Public Key Certification Authority

Course	Assignment Number
$CS350$, Network Security	3

Roll No	Name
2018007	Aditya Singh Rathore
2018237	Jaspreet Saka

Encryption / Decryption

```
def encrypt(plainText:str, e:int, n:int)->str:
    """ Encrypts each character of plain text using RSA."""
    l = [chr(pow(ord(M),e,n)) for M in plainText];
    return ''.join(l);

def decrypt(cypherText:str, d:int, n:int)->str:
    """Decrypts each character of cypher text which was encrypted using
    encrypt(...)."""
    l = [chr(pow(ord(C),d,n)) for C in cypherText];
    return ''.join(l);
```

- We have used `RSA` encryption.
- We convert every object and literal into its string representation (in python 3).
- Further, we convert each character to their `unicode-16` numerical value.
- Above values are concatenated and sent over string .
- On the decryption side, each character is decrypted and result concatenated to get the result.

Vulnerabilities

- 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.

Reasons for such choice

- 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 becomes way too large to be handled correctly.
- 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.

Client-to-CA

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

Application

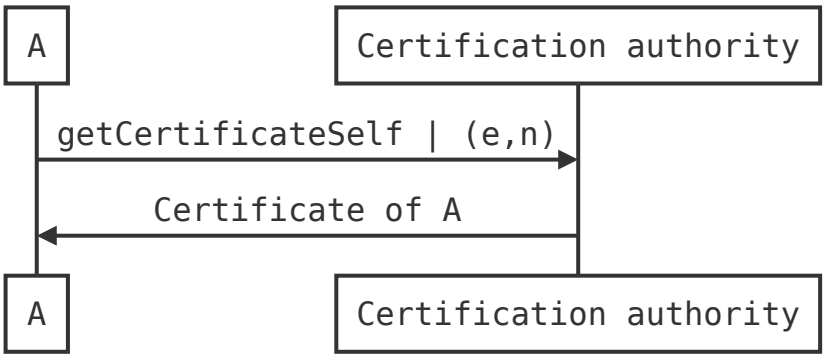
Certificate

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

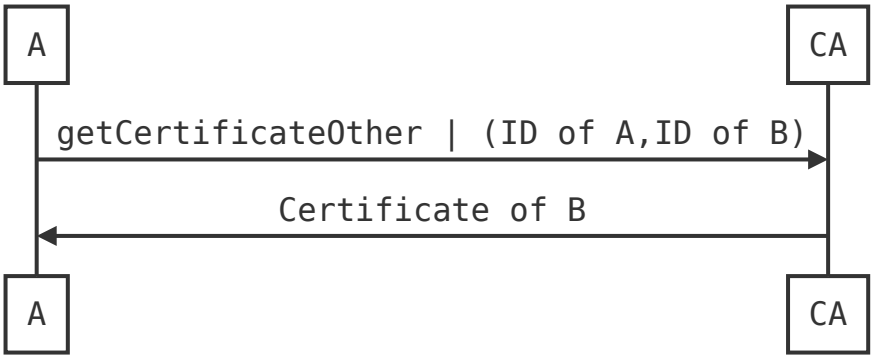
It handles three types of requests :

`getCertificateSelf`



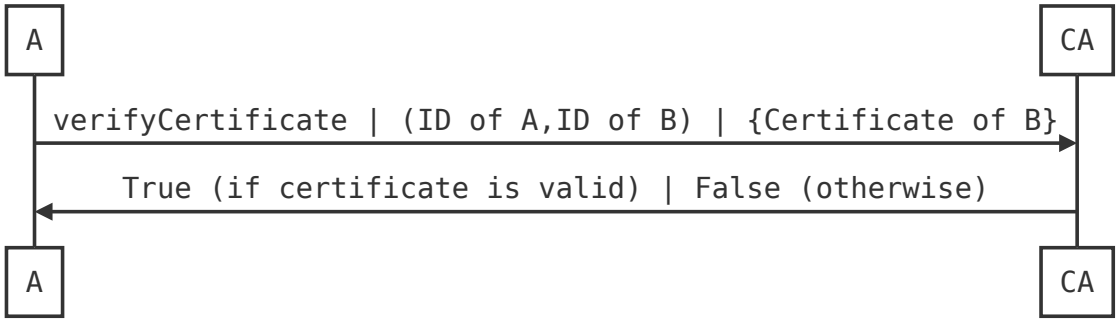
- With this request, client registers itself with Certification authority.
- Public key of A is sent with request.
- CA encrypts data with public key of A and sends back

`getCertificateOther`



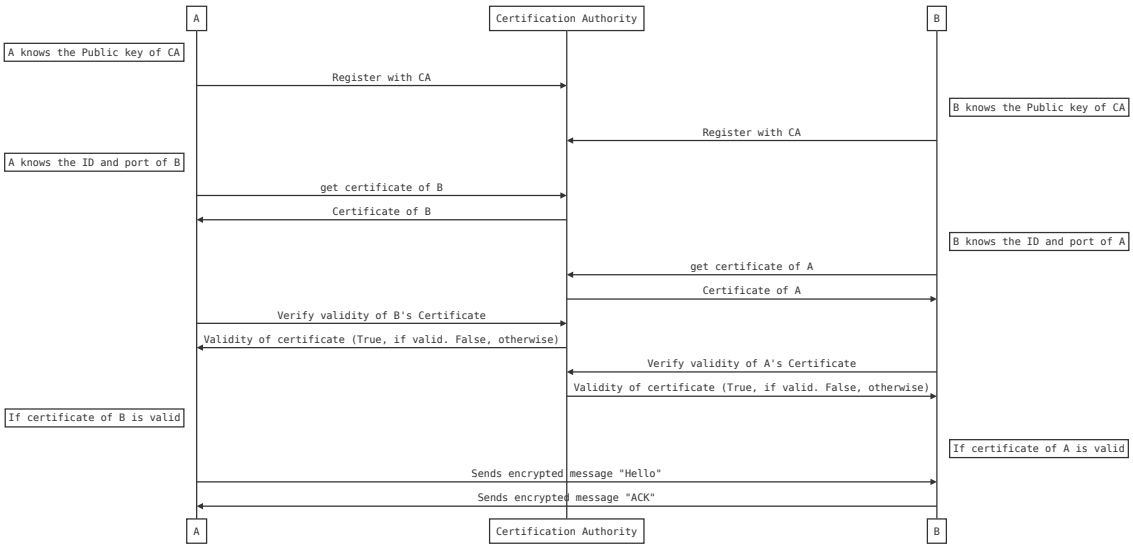
- With this request, client requests public certificate of B.
- It is assumed that B is already registered with CA.
- ID of B must be known to A.

`verifyCertificate`



- Used to verify certificate of B by A.
- B must be registered with CA.
- A must have ID and certificate of B.

Flow



- Public key of CA is known.
- A and B register themselves with CA.

- ID of B is known to A and vice-versa.
- A requests Public certificate of B from CA and B requests A's certificate from CA.
- A checks validity of B's certificate from CA and similarly B checks validity of A's certificate.
- After verifying, clients communicate with each other.
- **Encryption is described above.**