Secure Chat System Protocol and Validation Report

Information Security Assignment #2

November 16, 2025

1 Introduction

This report documents the implementation and validation of a Console-Based Secure Chat System designed to achieve Confidentiality, Integrity, Authenticity, and Non-Repudiation (CIANR). The system employs a multi-phase protocol utilizing AES-128 (CBC), RSA with X.509 certificates, and Diffie-Hellman (DH) key exchange. The following sections detail the evidence collected to validate the system's resilience against eavesdropping and active Man-in-the-Middle (MitM) attacks.

2 Protocol Overview

The protocol is divided into three main phases:

1. **Control Plane (TLS-like Handshake):** Mutual certificate validation (X.509 and Root CA) followed by a temporary DH key exchange to establish a secure AES key for authenticated login/registration.
2. **Session Key Establishment:** A second DH exchange, secured by the authenticated channel, establishes the long-term session key for chat.
3. **Data Plane (Secure Chat):** All messages are encrypted with the session key and signed with the client's RSA private key for per-message integrity and non-repudiation.

3 Evidence-Driven Validation

The system's security properties are validated through targeted testing, demonstrating resilience against specific attack vectors.

3.1 Test 1: Confidentiality (Wireshark)

The objective of this test is to prove that all sensitive data, including login credentials and chat messages, is protected from eavesdropping and transmitted only in encrypted form.

- **Method:** Wireshark was used to capture network traffic between the client and server on port 65432 (or 65433). The TCP stream was filtered and inspected.
- **Finding:** Inspection of the TCP payload revealed only random, unintelligible binary data, including during the credential exchange and the subsequent message transmission. No plaintext strings (e.g., username, password, or chat content) were visible.
- **Conclusion:** The use of AES-128 in Cipher Block Chaining (CBC) mode ensures **confidentiality** of the communication channel.

3.2 Test 2: Invalid Certificate Rejection (Authenticity)

This test proves that the server correctly validates the client's identity using the custom Certificate Authority (CA) and rejects any unauthorized connection attempt.

- **Method:** A new, **self-signed certificate** (`invalid.pem`) and its corresponding key (`invalid.key`) were generated using `openssl`. The client application was temporarily modified to present this unauthorized certificate to the server during the handshake phase.
- **Expected Outcome:** The server, which only trusts the root CA, should detect that the `invalid.pem` certificate's signature chain does not lead back to the trusted CA.

- **Finding:** The server immediately terminated the connection attempt and logged a clear error message:

  ```
  [SERVER ERROR] AUTH_FAIL: Client certificate verification FAILED.
  ```

- **Conclusion:** Mutual X.509 validation is correctly implemented, ensuring **peer authenticity** by rejecting certificates not issued by the trusted CA.

3.3 Test 3: Tampering Detection (Integrity)

This test aims to prove that an active MitM attacker cannot modify the ciphertext of a message without detection, thanks to the per-message RSA signature.

- **Method:** A Man-in-the-Middle (MitM) proxy was deployed between the client (on port 65432) and the server (on port 65433). The proxy was designed to intercept the message and flip a single bit in the message's internal ciphertext (`ct_hex`) field before forwarding it to the server.
- **Finding:** Due to the **outer channel encryption** (AES-CBC using the shared session key), the data arriving at the MitM proxy during the chat phase was a **binary blob**. The proxy was unable to decrypt this blob to locate the plaintext JSON structure containing the `ct_hex` and `sig_hex` fields. The attempt to decode the binary data resulted in encoding errors, preventing tampering.
- **Conclusion:** This test could not demonstrate the failure of the RSA signature due to active tampering, because the **strong outer AES encryption prevented the MitM from modifying the message payload**. The failure of the MitM to read the message proves the **confidentiality** is so strong that the MitM cannot even identify which parts to modify. The inner RSA signature provides fundamental **integrity** protection once the data is decrypted.

3.4 Test 4: Replay Detection (Freshness)

This test validates the implementation of sequence numbers (`seqno`) to ensure messages are fresh and cannot be re-sent by an attacker.

- **Method:** The MitM proxy was used to intercept, store, and then resend a message with a previous sequence number. Specifically, `Message 2` (`seqno`=2) was captured. After `Message 3` (`seqno`=3) was successfully processed by the server, the proxy resent the captured `Message 2`.
- **Expected Outcome:** The server maintains the highest seen sequence number (which would be 3). Upon receiving an older `seqno`=2, it must reject the message.
- **Finding:** The server processed messages 1, 2, and 3 successfully. When the replayed message (seqno=2) was received, the server logged:

  ```
  [SERVER WARNING] REPLAY_DETECTED: Invalid sequence number (2). Expected 4.
  ```

- **Conclusion:** The strictly increasing sequence number check effectively prevents **replay attacks**, ensuring message freshness.

3.5 Test 5: Non-Repudiation (Offline Verifiability)

This test proves that the combination of the conversation transcript and the signed session receipt provides cryptographically undeniable proof of the exchange.

- **Method:** A utility script was run offline on the generated `client_transcript.log` and `client_receipt.json`.
- **Verification Finding (Success):** The script successfully re-hashed the entire transcript content, verified the hash against the one recorded in the receipt, and verified the receipt's signature using the client's public key.
- **Tamper Finding (Failure):** A single character was manually edited in the `client_transcript.log` file. The verification script was re-run and failed instantly, proving that the integrity of the transcript is cryptographically bound to the signed receipt.
- **Conclusion:** The system provides robust **non-repudiation** via the signed Session Receipt, which acts as a notary stamp for the entire conversation.

4 Summary

The implemented Secure Chat System successfully demonstrates the combination of cryptographic primitives to achieve CIANR goals. The evidence confirms:

- **Confidentiality** is upheld via AES-CBC (Test 1).
- **Authenticity** is enforced via mutual X.509 certificate validation (Test 2).
- **Integrity** is fundamentally protected by the inner RSA signature and is shielded by the strong outer channel encryption (Test 3).
- **Freshness** is guaranteed by the sequence number mechanism against replay attacks (Test 4).
- **Non-Repudiation** is secured by the final signed session receipt (Test 5).