DEPARTMENT OF COMPUTER SCIENCE
IIT DELHI

Network and System Security

Assignment 3

2016MCS2681, 2016MCS2657

April 2017

1 Problem Statement

You are required to (a) build a CA, and (b) build clients that wish to send messages suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner.

2 Public Key Distribution

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:
- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

Any one can forge the Public Announcement(s). Some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A.

Stronger security for public-key distribution can be achieved by providing the distribution of public keys from the central directory. A central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. Each time a user A wants to communicate to user B, both A and B will take the public key of each other from KDC. Once they get the key, the keys are cached and updated periodically to absorb any change in the public keys.

But the public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering also.

3 Certificate Authority

In cryptography, a certificate authority or certification authority (CA) is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others (relying parties) to rely upon signatures or on assertions made about the private key that corresponds to the certified public key. A CA acts as a trusted third party i.e. trusted both by the subject (owner) of the certificate and by the party relying upon the certificate. The format of these certificates is specified by the X.509 standard.

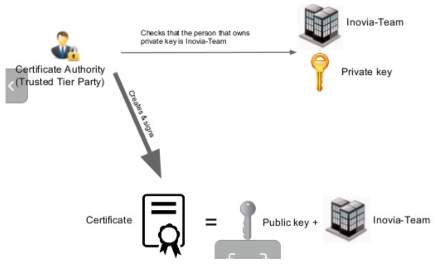


Figure 1: Certificate Authority

Digital certificates from the CA can be used by participants to exchange keys without contacting a public-key authority, in a way that as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted Certificate Authority. Certificate Authority is trusted by the user community. A

user can present his or her public key to the authority in a secure manner (through CSR) and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.
We can place the following requirements on this scheme:
1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.

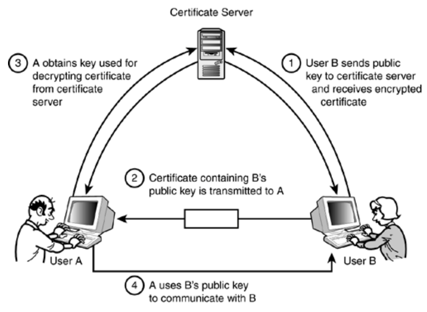

Certificate Server
3 A obtains key used for decrypting certificate from certificate server
1 User B sends public key to certificate server and receives encrypted certificate
2 Certificate containing B's public key is transmitted to A
User A
User B
4 A uses B's public key to communicate with B

Figure 2: Client Communication

A certificate scheme is illustrated in Figure2. Each participant applies to the certificate authority, supplying a public key and requesting a certificate. Application must be in some form of secure authenticated communication.
For participant A, the authority provides a certificate of the form

$$CA = E(PR_{auth}, [T||ID_A||PU_a])$$

where PRauth is the private key used by the authority and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

$$D(PU_{auth}, CA) = D(PU_{auth}, E(PR_{auth}, [T||ID_A||PU_a])) = (T||ID_A||PU_a)$$

The recipient uses the authority's public key, PUauth, to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements IDA and PUa provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate.

4 Implementation

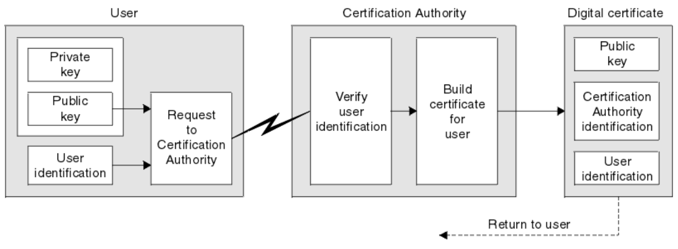
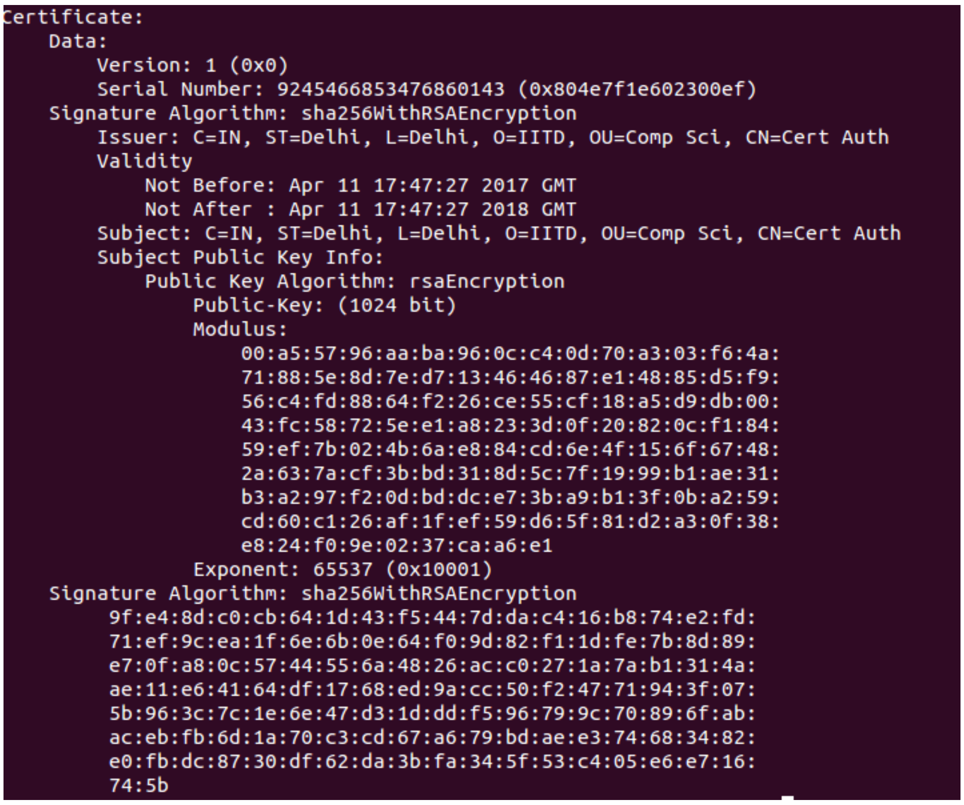


Figure 3: CA Implementation

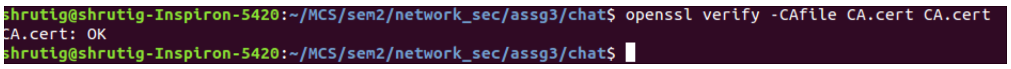
A local CA like ours will have their own certificate (a certificate to distribute the public key of CA)as a self signed certificate. By self signed we mean that the certificate content i.e. public key of CA and CA identity is digitally signed by private key of CA itself. This is done so that anyone can verify that CA public key certificate has not been tempered. The CA keeps his private key to himself only and is not available to outside world.



```
ertificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 9245466853476860143 (0x804e7f1e602300ef)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IN, ST=Delhi, L=Delhi, O=IITD, OU=Comp Sci, CN=Cert Auth
        Validity
            Not Before: Apr 11 17:47:27 2017 GMT
            Not After : Apr 11 17:47:27 2018 GMT
        Subject: C=IN, ST=Delhi, L=Delhi, O=IITD, OU=Comp Sci, CN=Cert Auth
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:a5:57:96:aa:ba:96:0c:c4:0d:70:a3:03:f6:4a:
                    71:88:5e:8d:7e:d7:13:46:46:87:e1:48:85:d5:f9:
                    56:c4:fd:88:64:f2:26:ce:55:cf:18:a5:d9:db:00:
                    43:fc:58:72:5e:e1:a8:23:3d:0f:20:82:0c:f1:84:
                    59:ef:7b:02:4b:6a:e8:84:cd:6e:4f:15:6f:67:48:
                    2a:63:7a:cf:3b:bd:31:8d:5c:7f:19:99:b1:ae:31:
                    b3:a2:97:f2:0d:bd:dc:e7:3b:a9:b1:3f:0b:a2:59:
                    cd:60:c1:26:af:1f:ef:59:d6:5f:81:d2:a3:0f:38:
                    e8:24:f0:9e:02:37:ca:a6:e1
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         9f:e4:8d:c0:cb:64:1d:43:f5:44:7d:da:c4:16:b8:74:e2:fd:
         71:ef:9c:ea:1f:6e:6b:0e:64:f0:9d:82:f1:1d:fe:7b:8d:89:
         e7:0f:a8:0c:57:44:55:6a:48:26:ac:c0:27:1a:7a:b1:31:4a:
         ae:11:e6:41:64:df:17:68:ed:9a:cc:50:f2:47:71:94:3f:07:
         5b:96:3c:7c:1e:6e:47:d3:1d:dd:f5:96:79:9c:70:89:6f:ab:
         ac:eb:fb:6d:1a:70:c3:cd:67:a6:79:bd:ae:e3:74:68:34:82:
         e0:fb:dc:87:30:df:62:da:3b:fa:34:5f:53:c4:05:e6:e7:16:
         74:5b
```

Figure 4: CA self signed certificate

Figure4 shows the self signed certificate of the CA whose corresponding private key is with the CA only. We can see the subject info to check that its the certificate for CA.

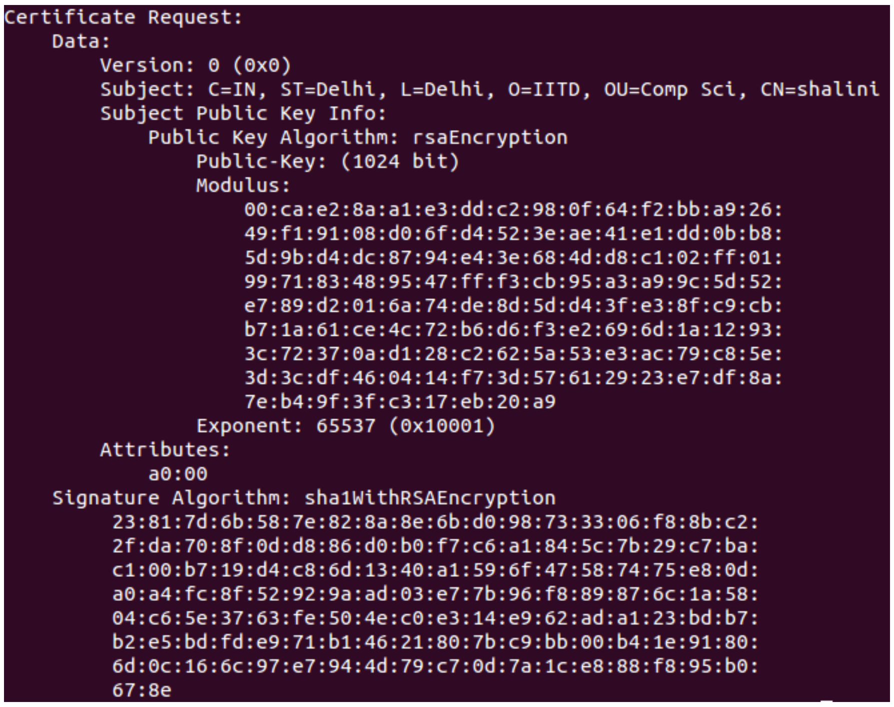


```
hrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/chat$ openssl verify -CAfile CA.cert CA.cert
A.cert: OK
hrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/chat$
```

Figure 5: CA Certificate verification

Figure5 shows that CA.cert is the CAfile to verify the CA.cert as its a self signed certificate. Thus, the certificate is verifiable at the client end. The OK message verifies the certificate.

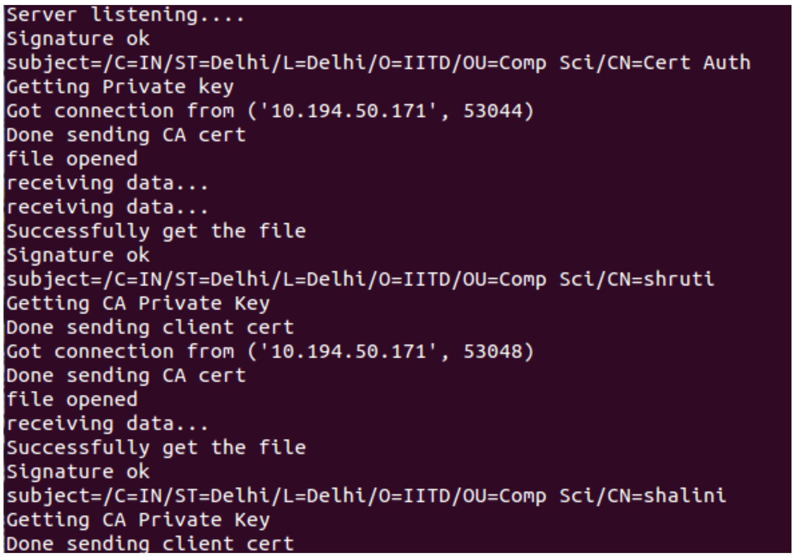
For any user A to chat or exchange messages with user B both of them have to have public key certificates from CA and share these among themselves to securely chat or exchange message. Both users do it the same way.
For each user you have to provide an identification name as a raw input. At the client side, the user generates a key pair, keeps the private key with itself in a .key file and creates a certificate signing request to be sent to CA to get a CA signed certificate for his public key.



```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=IN, ST=Delhi, L=Delhi, O=IITD, OU=Comp Sci, CN=shalini
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:ca:e2:8a:a1:e3:dd:c2:98:0f:64:f2:bb:a9:26:
                    49:f1:91:08:d0:6f:d4:52:3e:ae:41:e1:dd:0b:b8:
                    5d:9b:d4:dc:87:94:e4:3e:68:4d:d8:c1:02:ff:01:
                    99:71:83:48:95:47:ff:f3:cb:95:a3:a9:9c:5d:52:
                    e7:89:d2:01:6a:74:de:8d:5d:d4:3f:e3:8f:c9:cb:
                    b7:1a:61:ce:4c:72:b6:d6:f3:e2:69:6d:1a:12:93:
                    3c:72:37:0a:d1:28:c2:62:5a:53:e3:ac:79:c8:5e:
                    3d:3c:df:46:04:14:f7:3d:57:61:29:23:e7:df:8a:
                    7e:b4:9f:3f:c3:17:eb:20:a9
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha1WithRSAEncryption
         23:81:7d:6b:58:7e:82:8a:8e:6b:d0:98:73:33:06:f8:8b:c2:
         2f:da:70:8f:0d:d8:86:d0:b0:f7:c6:a1:84:5c:7b:29:c7:ba:
         c1:00:b7:19:d4:c8:6d:13:40:a1:59:6f:47:58:74:75:e8:0d:
         a0:a4:fc:8f:52:92:9a:ad:03:e7:7b:96:f8:89:87:6c:1a:58:
         04:c6:5e:37:63:fe:50:4e:c0:e3:14:e9:62:ad:a1:23:bd:b7:
         b2:e5:bd:fd:e9:71:b1:46:21:80:7b:c9:bb:00:b4:1e:91:80:
         6d:0c:16:6c:97:e7:94:4d:79:c7:0d:7a:1c:e8:88:f8:95:b0:
         67:8e
```

Figure 6: Certificate Signing request from client

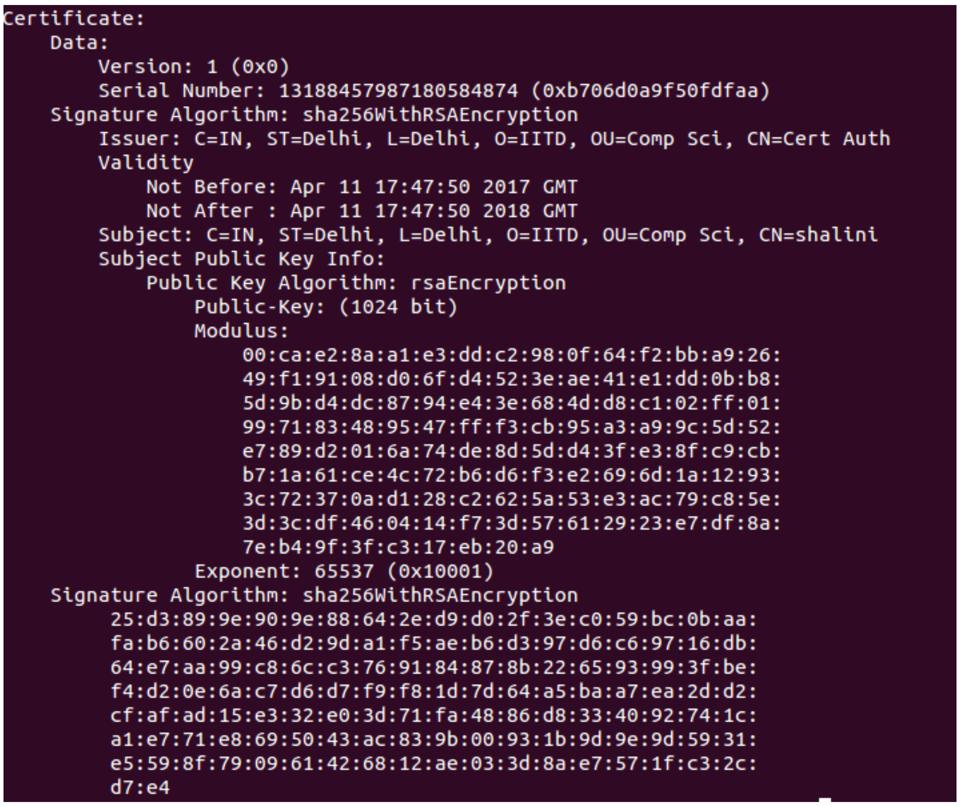
Figure6 shows the certificate signing request from client to CA. CSR contains all the information required for public key certificate generation by CA. The CSR is sent to CA encrypted using CA public key.



```
Server listening....
Signature ok
subject=/C=IN/ST=Delhi/L=Delhi/O=IITD/OU=Comp Sci/CN=Cert Auth
Getting Private key
Got connection from ('10.194.50.171', 53044)
Done sending CA cert
file opened
receiving data...
receiving data...
Successfully get the file
Signature ok
subject=/C=IN/ST=Delhi/L=Delhi/O=IITD/OU=Comp Sci/CN=shruti
Getting CA Private Key
Done sending client cert
Got connection from ('10.194.50.171', 53048)
Done sending CA cert
file opened
receiving data...
Successfully get the file
Signature ok
subject=/C=IN/ST=Delhi/L=Delhi/O=IITD/OU=Comp Sci/CN=shalini
Getting CA Private Key
Done sending client cert
```

Figure 7: Server verifying CSR and creating Certificates

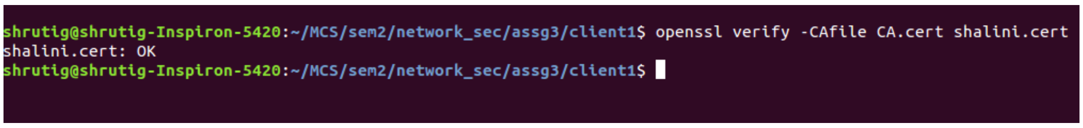
Figure 7 shows that CA after receiving the CSR verifies that it has not been tampered with, by verifying the signature in CSR after decrypting it using its own private key and then, create a public key certificate corresponding to decrypted CSR from client.



```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 13188457987180584874 (0xb706d0a9f50fdfaa)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IN, ST=Delhi, L=Delhi, O=IITD, OU=Comp Sci, CN=Cert Auth
        Validity
            Not Before: Apr 11 17:47:50 2017 GMT
            Not After : Apr 11 17:47:50 2018 GMT
        Subject: C=IN, ST=Delhi, L=Delhi, O=IITD, OU=Comp Sci, CN=shalini
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:ca:e2:8a:a1:e3:dd:c2:98:0f:64:f2:bb:a9:26:
                    49:f1:91:08:d0:6f:d4:52:3e:ae:41:e1:dd:0b:b8:
                    5d:9b:d4:dc:87:94:e4:3e:68:4d:d8:c1:02:ff:01:
                    99:71:83:48:95:47:ff:f3:cb:95:a3:a9:9c:5d:52:
                    e7:89:d2:01:6a:74:de:8d:5d:d4:3f:e3:8f:c9:cb:
                    b7:1a:61:ce:4c:72:b6:d6:f3:e2:69:6d:1a:12:93:
                    3c:72:37:0a:d1:28:c2:62:5a:53:e3:ac:79:c8:5e:
                    3d:3c:df:46:04:14:f7:3d:57:61:29:23:e7:df:8a:
                    7e:b4:9f:3f:c3:17:eb:20:a9
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         25:d3:89:9e:90:9e:88:64:2e:d9:d0:2f:3e:c0:59:bc:0b:aa:
         fa:b6:60:2a:46:d2:9d:a1:f5:ae:b6:d3:97:d6:c6:97:16:db:
         64:e7:aa:99:c8:6c:c3:76:91:84:87:8b:22:65:93:99:3f:be:
         f4:d2:0e:6a:c7:d6:d7:f9:f8:1d:7d:64:a5:ba:a7:ea:2d:d2:
         cf:af:ad:15:e3:32:e0:3d:71:fa:48:86:d8:33:40:92:74:1c:
         a1:e7:71:e8:69:50:43:ac:83:9b:00:93:1b:9d:9e:9d:59:31:
         e5:59:8f:79:09:61:42:68:12:ae:03:3d:8a:e7:57:1f:c3:2c:
         d7:e4
```

Figure 8: CA generated certificate

Figure 8 shows the certificate created by CA corresponding to CSR in figure 6. We can see that subject and public key info is same. This certificate is encrypted using client public key and then,sent to client so that he can share it (after decrypting it using its private key) with any one to whom he want to chat or exchange messages securely.

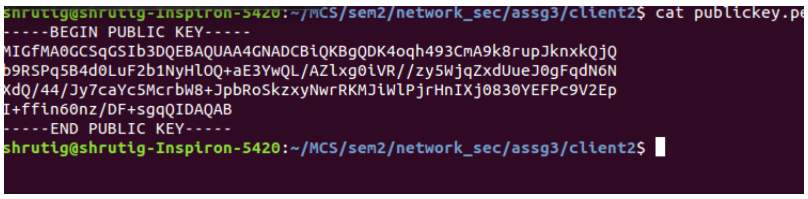


```
shrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/client1$ openssl verify -CAfile CA.cert shalini.cert
shalini.cert: OK
shrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/client1$ 
```

Figure 9: Client Certificate verification

Figure 9 shows that decrypted certificate can be verified at client side using CA.cert (public key certificate of CA available to all users)as CAfile.

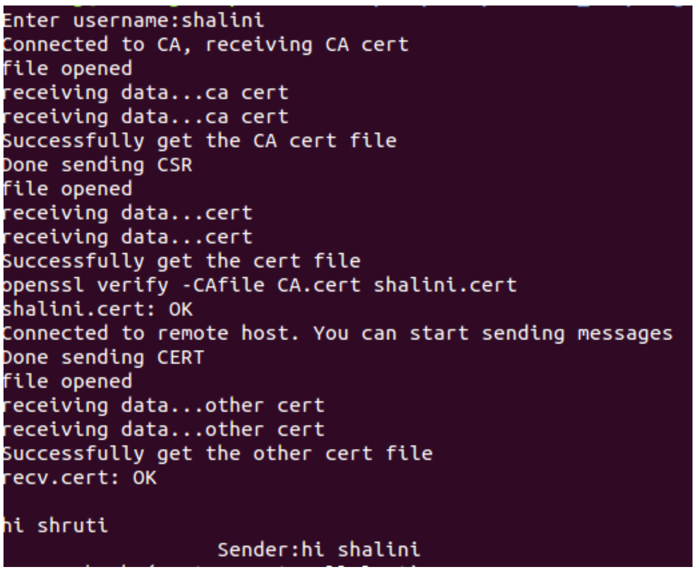
Similarly, another user will get his certificate from CA. In order to initiate chat between the users, the users have to exchange their certificate among themselves. Both users will verify the certificate received from other using CA.cert. Once the certificate is verified, user will extract the public key from the certificate.



```
shrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/client2$ cat publickey.pe
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDK4oqh493CmA9k8rupJknxkQjQ
b9RSPq5B4d0LuF2b1NyHlOQ+aE3YwQL/AZlxg0iVR//zy5WjqZxdUueJ0gFqdN6N
XdQ/44/Jy7caYc5McrbW8+JpbRoSkzxyNwrRKMJiWlPjrHnIXj0830YEFPc9V2Ep
I+ffin60nz/DF+sgqQIDAQAB
-----END PUBLIC KEY-----
shrutig@shrutig-Inspiron-5420:~/MCS/sem2/network_sec/assg3/client2$ 
```

Figure 10: Public key from certificate

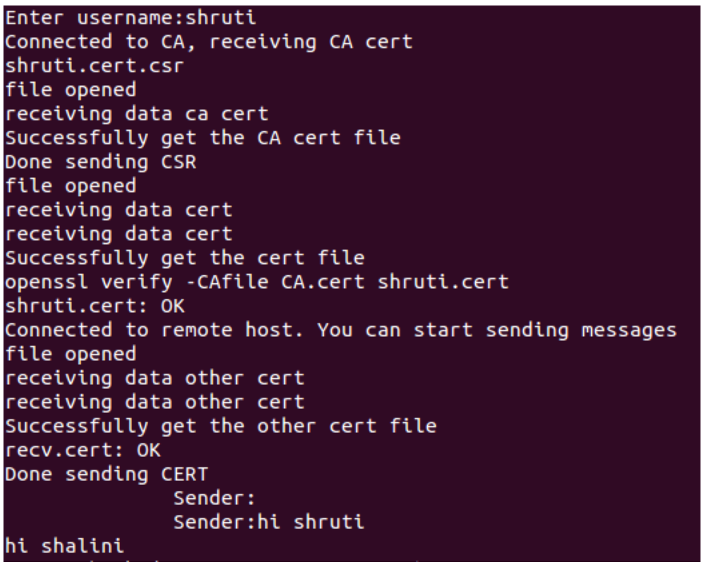
Figure 10 shows the public key extracted corresponding to certificate in figure8. Now, the public key of other user is used to encrypt messages for that user, so that only he can read the message correctly through decrypting it using his private key



```
Enter username:shalini
Connected to CA, receiving CA cert
file opened
receiving data...ca cert
receiving data...ca cert
Successfully get the CA cert file
Done sending CSR
file opened
receiving data...cert
receiving data...cert
Successfully get the cert file
openssl verify -CAfile CA.cert shalini.cert
shalini.cert: OK
Connected to remote host. You can start sending messages
Done sending CERT
file opened
receiving data...other cert
receiving data...other cert
Successfully get the other cert file
recv.cert: OK

hi shruti
                  Sender:hi shalini
```

Figure 11: Client 1



```
Enter username:shruti
Connected to CA, receiving CA cert
shruti.cert.csr
file opened
receiving data ca cert
Successfully get the CA cert file
Done sending CSR
file opened
receiving data cert
receiving data cert
Successfully get the cert file
openssl verify -CAfile CA.cert shruti.cert
shruti.cert: OK
Connected to remote host. You can start sending messages
file opened
receiving data other cert
receiving data other cert
Successfully get the other cert file
recv.cert: OK
Done sending CERT
              Sender:
              Sender:hi shruti
hi shalini
```

Figure 12: Client 2

Figure 11 and 12 shows that both the clients are able to successfully communicate, when the public keys are shared as certificate from CA. We can see that message typed at one end appear on the other end (decrypted successfully) as a message from sender.