CSE7101- Capstone Project
Review-2

Secure AI Chat System (End-to-End Encryption + AI Moderation)

Batch Number: COM_58

Roll Number :
20221COM0136

Name:
R. Charu Swathi Sree

Under the Supervision of,

Dr.Ruhin Kouser R
Assistant Professor – Senior Scale
School of Computer Science and Engineering
Presidency University

Name of the Program: B.tech. Computer Engineering (AI&ML)
Name of the HoD: Dr.Pallavi R
Name of the Program Project Coordinator: Dr.Debasmita Mishra
Name of the School Project Coordinators: Dr. Sampath A K , Dr. Geetha A

Content

Abstract

- Secure digital communication has become a fundamental requirement in modern society; however, it faces growing challenges such as **privacy violations, cyberbullying, hate speech, spam, and phishing attacks**. While popular messaging platforms employ **end-to-end encryption (E2EE)** to protect user privacy, they lack proactive mechanisms to prevent harmful content from being transmitted in real time.
- This project proposes a **Secure AI Chat System** that uniquely integrates **end-to-end encrypted communication with AI-based content moderation applied before encryption**. The system ensures that harmful messages are detected and blocked at the sender’s device while preserving complete privacy during transmission and storage.
- By leveraging **AES-256 and RSA-2048 encryption**, **real-time WebSocket communication**, and **Transformer-based AI models (BERT)** for toxicity detection, the proposed solution delivers a **privacy-preserving, safe, and scalable chat platform** suitable for academic, enterprise, and secure communication environments.

Algorithm

•**Client Login Request:**
User submits credentials (email/phone + password) via the app.
Request is sent over **TLS/SSL**.

•**Server Validation:**
Backend verifies credentials against the users database.
Passwords are validated using **bcrypt/argon2 hashing**.

•**Session Token Issuance:**
On success, server issues a **JWT** containing user_id, role, and expiration (exp).
Client stores token securely and includes it in all API calls.

•**Public Key Registration (One-time / On first login):**
Each user generates an **RSA key pair** on the device (or securely on server).
The **public key** is uploaded and stored in the server directory for key exchange.

- **Stage 2: Pre-Encryption AI Moderation (Safety Filter)** 🤖
- This stage ensures harmful content is blocked **before encryption**, preserving both privacy and safety.
- **Message Composition:**
  User types a message in the chat UI.
- **Local / Client-Side AI Check:**
  The message is passed to a **toxicity/spam/phishing classifier** (e.g., BERT-based).
  Output includes:
  - toxicity_score
  - spam_score
  - phishing_score
  - label (SAFE / UNSAFE)
- **Decision Rule:**
  If label == UNSAFE OR score exceeds threshold:
  - Show warning ("Message may be abusive/spam")
  - Block sending or require user confirmation (based on admin policy)
- **Safety Log (Optional):**
  Store only **non-sensitive metadata** (timestamp + label + scores) — **not plaintext**.

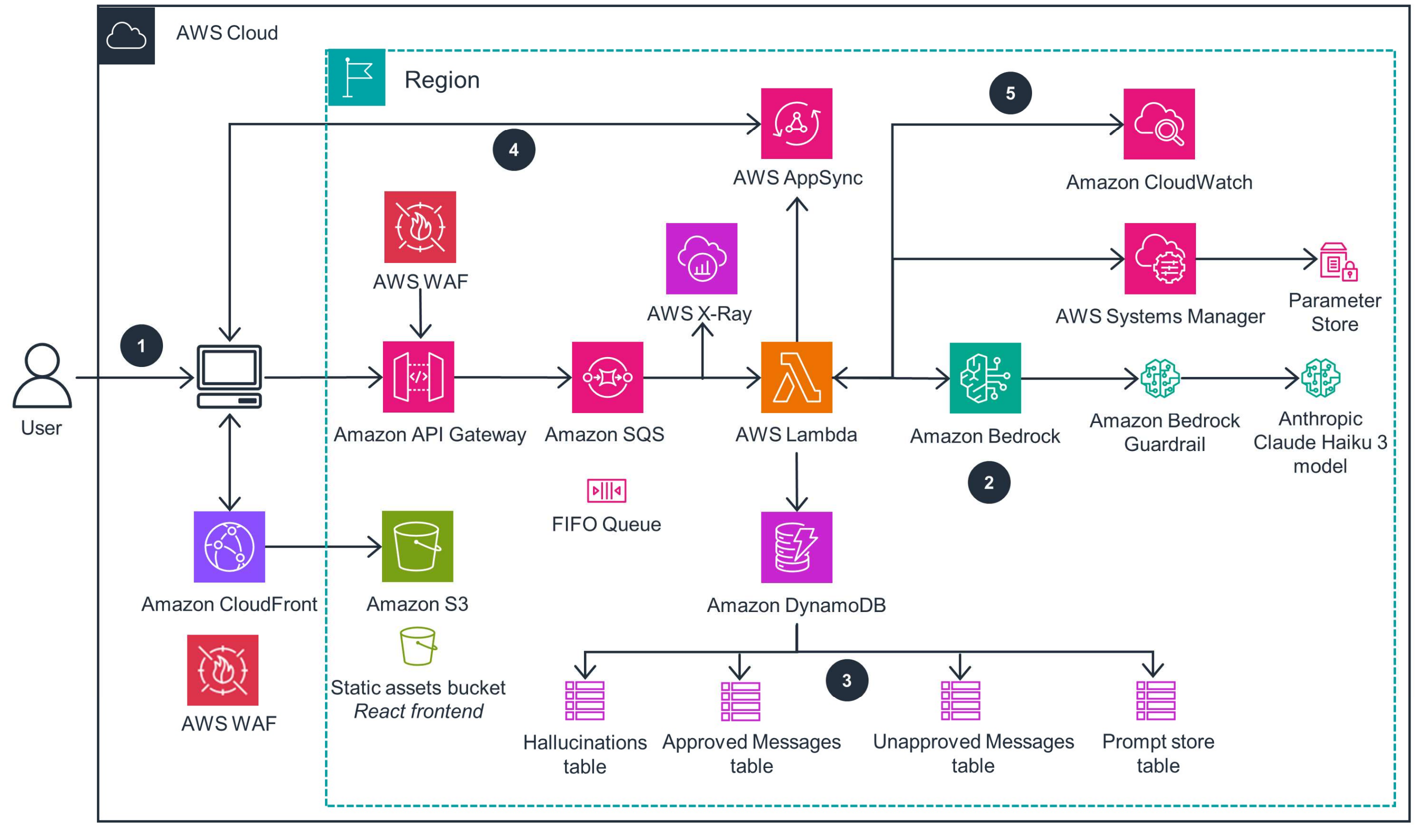
- **Stage 3: End-to-End Encryption & Key Exchange** 🔒
- This stage encrypts the approved message so only sender and receiver can read it.
- **Receiver Public Key Fetch:**
  Client requests receiver's RSA public key from server directory.
- **Symmetric Key Generation:**
  Client generates a random **AES-256 session key** and IV/nonce.
- **Message Encryption (AES):**
  Plaintext message is encrypted using **AES-GCM (preferred)** for confidentiality + integrity.
  Output: ciphertext, auth_tag, nonce.
- **Key Wrapping (RSA):**
  AES session key is encrypted using receiver's **RSA public key** → encrypted_aes_key.
- **Packet Formation:**
  Send payload:
  { encrypted_aes_key, ciphertext, nonce, auth_tag, sender_id, timestamp }

- **Stage 4: Real-Time Transmission & Secure Storage** 📩
- This stage ensures instant delivery and safe storage without exposing message content.
- **Real-Time Send:**
  Payload is transmitted via **WebSockets / Socket.IO** to server.
- **Server Relay (Zero-Trust):**
  Server does not decrypt anything. It only:
  - routes to receiver if online, OR
  - stores encrypted payload if offline
- **Encrypted Storage:**
  Database stores only encrypted payload fields (ciphertext, wrapped key, etc.).
  No plaintext is stored

- **Stage 5: Receiver Decryption & Integrity Verification** ✅
- This stage allows only the receiver to read the message securely.
- **Encrypted Payload Received:**
  Receiver gets the payload via WebSocket or fetches from server.
- **AES Key Recovery:**
  Receiver decrypts encrypted_aes_key using their **RSA private key** to get AES key.
- **Message Decryption (AES-GCM):**
  Receiver decrypts ciphertext and verifies integrity using auth_tag.
  If tag validation fails → message rejected (tampering detected).
- **Display Message:**
  Plaintext is shown in UI. (Optional local storage can be encrypted.)

- **Stage 6: Admin Controls & Reporting (Optional)** 📊
- Admin can configure moderation thresholds and policies (warn vs block).
- Dashboard shows aggregated stats: blocked count, spam attempts, toxicity trends.
- Reports contain **only analytics**, not plaintext messages.

Architecture Diagram



Proposed Method & Feasibility Study

Feasibility Study

1. Technical Feasibility

- AI models and crypto libraries are mature and open-source.
- Smartphones and web apps can handle required computation.

2. Economic Feasibility

- No special hardware required.
- Cloud-based deployment is cost-effective.

3. Operational Feasibility

- Simple UI similar to existing chat apps.
- Minimal user training required.

Literature Survey

- **2.1 Secure Messaging & End-to-End Encryption**
- [1] Secure Messaging Systems using E2EE (IEEE Communications Surveys)
  → Demonstrated that AES-RSA based encryption ensures confidentiality and integrity.
  → *Applied in our project:* Used as the cryptographic foundation for secure messaging.
- [2] Privacy-Preserving Communication Protocols (IEEE Access)
  → Highlighted the importance of zero-trust server architectures.
  → *Applied in our project:* Server never accesses plaintext messages.
- **AI-Based Content Moderation**
- [3] Hate Speech Detection using BERT (IEEE Access)
  → Proved transformer models outperform traditional ML in toxicity detection.
  → *Applied in our project:* Used for real-time AI moderation.
- [4] AI for Online Safety & Cyberbullying Prevention
  → Showed that proactive moderation reduces harmful exposure.
  → *Applied in our project:* Pre-encryption moderation pipeline.
- **Identified Research Gap**
- Existing platforms focus on **privacy OR moderation**, not both.
- Real-time moderation is absent due to encryption constraints.
- Our project bridges this gap using **client-side AI moderation**.

Objectives

- The primary objective of this project is to design and implement a **secure and intelligent real-time communication system**. The detailed objectives include:
- To ensure **confidential communication** using end-to-end encryption (AES & RSA).
- To prevent unauthorized access to message content by servers or third parties.
- To integrate **AI-based toxicity, spam, and phishing detection**.
- To apply **content moderation before encryption**, ensuring proactive safety.
- To maintain real-time performance with minimal latency.
- To design a scalable and user-friendly chat interface.

Existing Methods and Drawbacks

- **Existing Methods**
- WhatsApp, Signal, Telegram use E2EE.
- Moderation happens post-delivery via user reports.
- **Drawbacks**
- Harmful messages reach users before action.
- No real-time prevention of cyberbullying or scams.
- Servers cannot intervene due to encryption.

Proposed Method & Feasibility Study

- **Proposed Method**
- The proposed system integrates **AI moderation and cryptographic security** in a layered architecture:
- **1. Secure User Authentication**
- JWT-based login and session management.
- **2. Pre-Encryption AI Moderation**
- Message analyzed using BERT-based toxicity classifier.
- Harmful messages are blocked or warned before sending.
- **3. End-to-End Encryption**
- Approved messages encrypted using AES-256.
- AES key exchanged using RSA-2048.
- **4. Real-Time Communication**
- WebSockets enable instant message delivery.
- **5. Secure Storage**
- Only encrypted messages stored in the database.

Modules

•**User Authentication Module**
•Secure login, JWT session handling.

•**AI Moderation Module**
•Toxicity, hate speech, spam, phishing detection.

•**Encryption & Key Management Module**
•AES-256 encryption, RSA-2048 key exchange.

•**Real-Time Chat Module**
•WebSocket-based instant messaging.

•**Admin & Analytics Module**
•Moderation statistics, system monitoring.

Hardware and Software Details

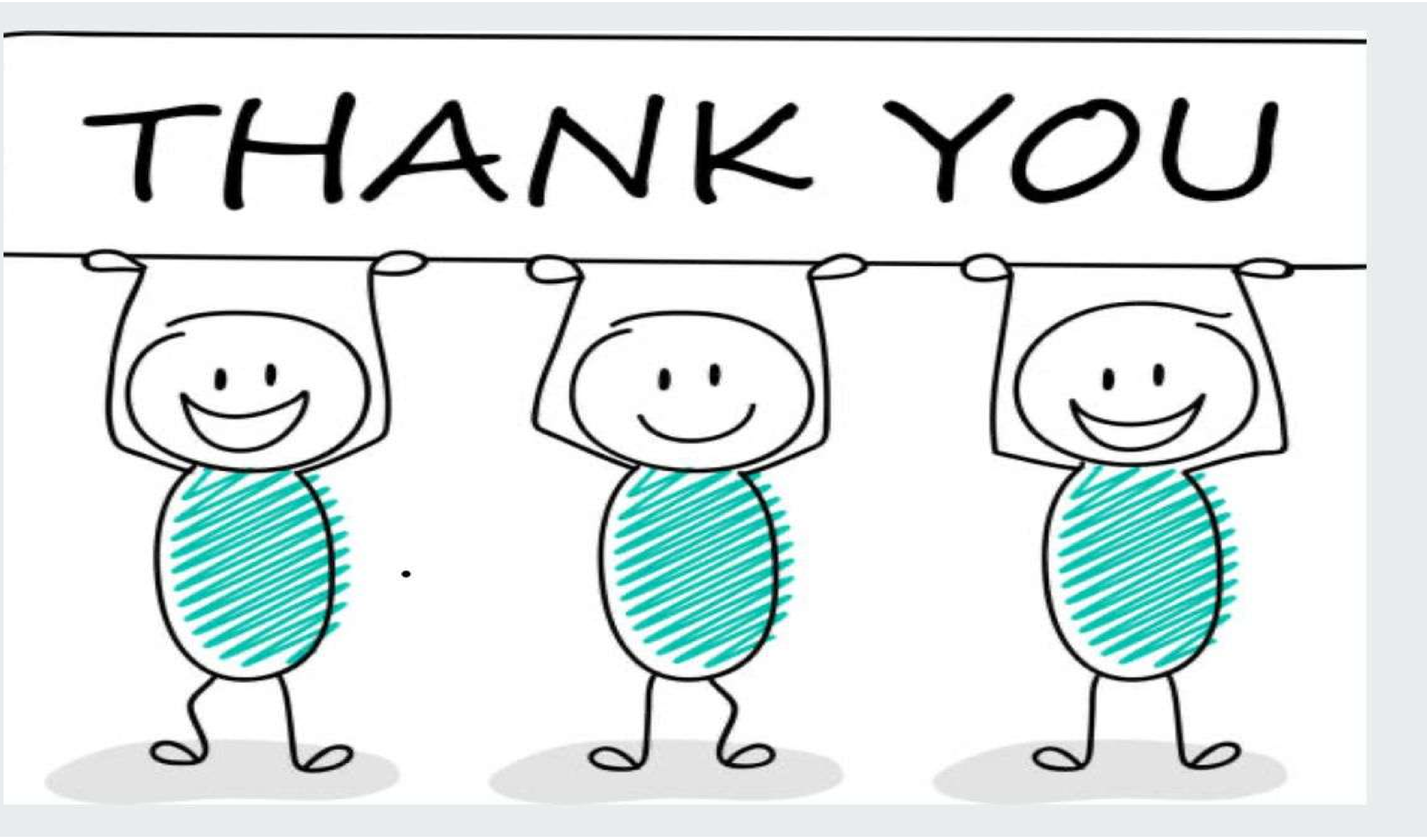
- **Hardware Requirements**
- Smartphone / Laptop
- Minimum 8 GB RAM (server)
- Internet connectivity
- **Software Requirements**
- Frontend: React / Flutter
- Backend: Flask / Node.js
- Database: MongoDB
- AI: TensorFlow / PyTorch
- Security: Crypto libraries, SSL/TLS

CONCLUSION

- **KEY INNOVATION**
- **Pre-Encryption AI Moderation (Core Novelty)**
- Combines privacy and safety without compromise
- Zero-trust server architecture
- Real-time harmful content prevention

- **FUTURE SCOPE**
- Voice & image moderation
- Federated learning for privacy-preserving AI
- Enterprise-grade secure communication
- Integration with secure healthcare & defense systems

- The Secure AI Chat System successfully demonstrates that **privacy and safety can coexist**. By combining **AI-based moderation with end-to-end encryption**, the system offers a practical and innovative solution to modern communication challenges.


THANK YOU