PUBLIC-KEY CERTIFICATION AUTHORITY

Aditya Singh Rathore, 2018007

Jaspreet Saka, 2018237

ENCRYPTION AND DECRYPTION

```
1  def encrypt(plainText:str, e:int, n:int)->str:
2      """ Encrypts each character of plain text using RSA."""
3      l = [chr(pow(ord(M),e,n)) for M in plainText];
4      return ''.join(l);
5
6  def decrypt(cypherText:str, d:int, n:int)->str:
7      """Decrypts each character of cypher text which was encrypted using
8      encrypt(...)."""
9      l = [chr(pow(ord(C),d,n)) for C in cypherText];
10     return ''.join(l);
```

CLIENT-TO-CA

- Everyone knows the public key of CA
- Client encrypts using this public key.

CA-TO-CLIENT

- Client registers its keys with CA and gets ID.
- For the first time, Client sends its public key along with request to register to CA.
- After that, CA can access the public key using ID.

CLIENT-TO-CLIENT

- A gets Public Certificate of B from CA.
- B gets public certificate of A from CA.
- They communicate using Public keys from these certificates.

ISSUES

- Obvious vulnerability is that the statistical nature of text remains. Thus, it can be subjected to character frequency analysis.
- Length of plain text is equal to the length of encrypted text.

WHY

- The reasons were purely implementation based.
- Correct way would have been to convert string to a byte array and convert that byte array into a number.
- But the size of string in certificate may become too long.
- If we were to implement this for a product that was to be used in reality rather than a proof of concept, we would have used Symmetric encryption like AES to encrypt data and use RSA to encrypt AES keys.

APPLICATION

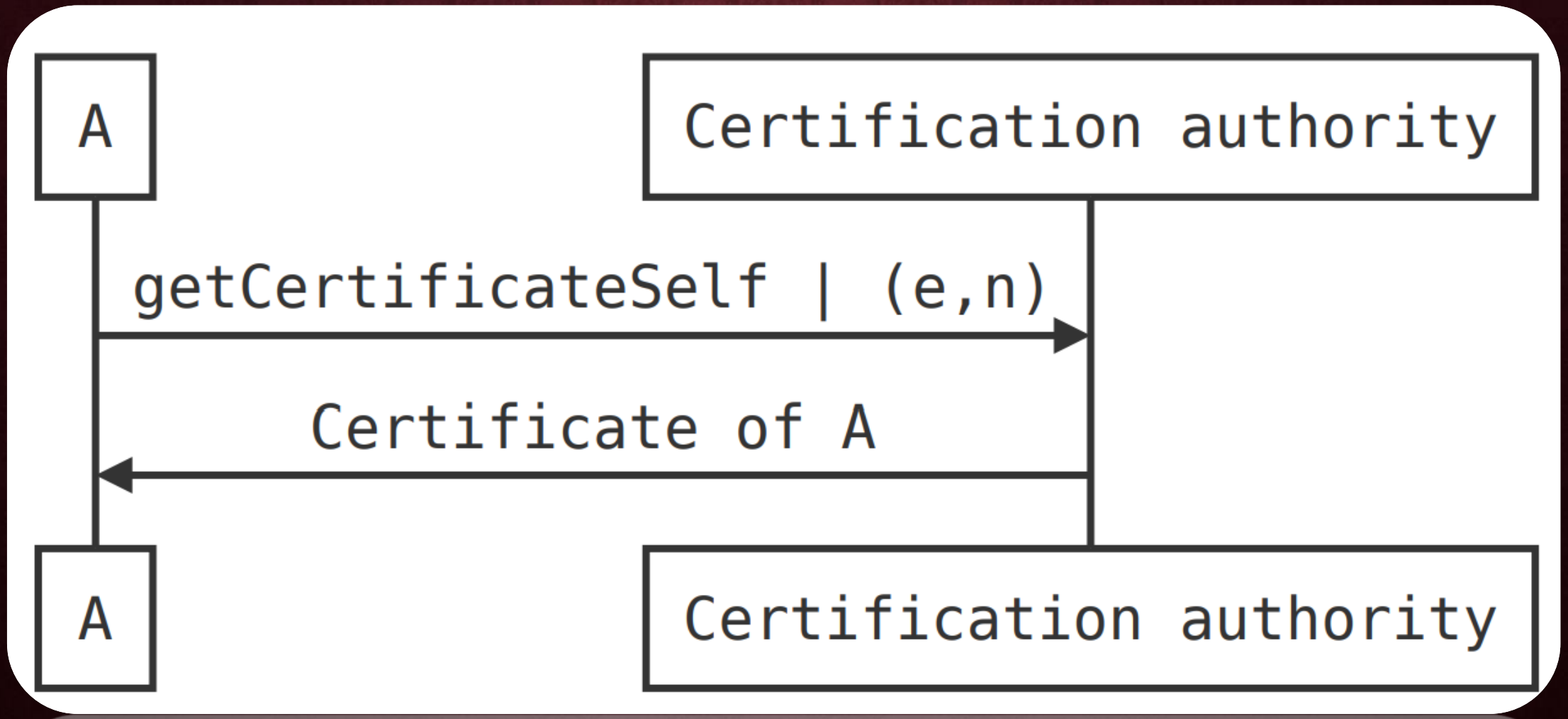
CERTIFICATE

Parameter	*Type*	*Description*
ID_A	int	Identification number of A
PU_A	$(e : int, n : int)$	Public Key of A
TIME	$unix - time$	Time at issuing of certificate
DURATION	$seconds$	How long is the certificate valid ?
ID_CA	int	Identification number of Certification Authority

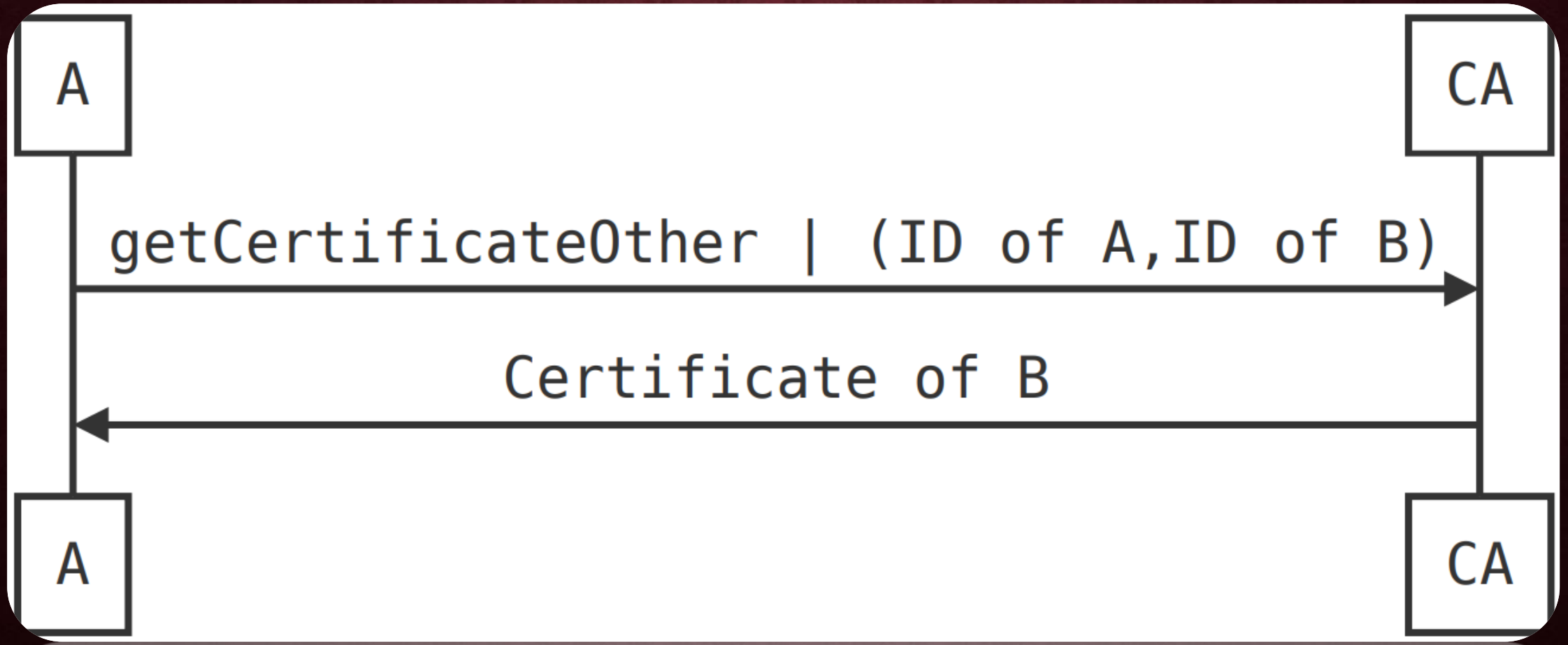
CERTIFICATION AUTHORITY

- Handles three kinds of queries:
  - getCertificateSelf
  - getCertificateOther
  - verifyCertificate

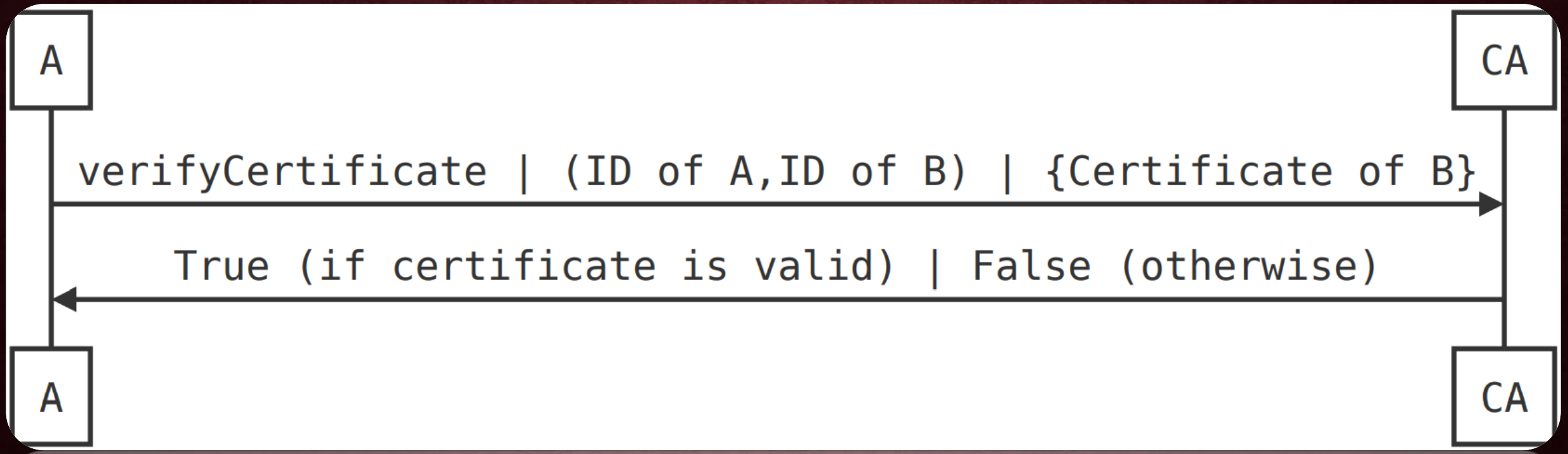
GET-CERTIFICATE-SELF



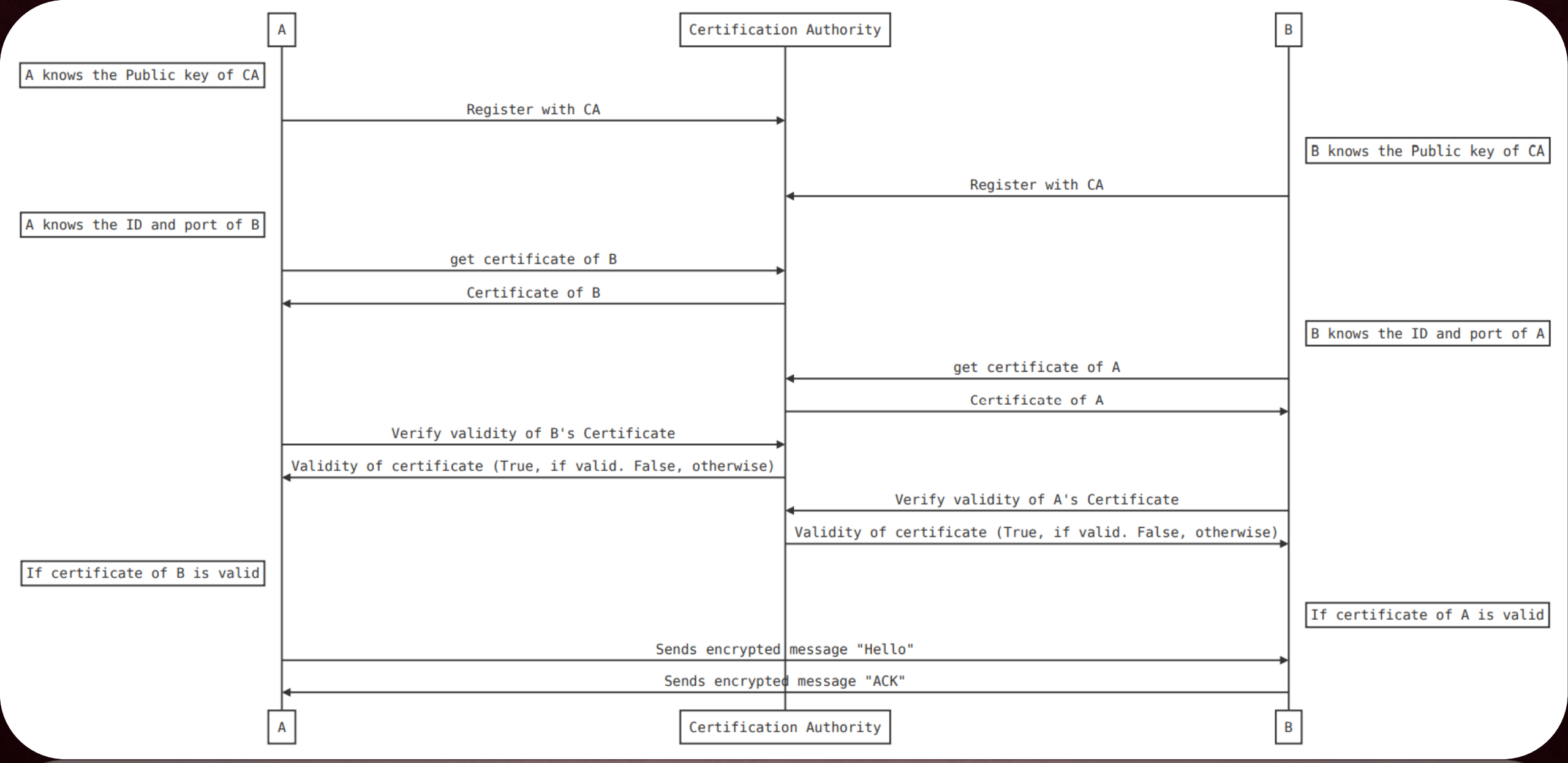
GET-CERTIFICATE-OTHER

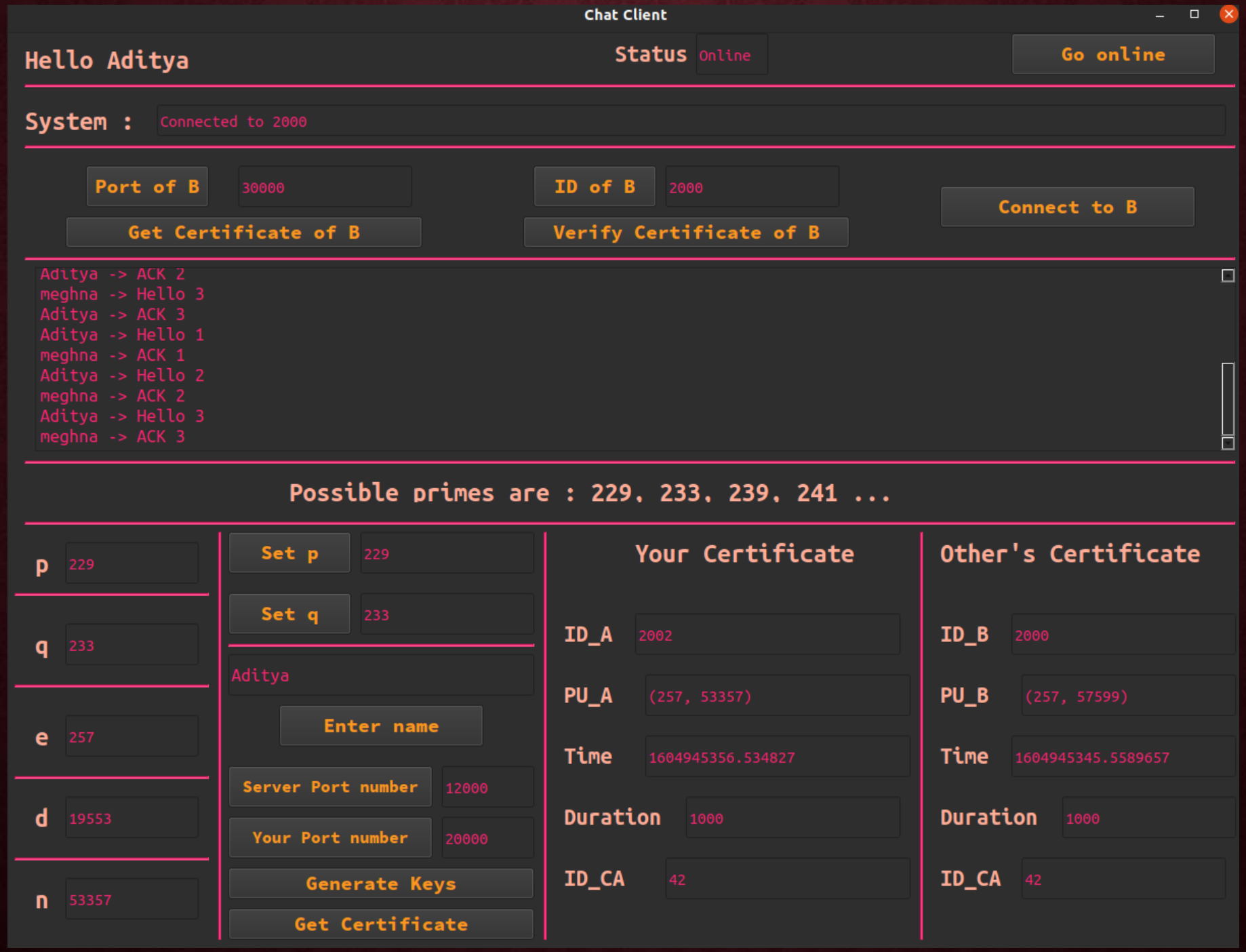


VERIFY-CERTIFICATE



APPLICATION FLOW


A
Certification Authority
B
A knows the Public key of CA
Register with CA
B knows the Public key of CA
Register with CA
A knows the ID and port of B
get certificate of B
Certificate of B
B knows the ID and port of A
get certificate of A
Certificate of A
Verify validity of B's Certificate
Validity of certificate (True, if valid. False, otherwise)
Verify validity of A's Certificate
Validity of certificate (True, if valid. False, otherwise)
If certificate of B is valid
If certificate of A is valid
Sends encrypted message "Hello"
Sends encrypted message "ACK"
A
Certification Authority
B


Chat Client
Hello Aditya
Status
Online
Go online
System :
Connected to 2000
Port of B
30000
ID of B
2000
Connect to B
Get Certificate of B
Verify Certificate of B
Aditya -> ACK 2
meghna -> Hello 3
Aditya -> ACK 3
Aditya -> Hello 1
meghna -> ACK 1
Aditya -> Hello 2
meghna -> ACK 2
Aditya -> Hello 3
meghna -> ACK 3
Possible primes are : 229, 233, 239, 241 ...
p
229
q
233
e
257
d
19553
n
53357
Set p
229
Set q
233
Aditya
Enter name
Server Port number
12000
Your Port number
20000
Generate Keys
Get Certificate
Your Certificate
ID_A
2002
PU_A
(257, 53357)
Time
1604945356.534827
Duration
1000
ID_CA
42
Other's Certificate
ID_B
2000
PU_B
(257, 57599)
Time
1604945345.5589657
Duration
1000
ID_CA
42