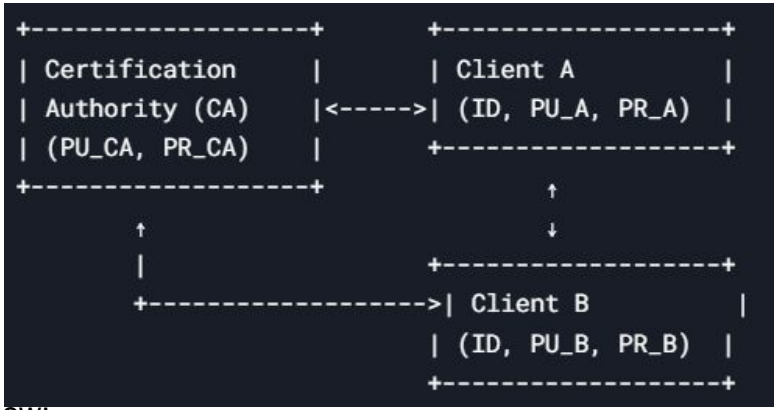
CSE350 - Network Security Assignment 3
Project 0

By:
Deepak Thappa (2021319)
Madhav Krishan Garg (2021333)

Introduction

- Objective:
  - Implement a PKI system with a CA for secure public key distribution.
  - Enable clients to exchange encrypted messages using RSA-based certificates.
- Key Requirements:
  - CA issues signed certificates containing client public keys.
  - Clients verify certificates using the CA's public key.
  - Secure message exchange via RSA encryption/decryption.
- Implementation:
  - Pure Python (no external crypto libraries).
  - Custom RSA key generation, signing, and verification.

System Architecture Diagram

1. Components:
   a. 

2. Data Flow:
   a. Clients register public keys with CA.
   b. CA issues signed certificates.
   c. Clients verify certificates and exchange messages.

Certificate Format

1. **Structure**:
   a. CERT_A = "ID_A|PU_A.n|PU_A.e|timestamp|duration|ID_CA||SIGN_CA"

2. **Fields**:
   a. `ID_A`: Client identifier (e.g., "ClientA").
   b. `PU_A`: Public key (modulus n and exponent e).
   c. `timestamp`: Unix epoch time of issuance.
   d. `duration`: Validity period in seconds.
   e. `SIGN_CA`: SHA-256 hash signed with CA’s private key.

Key Algorithms

1. **RSA Key Generation (custom_rsa.py):**
   a. Generate primes p and q (Miller-Rabin test).
   b. Compute $n = p*q$ and $\varphi(n) = (p-1)(q-1)$.
   c. Choose $e = 65537$, compute $d \equiv e^{-1} \bmod \varphi(n)$.
2. **Signing/Verification:**
   a. Sign: $S = hash(message)^d \bmod n$.
   b. Verify: $hash(message) \equiv S^e \bmod n$.

Workflow Example

1. **Setup:**

```
ca = CertificationAuthority()
clientA = Client("ClientA", ca.public_key)
```

2. **Certificate Issuance:**

```
ca.register_client(clientA.id, clientA.public_key)
certA = clientA.request_certificate(ca)  # Signed by CA
```

3. **Message Exchange:**

```
# ClientA → ClientB: Encrypt with PU_B
encrypted_msg = clientA.encrypt_message("Hello1", PU_B)
decrypted_msg = clientB.decrypt_message(encrypted_msg)  # "Hello1"
```

Verification Process

- **Certificate Validation:**
  - Split `certA` into `data` and `signature`.
  - Verify `signature` using `PU_CA`.
  - Check `timestamp` and `duration` for expiry.
- **Test Cases:**
  - Successful verification of valid certificates.
  - Rejection of tampered certificates.
  - Correct decryption of exchanged messages.