Detailed Security Analysis

The protocol we designed focuses on giving clients a secure way to authenticate themselves to the relay and then communicate safely with other clients. Since the relay sits in the middle of all communication, it is very important that we prevent attackers from pretending to be clients, replaying old messages, or trying to read messages that aren't meant for them. This section goes through the main parts of the protocol and explains how each one helps protect against different kinds of attacks.

---

1. Threat Model

When thinking about security, we assume there is someone on the network who can intercept everything being sent, replay old messages, or even try to modify messages before they reach the relay or the client. The attacker might also try to pretend to be a client or try to trick the relay into accepting fake information.

However, we assume the attacker **does not** have access to anyone's private RSA keys and **cannot** break RSA encryption. We also assume clients and the relay keep their private keys protected on their machines.

---

2. Confidentiality

Confidentiality means that no one except the intended recipient should be able to read the message.

In our protocol, we protect confidentiality by **secret key encryption** and **double encrypting** important messages:

1. The sender first encrypts the message with the secret key created by the receiver and themselves using Diffie-Hellman
2. Then the encrypted message is encrypted with their private key.
3. Then they encrypt that result again using the recipient's public key.

This ensures that:

- Only the relay or the intended client can decrypt the outer layers of the message providing safe transportation (because they have the corresponding private key).

- Even if someone sees the message in transit, they cannot read it.

Client-to-client communication also uses this double-encryption setup so that the relay can forward messages but not decrypt them. This keeps message contents hidden even from the server.

- An additional layer of confidentiality is the secret key shared between clients. Since it uses the Diffie-Hellman approach, it is considerably hard to guess and break into to read a message triple encrypted.
    - E_public key of relay(E _ public key of receiver(E_session key (Message)))

---

3. Integrity

Integrity means the message hasn't been changed while being sent.

Because messages are signed using the receiver's public key, the receiver can check the signature to confirm the message wasn't modified. If an attacker tries to change anything in the encrypted message, the signature check will fail immediately. In addition to that, the real message along with the whole formatted header is encrypted with the shared key between the clients, even if the attacker does alter the public key encryption, altering it would result in a garbage message. Since there is a higher difficulty of successfully guessing public key and secret key, the program is integrity safe. If the public key was to be guessed and something was to be al

---

4. Authentication

Authentication is handled through the three-message challenge–response system:

1. **Client → Relay:**
   Client sends a double-encrypted message proving it really owns its private key.

2. **Relay → Client:**
   Relay decrypts it, then sends back the message along with a challenge encrypted with its own private key.

3. **Client → Relay:**
   The client decrypts the challenge and returns it encrypted with the relay's public key.

If everything matches, the relay knows the client is legitimate.

This prevents attackers from pretending to be clients because they would need the client's private key to complete the challenge correctly. This creates an authentication based server, every client in the server entering will have to authenticate to be let in.

---

5. Replay Protection

Replay attacks happen when someone records a valid message and tries to send it again later. We defend against this in two ways:

- **Sequence numbers:**
  Messages are tagged with numbers that always increase. If the receiver gets a number they've already seen, they immediately drop the message.

- **Random challenges:**
  Every authentication attempt uses a brand-new challenge value, so reusing an old response will not work.

Together, these make replaying old messages useless.

---

6. Anonymity

Clients identify themselves using **pseudonyms** instead of real names. This helps hide real identities from other clients. The relay still sees which client is which, but clients do not expose personal info to each other.

Public keys act as long-term identifiers, so anonymity is limited, but pseudonyms at least keep user identities abstract.

---

7. Forward Secrecy

Our implementation of Diffie-Helman in Client-Client communication helps to ensure forward secrecy so that if any long-term RSA key is compromised past DH encrypted messages will be safe from malicious discovery.

---

8. Attacks the Protocol Defends Against

Man-in-the-Middle (MITM)

MITM attacks rely on tricking one or both sides into accepting modified or fake keys. Our protocol defends against this because:

- Double encryption makes forging signatures impossible.
- The client verifies the relay's challenge is correctly signed.
- The relay verifies the client's challenge response.

A MITM attacker cannot complete the challenge–response steps correctly without the actual private keys.

Replay Attacks

As noted earlier, sequence numbers, and fresh challenges protect against attackers trying to resend old messages.

Message Injection

If an attacker inserts their own message:

- The double encryption won't match the expected format
- The signature won't verify
- The hash (during session) won't match
- Sequence number could be overwritten

So injected messages are always detected.

Impersonation Attacks

An attacker cannot impersonate a client or relay because:

- They would need the private key to sign messages
- They can't generate valid responses to challenges
- The relay won't accept messages without proper encryption

This makes client identity very hard to fake.

---

9. Contributions

Dagmawet Zemedkun:

- Creation and full implementation of Client Class
- Creation and full implementation of Encryption and Decryption algorithm for RSA
- Creation and full implementation of Encryption and Decryption algorithms for Diffie-Hellman
- Creation and full implementation of Key creation algorithm
- Creation and full implementation of SendPublicKey and ReceivePublicKey class
- Creation and full implementation Diffie-Helman algorithm
- Editing and testing Relay/Server Class
    - Logger functionalities
    - Authentication
    - Client listing to new users
    - Client broadcasting of new users
- Edited Detailed Security Analysis

Myles Nelson:

- Creation of Relay/Server Class
- Implemented method of public key distribution after authentication
- Implemented Logger
- Second MSG challenge randomizer
- Facilitated client to client communication in server
- Wrote Detailed Security Analysis
- Wrote Summary of Final Protocol

---

10. Conclusion

Overall, the protocol provides strong protection against many common attacks, including MITM, replay attacks, message injection, and impersonation. The double-encryption setup ensures message confidentiality and integrity, while the challenge–response mechanism makes authentication reliable. Although there are areas that can be improved—especially around forward secrecy and relay trust—the design offers a solid and secure foundation for authenticated client-to-client communication through the relay.

---

11. Summary of Final Protocol

The protocol establishes a secure mutual-authentication process between clients through a central relay, using RSA-based cryptographic operations to create a trusted communication channel. Both the clients and the relay generate their own RSA key pairs and store each other's public keys. Clients identify themselves using pseudonyms, which helps maintain confidentiality and prevents real-identity exposure.

When a client first connects to the relay, the two parties exchange public keys and store them locally. They then begin the mutual-authentication handshake. Each authentication message is encrypted twice: first with the sender's private key (to provide integrity and authenticity), and then with the receiver's public key (to ensure confidentiality).

In the first authentication message, the client sends a randomly generated value that is double-encrypted and intended only for the relay. The relay responds with a message of the form:

Encrypt_clientPublicKey(decryptedFirstMsg | Encrypt_relayPrivateKey(challenge))

Upon receiving and decrypting this second message, the client extracts the challenge and returns it to the relay, encrypted only with the relay's public key. If the relay successfully decrypts this third message and verifies that the challenge matches the one it originally sent, the client is considered authenticated.

After successful authentication, the client shares its pseudonym with the relay, which then records the client as an active and trusted participant. The relay also distributes the new client's public key to all currently connected clients and informs them that a new authenticated user has joined. Likewise, the new client receives the current list of authenticated users and their public keys.

Client-to-client communication follows the same general pattern as the authentication steps: each message is double-encrypted and sent to the relay. The relay removes the outer

encryption layer to identify the message’s intended recipient, then signs the message with its private key before forwarding it. This ensures that messages remain confidential to the recipient while retaining verifiable authenticity.

The first message sent by the client will include the recipient's pseudonym along with a message of the sender's choice. The sender will store their message until they confirm whether or not the receiver possesses a session key. In the event that one doesn’t, Diffie-Helman will begin to create a session key. The initial message will be sent upon completion. Thus protecting communication from forward secrecy or the potential compromising of their RSA keys.