Secure Relay-Based Chat System

Overview

The Secure Relay-Based Chat System ensures multiple clients communicate security through a central relay server.

The Server Relay forwards messages only and is not trusted with message content.
The system ensures:

- Confidentiality: Relay cannot access messages
- Integrity: Messages are untampered
- Replay Protection: Prevents attackers from resending old messages
- Forward Secrecy: Past messages remain secure when long term key are breached

Threat Model

Attacker Capabilities

- Eavesdropping: listening network traffic between Clients and relay
- Message Tampering: alter message in transit
- Replay Attack: resend old messages
- Impersonation: pose as client or relay

3. Proposed Chat Protocol
This protocol is divided into four phases:

- Registration
- Authentication
- Session Setup
- Messages Exchange

Phase 1: Registration

Relay will get the Client's identity and public key.

a) Client → Relay

{ ClientID, ClientPubKey, Timestamp, Signature_Client }
The Relay will get the Client ID, Client Public Key, Timestamp, and Client Signature.
Signature_Client = Sign{ClientPriv}(ClientID || ClientPubKey || Timestamp)

b) Relay

The relay verifies the signature against known Client public key
The relay stores the digital signature that the client generates.

Security Feature:

- Ensure relay has the correct public key for each client
- Digital signature prevent impersonation during registration

Phase 2: Authentication

Clients will verify that they are communicating with the real relay. The relay confirms if the Client is legitimate.

a) The Client will encrypt a nonce with Relay's Public key and sends it
   $M_1 = E_{PR} (N_C)$
b) Relay decrypts, signs the nonce and returns it
   $N_C = D_{SK}(M_1)$
   $M_2 = Sign_{SK}(N_C)$
c) Client verifies the signed nonce using Relay's public key
   $Verify_{PR} (N_C, M_2) = True$

Client and Relay are Authenticated before the session setup.
Authentication is done using asymmetric encryption.

Phase 3: Session Setup & Key Agreement

Create a new session between the two clients through the Relay.

a) $C_a \rightarrow$ Relay
   {SessionRequest, ReceiverID, $Nonce_A$, $Ephemeral_{DHa}$, $Signature_{Ca}$}
   $Ephemeral_{DHa} = g^a \bmod p$ (public DH value)
   Signature authenticates message

b) Relay $\rightarrow C_b$: Forwards request
c) $C_b \rightarrow$ Relay
   {SessionResponse, SenderID, $Nonce_A$, $Nonce_B$, $Ephemeral_{DHb}$, $Signature_{Cb}$}
   $Ephmeral_{DHb} = g^b \bmod p$ (public DH value)
   Signed response binds C_b's DH to its identity
d) Relay $\rightarrow C_a$: Forwards response

e) Key Derivation
   Shared Secret: $K = (Ephemeral_{DHb})^a \bmod p = (Ephemeral_{DHa})^b \bmod p$
   Salt:
   $Salt = Hash(SessionID \,||\, Nonce_A \,||\, Nonce_B)$
   Session Keys:
   $K_{enc} = HKDF(Salt, K, \text{"encryption"})$
   $K_{mac} = HKDF(Salt, K, \text{"authentication"})$

Security Feature:

- The Relay cannot compute the shared secret which means no access to a or b
- Signatures prevent man in the middle attacks since relay cannot swap DH values

Phase 4: Secure Message Exchange

Since the session keys are derived, clients can now send each other messages securely.

Message Format:
{SessionID, SeqNo, Ciphertext, HMAC}
Ciphertext: XOR(Plaintext, KDF(K_{enc}, SeqNo))
HMAC: HMAC(K_{mac}, SessionID || SeqNo || Ciphertext)

Receiver Processing:

a) verify HMAC (integrity + authenticity)
b) Check SeqNo > last seen (no replay attack)
c) Decrypt using K_{enc}

Key Refresh: Periodically, Clients will re-run ephemeral DH to generate new keys to maintain forward secrecy.

Security Properties

Property	Policy
Confidentiality	End to end encryption using a session key.
Integrity	HMAC detects modification of ciphertext
Authentication	Header signatures and ephemeral sessions keys to identify clients
Replay Protection	Session ID, Nonces, DH key regenerations
Forward Secrecy	DH key regeneration so the Ephemeral key are not recoverable when leaked

Phase 5. Threat Handling

Attack	Defense
Eavesdropping	Encrypted messages with ephemeral key
Tampering or MITM	MHAC detects modification, signatures prevent DH substitution
Replay	Sequence numbers and Nonces
Impersonation	RSA signatures and session setup
Key compromise	Forward secrecy with DH key regen for the past messages