CS6903 - Network Security
Assignment 7: Secure Chat using OpenSSL and MITM attacks

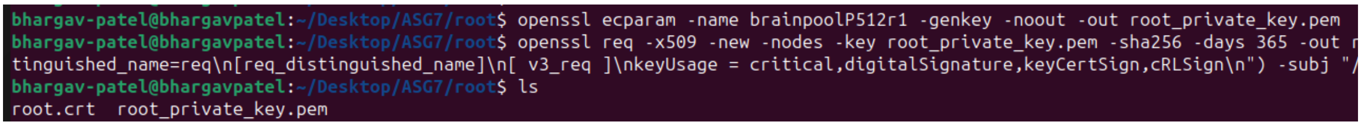
Group Details:

CS23MTECH14009 - Raj Popat	**ALICE**
CS23MTECH14015 - Sreyash Mohanty	**BOB**
CS23MTECH11026 - Bhargav Patel	**RootCA & IntermediateCA**

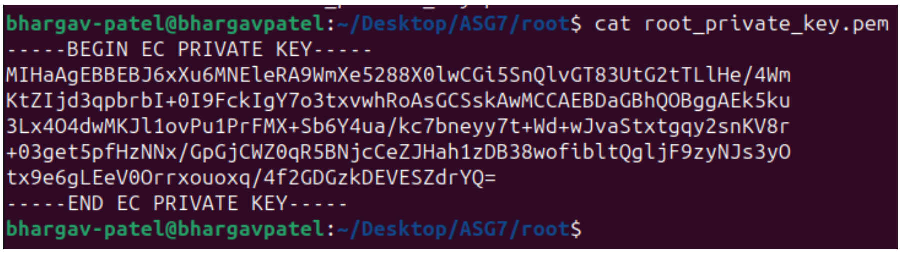
TASK1:

1. Key Generation:

- **RootCA:** Using brainpoolP512r1 (one of the 512-bit elliptic curves supported in my system) to generate the private key. Then, X.509 self-signed certificate is produced using a 512-bit ECC private key (root_private_key.pem) . The command used is shown in the image with all those required parameters. ***As the root generates a self-signed certificate (root.crt) so not required to generate CSR go for further steps (steps which intermediate & end entity follows).***

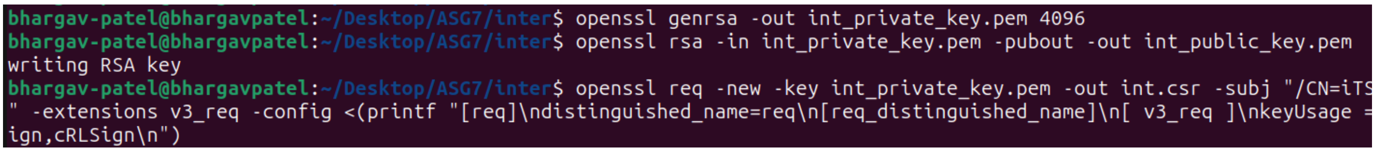


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ openssl ecparam -name brainpoolP512r1 -genkey -noout -out root_private_key.pem
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ openssl req -x509 -new -nodes -key root_private_key.pem -sha256 -days 365 -out r
tinguished_name=req\n[req_distinguished_name]\n[ v3_req ]\nkeyUsage = critical,digitalSignature,keyCertSign,cRLSign\n") -subj "/
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ ls
root.crt  root_private_key.pem
```

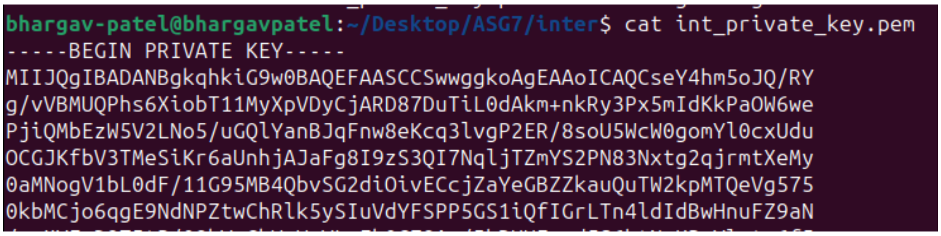


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ cat root_private_key.pem
-----BEGIN EC PRIVATE KEY-----
MIHaAgEBBEBJ6xXu6MNEleRA9WmXe5288X0lwCGi5SnQlvGT83UtG2tTLlHe/4Wm
KtZIjd3qpbrbI+0I9FckIgY7o3txvwhRoAsGCSskAwMCCAEBDaGBhQOBggAEk5ku
3Lx4O4dwMKJl1ovPu1PrFMX+Sb6Y4ua/kc7bneyy7t+Wd+wJvaStxtgqy2snKV8r
+03get5pfHzNNx/GpGjCWZ0qR5BNjcCeZJHah1zDB38wofibltQgljF9zyNJs3yO
tx9e6gLEeV0Orrxouoxq/4f2GDGzkDEVESZdrYQ=
-----END EC PRIVATE KEY-----
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$
```

- **IntermediateCA:** As asked, we have used the RSA algorithm to generate a private key of 4096-bit (int_private_key.pem) and then the corresponding generated public key (int_public_key.pem) using the following commands.

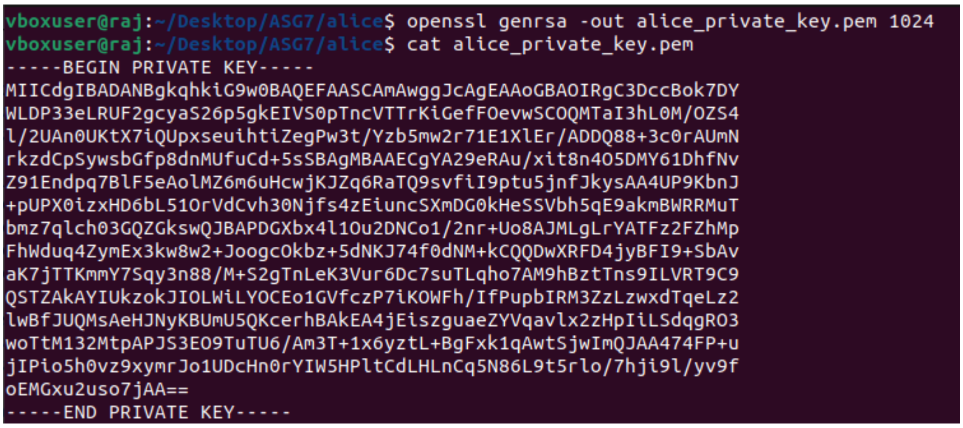


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl genrsa -out int_private_key.pem 4096
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl rsa -in int_private_key.pem -pubout -out int_public_key.pem
writing RSA key
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -new -key int_private_key.pem -out int.csr -subj "/CN=iTS
" -extensions v3_req -config <(printf "[req]\ndistinguished_name=req\n[req_distinguished_name]\n[ v3_req ]\nkeyUsage =
ign,cRLSign\n")
```



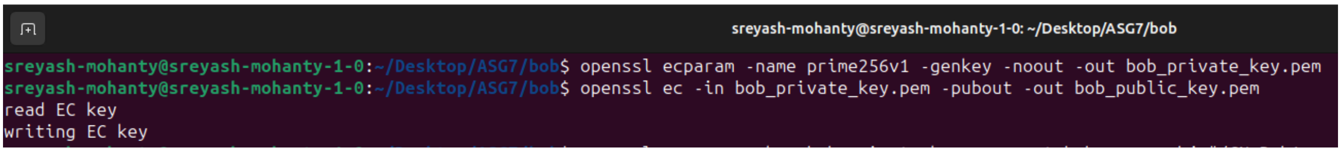
```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ cat int_private_key.pem
-----BEGIN PRIVATE KEY-----
MIIJQgIBADANBgkqhkiG9w0BAQEFAASCCSwwggkoAgEAAoICAQCseY4hm5oJQ/RY
g/vVBMUQPhs6XiobT11MyXpVDyCjARD87DuTiL0dAkm+nkRy3Px5mIdKkPaOW6we
PjiQMbEzW5V2LNo5/uGQlYanBJqFnw8eKcq3lvgP2ER/8soU5WcW0gomYl0cxUdu
OCGJKfbV3TMeSiKr6aUnhjAJaFg8I9zS3QI7NqljTZmYS2PN83Nxtg2qjrmtXeMy
0aMNogV1bL0dF/11G95MB4QbvSG2diOivECcjZaYeGBZZkauQuTW2kpMTQeVg575
0kbMCjo6qgE9NdNPZtwChRlk5ySIuVdYFSPP5GS1iQfIGrLTn4ldIdBwHnuFZ9aN
```

- **Alice:** Generating 1024-bit RSA private key (alice_private_key.pem) using the following command.

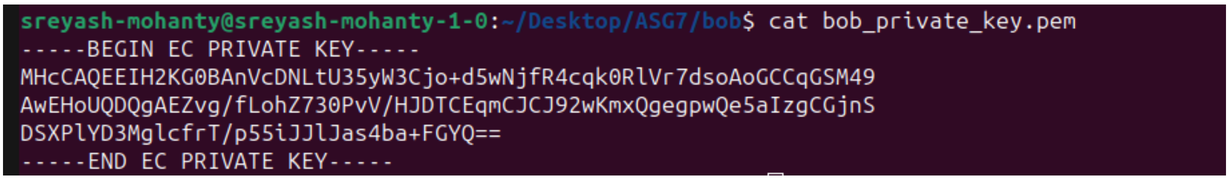


```
vboxuser@raj:~/Desktop/ASG7/alice$ openssl genrsa -out alice_private_key.pem 1024
vboxuser@raj:~/Desktop/ASG7/alice$ cat alice_private_key.pem
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAOIRgC3DccBok7DY
WLDP33eLRUF2gcyaS26p5gkEIVS0pTncVTTrKiGefFOevwSCOQMTaI3hL0M/OZS4
l/2UAn0UKtX7iQUpxseuihtiZegPw3t/Yzb5mw2r71E1XlEr/ADDQ88+3c0rAUmN
rkzdCpSywsbGfp8dnMUfuCd+5sSBAgMBAAECgYA29eRAu/xit8n4O5DMY61DhfNv
Z91Endpq7BlF5eAolMZ6m6uHcwjKJZq6RaTQ9svfiI9ptu5jnfJkysAA4UP9KbnJ
+pUPX0izxHD6bL51OrVdCvh30Njfs4zEiuncSXmDG0kHeSSVbh5qE9akmBWRRMuT
bmz7qlch03GQZGkswQJBAPDGXbx4l1Ou2DNCo1/2nr+Uo8AJMLgLrYATFz2FZhMp
FhWduq4ZymEx3kw8w2+JoogcOkbz+5dNKJ74f0dNM+kCQQDwXRFD4jyBFI9+SbAv
aK7jTTKmmY7Sqy3n88/M+S2gTnLeK3Vur6Dc7suTLqho7AM9hBztTns9ILVRT9C9
QSTZAkAYIUkzokJIOLWiLYOCEo1GVfczP7iKOWFh/IfPupbIRM3ZzLzwxdTqeLz2
lwBfJUQMsAeHJNyKBUmU5QKcerhBAkEA4jEiszguaeZYVqavlx2zHpIiLSdqgRO3
woTtM132MtpAPJS3EO9TuTU6/Am3T+1x6yztL+BgFxk1qAwtSjwImQJAA474FP+u
jIPio5h0vz9xymrJo1UDcHn0rYIW5HPltCdLHLnCq5N86L9t5rlo/7hji9l/yv9f
oEMGxu2uso7jAA==
-----END PRIVATE KEY-----
```

- **Bob:** Generating 256-bit ECC private key (bob_private_key.pem) using prime256v1 (one among other 256-bit elliptic curves supported in my system). Then, the corresponding public key is generated (bob_public_key.pem) using the following commands.



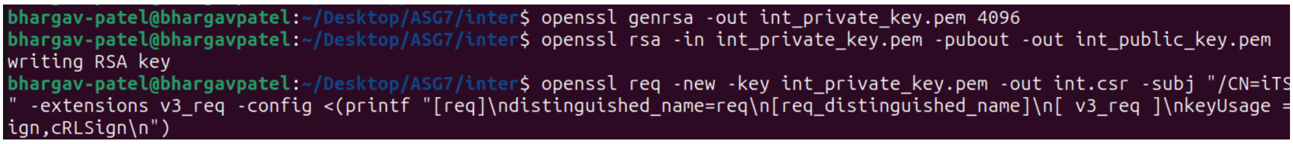
```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl ecparam -name prime256v1 -genkey -noout -out bob_private_key.pem
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl ec -in bob_private_key.pem -pubout -out bob_public_key.pem
read EC key
writing EC key
```



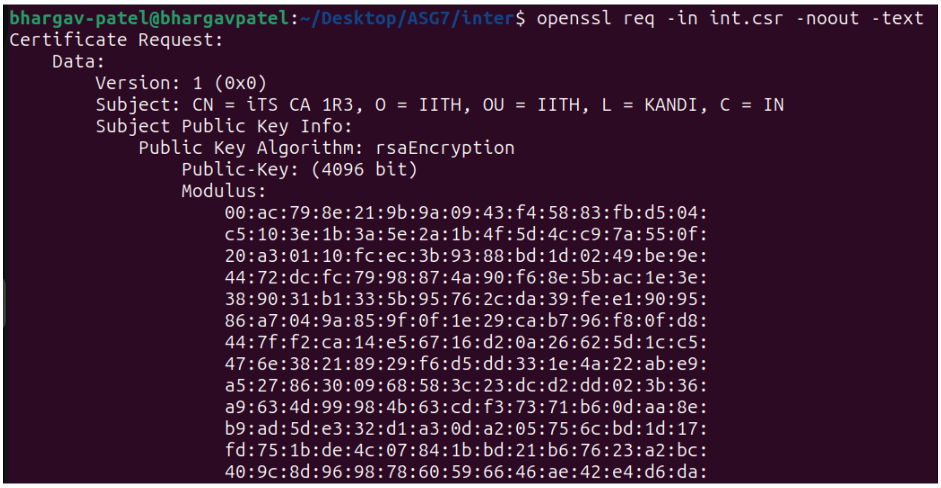
```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ cat bob_private_key.pem
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIH2KG0BAnVcDNLtU35yW3Cjo+d5wNjfR4cqk0RlVr7dsoAoGCCqGSM49
AwEHoUQDQgAEZvg/fLohZ730PvV/HJDTCEqmCJCJ92wKmxQgegpwQe5aIzgCGjnS
DSXPlYD3MglcfrT/p55iJJlJas4ba+FGYQ==
-----END EC PRIVATE KEY-----
```

2. Certificate Signing Request (CSR):

- **IntermediateCA:** using the intermediate's private key (int_private_key.pem), a certificate signing request (int.csr) is generated with the required parameters (as asked in TAS1) using the following commands.

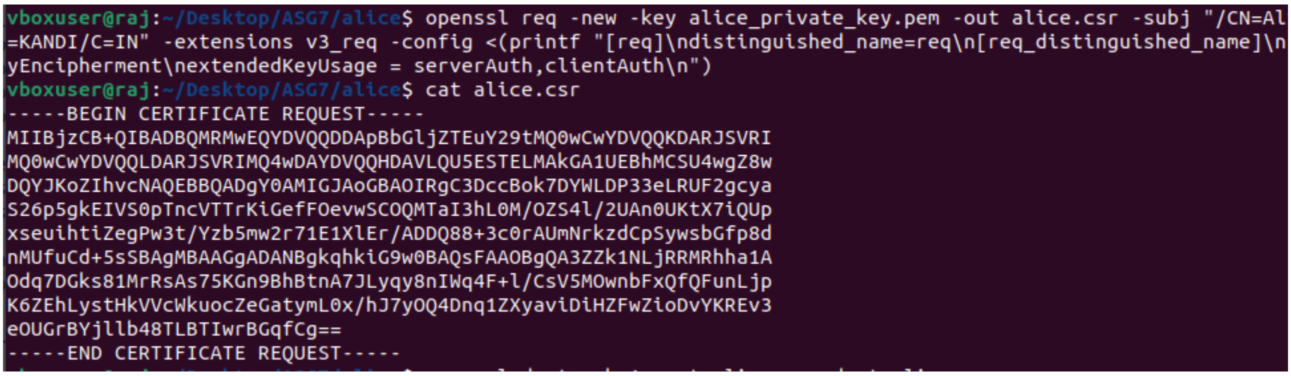


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl genrsa -out int_private_key.pem 4096
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl rsa -in int_private_key.pem -pubout -out int_public_key.pem
writing RSA key
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -new -key int_private_key.pem -out int.csr -subj "/CN=iTS
" -extensions v3_req -config <(printf "[req]\ndistinguished_name=req\n[req_distinguished_name]\n[ v3_req ]\nkeyUsage =
ign,cRLSign\n")
```



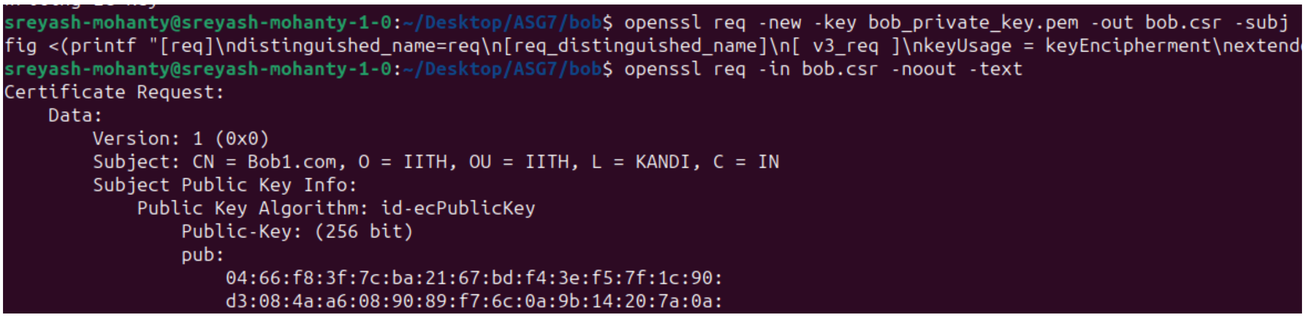
```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -in int.csr -noout -text
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = iTS CA 1R3, O = IITH, OU = IITH, L = KANDI, C = IN
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:ac:79:8e:21:9b:9a:09:43:f4:58:83:fb:d5:04:
                    c5:10:3e:1b:3a:5e:2a:1b:4f:5d:4c:c9:7a:55:0f:
                    20:a3:01:10:fc:ec:3b:93:88:bd:1d:02:49:be:9e:
                    44:72:dc:fc:79:98:87:4a:90:f6:8e:5b:ac:1e:3e:
                    38:90:31:b1:33:5b:95:76:2c:da:39:fe:e1:90:95:
                    86:a7:04:9a:85:9f:0f:1e:29:ca:b7:96:f8:0f:d8:
                    44:7f:f2:ca:14:e5:67:16:d2:0a:26:62:5d:1c:c5:
                    47:6e:38:21:89:29:f6:d5:dd:33:1e:4a:22:ab:e9:
                    a5:27:86:30:09:68:58:3c:23:dc:d2:dd:02:3b:36:
                    a9:63:4d:99:98:4b:63:cd:f3:73:71:b6:0d:aa:8e:
                    b9:ad:5d:e3:32:d1:a3:0d:a2:05:75:6c:bd:1d:17:
                    fd:75:1b:de:4c:07:84:1b:bd:21:b6:76:23:a2:bc:
                    40:9c:8d:96:98:78:60:59:66:46:ae:42:e4:d6:da:
```

- **Alice:** Generating the certificate signing request (alice.csr) using Alice's private key (alice_private_key.pem). Also, the required parameters, like CN, OU, O, etc., are given in the same commands as shown below.

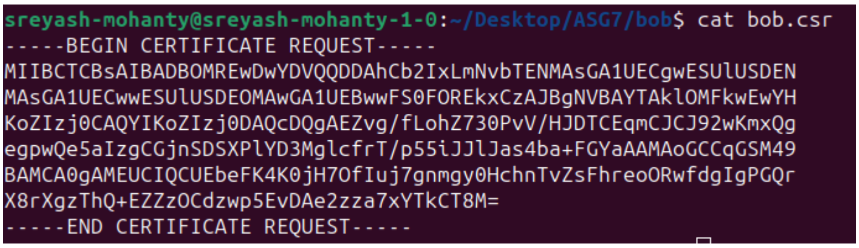


```
vboxuser@raj:~/Desktop/ASG7/alice$ openssl req -new -key alice_private_key.pem -out alice.csr -subj "/CN=Al
=KANDI/C=IN" -extensions v3_req -config <(printf "[req]\ndistinguished_name=req\n[req_distinguished_name]\n
yEncipherment\nextendedKeyUsage = serverAuth,clientAuth\n")
vboxuser@raj:~/Desktop/ASG7/alice$ cat alice.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIBjzCB+QIBADBQMRMwEQYDVQQDDApBbGljZTEuY29tMQ0wCwYDVQQKDARJSVRI
MQ0wCwYDVQQLDARJSVRIMQ4wDAYDVQQHDAVLQU5ESTELMAkGA1UEBhMCSU4wgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOIRgC3DccBok7DYWLDP33eLRUF2gcya
S26p5gkEIVS0pTncVTTrKiGefFOevwSCOQMTaI3hL0M/OZS4l/2UAn0UKtX7iQUp
xseuihtiZegPw3t/Yzb5mw2r71E1XlEr/ADDQ88+3c0rAUmNrkzdCpSywsbGfp8d
nMUfuCd+5sSBAgMBAAGgADANBgkqhkiG9w0BAQsFAAOBgQA3ZZk1NLjRRMRhha1A
Odq7DGks81MrRsAs75KGn9BhBtnA7JLyqy8nIWq4F+l/CsV5MOwnbFxQfQFunLjp
K6ZEhLystHkVVcWkuocZeGatymL0x/hJ7yOQ4Dnq1ZXyaviDiHZFwZioDvYKREv3
eOUGrBYjllb48TLBTIwrBGqfCg==
-----END CERTIFICATE REQUEST-----
```

- **Bob:** Created Bob's certificate signing request (bob.csr) using Bob's private (bob_private_key.pem) along with the required parameters (CN, OU, O, extension, etc.). The following commands are used.



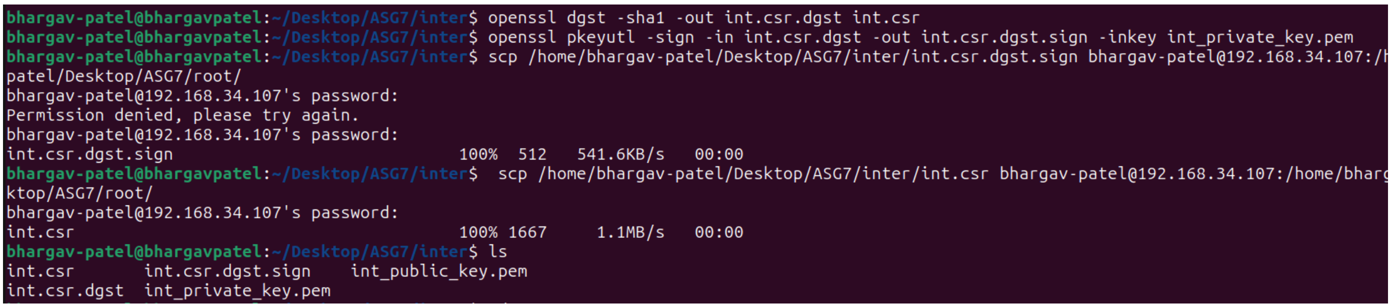
```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl req -new -key bob_private_key.pem -out bob.csr -subj
fig <(printf "[req]\ndistinguished_name=req\n[req_distinguished_name]\n[ v3_req ]\nkeyUsage = keyEncipherment\nextend
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl req -in bob.csr -noout -text
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = Bob1.com, O = IITH, OU = IITH, L = KANDI, C = IN
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:66:f8:3f:7c:ba:21:67:bd:f4:3e:f5:7f:1c:90:
                    d3:08:4a:a6:08:90:89:f7:6c:0a:9b:14:20:7a:0a:
```



```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ cat bob.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIBCTCBsAIBADBOMREwDwYDVQQDDAhCb2IxLmNvbTENMAsGA1UECgwESUlUSDEN
MAsGA1UECwwESUlUSDEOMAwGA1UEBwwFS0FOREkxCzAJBgNVBAYTAklOMFkwEwYH
KoZIzj0CAQYIKoZIzj0DAQcDQgAEZvg/fLohZ73OPvV/HJDTCEqmCJCJ92wKmxQg
egpwQe5aIzgCGjnSDSXPlYD3MglcfrT/p55iJJlJas4ba+FGYaAAMAoGCCqGSM49
BAMCA0gAMEUCIQCUEbeFK4K0jH7OfIuj7gnmgy0HchnTvZsFhreoORwfdgIgPGQr
X8rXgzThQ+EZZzOCdzwp5EvDAe2zza7xYTkCT8M=
-----END CERTIFICATE REQUEST-----
```

3. Signing the Digest:

- **IntermediateCA:** First create a digest (int.csr.dgst) of the csr (int.csr) and then sign the digest generated (int.csr.dgst.sign) using the mentioned commands. Then, securely send **int.csr.dgst.sign** and **int.csr (for verifying)** to the Root using scp (secure copy) commands.

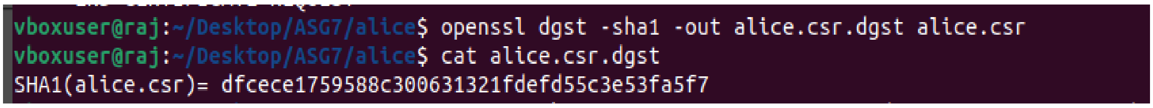


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl dgst -sha1 -out int.csr.dgst int.csr
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl pkeyutl -sign -in int.csr.dgst -out int.csr.dgst.sign -inkey int_private_key.pem
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ scp /home/bhargav-patel/Desktop/ASG7/inter/int.csr.dgst.sign bhargav-patel@192.168.34.107:/h
patel/Desktop/ASG7/root/
bhargav-patel@192.168.34.107's password:
Permission denied, please try again.
bhargav-patel@192.168.34.107's password:
int.csr.dgst.sign                             100%  512   541.6KB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$  scp /home/bhargav-patel/Desktop/ASG7/inter/int.csr bhargav-patel@192.168.34.107:/home/bharg
ktop/ASG7/root/
bhargav-patel@192.168.34.107's password:
int.csr                                       100% 1667     1.1MB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ ls
int.csr       int.csr.dgst.sign    int_public_key.pem
int.csr.dgst  int_private_key.pem
```

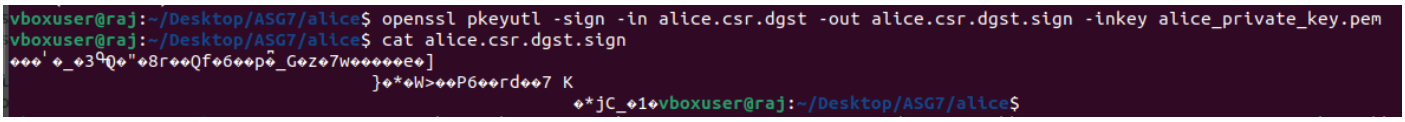
- **Alice:** The same steps were followed as intermediateCA.

1. create digest - alice.csr.dgst
2. sign that digest - alice.csr.dgst.sign

The following commands are used to do above steps. The using scp commands send csr (alice.csr) and signed digest (alice.csr.dgst.sign) to intermediate

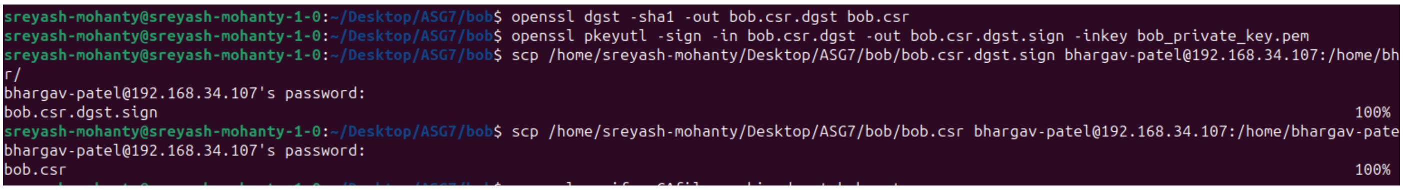


```
vboxuser@raj:~/Desktop/ASG7/alice$ openssl dgst -sha1 -out alice.csr.dgst alice.csr
vboxuser@raj:~/Desktop/ASG7/alice$ cat alice.csr.dgst
SHA1(alice.csr)= dfcece1759588c300631321fdefd55c3e53fa5f7
```



```
vboxuser@raj:~/Desktop/ASG7/alice$ openssl pkeyutl -sign -in alice.csr.dgst -out alice.csr.dgst.sign -inkey alice_private_key.pem
vboxuser@raj:~/Desktop/ASG7/alice$ cat alice.csr.dgst.sign
���'�_�3⁹Q�"�8r��Qf�6��p�_G�z�7w������e�]
                                    }�*�W>��P6��rd��7 K
                                                   �*jC_�1�vboxuser@raj:~/Desktop/ASG7/alice$
```

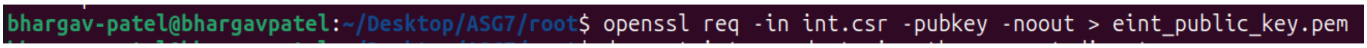
- **Bob:** Same steps as Alice’s are done at Bob’s end.



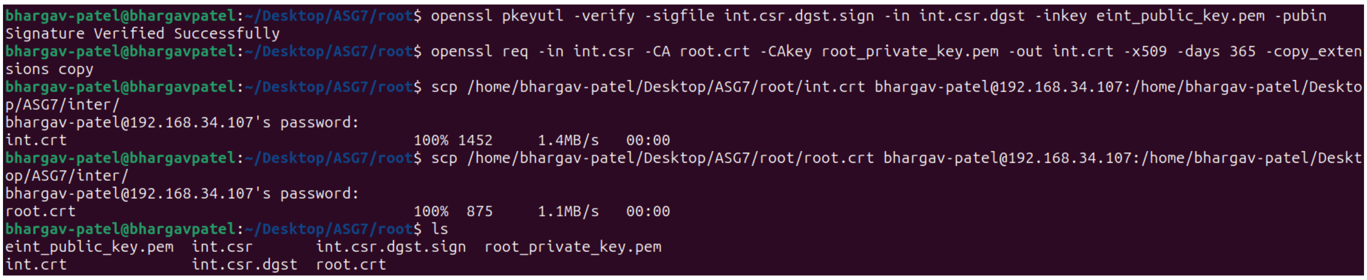
```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl dgst -sha1 -out bob.csr.dgst bob.csr
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl pkeyutl -sign -in bob.csr.dgst -out bob.csr.dgst.sign -inkey bob_private_key.pem
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ scp /home/sreyash-mohanty/Desktop/ASG7/bob/bob.csr.dgst.sign bhargav-patel@192.168.34.107:/home/bh
r/
bhargav-patel@192.168.34.107's password:
bob.csr.dgst.sign                                                                                                                              100%
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ scp /home/sreyash-mohanty/Desktop/ASG7/bob/bob.csr bhargav-patel@192.168.34.107:/home/bhargav-pate
bhargav-patel@192.168.34.107's password:
bob.csr                                                                                                                                        100%
```

4. **<u>Extract - Verify - create certificate - send to respective client:</u>**

- **IntermediateCA:** Extract - verify - create a certificate (steps done at RootCA)
   1. Extract key form int.csr
   2. Create digest form that extracted key (eint_public_key.pem)
   3. Verify the signature by comparing the digest obtained from intermediateCA and the digest created by RootCA using the extracted key
   4. After verifying the signature, RootCA creates a certificate of intermediateCA with the parameter mentioned in the csr (CN, OU, O, extensions, etc)
   5. Securely sending the intermediateCA’s certificate (int.crt) using scp commands.

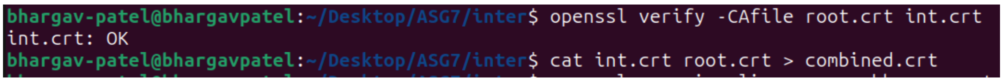


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ openssl req -in int.csr -pubkey -noout > eint_public_key.pem
```



```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ openssl pkeyutl -verify -sigfile int.csr.dgst.sign -in int.csr.dgst -inkey eint_public_key.pem -pubin
Signature Verified Successfully
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ openssl req -in int.csr -CA root.crt -CAkey root_private_key.pem -out int.crt -x509 -days 365 -copy_exten
sions copy
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ scp /home/bhargav-patel/Desktop/ASG7/root/int.crt bhargav-patel@192.168.34.107:/home/bhargav-patel/Deskto
p/ASG7/inter/
bhargav-patel@192.168.34.107's password:
int.crt                                       100% 1452     1.4MB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ scp /home/bhargav-patel/Desktop/ASG7/root/root.crt bhargav-patel@192.168.34.107:/home/bhargav-patel/Deskt
op/ASG7/inter/
bhargav-patel@192.168.34.107's password:
root.crt                                      100%  875     1.1MB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/root$ ls
eint_public_key.pem  int.csr       int.csr.dgst.sign  root_private_key.pem
int.crt              int.csr.dgst  root.crt
```

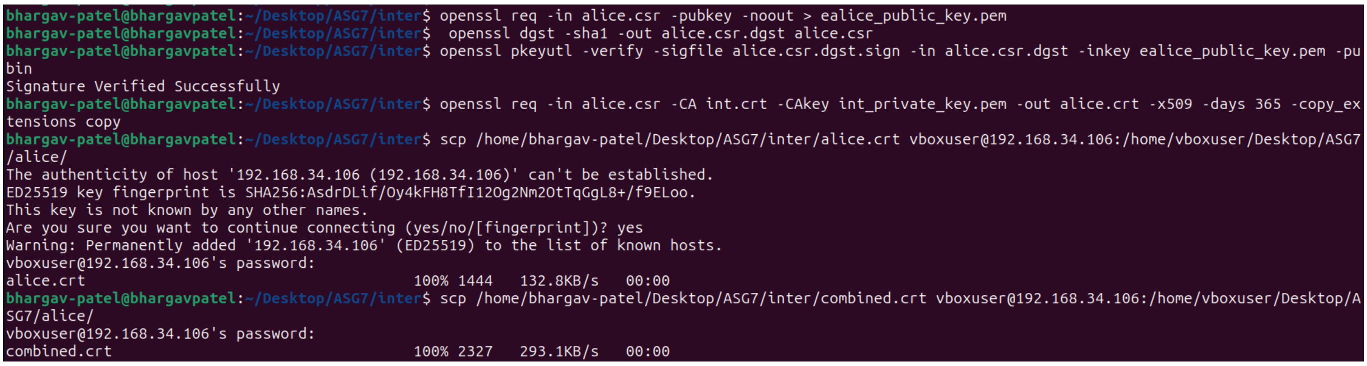
As intermediateCA gets it’s certificate and root certificate. The intermediateCA verifies it’s certificate (int.crt) and combines the root.crt (RootCA certificate) and int.crt (IntermediateCA). After verification of int.crt then only create the end entity certificates (for Alice and Bob).



```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl verify -CAfile root.crt int.crt
int.crt: OK
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ cat int.crt root.crt > combined.crt
```

- **Alice:** Extract - verify - create a certificate (steps done at IntermediateCA)

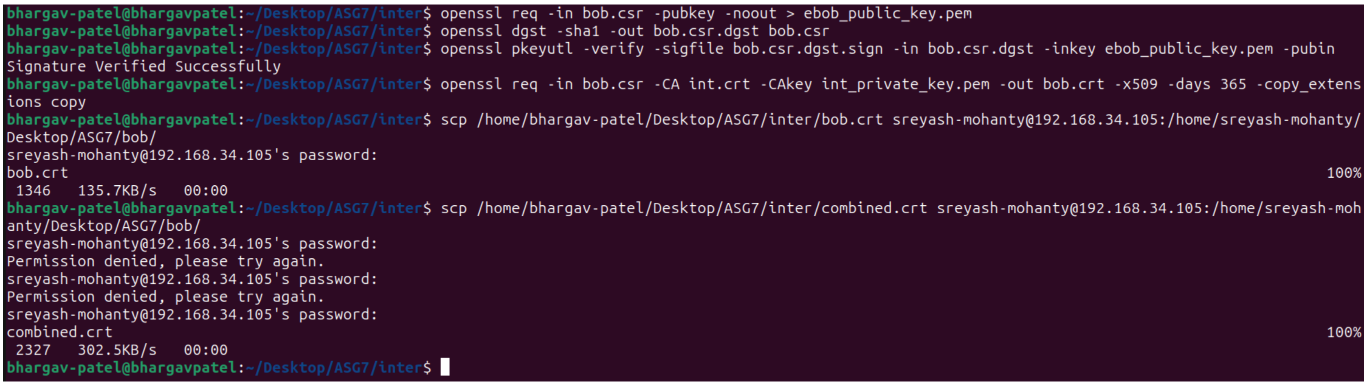
Similar steps followed at the IntermediateCA as was done at RootCA for int.crt
At the end the IntermediateCA sends the combined.crt and Alice.crt to Alice



```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -in alice.csr -pubkey -noout > ealice_public_key.pem
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$  openssl dgst -sha1 -out alice.csr.dgst alice.csr
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl pkeyutl -verify -sigfile alice.csr.dgst.sign -in alice.csr.dgst -inkey ealice_public_key.pem -pu
bin
Signature Verified Successfully
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -in alice.csr -CA int.crt -CAkey int_private_key.pem -out alice.crt -x509 -days 365 -copy_ex
tensions copy
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ scp /home/bhargav-patel/Desktop/ASG7/inter/alice.crt vboxuser@192.168.34.106:/home/vboxuser/Desktop/ASG7
/alice/
The authenticity of host '192.168.34.106 (192.168.34.106)' can't be established.
ED25519 key fingerprint is SHA256:AsdrDLif/Oy4kFH8TfI12Og2Nm2OtTqGgL8+/f9ELoo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.34.106' (ED25519) to the list of known hosts.
vboxuser@192.168.34.106's password:
alice.crt                                     100% 1444   132.8KB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ scp /home/bhargav-patel/Desktop/ASG7/inter/combined.crt vboxuser@192.168.34.106:/home/vboxuser/Desktop/A
SG7/alice/
vboxuser@192.168.34.106's password:
combined.crt                                  100% 2327   293.1KB/s   00:00
```

- **Bob:** Extract - verify - create a certificate (steps done at IntermediateCA)

Similar steps followed at the IntermediateCA as was done at RootCA for int.crt
At the end the IntermediateCA sends the combined.crt and Bob.crt to Bob

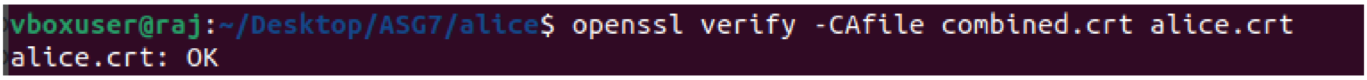


```
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -in bob.csr -pubkey -noout > ebob_public_key.pem
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl dgst -sha1 -out bob.csr.dgst bob.csr
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl pkeyutl -verify -sigfile bob.csr.dgst.sign -in bob.csr.dgst -inkey ebob_public_key.pem -pubin
Signature Verified Successfully
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ openssl req -in bob.csr -CA int.crt -CAkey int_private_key.pem -out bob.crt -x509 -days 365 -copy_extens
ions copy
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ scp /home/bhargav-patel/Desktop/ASG7/inter/bob.crt sreyash-mohanty@192.168.34.105:/home/sreyash-mohanty/
Desktop/ASG7/bob/
sreyash-mohanty@192.168.34.105's password:
bob.crt                                                                                                                                         100%
 1346   135.7KB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ scp /home/bhargav-patel/Desktop/ASG7/inter/combined.crt sreyash-mohanty@192.168.34.105:/home/sreyash-moh
anty/Desktop/ASG7/bob/
sreyash-mohanty@192.168.34.105's password:
Permission denied, please try again.
sreyash-mohanty@192.168.34.105's password:
Permission denied, please try again.
sreyash-mohanty@192.168.34.105's password:
combined.crt                                                                                                                                    100%
 2327   302.5KB/s   00:00
bhargav-patel@bhargavpatel:~/Desktop/ASG7/inter$ 
```

5. Certificate Verification:

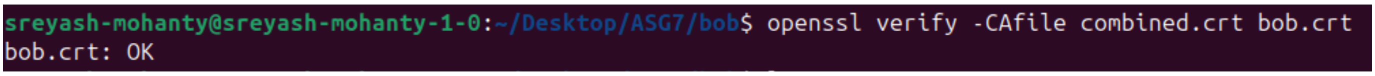
- **IntermediateCA:** Already verified before the creation of Alice.crt & Bob.crt. Shown in the previous step (before the generation of combined.crt).

- **Alice:** Alice verifies the alice.crt using combine.crt (ensuring the chain of trust)



```
vboxuser@raj:~/Desktop/ASG7/alice$ openssl verify -CAfile combined.crt alice.crt
alice.crt: OK
```

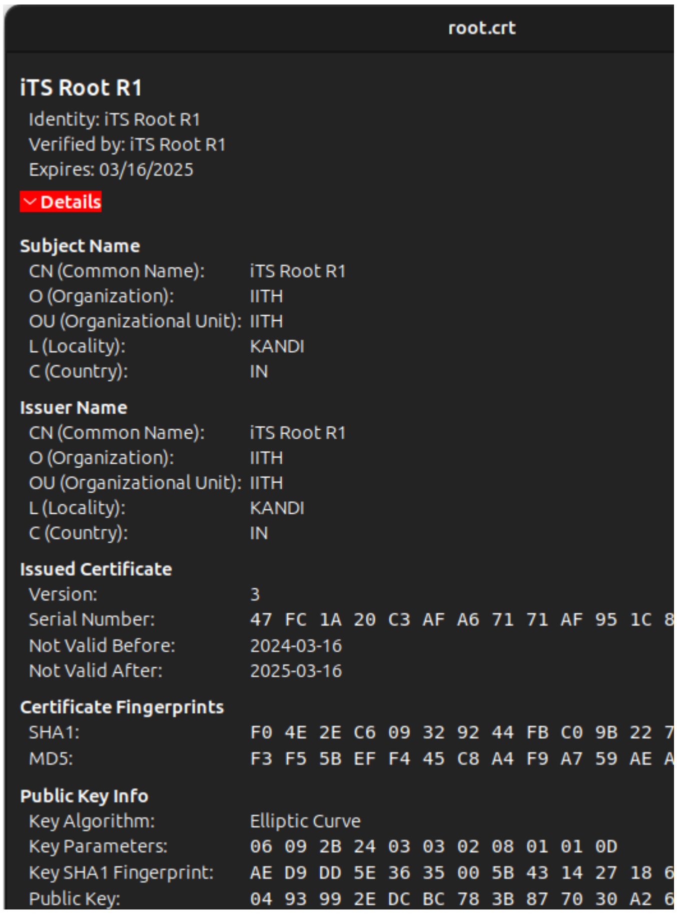
- **Bob:** Bob verifies the bob.crt using combine.crt (ensuring the chain of trust)



```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ openssl verify -CAfile combined.crt bob.crt
bob.crt: OK
```

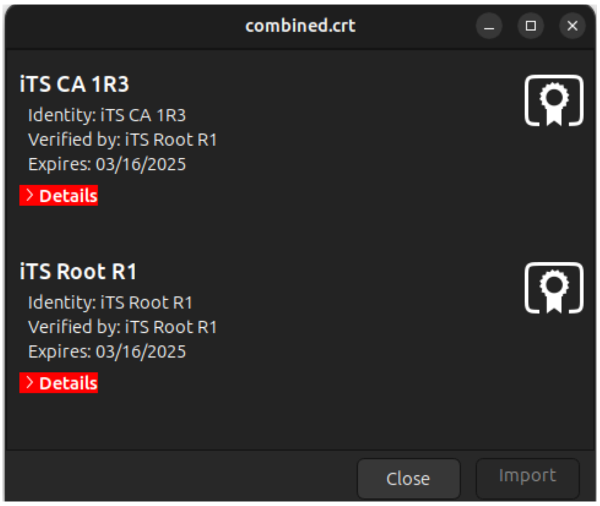
6. RootCA, Combined, IntermediateCA, Alice and Bob's certificate, respectively:

RootCA



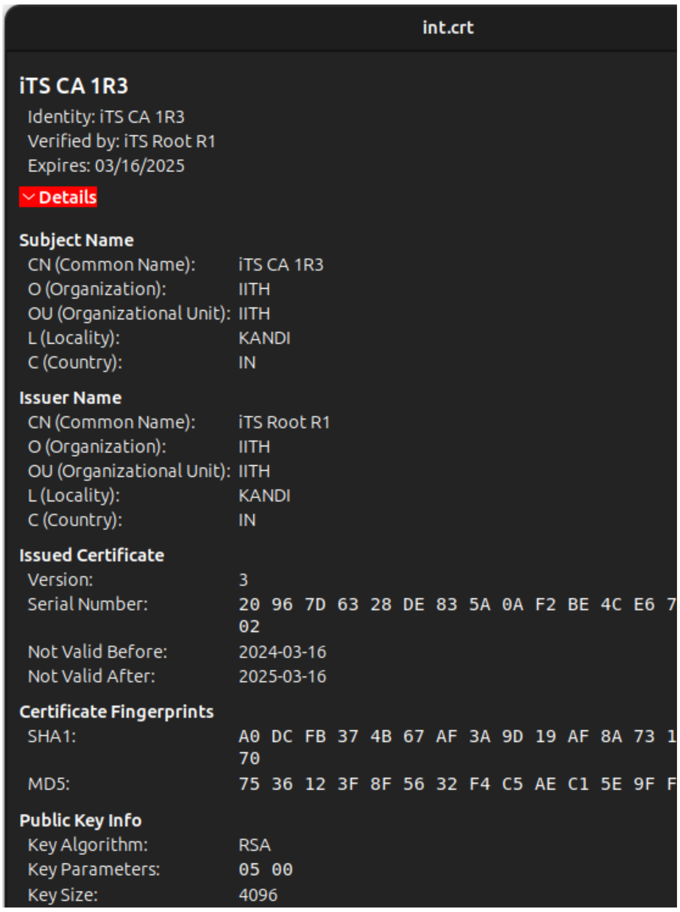
```
root.crt

iTS Root R1
  Identity: iTS Root R1
  Verified by: iTS Root R1
  Expires: 03/16/2025

Details

Subject Name
  CN (Common Name):           iTS Root R1
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issuer Name
  CN (Common Name):           iTS Root R1
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issued Certificate
  Version:                    3
  Serial Number:              47 FC 1A 20 C3 AF A6 71 71 AF 95 1C 8
  Not Valid Before:           2024-03-16
  Not Valid After:            2025-03-16

Certificate Fingerprints
  SHA1:                       F0 4E 2E C6 09 32 92 44 FB C0 9B 22 7
  MD5:                        F3 F5 5B EF F4 45 C8 A4 F9 A7 59 AE A

Public Key Info
  Key Algorithm:              Elliptic Curve
  Key Parameters:             06 09 2B 24 03 03 02 08 01 01 0D
  Key SHA1 Fingerprint:       AE D9 DD 5E 36 35 00 5B 43 14 27 18 6
  Public Key:                 04 93 99 2E DC BC 78 3B 87 70 30 A2 6
```

Combined CRT (Root + Intermediate)



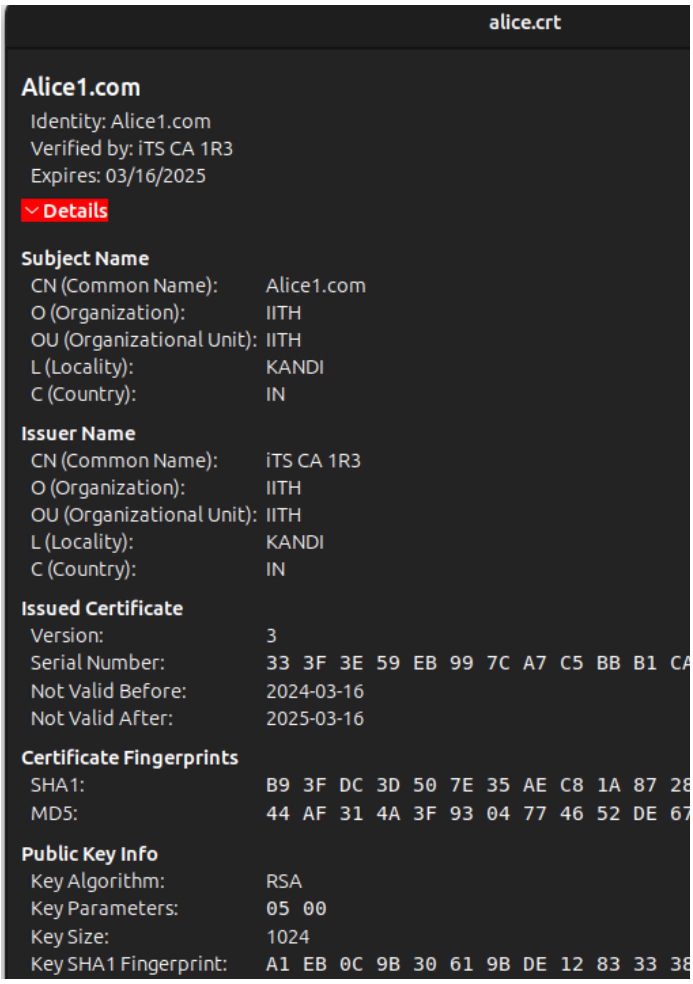
```
combined.crt

iTS CA 1R3
  Identity: iTS CA 1R3
  Verified by: iTS Root R1
  Expires: 03/16/2025

Details

iTS Root R1
  Identity: iTS Root R1
  Verified by: iTS Root R1
  Expires: 03/16/2025

Details
```

IntermediateCA:



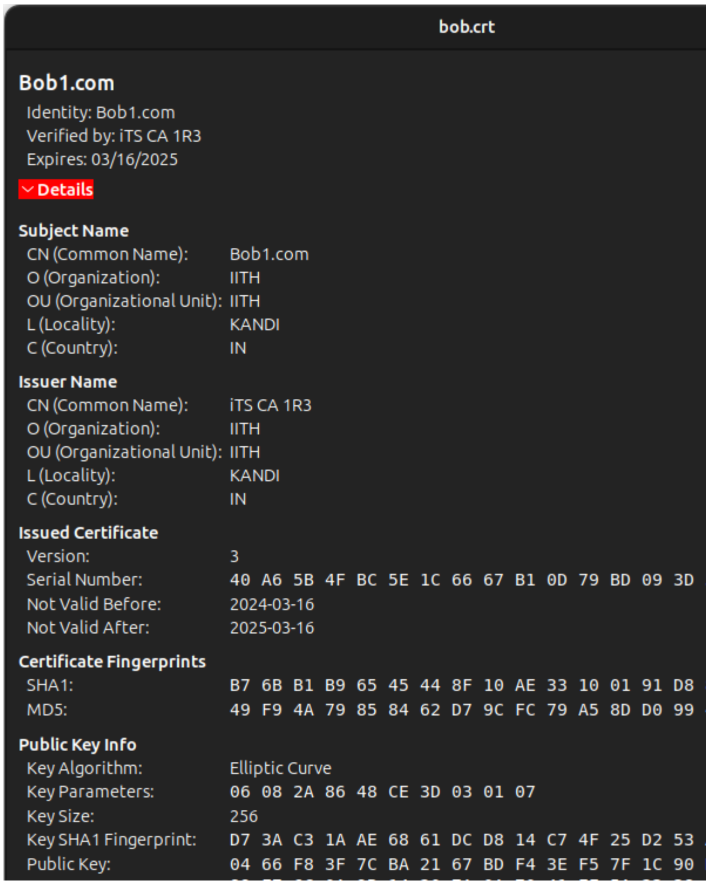
```
int.crt

iTS CA 1R3
  Identity: iTS CA 1R3
  Verified by: iTS Root R1
  Expires: 03/16/2025
Details

Subject Name
  CN (Common Name):           iTS CA 1R3
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issuer Name
  CN (Common Name):           iTS Root R1
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issued Certificate
  Version:                    3
  Serial Number:              20 96 7D 63 28 DE 83 5A 0A F2 BE 4C E6 7
                              02
  Not Valid Before:           2024-03-16
  Not Valid After:            2025-03-16

Certificate Fingerprints
  SHA1:                       A0 DC FB 37 4B 67 AF 3A 9D 19 AF 8A 73 1
                              70
  MD5:                        75 36 12 3F 8F 56 32 F4 C5 AE C1 5E 9F F

Public Key Info
  Key Algorithm:              RSA
  Key Parameters:             05 00
  Key Size:                   4096
```

Alice:



```
alice.crt

Alice1.com
  Identity: Alice1.com
  Verified by: iTS CA 1R3
  Expires: 03/16/2025
Details

Subject Name
  CN (Common Name):           Alice1.com
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issuer Name
  CN (Common Name):           iTS CA 1R3
  O (Organization):           IITH
  OU (Organizational Unit):   IITH
  L (Locality):               KANDI
  C (Country):                IN

Issued Certificate
  Version:                    3
  Serial Number:              33 3F 3E 59 EB 99 7C A7 C5 BB B1 CA
  Not Valid Before:           2024-03-16
  Not Valid After:            2025-03-16

Certificate Fingerprints
  SHA1:                       B9 3F DC 3D 50 7E 35 AE C8 1A 87 28
  MD5:                        44 AF 31 4A 3F 93 04 77 46 52 DE 67

Public Key Info
  Key Algorithm:              RSA
  Key Parameters:             05 00
  Key Size:                   1024
  Key SHA1 Fingerprint:       A1 EB 0C 9B 30 61 9B DE 12 83 33 38
```

Bob:



bob.crt

Bob1.com

Identity: Bob1.com
Verified by: iTS CA 1R3
Expires: 03/16/2025

Details

Subject Name

```
CN (Common Name):              Bob1.com
O (Organization):              IITH
OU (Organizational Unit):      IITH
L (Locality):                  KANDI
C (Country):                   IN
```

Issuer Name

```
CN (Common Name):              iTS CA 1R3
O (Organization):              IITH
OU (Organizational Unit):      IITH
L (Locality):                  KANDI
C (Country):                   IN
```

Issued Certificate

```
Version:                       3
Serial Number:                 40 A6 5B 4F BC 5E 1C 66 67 B1 0D 79 BD 09 3D
Not Valid Before:              2024-03-16
Not Valid After:               2025-03-16
```

Certificate Fingerprints

```
SHA1:                          B7 6B B1 B9 65 45 44 8F 10 AE 33 10 01 91 D8
MD5:                           49 F9 4A 79 85 84 62 D7 9C FC 79 A5 8D D0 99
```

Public Key Info

```
Key Algorithm:                 Elliptic Curve
Key Parameters:                06 08 2A 86 48 CE 3D 03 01 07
Key Size:                      256
Key SHA1 Fingerprint:          D7 3A C3 1A AE 68 61 DC D8 14 C7 4F 25 D2 53
Public Key:                    04 66 F8 3F 7C BA 21 67 BD F4 3E F5 7F 1C 90
```

TASK2: Secure Chat App using DTLSv1.2 and UDP in C++

This part consists of two peers communicating over a secure channel, namely Alice and Bob.

-> The flow of the chat is as given below:

—--------------- Application Control Messages —-------------

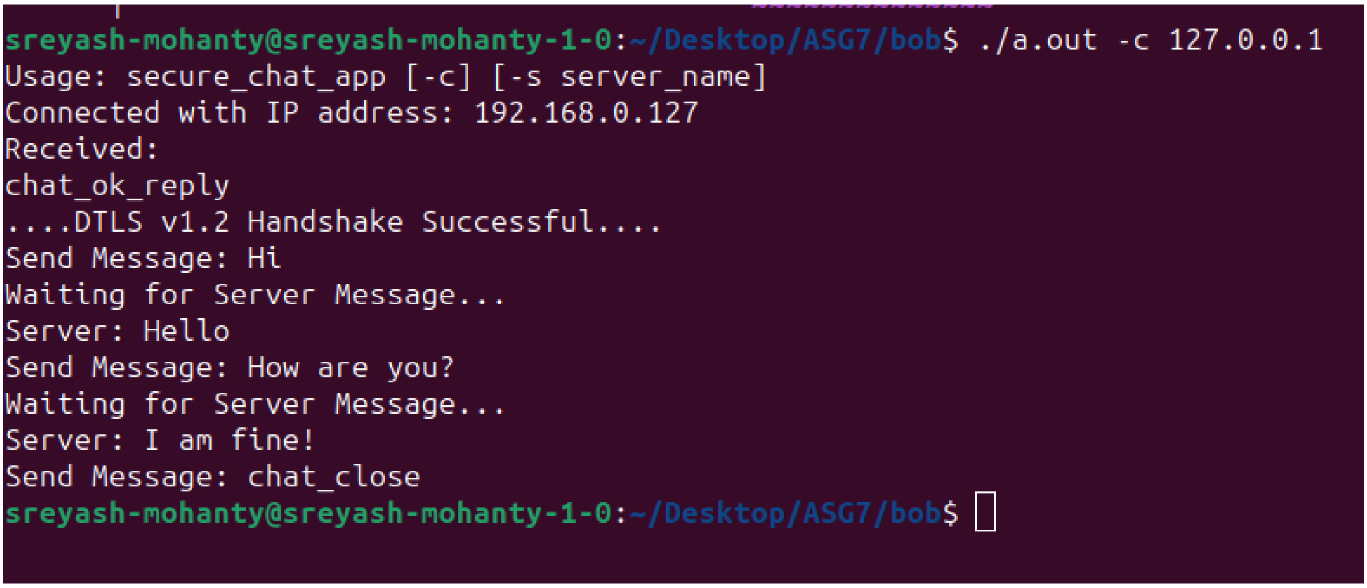
1) The client initiates a chat_hello message.
2) The server responds with chat_ok_reply.
3) The client sends chat_START_SSL
4) The server sends chat_START_TLS_ACK

—---------------DTLSv1.2 Handshake —--------------------------

Regular Chat Messages

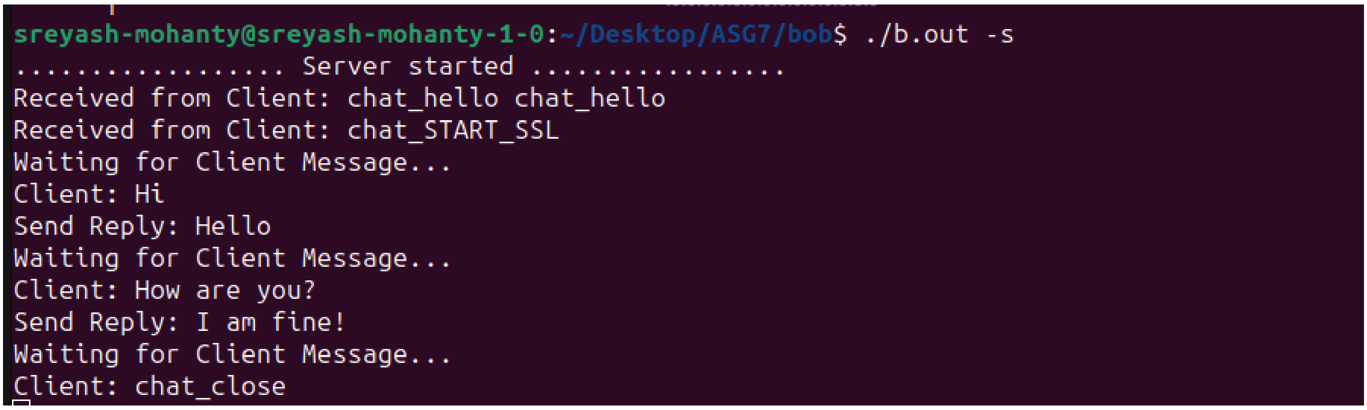
—--

Output (Chat History):



```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ ./a.out -c 127.0.0.1
Usage: secure_chat_app [-c] [-s server_name]
Connected with IP address: 192.168.0.127
Received:
chat_ok_reply
....DTLS v1.2 Handshake Successful....
Send Message: Hi
Waiting for Server Message...
Server: Hello
Send Message: How are you?
Waiting for Server Message...
Server: I am fine!
Send Message: chat_close
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ 
```

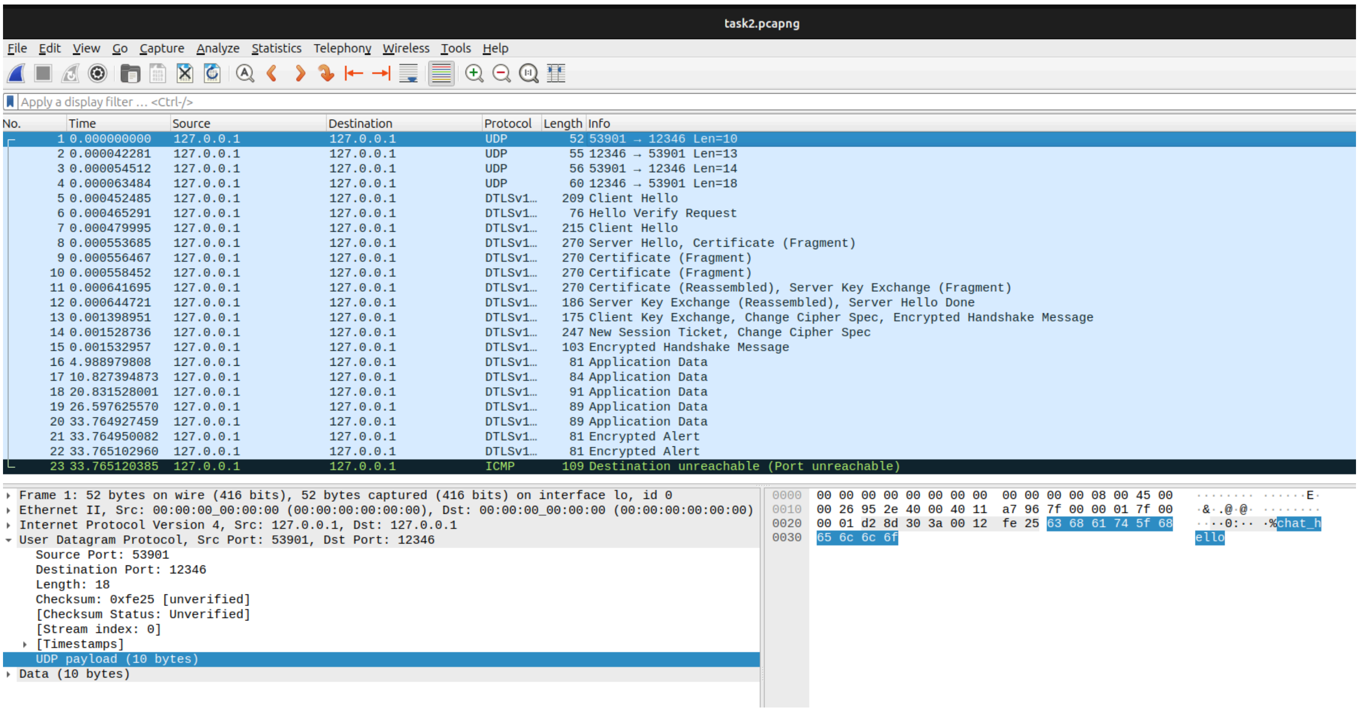
Client View



```
sreyash-mohanty@sreyash-mohanty-1-0:~/Desktop/ASG7/bob$ ./b.out -s
.................. Server started .................
Received from Client: chat_hello chat_hello
Received from Client: chat_START_SSL
Waiting for Client Message...
Client: Hi
Send Reply: Hello
Waiting for Client Message...
Client: How are you?
Send Reply: I am fine!
Waiting for Client Message...
Client: chat_close
```

Server View

Output (Wireshark) :

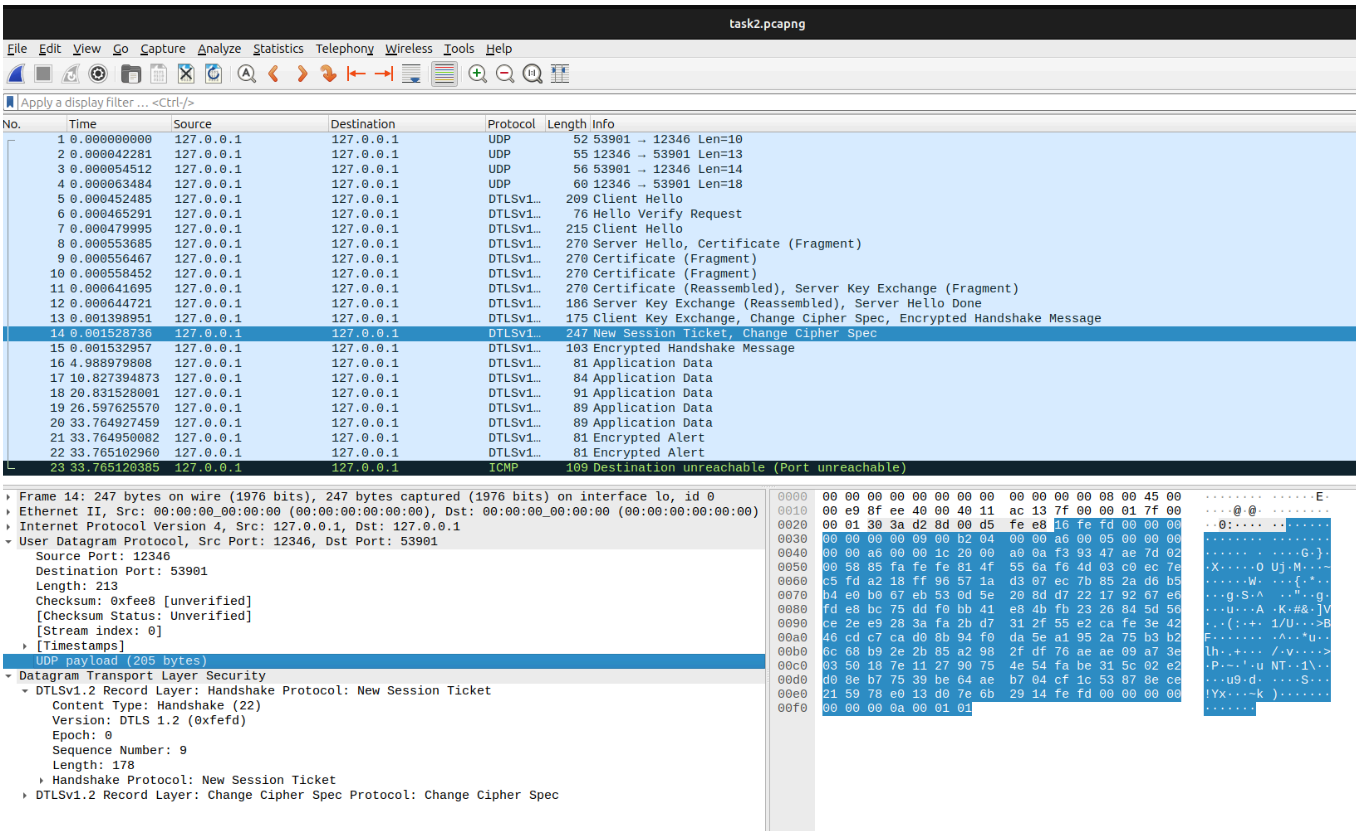

task2.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.    Time           Source        Destination   Protocol  Length Info
     1 0.000000000    127.0.0.1     127.0.0.1     UDP        52 53901 → 12346 Len=10
     2 0.000042281    127.0.0.1     127.0.0.1     UDP        55 12346 → 53901 Len=13
     3 0.000054512    127.0.0.1     127.0.0.1     UDP        56 53901 → 12346 Len=14
     4 0.000063484    127.0.0.1     127.0.0.1     UDP        60 12346 → 53901 Len=18
     5 0.000452485    127.0.0.1     127.0.0.1     DTLSv1…   209 Client Hello
     6 0.000465291    127.0.0.1     127.0.0.1     DTLSv1…    76 Hello Verify Request
     7 0.000479995    127.0.0.1     127.0.0.1     DTLSv1…   215 Client Hello
     8 0.000553685    127.0.0.1     127.0.0.1     DTLSv1…   270 Server Hello, Certificate (Fragment)
     9 0.000556467    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Fragment)
    10 0.000558452    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Fragment)
    11 0.000641695    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Reassembled), Server Key Exchange (Fragment)
    12 0.000644721    127.0.0.1     127.0.0.1     DTLSv1…   186 Server Key Exchange (Reassembled), Server Hello Done
    13 0.001398951    127.0.0.1     127.0.0.1     DTLSv1…   175 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
    14 0.001528736    127.0.0.1     127.0.0.1     DTLSv1…   247 New Session Ticket, Change Cipher Spec
    15 0.001532957    127.0.0.1     127.0.0.1     DTLSv1…   103 Encrypted Handshake Message
    16 4.988979808    127.0.0.1     127.0.0.1     DTLSv1…    81 Application Data
    17 10.827394873   127.0.0.1     127.0.0.1     DTLSv1…    84 Application Data
    18 20.831528001   127.0.0.1     127.0.0.1     DTLSv1…    91 Application Data
    19 26.597625570   127.0.0.1     127.0.0.1     DTLSv1…    89 Application Data
    20 33.764927459   127.0.0.1     127.0.0.1     DTLSv1…    89 Application Data
    21 33.764950082   127.0.0.1     127.0.0.1     DTLSv1…    81 Encrypted Alert
    22 33.765102960   127.0.0.1     127.0.0.1     DTLSv1…    81 Encrypted Alert
    23 33.765120385   127.0.0.1     127.0.0.1     ICMP      109 Destination unreachable (Port unreachable)
```

```
› Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface lo, id 0
› Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
⌄ User Datagram Protocol, Src Port: 53901, Dst Port: 12346
    Source Port: 53901
    Destination Port: 12346
    Length: 18
    Checksum: 0xfe25 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  › [Timestamps]
    UDP payload (10 bytes)
› Data (10 bytes)
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 26 95 2e 40 00 40 11  a7 96 7f 00 00 01 7f 00   ·&·.@·@· ········
0020  00 01 d2 8d 30 3a 00 12  fe 25 63 68 61 74 5f 68   ····0:·· ·%chat_h
0030  65 6c 6c 6f                                        ello
```

Chat_hello (plain-text)


task2.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.    Time           Source        Destination   Protocol  Length Info
     1 0.000000000    127.0.0.1     127.0.0.1     UDP        52 53901 → 12346 Len=10
     2 0.000042281    127.0.0.1     127.0.0.1     UDP        55 12346 → 53901 Len=13
     3 0.000054512    127.0.0.1     127.0.0.1     UDP        56 53901 → 12346 Len=14
     4 0.000063484    127.0.0.1     127.0.0.1     UDP        60 12346 → 53901 Len=18
     5 0.000452485    127.0.0.1     127.0.0.1     DTLSv1…   209 Client Hello
     6 0.000465291    127.0.0.1     127.0.0.1     DTLSv1…    76 Hello Verify Request
     7 0.000479995    127.0.0.1     127.0.0.1     DTLSv1…   215 Client Hello
     8 0.000553685    127.0.0.1     127.0.0.1     DTLSv1…   270 Server Hello, Certificate (Fragment)
     9 0.000556467    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Fragment)
    10 0.000558452    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Fragment)
    11 0.000641695    127.0.0.1     127.0.0.1     DTLSv1…   270 Certificate (Reassembled), Server Key Exchange (Fragment)
    12 0.000644721    127.0.0.1     127.0.0.1     DTLSv1…   186 Server Key Exchange (Reassembled), Server Hello Done
    13 0.001398951    127.0.0.1     127.0.0.1     DTLSv1…   175 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
    14 0.001528736    127.0.0.1     127.0.0.1     DTLSv1…   247 New Session Ticket, Change Cipher Spec
    15 0.001532957    127.0.0.1     127.0.0.1     DTLSv1…   103 Encrypted Handshake Message
    16 4.988979808    127.0.0.1     127.0.0.1     DTLSv1…    81 Application Data
    17 10.827394873   127.0.0.1     127.0.0.1     DTLSv1…    84 Application Data
    18 20.831528001   127.0.0.1     127.0.0.1     DTLSv1…    91 Application Data
    19 26.597625570   127.0.0.1     127.0.0.1     DTLSv1…    89 Application Data
    20 33.764927459   127.0.0.1     127.0.0.1     DTLSv1…    89 Application Data
    21 33.764950082   127.0.0.1     127.0.0.1     DTLSv1…    81 Encrypted Alert
    22 33.765102960   127.0.0.1     127.0.0.1     DTLSv1…    81 Encrypted Alert
    23 33.765120385   127.0.0.1     127.0.0.1     ICMP      109 Destination unreachable (Port unreachable)
```

```
› Frame 14: 247 bytes on wire (1976 bits), 247 bytes captured (1976 bits) on interface lo, id 0
› Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
⌄ User Datagram Protocol, Src Port: 12346, Dst Port: 53901
    Source Port: 12346
    Destination Port: 53901
    Length: 213
    Checksum: 0xfee8 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  › [Timestamps]
    UDP payload (205 bytes)
⌄ Datagram Transport Layer Security
  ⌄ DTLSv1.2 Record Layer: Handshake Protocol: New Session Ticket
      Content Type: Handshake (22)
      Version: DTLS 1.2 (0xfefd)
      Epoch: 0
      Sequence Number: 9
      Length: 178
    › Handshake Protocol: New Session Ticket
  › DTLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 e9 8f ee 40 00 40 11  ac 13 7f 00 00 01 7f 00   ····@·@· ········
0020  00 01 30 3a d2 8d 00 d5  fe e8 16 fe fd 00 00 00   ··0:···· ········
0030  00 00 00 00 09 00 b2 04  00 00 a6 00 05 00 00 00   ········ ········
0040  00 00 a6 00 00 1c 20 00  a0 0a f3 93 47 ae 7d 02   ······ · ····G·}·
0050  00 58 85 fa fe fe 81 4f  55 6a f6 4d 03 c0 ec 7e   ·X·····O Uj·M···~
0060  c5 fd a2 18 ff 96 57 1a  d3 07 ec 7b 85 2a d6 b5   ······W· ···{·*··
0070  b4 e0 b0 67 eb 53 0d 5e  20 8d d7 22 17 92 67 e6   ···g·S·^  ··"··g·
0080  fd e8 bc 75 dd f0 bb 41  e8 4b fb 23 26 84 5d 56   ···u···A ·K·#&·]V
0090  ce 2e e9 28 3a fa 2b d7  31 2f 55 e2 ca fe 3e 42   ·.·(:·+· 1/U···>B
00a0  46 cd c7 ca d0 8b 94 f0  da 5e a1 95 2a 75 b3 b2   F······· ·^··*u··
00b0  6c 68 b9 2e 2b 85 a2 98  2f df 76 ae ae 09 a7 3e   lh·.+··· /·v····>
00c0  03 50 18 7e 11 27 90 75  4e 54 fa be 31 5c 02 e2   ·P·~·'·u NT··1\··
00d0  d0 8e b7 75 39 be 64 ae  b7 04 cf 1c 53 87 8e ce   ···u9·d· ····S···
00e0  21 59 78 e0 13 d0 7e 6b  29 14 fe fd 00 00 00 00   !Yx···~k )·······
00f0  00 00 00 0a 00 01 01                               ·······
```

Session Ticket for Resumption

ANTI-PLAGIARISM STATEMENT

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other students if we become aware of it.

Names: **Raj Popat**
Sreyash Mohanty
Bhargav Patel
Date: **18-03-24**
Signature: **RP, SM, BP**

References:

1. OpenSSL Cookbook: Chapter 1. OpenSSL Command Line (feistyduck.com)
2. /docs/man1.1.1/man3/index.html (openssl.org)
3. OpenSSL client and server from scratch, part 1 – Arthur O'Dwyer – Stuff mostly about C++ (quuxplusone.github.io)
4. ssl — TLS/SSL wrapper for socket objects — Python 3.9.2 documentation
5. Secure programming with the OpenSSL API – IBM Developer
6. Simple TLS Server - OpenSSLWiki
7. The /etc/hosts file (tldp.org)
8. PowerPoint Presentation (owasp.org)
9. SEED Project (seedsecuritylabs.org)
10. https://github.com/ManishaMahapatra1/Secure-chat-using-openssl-and-MITM-attacks **(Using Certificates for Successful DTLSv1.2 handshake)**