CS6349: Project Report

Mashroor Hasan Bhuiyan — Net ID: dal650738
Fatema Tuj Johora — Net ID: dal937703

October 10, 2025

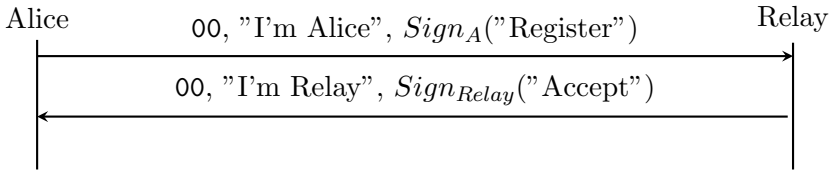
Problem Statement

The goal of this project is to design and implement a secure relay-based chat system in which multiple clients communicate through an untrusted Relay Server. The system must allow clients to establish authenticated, confidential, and tamper-evident sessions while ensuring forward secrecy and protection against replay attacks.

System Design

Registration & Authentication

`00` = Flag for Registration message
$Sign_A(x) = x$ signed by Alice i.e., x encrypted using Alice's private key.

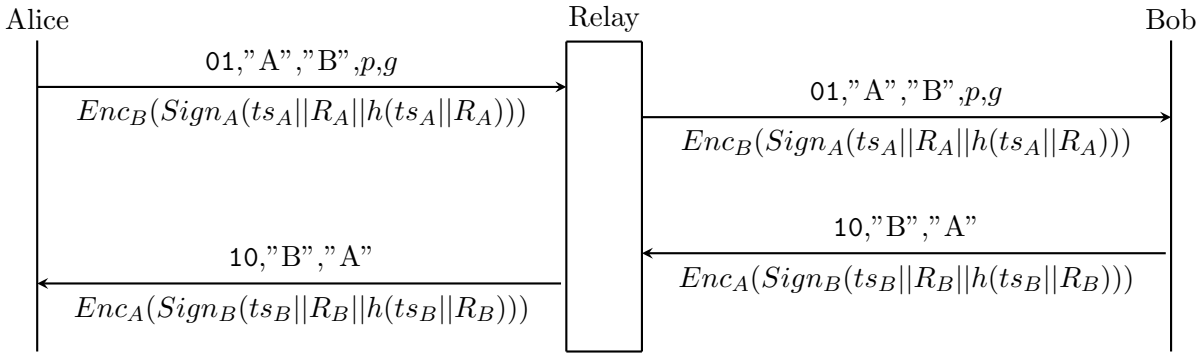


Alice initiates a registration with the Relay and the Relay send a reply back. Both authenticate each other using their signatures.

Session Setup

Session Establishment

$$\begin{aligned}
\texttt{01} &= \text{Flag for initiating session} \\
\texttt{10} &= \text{Flag for session acknowledgment} \\
Enc_A(x) &= x\text{encrypted using Alice's public key} \\
ts &= \text{timestamp} \\
r_A &= \text{private session key for Alice} \\
R_A &= g^{r_A} \text{mod} p \\
h() &= \text{hash function}
\end{aligned}$$



Alice sends all the necessary data to contact Bob via the Relay server. The Relay identifies who the receiver is and passes the message bytes as it is to Bob. Bob replies with messages of his own to Alice, via the Relay.

Session Key Generation

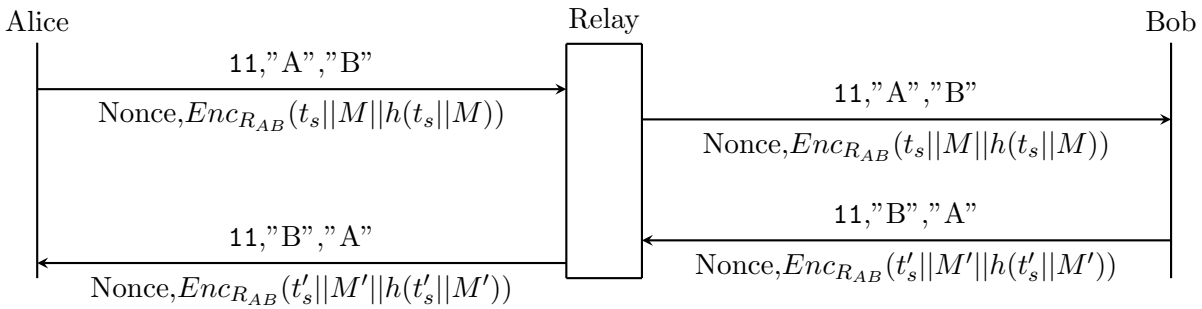
Alice and Bob will now generate their shared session key using R_A and R_B.
We will calculate the shared session key using the Diffie-Hellman algorithm. Alice has her private session key r_A, and Bob's public session key $R_B = g^{r_b} \text{mod} p$.
$\therefore$ For Alice,

$$R_{AB} = (R_B^{r_a}) \text{mod} p = g^{r_A \cdot r_B} \text{mod} p$$

Similarly, for Bob,

$$R_{AB} = (r_B^{R_A}) \text{mod} p = g^{r_A \cdot r_B} \text{mod} p$$

Secure Messaging



We will implement the **HMAC-CTR algorithm**, which is a keyed-hash based encryption scheme. The algorithm combines the confidentiality of the counter (CTR) mode with the authentication and pseudorandom properties of HMAC.

Message Construction

Let:

$$M = \text{Plaintext of arbitrary length},$$
$$ts = \text{Timestamp of 6 bytes}.$$

The message body is constructed as:

$$\text{Message} = ts \,||\, M$$

where the timestamp precedes the plain text to prevent replay attacks.

We divide the concatenated message $(ts||M)$ into n blocks, each 32 bytes long (the output size of HMAC-SHA256):

$$M = M_1||M_2||\ldots||M_n$$

Keystream Generation

The keystream for each block is generated using HMAC as follows:

$$Ks_i = \text{HMAC}(R_{AB}, N_e||\text{count}_i)$$

where:

- R_{AB} : Shared session key between entities A and B,
- N_e : Nonce (random or unique per-encryption value to prevent keystream reuse),
- count_i : Integer counter for block i. The initial counter count_0 is derived from the session key and the message index.

HMAC Definition: Let the following constants be defined for SHA-256 (block size $B = 64$ bytes):

$$\text{ipad} = 0x36 \text{ repeated } B \text{ times}, \quad \text{opad} = 0x5C \text{ repeated } B \text{ times}$$

and let $h()$ denote the hash function, i.e., $h() = \text{SHA-256}()$. Then, the HMAC computation proceeds as:

$$\begin{aligned} \text{innerhash} &= h\big((R_{AB} \oplus \text{ipad}) \,||\, N_e||\text{count}_i\big) \\ \text{HMAC}(R_{AB}, N_e||\text{count}_i) &= h\big((R_{AB} \oplus \text{opad}) \,||\, \text{innerhash}\big) \end{aligned}$$

Thus,

$$\boxed{\text{HMAC}(R_{AB}, N_e||\text{count}_i) = H\Big((R_{AB} \oplus \text{opad}) \,||\, H\big((R_{AB} \oplus \text{ipad}) \,||\, N_e||\text{count}_i\big)\Big)}$$

Encryption (Sender Side)
For each message block M_i, the ciphertext block C_i is computed as:

$$C_i = Ks_i \oplus M_i$$

The sender transmits the nonce N_e along with the ciphertext to the receiver.
Decryption (Receiver Side)
Upon receiving (C_i, N_e), the receiver recomputes the same keystream Ks_i and retrieves the plaintext:

$$M_i = Ks_i \oplus C_i$$

Security Analysis

Confidentiality

Confidentiality means that only the sender and receiver will know the content of messages.

- During session establishment, only the sender and receiver will know the content of the message if his key is not compromised.
- For message exchange, only the sender and receiver know the session key, so except for them, no one else can decrypt the messages.

Integrity

Integrity ensures that data are accurate, reliable and have not been altered by unauthorized users.

- During session establishment, the sender sends the encrypted value of the signed message contents. If an attacker alters some bits, the receiver will not be able to match the hash value.
- For message exchange, the sender sends the hash value of the message contents. If an attacker alters some bits, the receiver will not be able to match the hash value.

Forward Secrecy

We use the Diffie–Hellman key exchange to generate session keys for message communication. Therefore, even if one session key is compromised, the attacker cannot access or decrypt messages from any other session.

Threat Analysis

Replay Attack

- In this protocol, timestamp is sent with every message packet except during the "Registration" process. So, even if the attacker replays any messages, Bob will notice this by comparing the timestamp with his most recent timestamp.
- During session establishment, Bob will decrypt the message with his secret key and then verify Alice's sign. Then he will obtain the timestamp of Alice and R_A, which is the public key for this session, and the hashed value of this. Since the timestamp and its hash value are attached, Bob can easily verify whether a message is a replay.
- For secure message, the timestamp is sent along with the message content including the hash value, which can be decrypted only using the session key.

Message Injection & Tempering

We have already discussed that no attacker could inject messages without being noticed by the receiver. So, even if an attacker injects or tampers with any message, the receiver will eventually detect it.

Impersonating

In the registration step, an attacker cannot pretend to be Alice because she sends a signed message.
In the session setup, the same protection applies since the sender sends an encrypted version of the signed message.
During message exchange, R_{AB} is known only to the sender and receiver, preventing the attacker from impersonating the sender without obtaining the session key. Therefore, in all cases, the attacker cannot impersonate the sender without having access to their cryptographic key.