TLS Programming Assignment

Problem statement: Let us assume that IITH is going to have many web services inside the campus which would be accessed among the IITH community. In this case, we may not need to have server certificates to be issued by a public Certificate Authority, we can have our own Certificate Authority to have secured services. This assignment is about creating such infrastructure and services by completing the following four tasks.

Before starting with the assignment, we need to install OpenSSL[1] and apache2[2] as a prerequisite.

```
$ sudo apt-get install openssl
$ sudo apt-get install apache2
```

1. Creating a Public Key Infrastructure (PKI) for IITH

Problem Statement: Let us have one root CA for IITH who would certify the intermediate CA for each department i.e., CSE, EE, ISAC. When each department wants to host their web servers then can get their server certificate signed by the intermediate CA of their own department. All the IITH community can configure their system to trust the IITH root CA. You need to create this PKI for IITH. You can use any cloud services such as Google Cloud or your own laptop to create VMs for each entity i.e., root CA, CSE CA, EE CA and ISAC CA. Server certificates for each department can be certified by only the respective intermediate CAs. Ensure that you provide realistic meta data while creating certificates like values for CN, OU, L, Country, etc.

1.1 Creating the Root CA

In this section, we will be creating the root pair which consists of the Root Key and Root Certificate. This pair will form the identity of our Root CA. Follow the below steps in order to create the Root CA of your Public Key Infrastructure,

1.1.1 Preparing the Directory

This directory will store all the keys and certificates for the Root CA and the Intermediate CAs. ***index.txt*** and ***serial*** will act as a flat file database to keep track of signed certificates.

```
$ mkdir /root/ca
$ cd /root/ca
$ mkdir certs crl newcerts private
$ chmod 777 private
$ touch index.txt
$ echo 1000 \> serial
```

1.1.2 Preparing the Configuration file

We will also create a configuration file for OpenSSL to use. If the OpenSSL configuration file is not specified, OpenSSL uses the default configuration file situated at ***/etc/ssl/openssl.cnf***.

In the OpenSSL configuration file, we will specify the **[ca]** section which is mandatory and tells OpenSSL which defaults to use. We will then specify the **[CA_default]** which contains a long range of defaults like default directory, sub-directory, database file etc. We will also have a **[policy_strict]** section which is required by all the Root CAs, as the Root CAs are only used to create intermediate CAs. We will also have a **[req]** section which is used for certificate signing requests, **[v3_ca]** which is usually used when creating the root certificate along with **[usr_cert]**, **[server_cert]** and **[oscp]**.

```
$ gedit openssl.conf
```

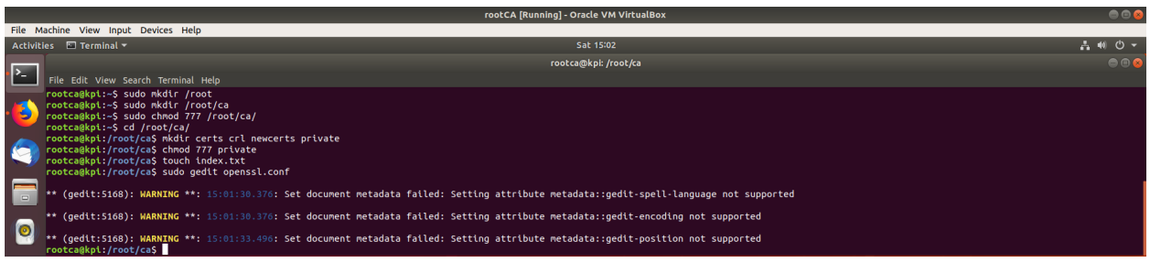

Figure 1: Root CA - Preparing the Directory & OpenSSL Configuration file.

1.1.3 Creating the Root Key

We will then create the Root Key which will be used to sign the Root Certificate. We will be using AES 256 bit encryption to create the Root Key and store it in ***/root/ca/private***.

```
$ cd /root/ca
$ openssl genrsa -aes256 -out private/ca.key.pem 4096
$ chmod 777 private/ca.key.pem
```

1.1.4 Creating the Root Certificate

We will use the Root Key to sign and create the Root Certificate. We will provide the Root Certificate a long expiry date, because, once the root certificate expires all the certificates signed by the CA becomes invalid.

```
$ cd /root/ca
$ openssl req -config openssl.conf -key private/ca.key.pem \
    -new -x509 -days 7300 -sha256 -extensions v3_ca -out certs/ca.cert.pem
$ chmod 777 certs/ca.cert.pem
```

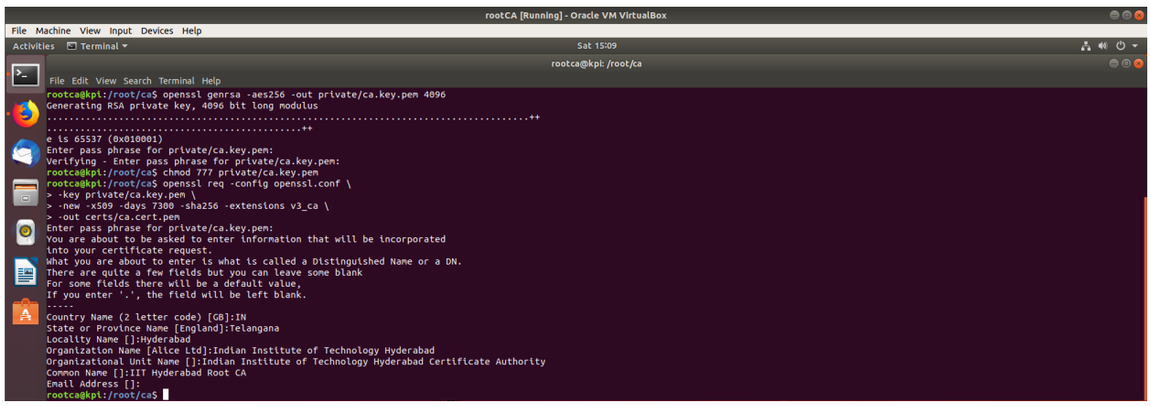

Figure 2: Root CA - Creating the Root Key & Certificate.

1.1.5 Content of Root Certificate

We can verify the contents of the Root Certificate by using the below command. The Root Certificate will contain Signature Algorithm, Validity, Public-key length, Issuer, Subject among other fields. It is to note that for Root Certificates *Issuer* and *Subject* will be identical.

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

1.2 Creating the CSE Intermediate CA

In this section, we will be creating the intermediate pair for CSE which consists of the CSE Intermediate Key and Certificate. This pair will form the identity of our Intermediate CA for CSE Department. Follow the below steps in order to create the Intermediate CSE CA of your Public Key Infrastructure,

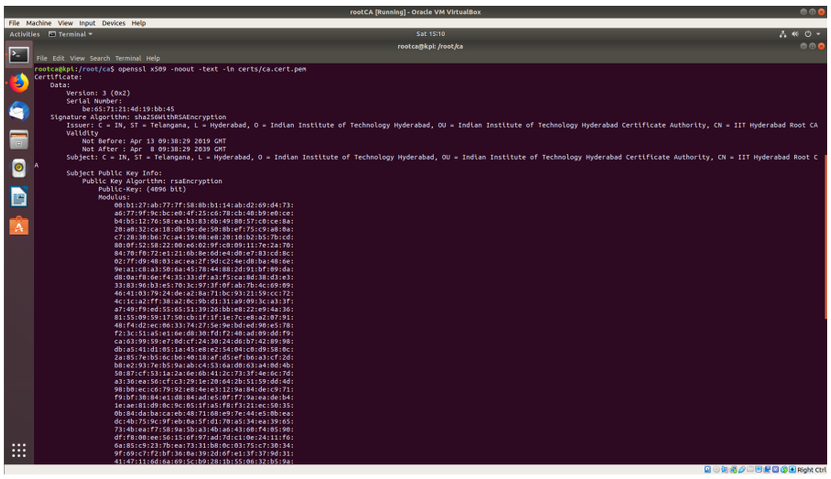

Figure 3: Root CA - Content of Root Certificate.

1.2.1 Preparing the Directory

This directory will store all the keys and certificates for the Root CA and the Intermediate CAs. ***index.txt*** and ***serial*** will act as a flat file database to keep track of signed certificates. And ***csr*** will be used to hold the certificate signing requests and ***crlnumber*** to keep track of certificate revocation lists.

```
$ mkdir /root/ca/intermediate_cse
$ cd /root/ca/intermediate_cse
$ mkdir certs crl csr newcerts private
$ chmod 777 private
$ touch index.txt
$ echo 1000 \> serial
$ echo 1000 \> crlnumber
```

1.2.2 Preparing the Configuration file

We will also create a configuration file for OpenSSL to use. If the OpenSSL configuration file is not specified, OpenSSL uses the default configuration file situated at ***/etc/ssl/openssl.cnf***.

The OpenSSL configuration file for Intermediate CA would be same as for Root CA except that the ***directory*** and ***policy*** would be different in ***[CA_default]***. The policy for Root CA was ***strict_policy*** whereas the policy for Intermediate CA would be ***loose_policy***.

```
$ gedit openssl.conf
```

Figure 4: Intermediate CSE CA - Preparing the Directory & OpenSSL Configuration file.

1.2.3 Creating the CSE Intermediate Key

We will then create the Intermediate CSE Key which will be used to sign the Intermediate CSE Certificate. We will be using AES 256 bit encryption to create the Intermediate CSE Key and store it in ***/root/ca/intermediate_cse/private***.

```
$ cd /root/ca
$ openssl genrsa -aes256 -out intermediate_cse/private/intermediate.key.pem 4096
$ chmod 777 intermediate_cse/private/intermediate.key.pem
```

1.2.4 Creating the CSE Intermediate Certificate

We will use the Intermediate CSE Key to sign and create the Certificate Signing Request (CSR). The details of the CSR should generally match with the Root CA. The ***Common Name***, however, must be different.

```
$ cd /root/ca
$ openssl req -config intermediate_cse/openssl.conf -new sha256 \
    -key intermediate_cse/private/intermediate.key.pem \
    -out intermediate_cse/csr/intermediate.csr.pem
```

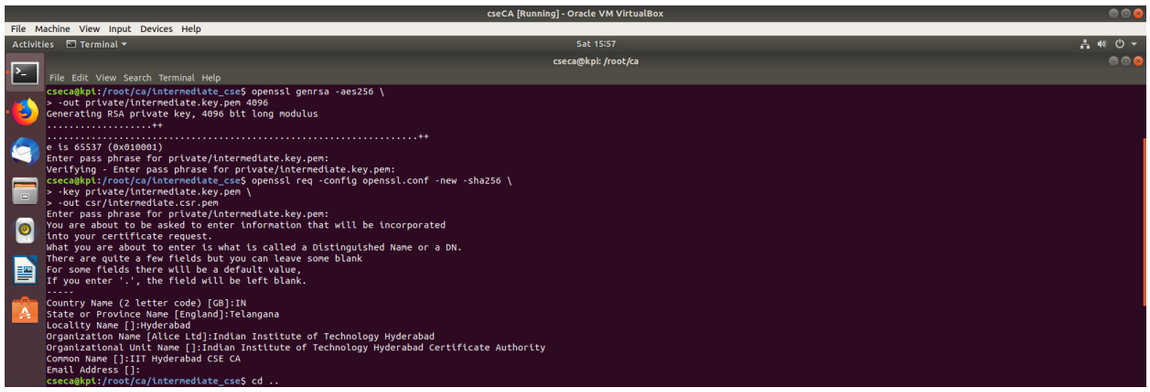

Figure 5: Intermediate CSE CA - Creating the Intermediate CSE Key & Certificate Signing Request.

To create the intermediate certificate, we will use the Root CA to sign the Intermediate CSR.

```
$ cd /root/ca
$ openssl ca -config openssl.conf -extensions v3_intermediate_ca \
    -days 3650 -notext -md sha256 \
    -in intermediate_cse/csr/intermediate.csr.pem \
    -out intermediate_cse/certs/intermediate.cert.pem
$ chmod 777 intermediate_cse/certs/intermediate.cert.pem
```

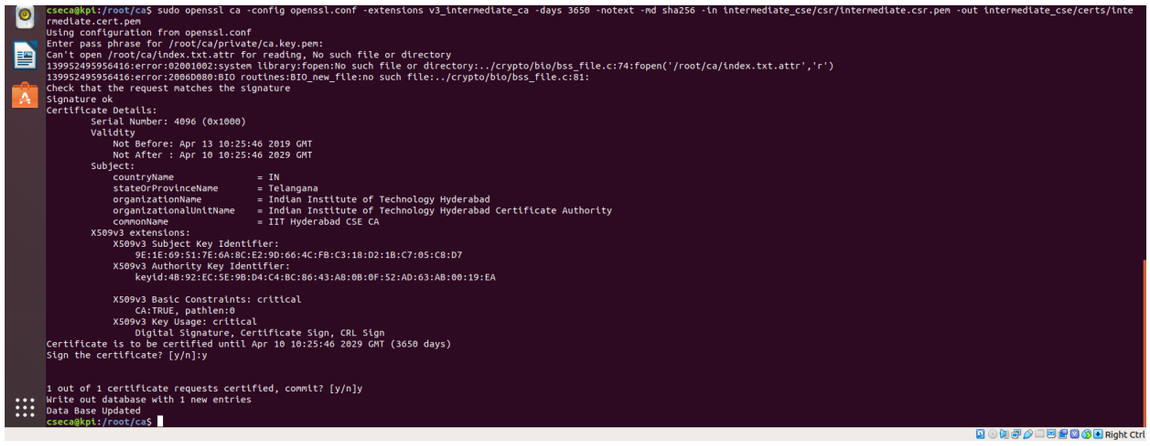

Figure 6: Intermediate CSE CA - Creating the Intermediate CSE Certificate.

1.2.5 Verifying the Intermediate CSE Certificate

We can verify the contents of the Intermediate EE Certificate by using the below command.

```
$ openssl x509 -noout -text -in intermediate_cse/certs/intermediate.cert.pem
```

We will also verify the Intermediate CSE Certificate against that of the Root Certificate. An ***OK*** will indicate that the chain of trust is intact.

```
$ Copy the Root Certificate \& Intermediate CSE Cert using SCP
$ openssl verify -CAfile certs/ca.cert.pem\
    intermediate_cse/certs/intermediate.cert.pem
```

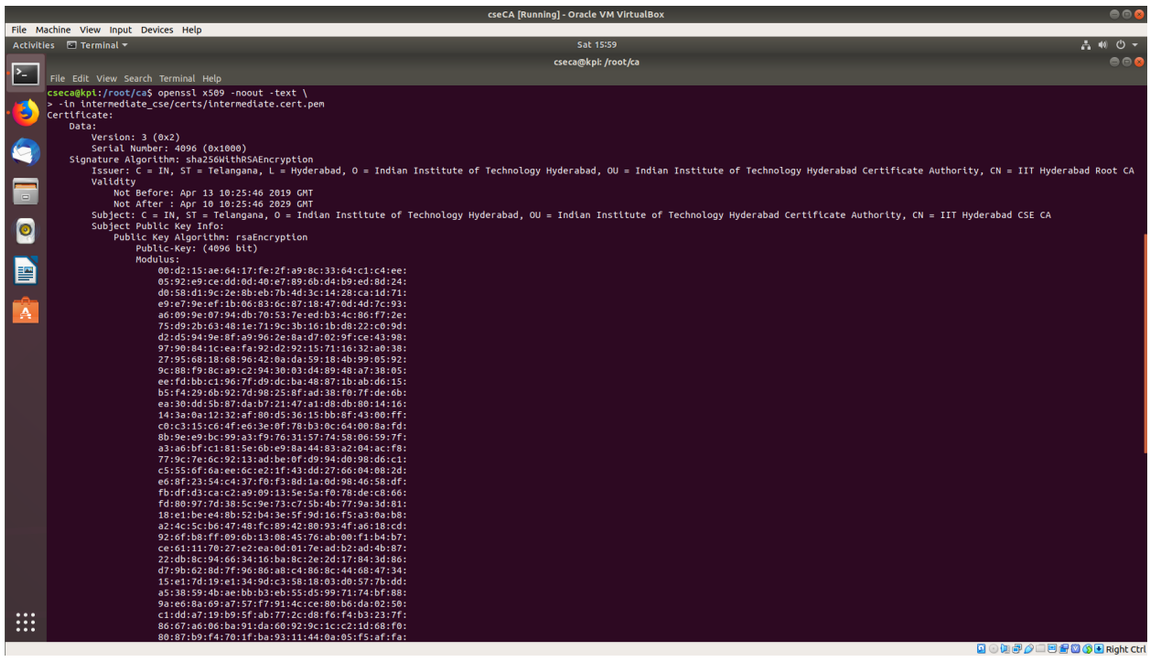

Figure 7: Intermediate CSE CA - Content of Intermediate CSE Certificate.

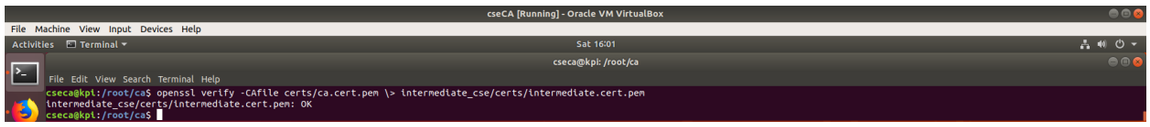

Figure 8: Intermediate CSE CA - Verifying the Intermediate CSE Certificate against Root Certificate.

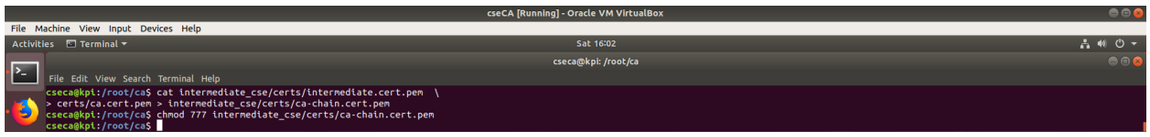

Figure 9: Intermediate CSE CA - Creating the Chain of Trust.

1.2.6 Creating the Certificate Chain

Whenever an application verifies a certificate signed by the Intermediate CA, it also verifies the Intermediate Certificate against the Root Certificate. For that, we need to complete the chain of trust by concatenating the Intermediate and Root Certificates together.

```
$ cat intermediate_cse/certs/intermediate.cert.pem \
    certs/ca.cert.pem > intermediate_cse/certs/ca-chain.cert.pem
$ chmod 777 intermediate_cse/certs/ca-chain.cert.pem
```

1.3 Creating the EE Intermediate CA

In this section, we will be creating the intermediate pair for EE which consists of the EE Intermediate Key and Certificate. This pair will form the identity of our Intermediate CA for EE Department. Follow the below steps in order to create the Intermediate EE CA of your Public Key Infrastructure,

1.3.1 Preparing the Directory

This directory will store all the keys and certificates for the Root CA and the Intermediate CAs. ***index.txt*** and ***serial*** will act as a flat file database to keep track of signed certificates. And ***csr*** will be used to hold the certificate signing requests and ***crlnumber*** to keep track of certificate revocation lists.

```
$ mkdir /root/ca/intermediate_ee
$ cd /root/ca/intermediate_ee
```

```
$ mkdir certs crl csr newcerts private
$ chmod 777 private
$ touch index.txt
$ echo 1000 \> serial
$ echo 1000 \> crlnumber
```

1.3.2 Preparing the Configuration file

We will also create a configuration file for OpenSSL to use. If the OpenSSL configuration file is not specified, OpenSSL uses the default configuration file situated at ***/etc/ssl/openssl.cnf***.

The OpenSSL configuration file for Intermediate CA would be same as for Root CA except that the ***directory*** and ***policy*** would be different in ***[CA_default]***. The policy for Root CA was ***strict_policy*** whereas the policy for Intermediate CA would be ***loose_policy***.

```
$ gedit openssl.conf
```

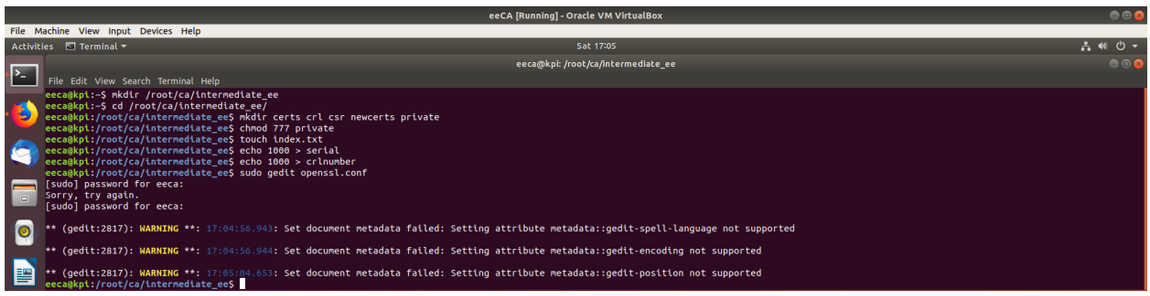

Figure 10: Intermediate EE CA - Preparing the Directory & OpenSSL Configuration file.

1.3.3 Creating the EE Intermediate Key

We will then create the Intermediate EE Key which will be used to sign the Intermediate EE Certificate. We will be using AES 256 bit encryption to create the Intermediate EE Key and store it in ***/root/ca/intermediate_ee/private***.

```
$ cd /root/ca
$ openssl genrsa -aes256 -out intermediate_ee/private/intermediate.key.pem 4096
$ chmod 777 intermediate_ee/private/intermediate.key.pem
```

1.3.4 Creating the EE Intermediate Certificate

We will use the Intermediate EE Key to sign and create the Certificate Signing Request (CSR). The details of the CSR should generally match with the Root CA. The ***Common Name***, however, must be different.

```
$ cd /root/ca
$ openssl req -config intermediate_ee/openssl.conf -new sha256 \
    -key intermediate_ee/private/intermediate.key.pem \
    -out intermediate_ee/csr/intermediate.csr.pem
```

To create the intermediate certificate, we will use the Root CA to sign the Intermediate CSR.

```
$ cd /root/ca
$ openssl ca -config openssl.conf -extensions v3_intermediate_ca \
    -days 3650 -notext -md sha256 \
    -in intermediate_ee/csr/intermediate.csr.pem \
    -out intermediate_ee/certs/intermediate.cert.pem
$ chmod 777 intermediate_ee/certs/intermediate.cert.pem
```

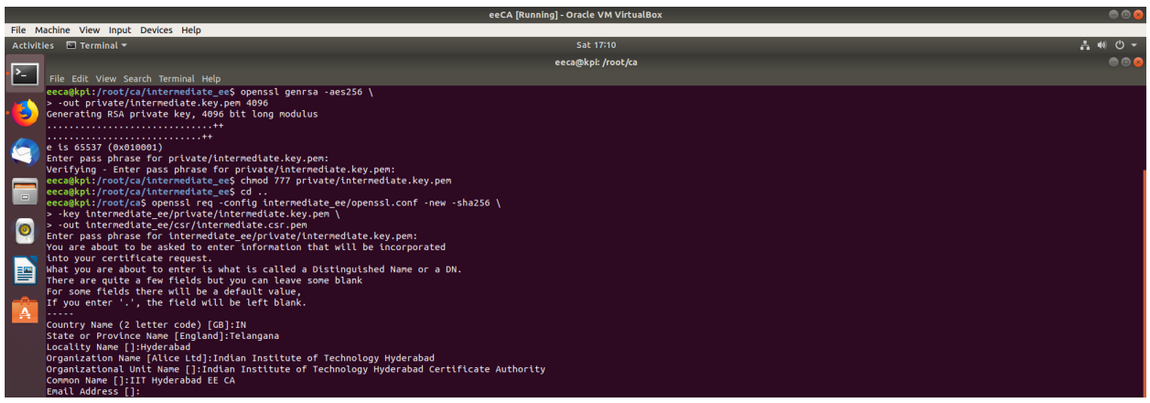

Figure 11: Intermediate EE CA - Creating the Intermediate EE Key & Certificate Signing Request.

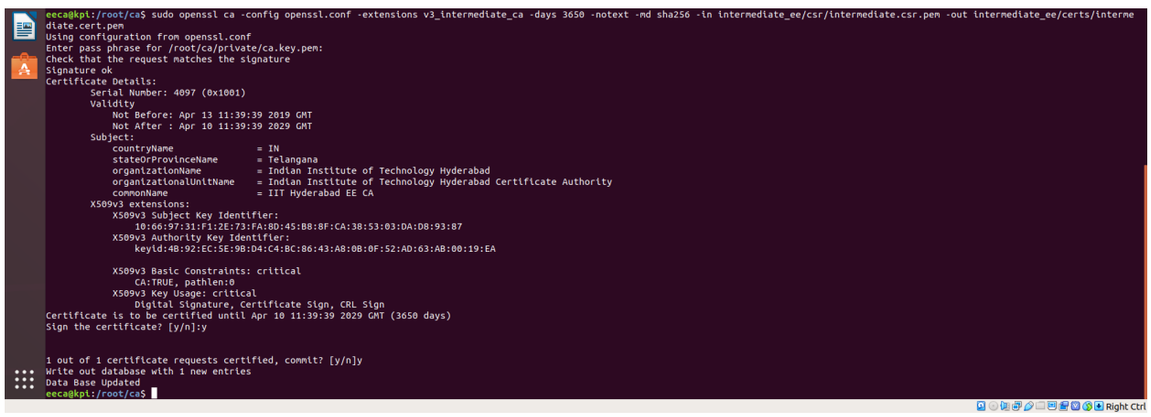

Figure 12: Intermediate EE CA - Creating the Intermediate EE Certificate.

1.3.5 Verifying the Intermediate EE Certificate

We can verify the contents of the Intermediate EE Certificate by using the below command.

```
$ openssl x509 -noout -text -in intermediate_ee/certs/intermediate.cert.pem
```

We will also verify the Intermediate EE Certificate against that of the Root Certificate. An ***OK*** will indicate that the chain of trust is intact.

```
$ Copy the Root Certificate \& Intermediate CSE Cert using SCP
$ openssl verify -CAfile certs/ca.cert.pem\
    intermediate_ee/certs/intermediate.cert.pem
```

1.3.6 Creating the Certificate Chain

Whenever an application verifies a certificate signed by the Intermediate CA, it also verifies the Intermediate Certificate against the Root Certificate. For that, we need to complete the chain of trust by concatenating the Intermediate and Root Certificates together.

```
$ cat intermediate_ee/certs/intermediate.cert.pem \
    certs/ca.cert.pem > intermediate_ee/certs/ca-chain.cert.pem
$ chmod 777 intermediate_ee/certs/ca-chain.cert.pem
```

1.4 Creating the ISAC Intermediate CA

In this section, we will be creating the intermediate pair for ISAC which consists of the ISAC Intermediate Key and Certificate. This pair will form the identity of our Intermediate CA for ISAC Department. Follow the below steps in order to create the Intermediate ISAC CA of your Public Key Infrastructure,

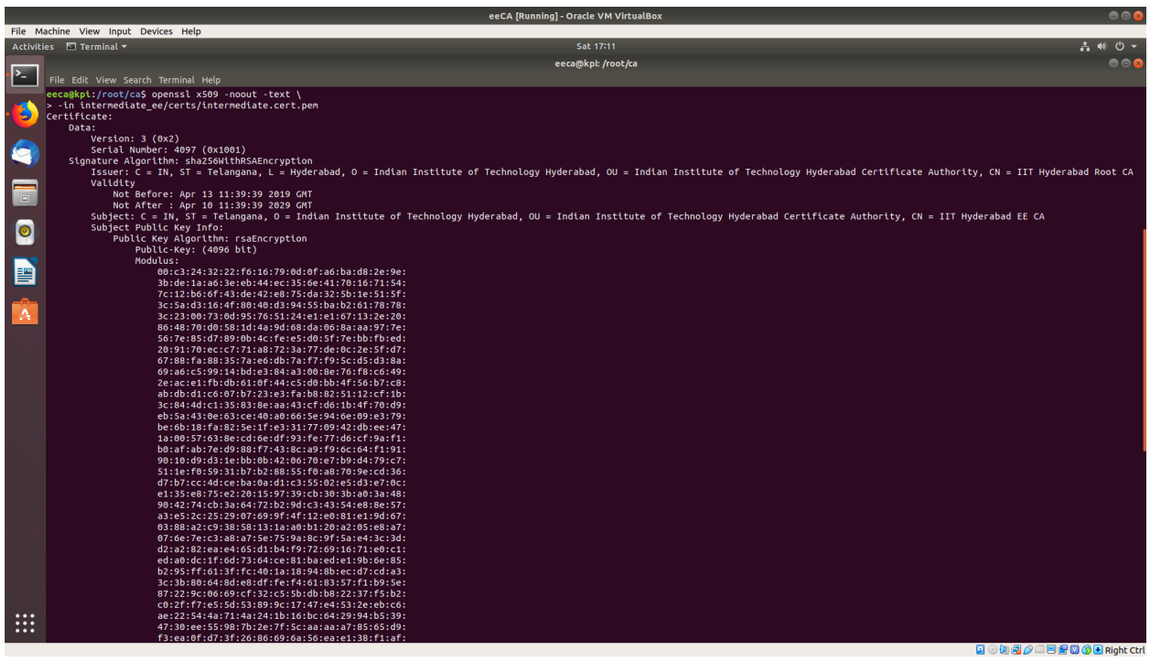

Figure 13: Intermediate EE CA - Content of Intermediate EE Certificate.

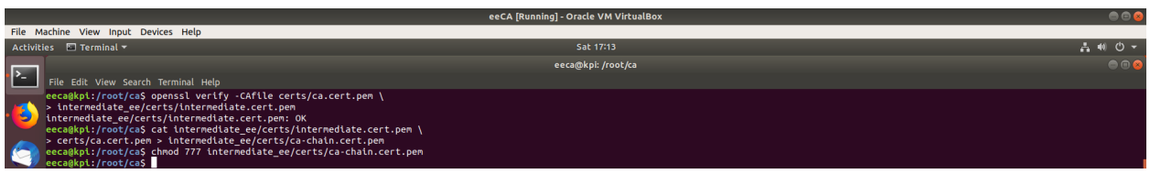

Figure 14: Intermediate EE CA - Verifying the Intermediate EE Certificate against Root Certificate.

1.4.1 Preparing the Directory

This directory will store all the keys and certificates for the Root CA and the Intermediate CAs. ***index.txt*** and ***serial*** will act as a flat file database to keep track of signed certificates. And ***csr*** will be used to hold the certificate signing requests and ***crlnumber*** to keep track of certificate revocation lists.

```
$ mkdir /root/ca/intermediate_cse
$ cd /root/ca/intermediate_cse
$ mkdir certs crl csr newcerts private
$ chmod 777 private
$ touch index.txt
$ echo 1000 \> serial
$ echo 1000 \> crlnumber
```

1.4.2 Preparing the Configuration file

We will also create a configuration file for OpenSSL to use. If the OpenSSL configuration file is not specified, OpenSSL uses the default configuration file situated at ***/etc/ssl/openssl.cnf***.

The OpenSSL configuration file for Intermediate CA would be same as for Root CA except that the ***directory*** and ***policy*** would be different in ***[CA_default]***. The policy for Root CA was ***strict_policy*** whereas the policy for Intermediate CA would be ***loose_policy***.

```
$ gedit openssl.conf
```

1.4.3 Creating the ISAC Intermediate Key

We will then create the Intermediate ISAC Key which will be used to sign the Intermediate ISAC Certificate. We will be using AES 256 bit encryption to create the Intermediate ISAC Key and store it in ***/root/ca/intermediate_isac/private***.

Figure 15: Intermediate ISAC CA - Preparing the Directory & OpenSSL Configuration file.

```
$ cd /root/ca
$ openssl genrsa -aes256 -out intermediate_isac/private/intermediate.key.pem 4096
$ chmod 777 intermediate_isac/private/intermediate.key.pem
```

1.4.4 Creating the ISAC Intermediate Certificate

We will use the Intermediate ISAC Key to sign and create the Certificate Signing Request (CSR). The details of the CSR should generally match with the Root CA. The ***Common Name***, however, must be different.

```
$ cd /root/ca
$ openssl req -config intermediate_isac/openssl.conf -new sha256 \
    -key intermediate_isac/private/intermediate.key.pem \
    -out intermediate_isac/csr/intermediate.csr.pem
```

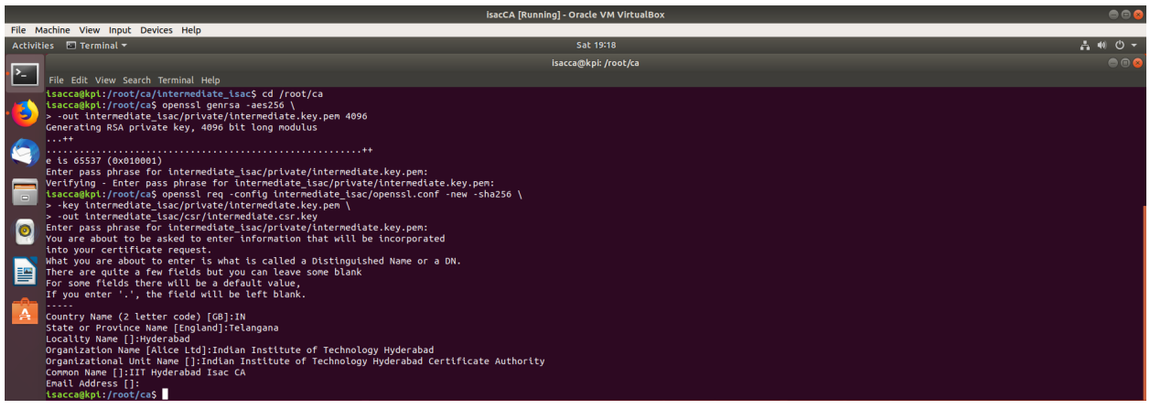

Figure 16: Intermediate ISAC CA - Creating the Intermediate ISAC Key & Certificate Signing Request.

To create the intermediate certificate, we will use the Root CA to sign the Intermediate CSR.

```
$ cd /root/ca
$ openssl ca -config openssl.conf -extensions v3_intermediate_ca \
    -days 3650 -notext -md sha256 \
    -in intermediate_isac/csr/intermediate.csr.pem \
    -out intermediate_isac/certs/intermediate.cert.pem
$ chmod 777 intermediate_isac/certs/intermediate.cert.pem
```

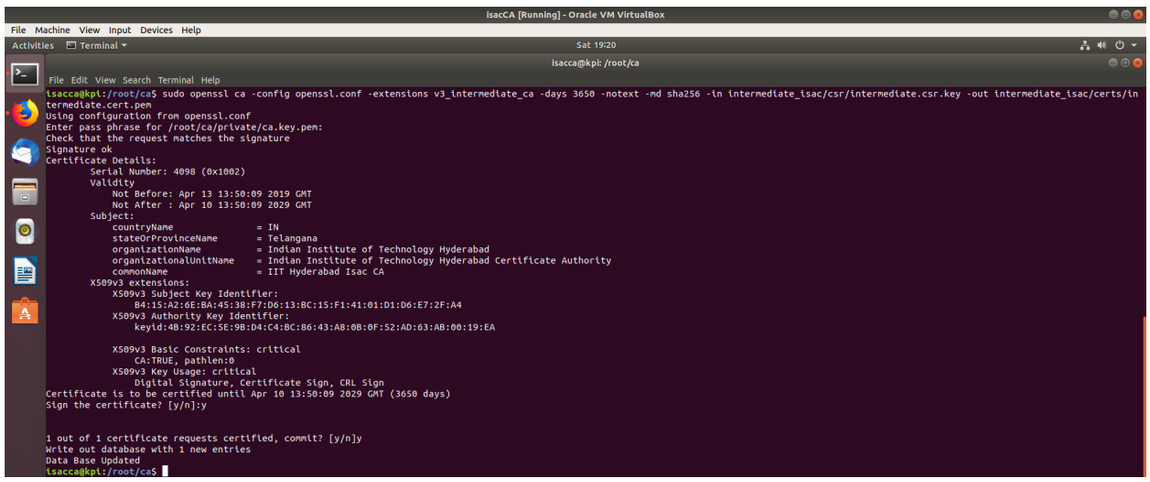

Figure 17: Intermediate ISAC CA - Creating the Intermediate ISAC Certificate.

1.4.5 Verifying the Intermediate ISAC Certificate

We can verify the contents of the Intermediate ISAC Certificate by using the below command.

```
$ openssl x509 -noout -text -in intermediate_isac/certs/intermediate.cert.pem
```

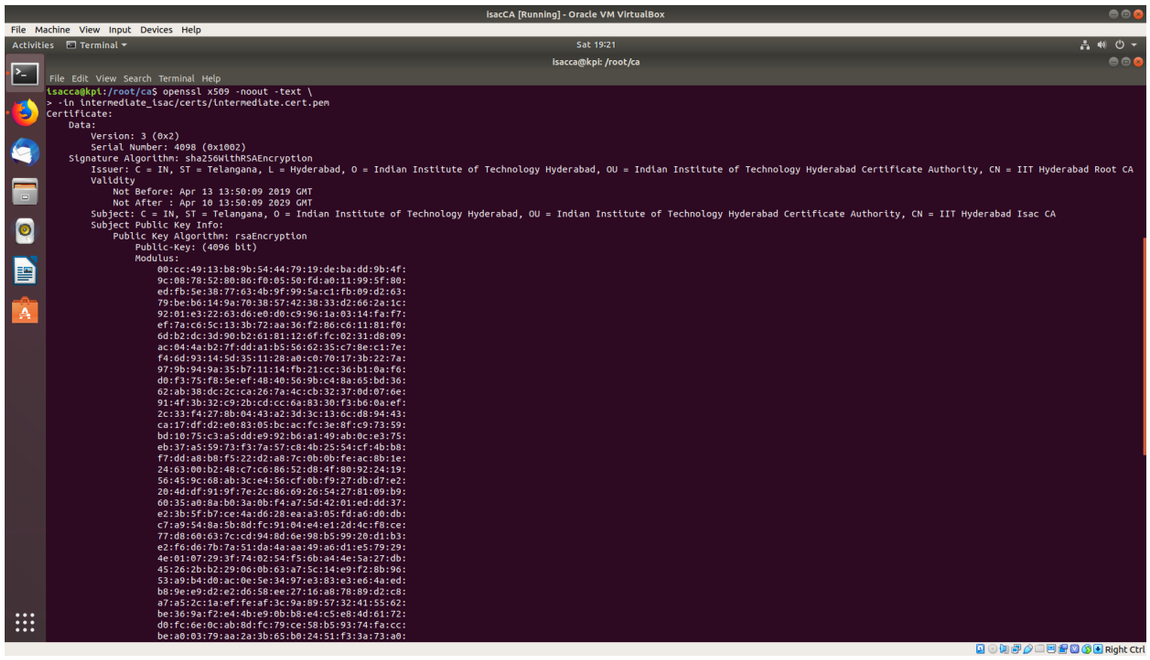

Figure 18: Intermediate ISAC CA - Content of Intermediate ISAC Certificate.

We will also verify the Intermediate ISAC Certificate against that of the Root Certificate. An ***OK*** will indicate that the chain of trust is intact.

```
$ Copy the Root Certificate \& Intermediate CSE Cert using SCP
$ openssl verify -CAfile certs/ca.cert.pem\
    intermediate_isac/certs/intermediate.cert.pem
```

1.4.6 Creating the Certificate Chain

Whenever an application verifies a certificate signed by the Intermediate CA, it also verifies the Intermediate Certificate against the Root Certificate. For that, we need to complete the chain of trust by concatenating the Intermediate and Root Certificates together.

```
$ cat intermediate_isac/certs/intermediate.cert.pem \
    certs/ca.cert.pem > intermediate_isac/certs/ca-chain.cert.pem
$ chmod 777 intermediate_isac/certs/ca-chain.cert.pem
```

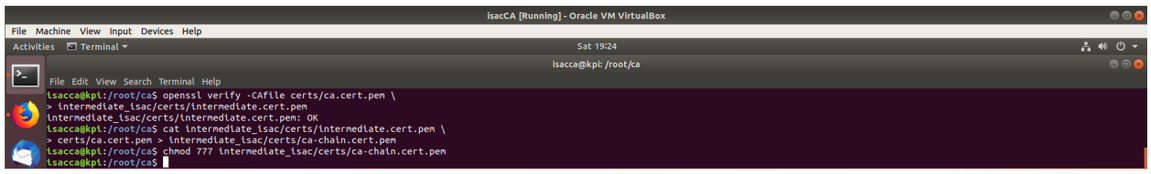

Figure 19: Intermediate ISAC CA - Verifying the Intermediate ISAC Certificate against Root Certificate.

2. Creating a Webserver with HTTPS Support

Problem Statement: The web server inside a particular department creates their own certificate and get it signed by the department's intermediate CA. The web server should allow only HTTPS connections to it. Configure the web server accordingly and configure the browser to trust the root CA that you have created in Task 1. Host a web service that collects user profile for registration for a particular service. The user suppose to fill a form with some personal information. You need to show using Wireshark[3] that the connection between the client and server happens over HTTPS. You can either use a public cloud service such as Google Cloud or VMs on your own laptop/desktop.

2.1 Configuring Browser to Trust Root CA

For proper functioning of Server Certificates signed by Root/Intermediate Certificate Authority, we need to configure our browser to trust Root CA Certificate.

```
$ Open Firefox and go to "Preferences -> Privacy and Security"
$ At the end of the page, click on "View Certificates" button.
$ Click on the "Authorities" tab.
$ Click on the "Import" button.
$ Select the downloaded certificate.
$ Check "Trust this CA to identify websites".
$ Click OK.
$ Restart Firefox.
```

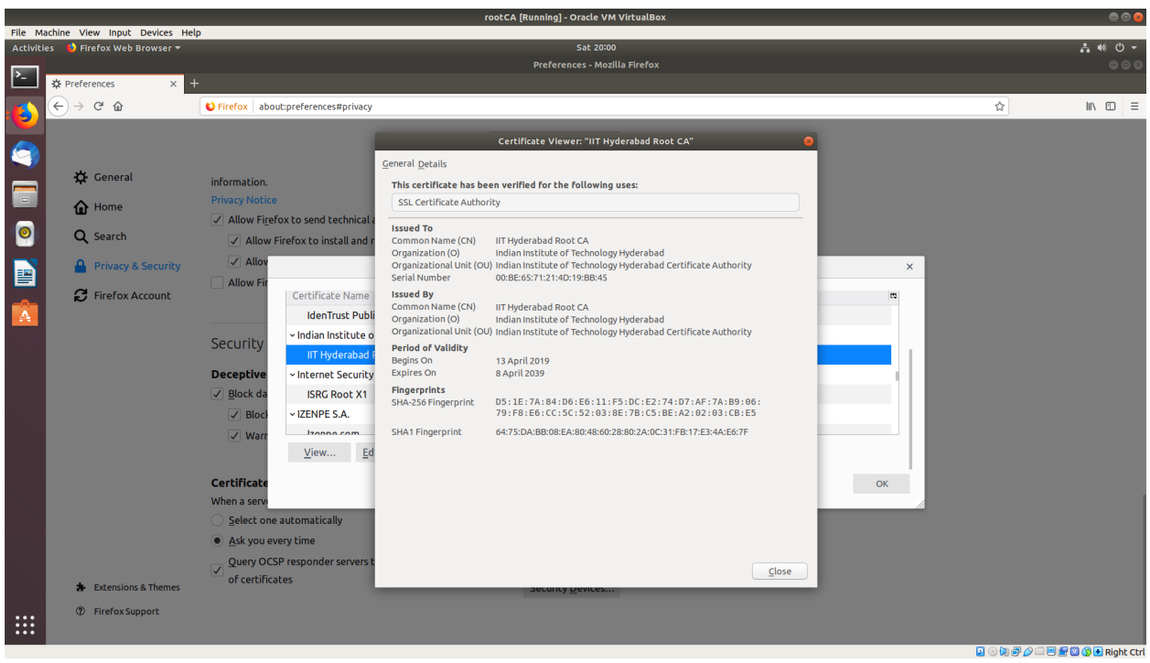

Figure 20: Configuring Browser to Trust Root Certificate.

2.2 Creating the CSE Webserver

In this section we will be creating a Server Certificate signed by the Intermediate CSE CA and set up the Apache2 Webserver to only allow HTTPS connection to it. We will also be showing

the same i.e connection over HTTPS using Wireshark.

2.2.1 Creating the CSE Server Key & Certificate

We will create the CSE Server Key which will be used to sign the CSE Server Certificate. We will be using AES 256 bit encryption to create the same.

```
$ openssl genrsa -aes256 \
    -out intermediate_cse/private/www.cse.iith.com.key.pem 2048
$ chmod 777 intermediate_cse/private/www.cse.iith.com.key.pem
```

The generated Server Key will be used to generate the Certificate Signing Request (CSR). The common name for the server certificate must be a ***Fully Qualified DOmain Name (FQDN)***.

```
$ openssl req -config intermediate_cse/openssl.conf \
    -key intermediate_cse/private/www.cse.iith.com.key.pem \
    -new -sha256 -out intermediate_cse/csr/www.cse.iith.com.csr.pem
```

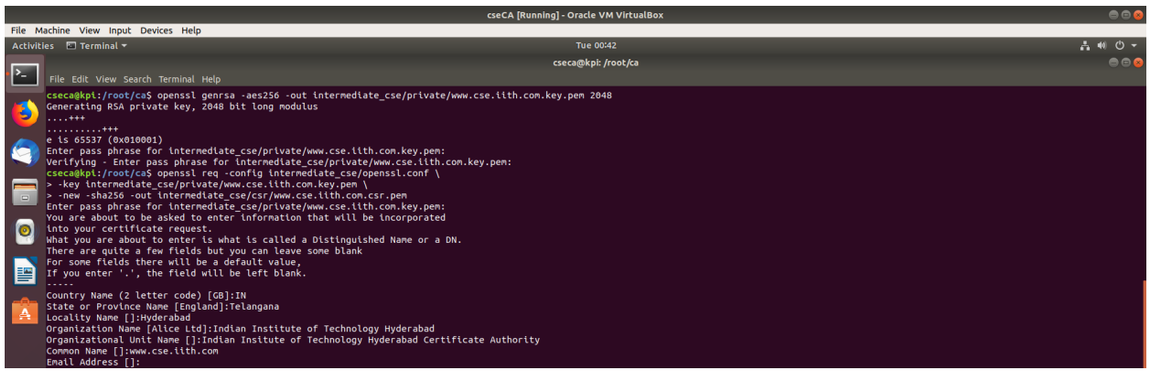

Figure 21: CSE Server Certificate - Creating the CSE Server Key & CSR.

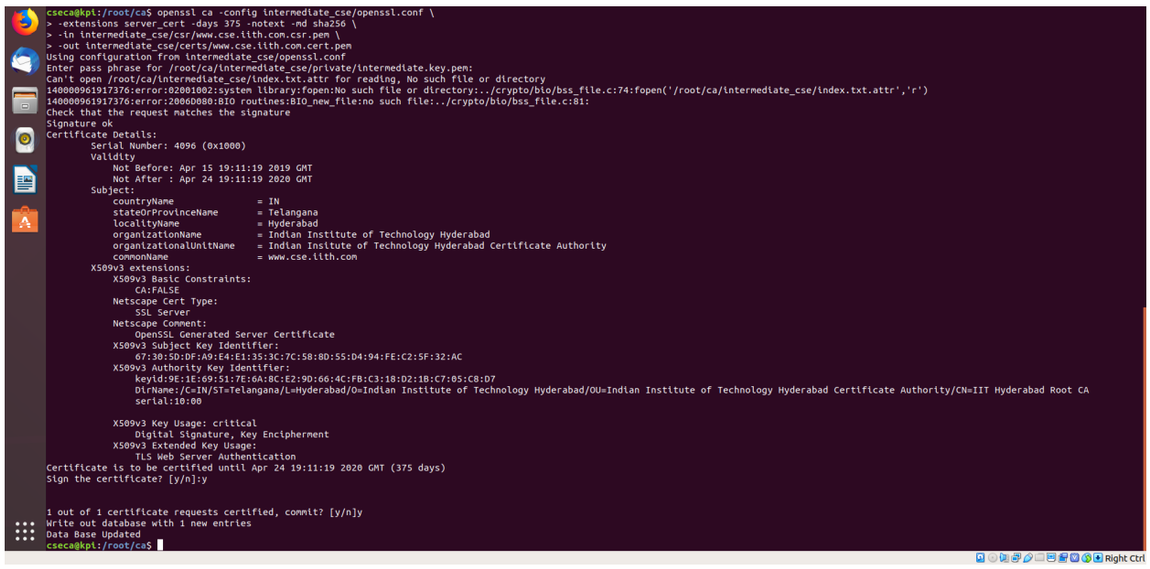

Figure 22: CSE Server Certificate - Creating the CSE Server Certificate.

This CSR will be signed by the CSE Intermediate CA in order to generate the CSE Server Certificate.

```
$ openssl ca -config intermediate_cse/openssl.conf \
    -extensions server_cert -days 375 -notext -md -sha256 \
    -in intermediate_cse/csr/www.cse.iith.com.csr.pem \
    -out intermediate_cse/certs/www.cse.iith.com.cert.pem
$ chmod 777 intermediate_cse/certs/www.cse.iith.com.cert.pem
```

The content of the Server Certificate can be viewed by using the below command. Here, the Issuer will be the CSE Intermediate CA and the Subject will refer to the Server Certificate itself.

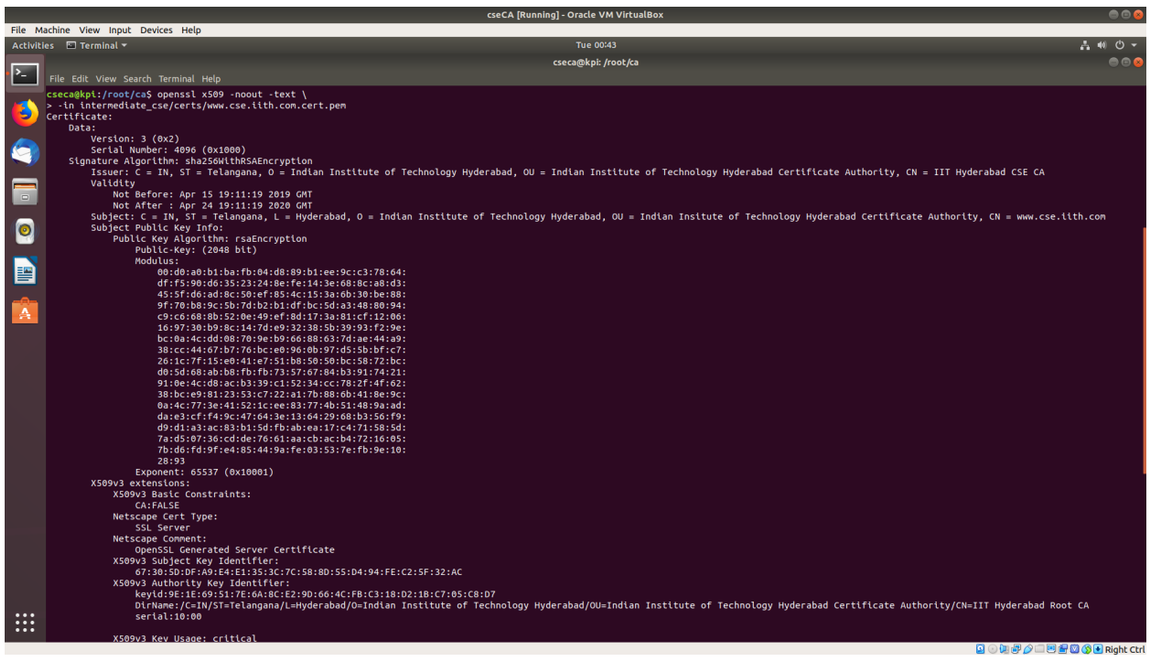

Figure 23: CSE Server Certificate - Content of the Server Certificate.

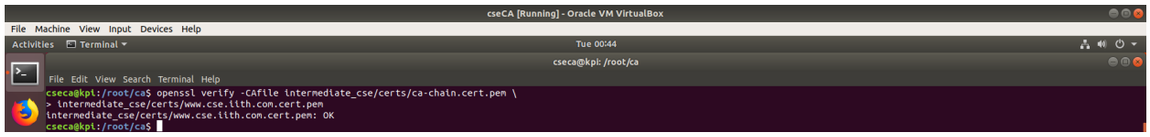

Figure 24: CSE Server Certificate - Verifying the Server Certificate against the Certificate Chain.

```
$ openssl x509 -noout -text \
    -in intermediate_cse/certs/www.cse.iith.com.cert.pem
```

We can also use the CA certificate chain to verify the Server Certificate, whether it has a valid chain of trust or not. An ***OK*** indicates that it has a valid chain of trust.

```
$ openssl verify -CAfile intermediate_cse/certs/ca-chain.cert.pem \
    intermediate_cse/certs/www.cse.iith.com.cert.pem
```

We can use the below mentioned files to configure the Apache Webserver and provide HTTPS connection to users.

```
$ ca-chain.cert.pem
$ www.cse.iith.com.key.pem
$ www.cse.iith.com.cert.pem
```

2.2.2 Configuring the Apache Webserver

We can configure the Apache Webserver to provide HTTPS connection to it by using the CSE Server Key, Certificate and the Chain Certificate of the CSE Intermediate CA.

```
$ sudo a2enmod ssl
$ sudo service apache2 reload
$ sudo service apache2 restart
$ gedit /etc/apache2/sites-available/default-ssl.conf
    # Add ServerName www.cse.iith.com below <VirtualHost_default_:443>
    # Look for SSLCertificateFile and change its path to
        /root/ca/intermediate_cse/certs/www.cse.iith.com.cert.pem
    # Look for SSLCertificateKeyFile and change its path to
        /root/ca/intermediate_cse/private/www.cse.iith.com.key.pem
    # Add SSLCertificateChainFile /root/ca/intermediate_cse/certs/ca-chain.cert.pem
        below SSLCertificateKeyFile
$ sudo a2ensite default-ssl.conf
$ gedit /etc/hosts
```

```
    # Add 127.0.0.1 www.cse.iith.com
$ gedit /etc/apache2/sites-available/000-default.conf
    # Add ServerName www.cse.iith.com below ServerAdmin
    # Add Redirect / https://www.cse.iith.com below ServerName
$ sudo service apache2 reload
$ sudo service apache2 restart
```

We also created a Webpage consisting of a registration form and replaced it with the contents of ***/var/www/html***, We can now observe the webpage over HTTPS connection in the browser by going to ***https://www.cse.iith.com***.

We can verify that the communication with the CSE Webserver is over HTTPS connection by taking Wireshark trace for the same. Below we can see that the communication is HTTP over TLS.

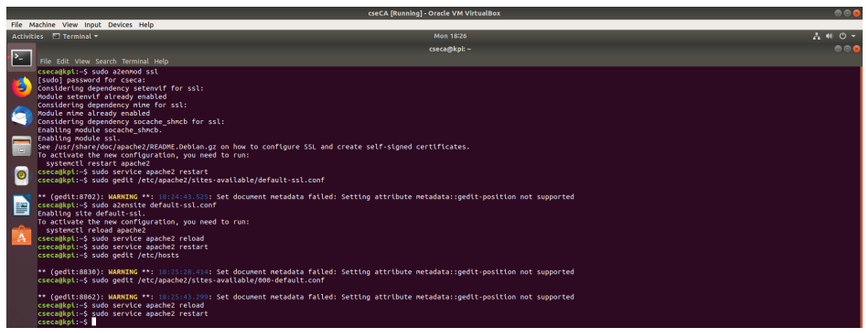

Figure 25: Configuring the Apache Webserver for CSE Webserver.

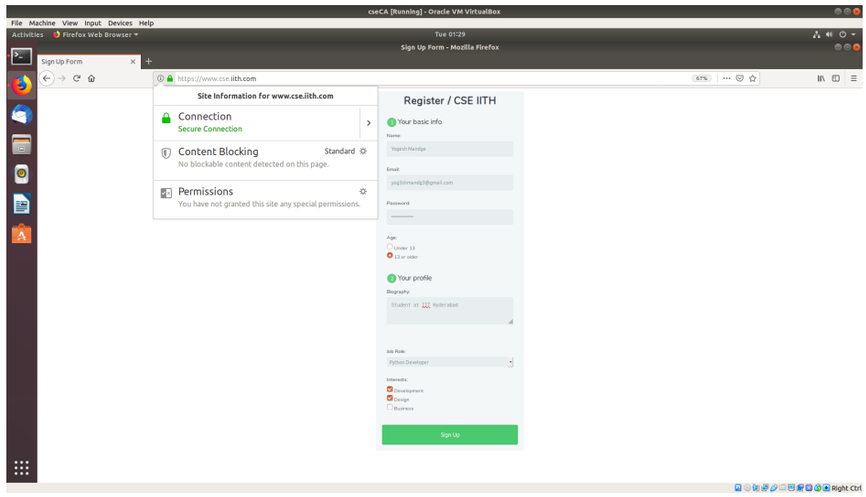

Figure 26: Registration page of CSE Webserver over HTTPS connection.

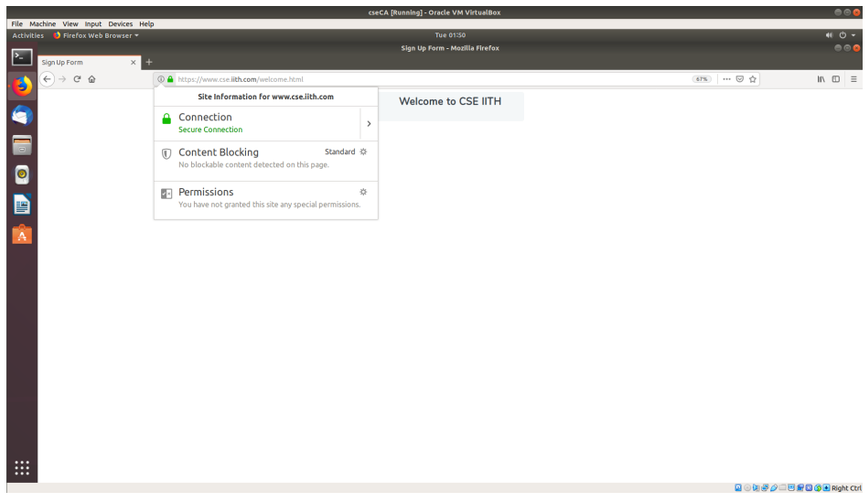

Figure 27: Welcome page of CSE Webserver over HTTPS connection.

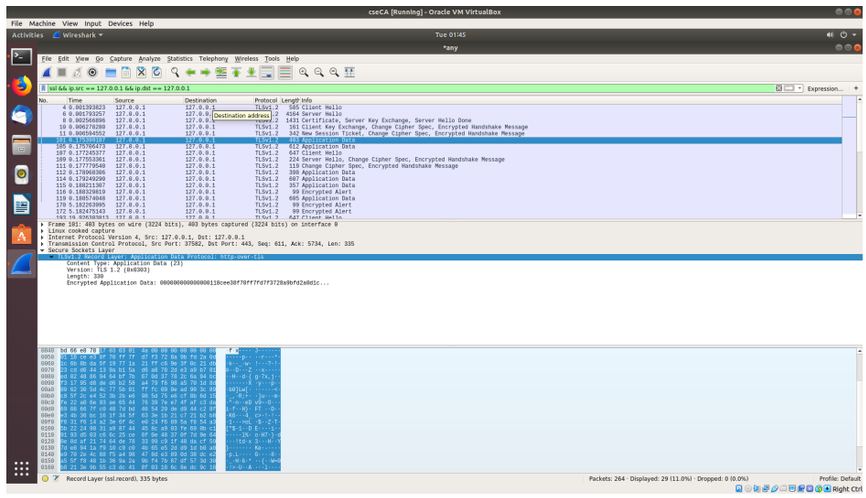

Figure 28: Wireshark trace of the communication between CSE Webserver

2.2 Creating the EE & ISAC Webserver

We also created the Server Key, CSR and Server Certificate for EE & ISAC using their respective EE Intermediate CA and ISAC Intermediate CA. We configured the Webserver for both of them using the same steps as mentioned in the ***2.1 Creating the CSE Webserver***. We can confirm the HTTPS enabled Webserver for EE using the below figures,

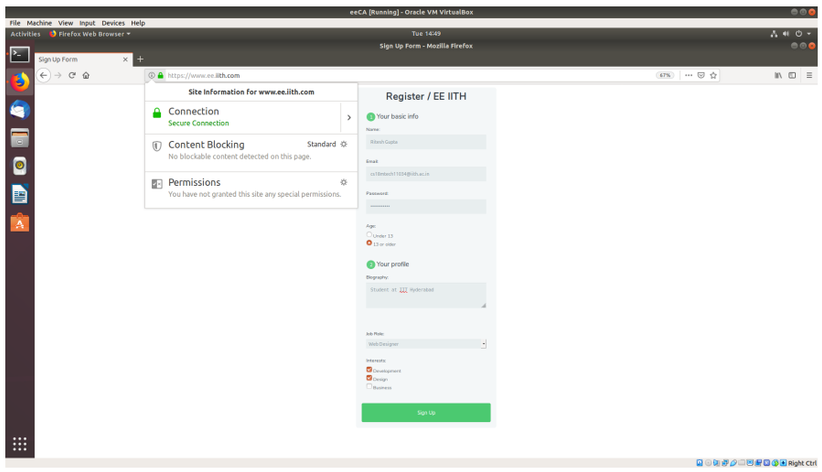

Figure 29: Registration page of EE Webserver over HTTPS connection.

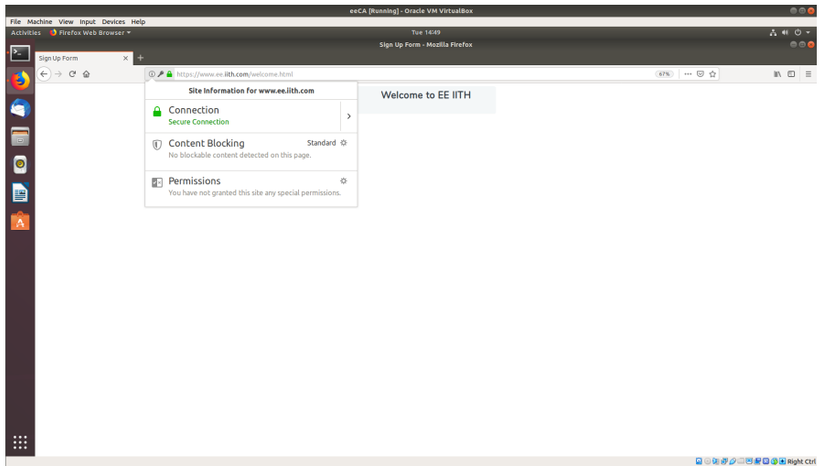

Figure 30: Welcome page of EE Webserver over HTTPS connection.

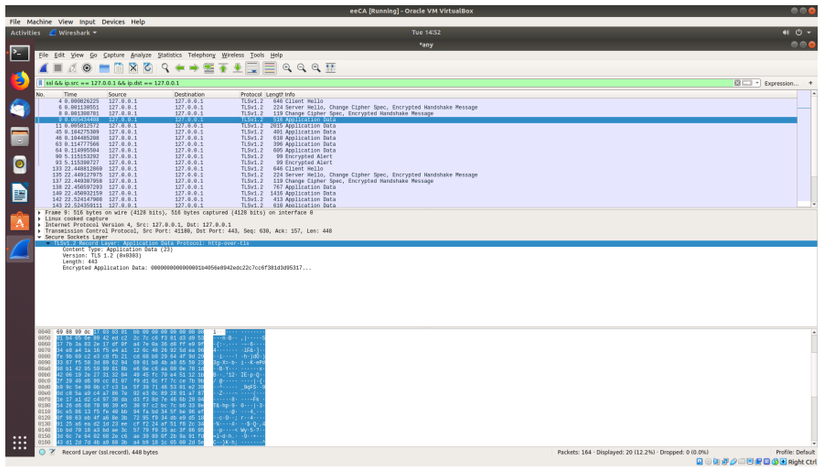

Figure 31: Wireshark trace of the communication between EE Webserver

We can also confirm the HTTPS enabled Webserver for ISAC using the below figures,

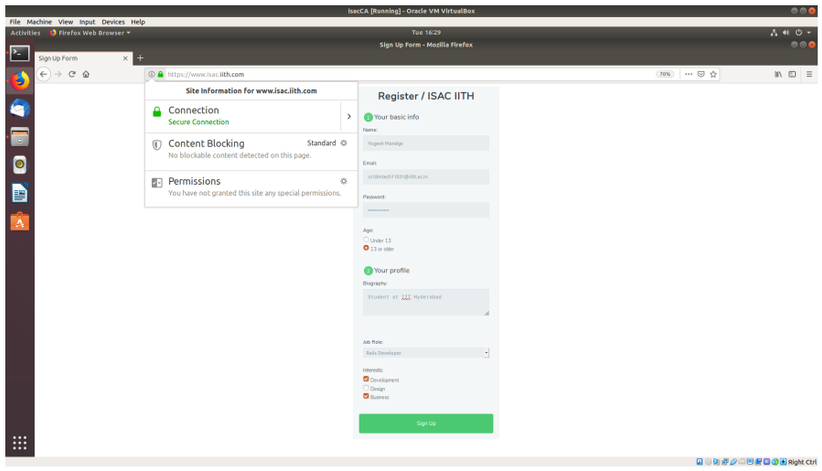

Figure 32: Registration page of ISAC Webserver over HTTPS connection.

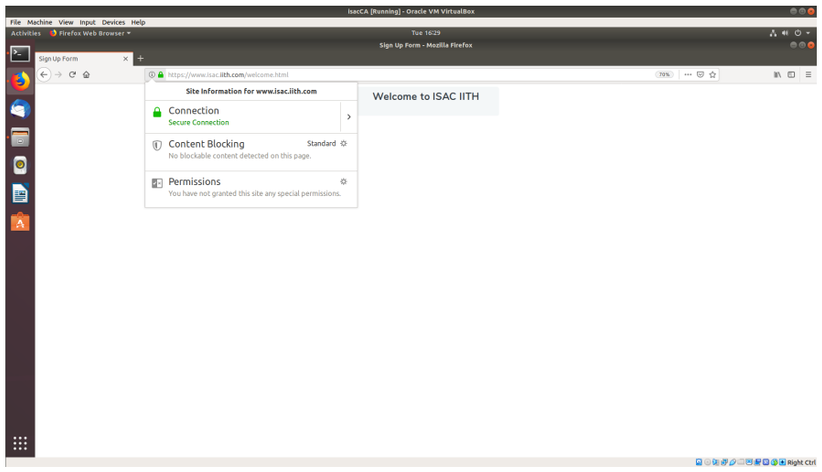

Figure 33: Welcome page of ISAC Webserver over HTTPS connection.

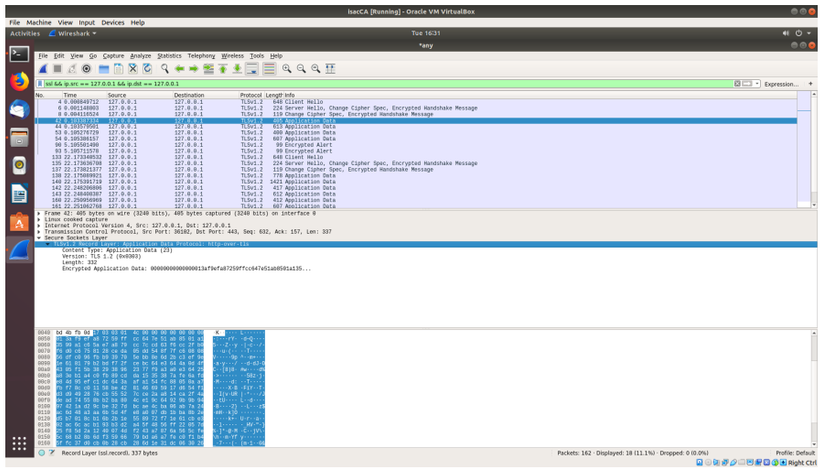

Figure 34: Wireshark trace of the communication between ISAC Webserver

3. Secure Peer-to-Peer application using PKI

Problem Statement: Let us assume that the IITH community wants to create a secure application (e.g., chat application/ gaming/ file transfer/ etc.) using the PKI of IITH in Task 1. You can pick up any application of your choice and code it in C/C++ by using OpenSSL or GnuTLS library. You need to use the PKI of IITH to get the user certificates and use SSL based communication between the users. Create your application using socket programming using SSL socket and show that the communication between the users is secure by using either TLS 1.2 or 1.3. You get bonus marks if you show performance improvement in TLS 1.3 compared to TLS 1.2 for your application.

3.1 Peer-to-peer Chat Application

We created a Peer-to-peer chat application using the OpenSSL library, for turning a socket into SSL socket and our Public key infrastructure of IITH for encryption.

The application requires three parameters passed as command line arguments, <hostname> & <port_server> which is the IP address and Port address on which the server method of P2P application will create the socket, & <port_client> which is required by the client method of P2P application to connect to the server. We can compile and execute the P2P chat application using the below commands,

```
$ gcc -Wall -o peerA peer.c -L/usr/lib -lssl -crypto -lpthread
$ gcc -Wall -o peerB peer.c -L/usr/lib -lssl -crypto -lpthread
$ ./peerA 127.0.0.1 9999 9998
$ ./peerB 127.0.0.1 9998 9999
```

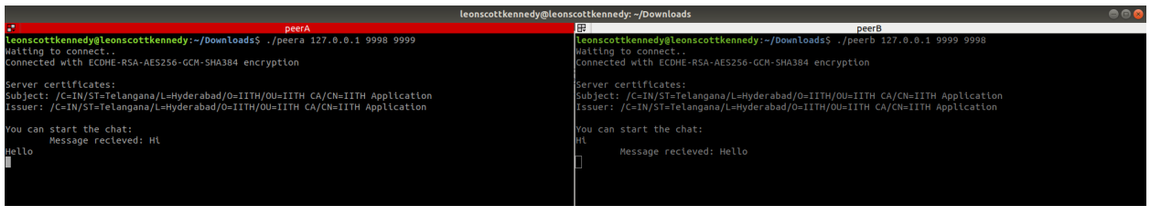

Figure 35: Working of P2P Chat Application.

We also collected the packet trace of our P2P chat application, we can see that the application used TLSv1.2 protocol for packet transfer.

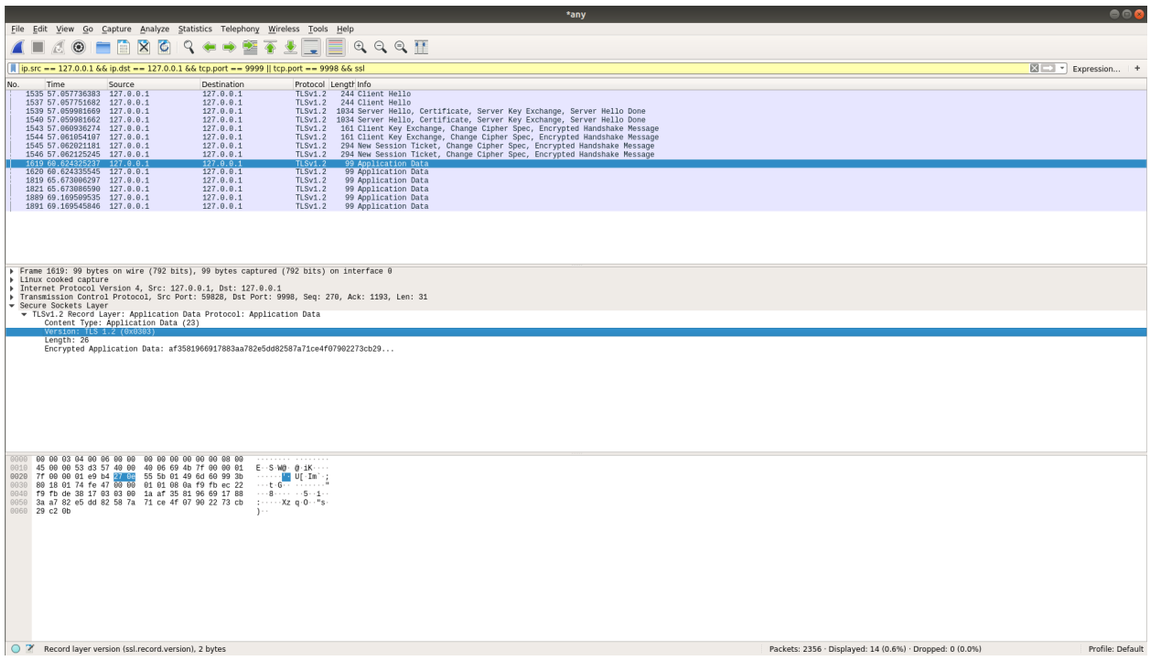

Figure 36: Wireshark trace of the P2P Chat Application.

4. References

[1] **OpenSSL** - Used for creating and verifying OpenSSL certificates.
[2] **Apache** - Used for providing Webserver capabilites.
[3] **Wireshark** - Used for capturing packet traces of Webserver requests & P2P application.
[4] **scp** - Used for secure copy-paste of OpenSSL certificates among different VMs.
[5] **VirtualBox** - Used for hosting and maintaining different Virtual machines.

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 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 honour violations by other students if we become aware of it.

Names of group members: Yogesh Mandge, Ritesh Gupta
Date: April 23, 2019
Signature: YM, RG