Assignment Report

We use the OpenSSL library

OpenSSL is a free and open-source cryptographic library that provides several command-line tools for handling digital certificates.

As a CA we deal with cryptographic pairs of private keys and public certificates. We'll create is the root pair. This consists of the root key (`ca.key.pem`) and root certificate (`ca.cert.pem`). This pair is the identity of our CA.

1) Creating root :

For that we create folders like

```
mkdir certs crl newcerts private
```

Then we configure the openssl.cnf which has information regarding certificate.

For more info refer the openssl.cnf file

Now generating our root key

```
openssl genrsa -aes256 -out private/ca.key.pem 4096

chmod 400 private/ca.key.pem
```

Creating root certificate

```
openssl req -config openssl.cnf \
        -key private/ca.key.pem \
        -new -x509 -days 7300 -sha256 -extensions v3_ca \
        -out certs/ca.cert.pem
```

```
Country Name (2 letter code) [GB]:IN
State or Province Name [England]:TG
Locality Name []:Hyd
Organization Name [Alice Ltd]:RealMadrid
Organizational Unit Name []:Captain
Common Name []:Ronaldo
Email Address []:es14btech11017@iith.ac.in
```

Fill details as above

Then we verify the certificate

```
openssl x509 -noout -text -in certs/ca.cert.pem
```

2) Creating the IntermediateCA

we follow the above procedure but only the change comes where the rootCA certifies and verifies the Intermediate.

We verify as follows:

```
openssl verify -CAfile certs/ca.cert.pem \
        intermediate/certs/intermediate.cert.pem
```

3) Creating Chain

We create a CA chain and store it.

```
cat intermediate/certs/intermediate.cert.pem \
        certs/ca.cert.pem > intermediate/certs/ca-chain.cert.pem
```

Example above.

4) Creating Client/Server

Follow the above procedure to create the key and certificate.

Get it signed by the intermediate CA.

We verify the certifcate using the CA chain.

```
openssl verify -CAfile intermediate/certs/ca-chain.cert.pem \
        intermediate/certs/www.example.com.cert.pem
```

Similarly, we can create various departments/intermediates and servers signed respectively by them.

Task2:

Now we have the server certificate, we deploy our webserver. We include the CA chain and deploy the rootCA to Chrome.

Now we open brower to type the IP of server.

My server was deployed on the AWS, so I had to send all my server certificates, key and chain by ssh. Obviously using the secret key.

We copy the files using scp to the server.

I used the below libraries for secure HTTPS connection.

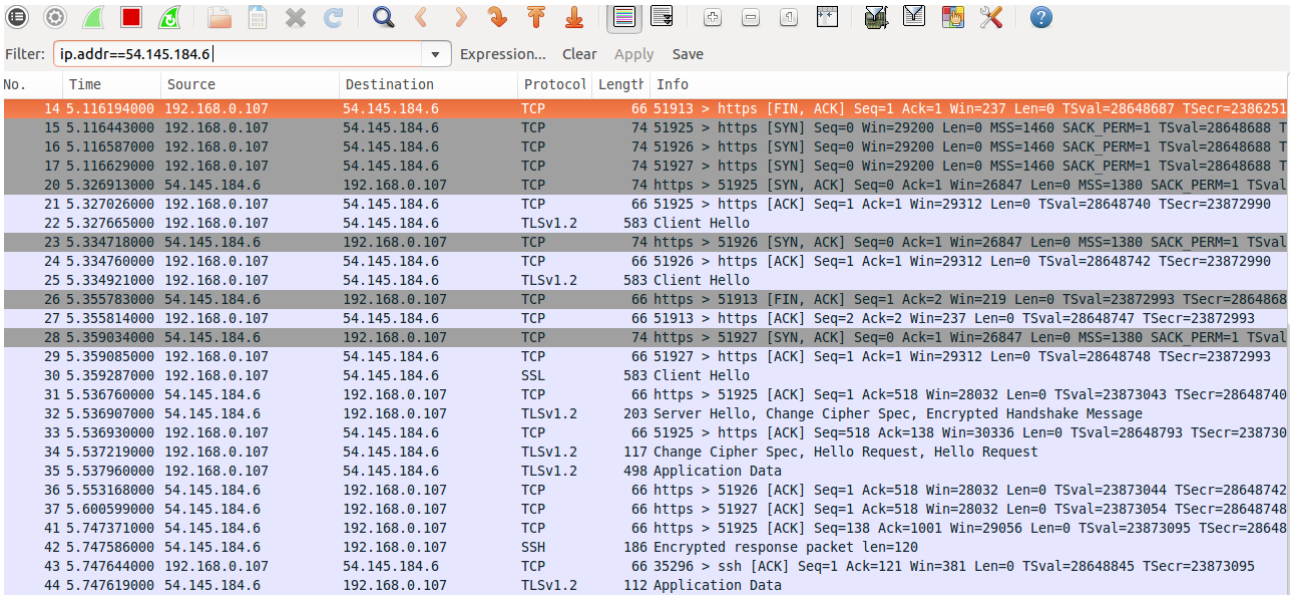
import ssl

from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer

I used the simple HTML form.

The server included ca_certs which verified the heirarchy using CA chain.

This worked gracefully for me and I could capture packets on wireshark.



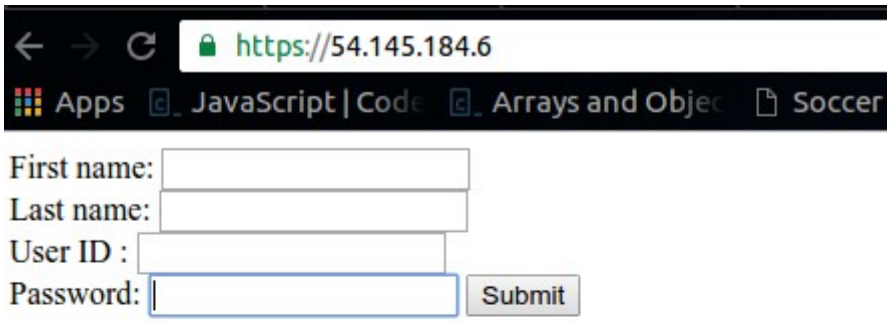
Filter: ip.addr==54.145.184.6 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
14	5.116194000	192.168.0.107	54.145.184.6	TCP	66	51913 > https [FIN, ACK] Seq=1 Ack=1 Win=237 Len=0 TSval=28648687 TSecr=2386251
15	5.116443000	192.168.0.107	54.145.184.6	TCP	74	51925 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=28648688 T
16	5.116587000	192.168.0.107	54.145.184.6	TCP	74	51926 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=28648688 T
17	5.116629000	192.168.0.107	54.145.184.6	TCP	74	51927 > https [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=28648688 T
20	5.326913000	54.145.184.6	192.168.0.107	TCP	74	https > 51925 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1380 SACK_PERM=1 TSval
21	5.327026000	192.168.0.107	54.145.184.6	TCP	66	51925 > https [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=28648740 TSecr=23872990
22	5.327665000	192.168.0.107	54.145.184.6	TLSv1.2	583	Client Hello
23	5.334718000	54.145.184.6	192.168.0.107	TCP	74	https > 51926 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1380 SACK_PERM=1 TSval
24	5.334760000	192.168.0.107	54.145.184.6	TCP	66	51926 > https [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=28648742 TSecr=23872990
25	5.334921000	192.168.0.107	54.145.184.6	TLSv1.2	583	Client Hello
26	5.355783000	54.145.184.6	192.168.0.107	TCP	66	https > 51913 [FIN, ACK] Seq=1 Ack=2 Win=219 Len=0 TSval=23872993 TSecr=2864868
27	5.355814000	192.168.0.107	54.145.184.6	TCP	66	51913 > https [ACK] Seq=2 Ack=2 Win=237 Len=0 TSval=28648747 TSecr=23872993
28	5.359034000	54.145.184.6	192.168.0.107	TCP	74	https > 51927 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1380 SACK_PERM=1 TSval
29	5.359085000	192.168.0.107	54.145.184.6	TCP	66	51927 > https [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=28648748 TSecr=23872993
30	5.359287000	192.168.0.107	54.145.184.6	SSL	583	Client Hello
31	5.536760000	54.145.184.6	192.168.0.107	TCP	66	https > 51925 [ACK] Seq=1 Ack=518 Win=28032 Len=0 TSval=23873043 TSecr=28648740
32	5.536907000	54.145.184.6	192.168.0.107	TLSv1.2	203	Server Hello, Change Cipher Spec, Encrypted Handshake Message
33	5.536930000	192.168.0.107	54.145.184.6	TCP	66	51925 > https [ACK] Seq=518 Ack=138 Win=30336 Len=0 TSval=28648793 TSecr=238730
34	5.537219000	192.168.0.107	54.145.184.6	TLSv1.2	117	Change Cipher Spec, Hello Request, Hello Request
35	5.537960000	192.168.0.107	54.145.184.6	TLSv1.2	498	Application Data
36	5.553168000	54.145.184.6	192.168.0.107	TCP	66	https > 51926 [ACK] Seq=1 Ack=518 Win=28032 Len=0 TSval=23873044 TSecr=28648742
37	5.600599000	54.145.184.6	192.168.0.107	TCP	66	https > 51927 [ACK] Seq=1 Ack=518 Win=28032 Len=0 TSval=23873054 TSecr=28648748
41	5.747371000	54.145.184.6	192.168.0.107	TCP	66	https > 51925 [ACK] Seq=138 Ack=1001 Win=29056 Len=0 TSval=23873095 TSecr=28648
42	5.747586000	54.145.184.6	192.168.0.107	SSH	186	Encrypted response packet len=120
43	5.747644000	192.168.0.107	54.145.184.6	TCP	66	35296 > ssh [ACK] Seq=1 Ack=121 Win=381 Len=0 TSval=28648845 TSecr=23873095
44	5.747619000	54.145.184.6	192.168.0.107	TLSv1.2	112	Application Data

I have attached the wireshark file to verify.

DONT FORGET TO ADD ROOTCA to the server.

I was able to get the secure lock after the above procedure.



The above shows the green secure lock on myserver.

TASK 3:

We now make a secure peer to peer chat application.

For this I created another server signed from the intermediate CA. I got the key and certificate as above and got signed it by Intermediate CA.

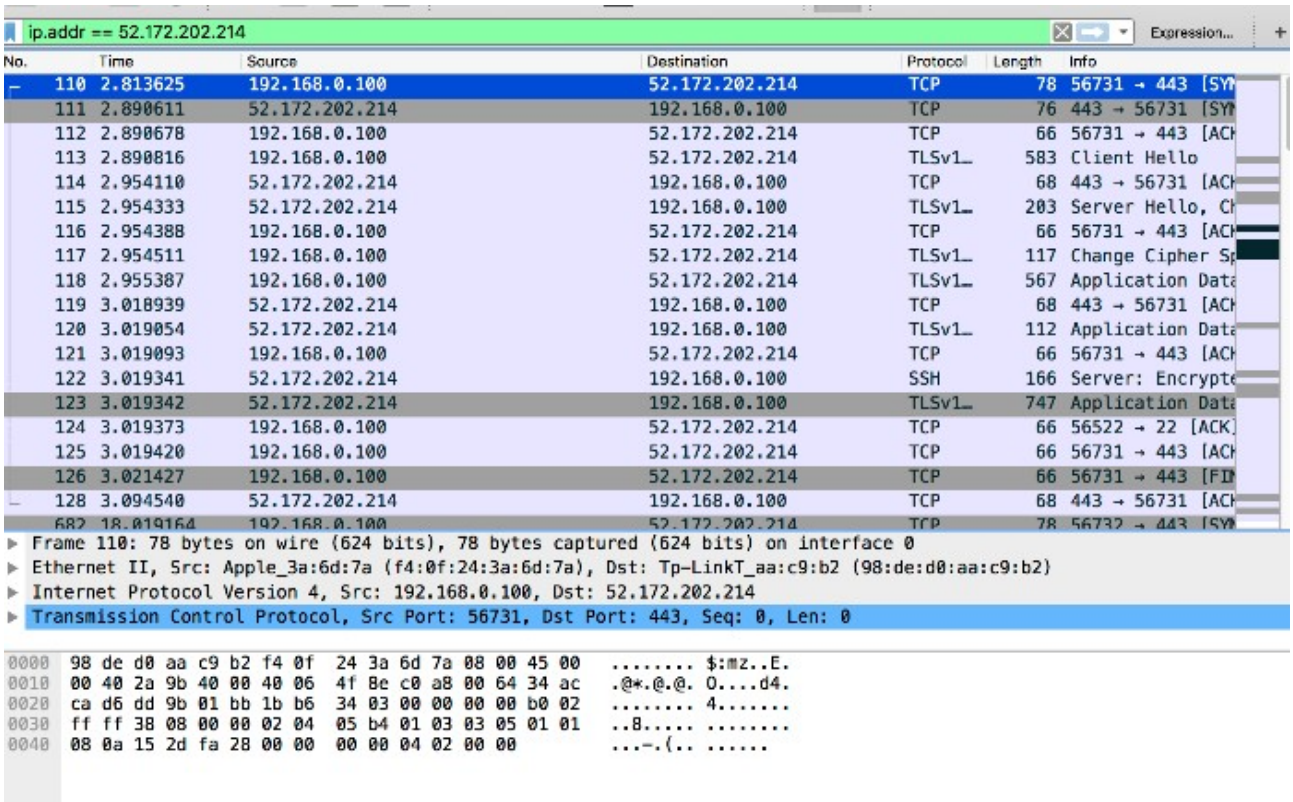
Now Creating the server,

We created a normal socket client server with additional

ssl_wrap function which allows input of Certificate, key and the Ca-chain of the server.

The Certificate is verified by matching the root CA with the heirachy of the CA chain with the server. If this is certified then it establishes the connection and askes fot the key.

The message can be sent and received from both sides.



I created a server and captured packets to show the secure connection.

Task 4:

We create another root CA like we did before. Now we created its intermediate and servers as before. Now the only change we do is we concatenated the RootCA and RootCA1 and store it. Now we put this conacatenated certificate into the browser and compare it with the CA chains.

I have submitted the heirarchy for referal.