryption module implements Data Encryption Standard (DES) for encrypting outgoing messages and decrypting incoming ones.

- When a message is typed, it is first passed through the DES encryption function before being transmitted over the network.
- On the receiving end, the encrypted message is decrypted using the same DES key before being displayed to the user.
- Since DES is a symmetric encryption algorithm, both sender and receiver must use the same encryption key.
- This ensures end-to-end encryption, preventing unauthorized access to messages even if intercepted.

The encryption module seamlessly integrates with the chat module, ensuring that every message sent between peers is secure and unreadable by third parties.

3. Spam Detection Module

Spam messages can disrupt meaningful conversations, which is why a spam detection mechanism is integrated into the project. This module uses Naive-based analysis to detect and block potential spam messages before they are sent.

- The system maintains a list of common spam keywords and patterns (e.g., "win money", "click here", "subscribe now").
- When a user attempts to send a message, the module analysis the content using regular expressions and probabilistic filtering techniques.
- If the message contains a high probability of being spam, it is blocked, and the user is notified with a warning.
- This prevents unnecessary clutter and ensures that communication remains clean and meaningful.

By incorporating automated spam filtering, the project adds another layer of security and usability, making it ideal for use in closed networks such as campuses or organizations where professional communication is required.

References

1. Diffie, W., & Hellman, M. E. (1976). *New directions in cryptography*. IEEE Transactions on Information Theory, 22(6), 644–654.

2. Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson.

3. Guzella, T. S., & Caminhas, W. M. (2009). *A review of machine learning approaches to spam filtering*. Expert Systems with Applications, 36(7), 10206–10222.

4. Zhang, Y., Jin, Y., & Wang, L. (2019). *A survey on secure messaging: Requirements, technologies, and challenges*. Future Generation Computer Systems, 100, 348–364.