🧾 Title Page

Secure Chat App

Developed by: Golla Venkatesh

Date: October 2025

Internship Project – Cybersecurity & Web Development

--------------------------------------------------------------------------------------------------------------

## 📘 Introduction

The Secure Chat App is a lightweight, real-time messaging platform designed to ensure confidentiality and integrity of communication between users. Built using Flask and SocketIO, the app leverages RSA for secure key exchange and AES for encrypting message content.

In today’s digital world, privacy is paramount. This project demonstrates how end-to-end encryption can be implemented in a simple web-based chat system, making it ideal for learning and showcasing secure communication principles.

The app allows users to register, select a recipient, and send encrypted messages. Each message is protected using a randomly generated AES key, which is itself encrypted using the recipient’s RSA public key. This ensures that only the intended recipient can decrypt and read the message.

--------------------------------------------------------------------------------------------------------------

## 🧠 Architecture Description

1. User registers → RSA key pair is generated

2. User sends message → AES key is generated

3. Message is encrypted with AES

4. AES key is encrypted with recipient’s RSA public key

5. Encrypted message and key are sent via SocketIO

6. Recipient decrypts AES key using their RSA private key

7. Message is decrypted using AES key

## 🧩 Features

* RSA key generation for each user
* AES encryption for message confidentiality
* Real-time messaging via Flask-SocketIO
* Decryption endpoint for testing
* Simple and intuitive UI
* Encrypted values printed for transparency and debugging

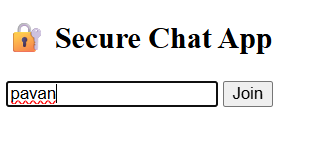
## 📸 Screenshots

A screenshot of a chat box

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.



A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 🧪 Testing & Output

To verify encryption and decryption, the app prints the AES-encrypted message and RSA-encrypted key in the terminal. These values can be copied into a test script (`test\_decrypt.py`) to simulate secure decryption.

Sample Output:

Encrypted AES Key: MIIBIjANBgkqhkiG9...

Encrypted Message: U2FsdGVkX1+9a2V...

Decryption Response:

{'decrypted\_message': 'Hi pavan'}

## 💻 Code Snippets

app.py – Register User

@socketio.on('register')

def handle\_register(data):

username = data['username']

private\_key, public\_key = generate\_rsa\_keys()

users[username] = {'private': private\_key, 'public': public\_key}

crypto\_utils.py – AES Encrypt

def aes\_encrypt(message, key):

iv = os.urandom(16)

cipher = Cipher(algorithms.AES(key), modes.CFB(iv))

encryptor = cipher.encryptor()

return iv + encryptor.update(message.encode()) + encryptor.finalize()

## script.js – Send Message

function sendMessage() {

const msg = document.getElementById("message").value;

socket.emit('send\_message', {

username: username,

recipient: recipient,

message: msg

});

}

# 📝 Conclusion & Future Scope

This project demonstrates how secure communication can be achieved using a combination of RSA and AES encryption. By integrating Flask-SocketIO, real-time messaging becomes both efficient and private.

Challenges faced included managing key exchange securely and debugging encryption workflows. These were overcome through step-by-step testing and modular design.

Future improvements:

- Add user authentication and login

- Store chat history securely

- Add a “Decrypt” button in the UI

- Use HTTPS for secure transport