# Network Security (CSE350) Programming Assignment-3

Abhinav Ujjawal (2021120)
Nikhil Suri (2021268)

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
**DELHI**

# RSA-based Public-key Certification Authority

- Each user has a “public-key certificate”, issued by a trusted “certification authority” (or CA). This can be shared with anyone.
- After a certain duration of time, user has to renew the certificate. The certificate has this duration stored as well.
- A receiver must check the certificate, and confirm that received public key is genuine.
- $CERT_A=[(ID_A, PU_A, T_A, DUR_A, INFO_{CA})||ENC_{PR\text{-}CA}(Hash(ID_A, PU_A, T_A, DUR_A, INFO_{CA}))]$

# RSA Algorithm

- Get two prime numbers p and q.
- Now, ***n = p*q*** and ***ϕ= (p-1)*(q-1)***.
- Generate the public and private keys. Select e such that ***1 < e < ϕ*** and ***gcd(e,ϕ) = 1***. Get d such that ***ed = 1 mod ϕ***.
- Private key and public key are {e,n} and {d,n} respectively.
- Next we have to use these keys for encryption and decryption:
  - **$C = M^e \pmod n$**
  - **$M = C^d \pmod n$**

# Encryption and Decryption

- We use RSA algorithm for encryption and decryption. Let:

  $$message=(ID_A, PU_A, T_A, DUR_A, INFO_{CA})$$

- Now using SHA-256, we get the ***h = hash(message)***. Concatenate ‘message’ and ***encrypt(h)*** using RSA and send them to the client.
- The client decrypts the encrypted part of the certificate and verifies that the hash of the message is equal to the decrypted value.
- After this, the clients exchange messages in between them using gRPC. This also involves encryption and decryption.

# RSA-based Public-key Certification Authority

A client will get the public key of another client through the help of a CA. Now, our aim is to:

- build a public-key CA, that responds to requests from clients that seek their own RSA-based public-key certificates OR that of other clients,
- build 2 clients that:
  - send requests to the CA for their own public-key certificates OR that of other clients, and
  - exchange messages with each other in a confidential manner, suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner.

# Algorithm (at CA)

At CA:

- Start listening on the port using gRPC server.
- Connect to the two clients and respond to their messages.
- Do the following for each of the clients:
  - Send CA's public key to the client.
  - Secondly, register the client by adding it in the dictionary of clients.
  - Lastly, it issues the certificate of the other client and sends it to the current client. (i.e. Certificate of B is sent to A and vice versa).
- The CA stops execution after both the clients have been handled.

# Algorithm (at Client)

At Client (A/B):

- The client connects to one of the ports of the server.
- It starts sending and receiving messages using gRPC.
- Depending on the messages:
  - First, it asks for the public key of CA and stores it in an instance of the client class.
  - Second, clients sends the request for registering itself.
  - Lastly, it asks for the Certificate of the other client. (i.e. Certificate B requested by A and vice versa)
- After receiving the certificate, client starts the message conversation with other client on receiving the input, via RSA-based encryption and decry

# Output

```
PS C:\Users\suris\Desktop\NSC_A3> python a.py
Client A registered successfully

Register client B with the CA server and press Enter to continue
Received certificate of B from CA server

Certificate verified

Press Enter to send encrypted message from A to B
Certificate of B is still active.
Decrypted Response: acked1
Certificate of B is still active.
Decrypted Response: acked2
Certificate of B is still active.
Decrypted Response: acked3
PS C:\Users\suris\Desktop\NSC_A3>
```

```
PS C:\Users\suris\Desktop\NSC_A3> python b.py
Client B registered successfully

Register client A with the CA server and press Enter to continue
Received certificate of A from CA server

Certificate verified

Server B is listening on port 50053...
Decrypted message received: hello1
Certificate of B is still active.
Decrypted message received: hello2
Certificate of B is still active.
Decrypted message received: hello3
Certificate of B is still active.
```

```
PS C:\Users\suris\Desktop\NSC_A3> python ca.py
Server is listening on port 50051...
Client A registered
Client B registered
Certificate of B signed
Certificate of A signed
```