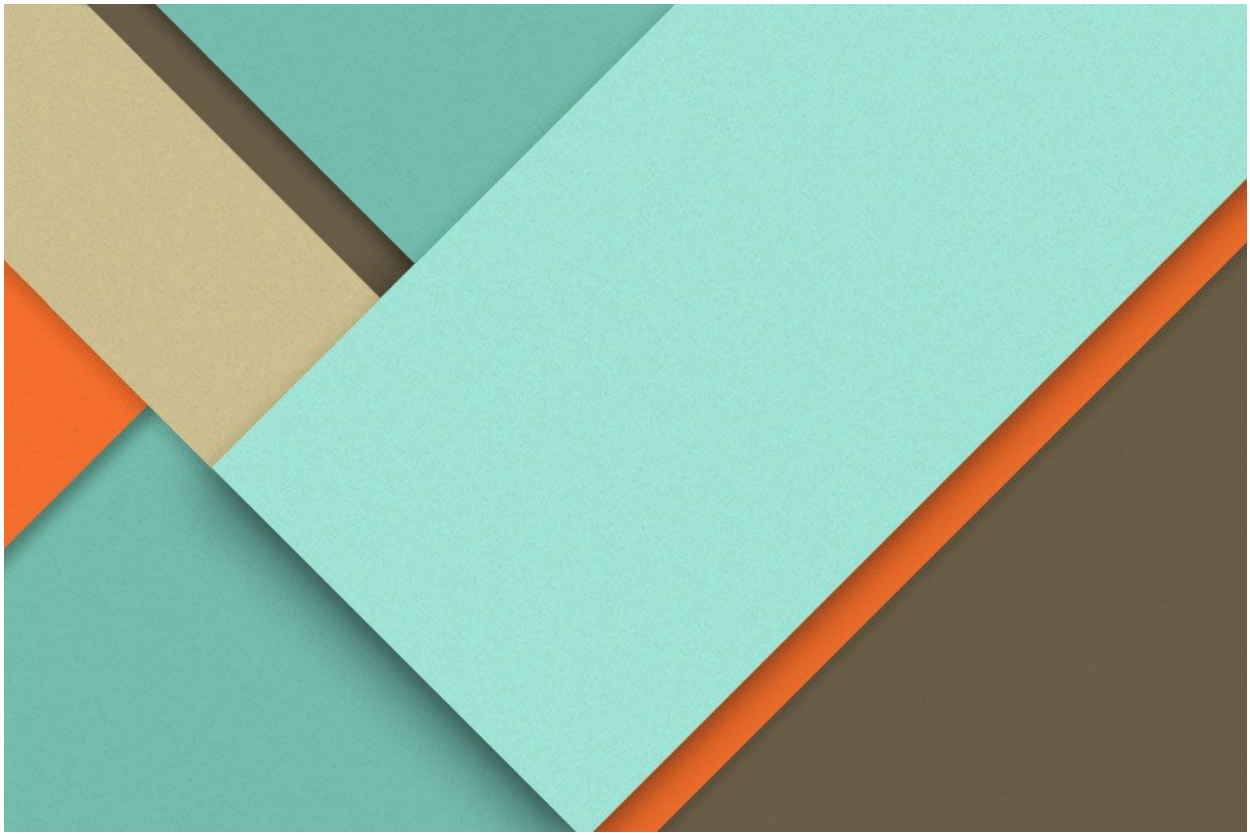


TLS Assignment

23.04.2019

Om Sitapara
CS16BTECH11036

Harshit Patel
CS16BTECH11017

Plagiarism Statement:

We certify that this assignment/report is our own group's 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, 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 in any other course, 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 plagiarised 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 honour violations by other students if we become aware of it.

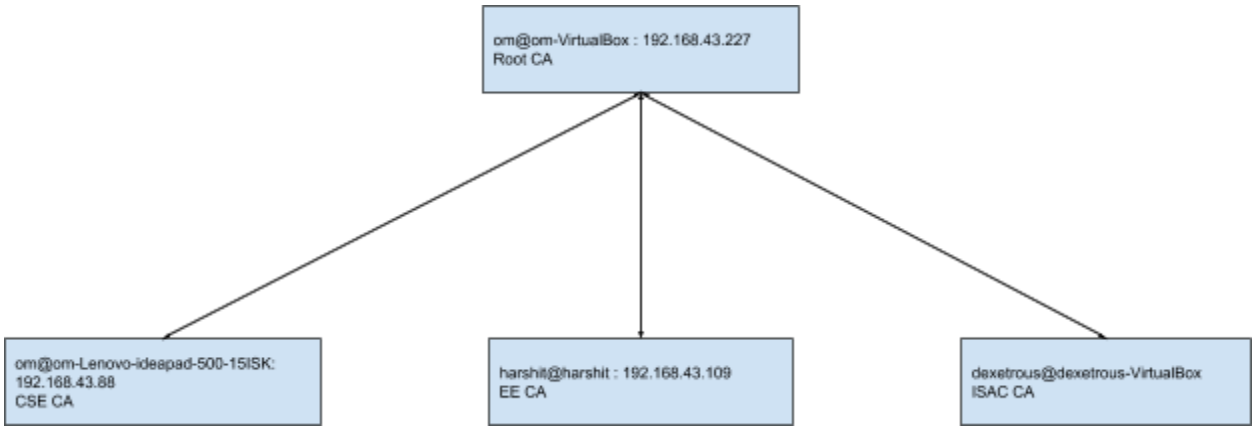
Names of the Students in the group: Om Sitapara, Harshit Patel

Date: 23/04/2019

Signature: OS, HP

Question - 1:

Here



Here are the screenshots for the transfer.

- Sending EE csr:

```
harshit@harshit:~$ scp ee.csr om@192.168.43.227:/home/om/
The authenticity of host '192.168.43.227 (192.168.43.227)' can't be established.
ECDSA key fingerprint is SHA256:19nngAJ416PCqvpzn8SFPSKfYF+a7e6RXRLyxWKjyNc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.227' (ECDSA) to the list of known hosts.
om@192.168.43.227's password:
ee.csr                                        100% 1029    72.3KB/s   00:00
```

- Receiving EE.pem by root:

```
om@om-VirtualBox:~$ scp ee.pem harshit@192.168.43.109:/home/harshit/
The authenticity of host '192.168.43.109 (192.168.43.109)' can't be established.
ECDSA key fingerprint is SHA256:YZcl45iIf1PVb4i2eZWihp8p8Dn59FCm7XnyOFvmbGg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.109' (ECDSA) to the list of known hosts.
harshit@192.168.43.109's password:
ee.pem                                        100% 1314    18.7KB/s   00:00
```

Similarly here are the other screenshots of transfer and signing:

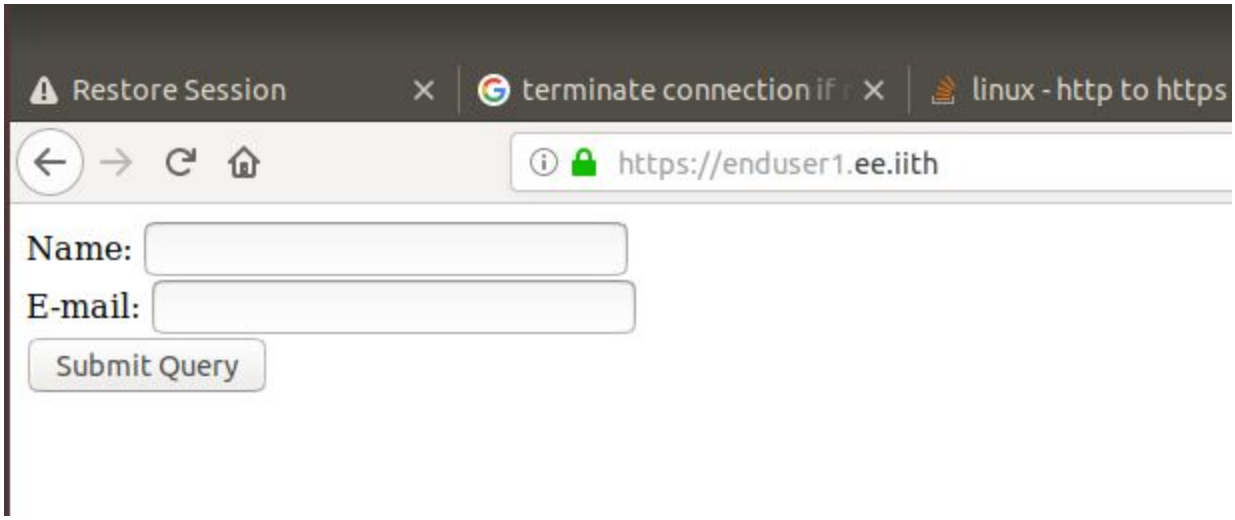
```
om@om-VirtualBox:~$ scp isac.pem dexetrous@192.168.43.95:/home/dexetrous/
The authenticity of host '192.168.43.95 (192.168.43.95)' can't be established.
ECDSA key fingerprint is SHA256:SxgNwd36lQKNRhuiuNeoJkH4eL07Fk3lW3taxP848SE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.95' (ECDSA) to the list of known hosts.
dexetrous@192.168.43.95's password:
isac.pem                                      100% 1322    65.6KB/s   00:00
```

```
dexetrous@dexetrous-VirtualBox:~$ scp isac.csr om@192.168.43.227:/home/om/
The authenticity of host '192.168.43.227 (192.168.43.227)' can't be established.
ECDSA key fingerprint is f3:2d:88:65:69:37:2d:0d:77:7a:b9:28:be:6c:24:ed.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.43.227' (ECDSA) to the list of known hosts.
om@192.168.43.227's password:
isac.csr                                      100% 1041     1.0KB/s   00:00
```

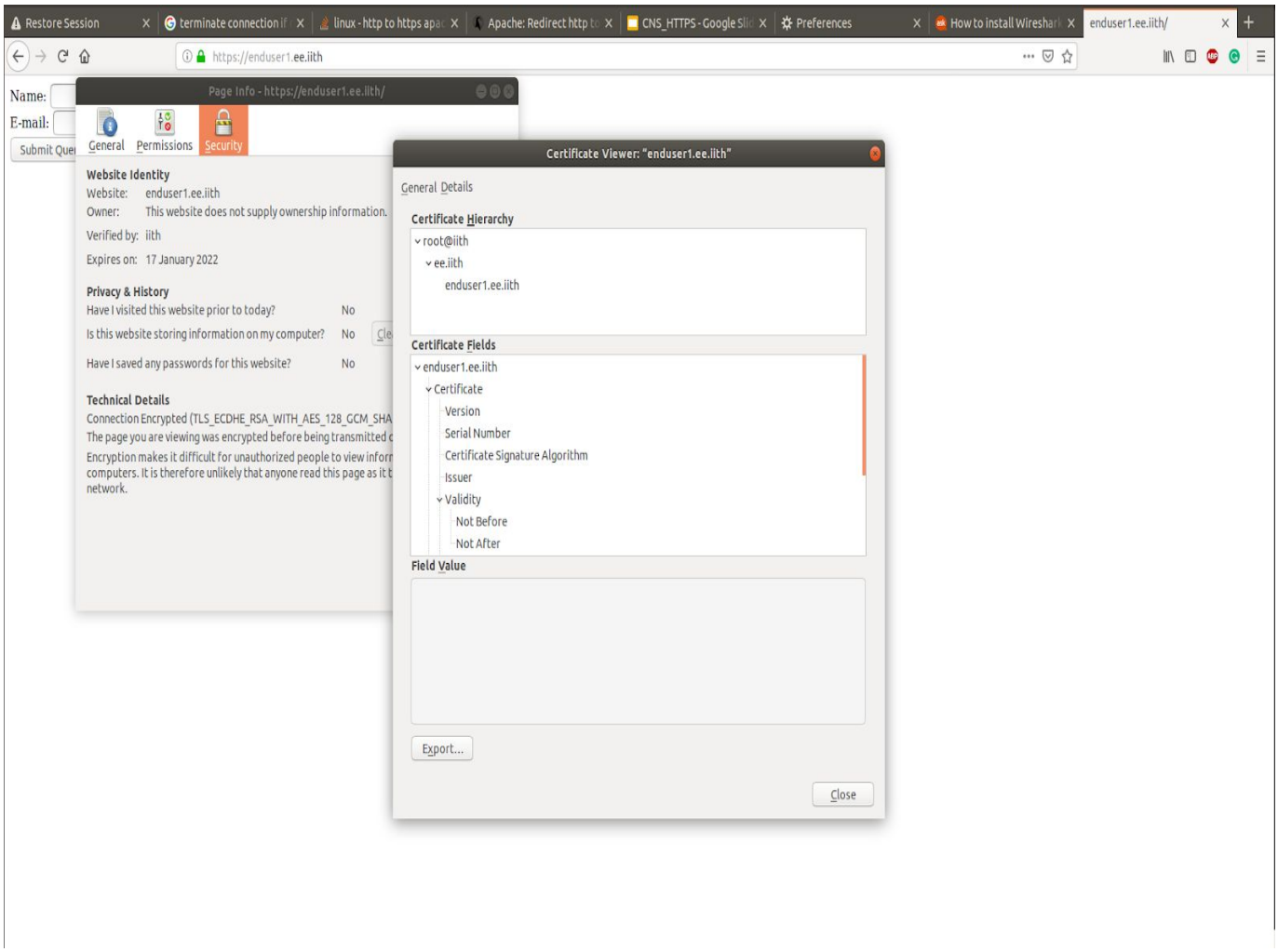
Question-2

For this part we have hosted a website called enduser1.ee.iith on ip 192.168.43.109. We have made this server to redirect all traffic to https. Now on ip 192.168.43.88 we did the following things:

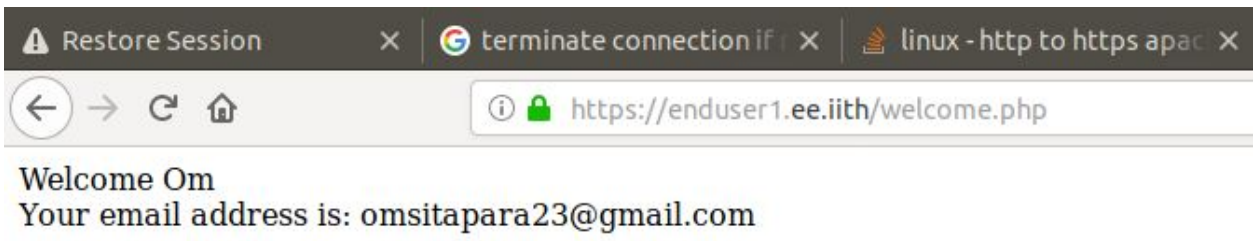
- Installed the **root CA** of iith
- Added the entry **192.168.43.109 enduser1.ee.iith** in **/etc/hosts/** for accessing the site using name (performing local dns)
- Now accessing the site it opens like this:



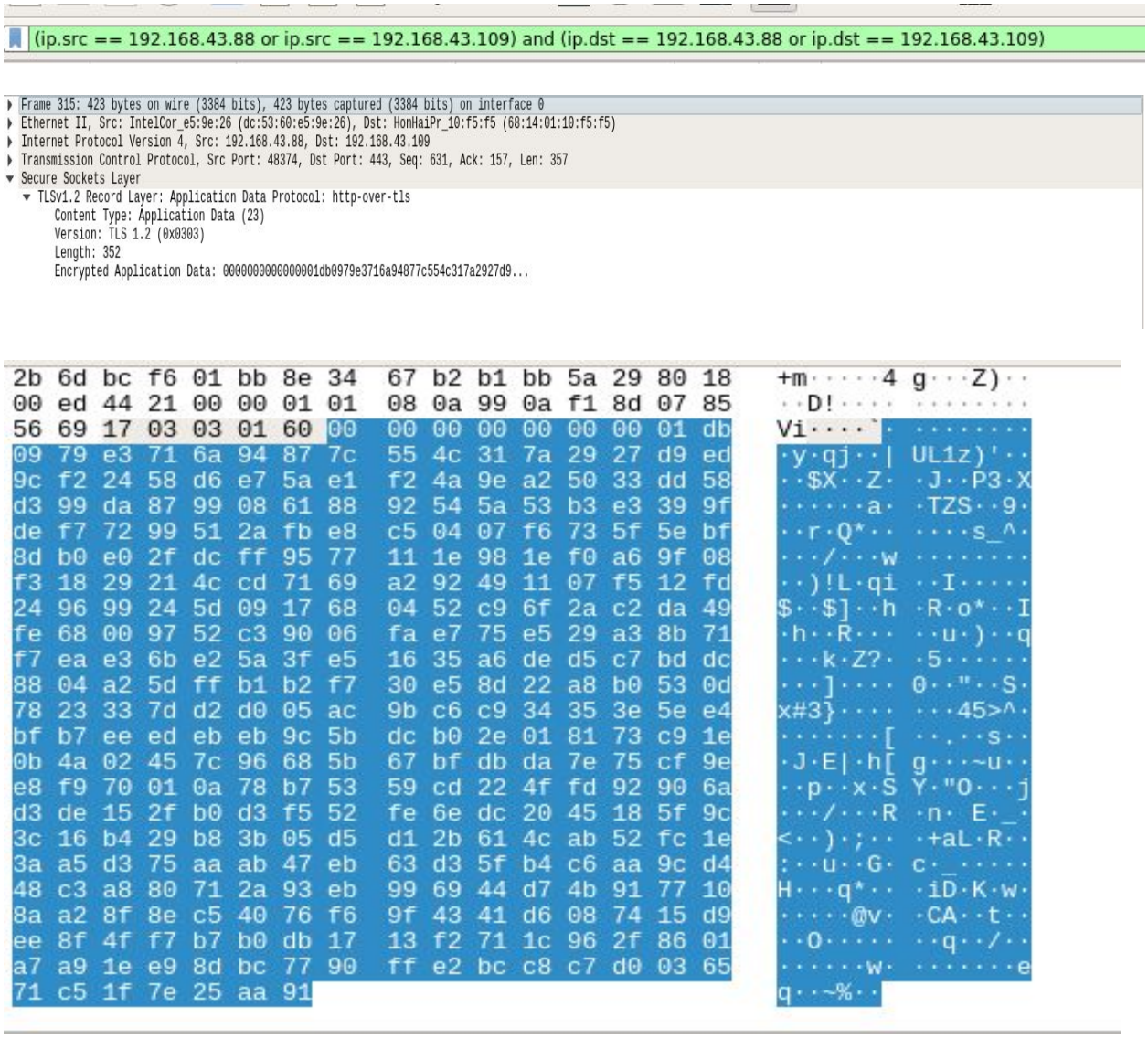
- The certificate can be viewed :



- Now entering the name and password : This shows that the server has received the entered data



- The messages passed are encrypted :



Question-3

We have 3 code files:

Peer.cpp

clientSSL.cpp

serverSSL.cpp

Command to run : g++ peer.cpp -o peer -lssl -lcrypto -lpthread

./peer

This will ask to be server or client. Now set the fields asked accordingly.

In the code for each peer, I have loaded openssl algorithms:

OpenSSL_add_all_algorithms();

And the error strings using

SSL_load_error_strings();

Then we have created a **SSL_CTX*** named **ctx** and passed the method created for each peer into this context. For each peer, mode for **SSL_CTX_set_verify()**

Is set to **SSL_VERIFY_PEER.**

Verify location is set using: **SSL_CTX_load_verify_locations()**

Which will be used to verify the received certificate.

Next certificate file and key(to check if certificate matches the key) are loaded into the SSL context.

Then a TCP connection is created using connect() and accept().

After the TCP connection is established we create a SSL connection using this tcp connection using **SSL_connect()** and **SSL_accept()**.

Then the certificates are verified for the other peer. Suitable errors are generated if the verification process fails.

After the verification succeeds, each peer creates 2 threads for sending and receiving messages.

Messages are send using SSL_write() and received using SSL_read().

To terminate the SSL session and close it both the peers have to send “exit” message.

Demonstrating the chat application:

PEER1 sends certificate : enduserEE1.pem , uses key enduserEE1.key

Certificate to verify : eechain.pem

PEER1 sends certificate : enduserEE.pem , uses key enduserEE.key

Certificate to verify : eechain.pem

Note: eechain.pem is a chain of EE CA and root CA

Here are the screenshots for the chat:

```
harshit@harshit:~/Documents/CNS Prog$ g++ peer.cpp -o peer -lssl -lcrypto -lpthread
harshit@harshit:~/Documents/CNS Prog$ ./peer
Enter your choice :
1 for Server, 2 for Receiver
1
Enter Port No :3000
Enter certificate(enduser) to send :enduserEE1.pem
Enter private key(enduser) path :enduserEE1.key
Enter certificate to verify with :eechain.pem
Enter PEM pass phrase:
4
Certificate validation completed :
Start chatting:


hi I am peer 1
   hi i am peer 2

how are you
   ca va. et toi?

Nice, CNS is fun
exit   exit
```

```
om@om-Lenovo-ideapad-500-15ISK:~/Downloads$ ./peer
Enter your choice :
1 for Server, 2 for Receiver
2
Enter Port No :3000
Enter certificate(client) to send :enduserEE.pem
Enter private key(enduser) path :enduserEE.key
Enter certificate to verify with :eechain.pem
Enter PEM pass phrase:
connected
Certificate validation completed :
Start chatting:


   hi I am peer 1

hi i am peer 2
   how are you

ca va. et toi?
   Nice, CNS is fun

exit
   exit
```

Now here we can see that the peers send the request for others certificate after they send their own certificate:

```
192.168.43.109  192.168.43.88   TLSv1.2  3363 Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
192.168.43.88   192.168.43.109  TCP        66 50460 → 3000 [ACK] Seq=177 Ack=3298 Win=35840 Len=0 TSval=2571756416 TSecr=130304108
192.168.43.88   192.168.43.109  TCP      1514 50460 → 3000 [ACK] Seq=177 Ack=3298 Win=35840 Len=1448 TSval=2571756430 TSecr=130304108
192.168.43.88   192.168.43.109  TLSv1.2  1514 Certificate [TCP segment of a reassembled PDU]
```

Here we can see that the message are transfered in an encrypted fashion: