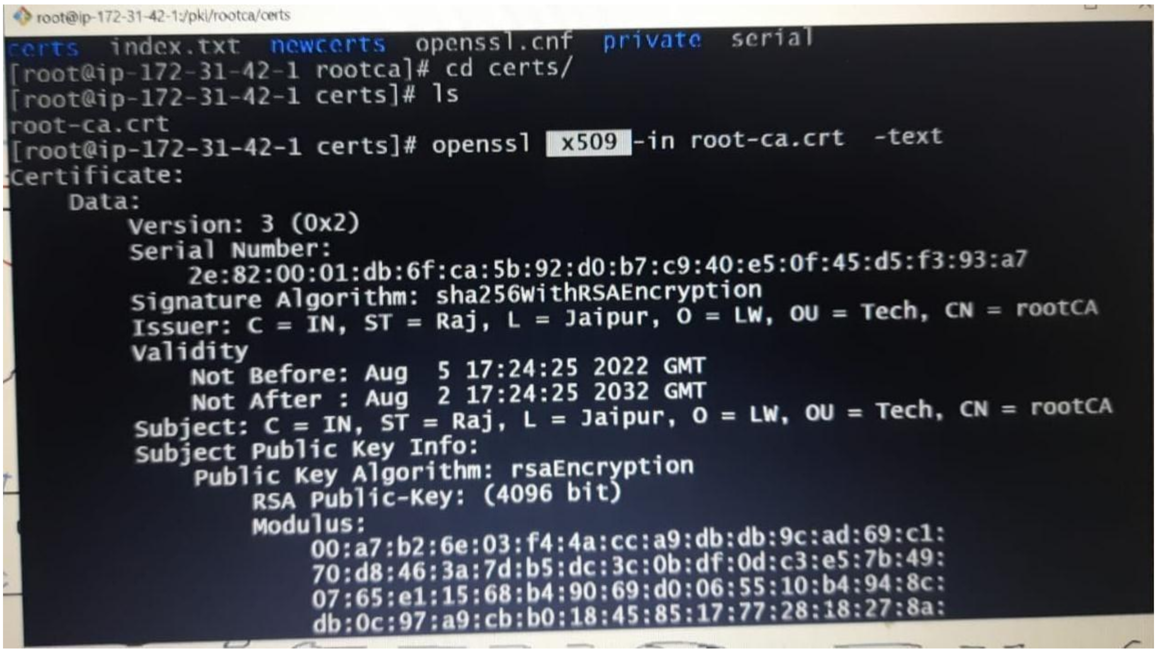
[Cryptography]

Cryptography Session No.11
Summary 12-08-2022

Detailed Discussion on the below points –

- Launch an Apache Webserver on Linux instances
- Implement SSL/TLS client
- The webserver consists of a private key and CSR
- CSR of the web server consists of company information and public key
- SSL / TLS handshaking – when a client hits the Webserver – the server certificate is sent to the client - client checks the signature of the certificate
- But the only challenge here is for the client to verify the signature of the CA, it requires the CA public key
- The client believes the root CA – but there are a limited root CA
- So we can create sub-CA – but it should be authorized by root CA
- Concept of self-signing of root CA – build a root CA

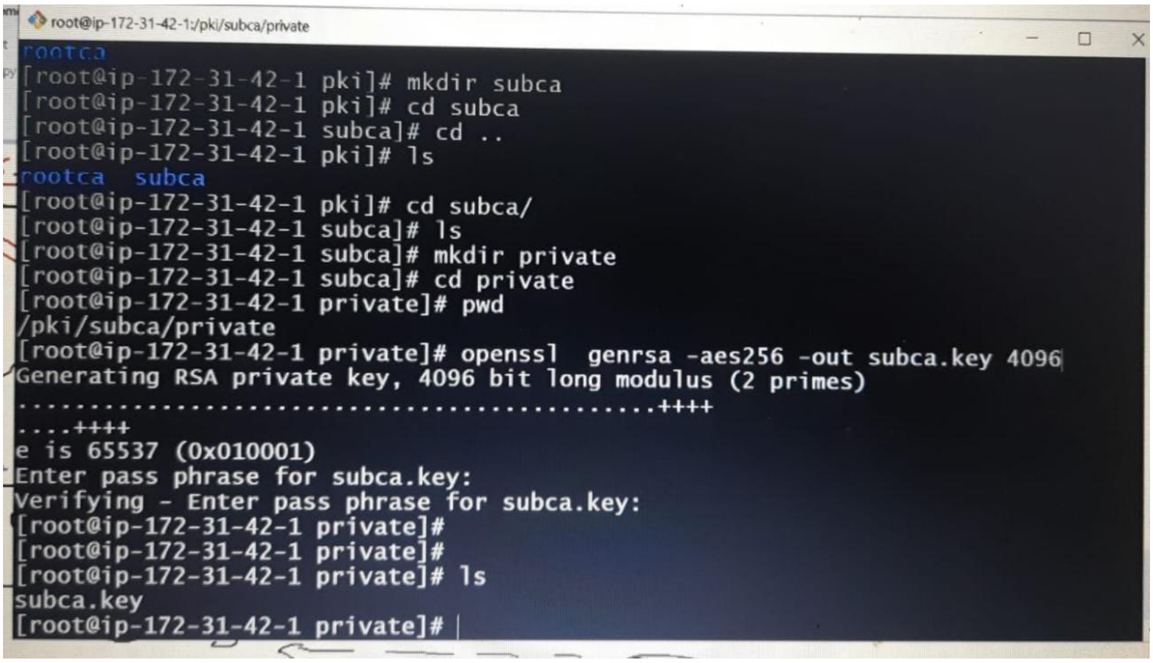


root@ip-172-31-42-1:/pki/rootca/certs

```
certs  index.txt  newcerts  openssl.cnf  private  serial
[root@ip-172-31-42-1 rootca]# cd certs/
[root@ip-172-31-42-1 certs]# ls
root-ca.crt
[root@ip-172-31-42-1 certs]# openssl  x509 -in root-ca.crt  -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2e:82:00:01:db:6f:ca:5b:92:d0:b7:c9:40:e5:0f:45:d5:f3:93:a7
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = IN, ST = Raj, L = Jaipur, O = LW, OU = Tech, CN = rootCA
        Validity
            Not Before: Aug  5 17:24:25 2022 GMT
            Not After : Aug  2 17:24:25 2032 GMT
        Subject: C = IN, ST = Raj, L = Jaipur, O = LW, OU = Tech, CN = rootCA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:a7:b2:6e:03:f4:4a:cc:a9:db:db:9c:ad:69:c1:
                    70:d8:46:3a:7d:b5:dc:3c:0b:df:0d:c3:e5:7b:49:
                    07:65:e1:15:68:b4:90:69:d0:06:55:10:b4:94:8c:
                    db:0c:97:a9:cb:b0:18:45:85:17:77:28:18:27:8a:
```

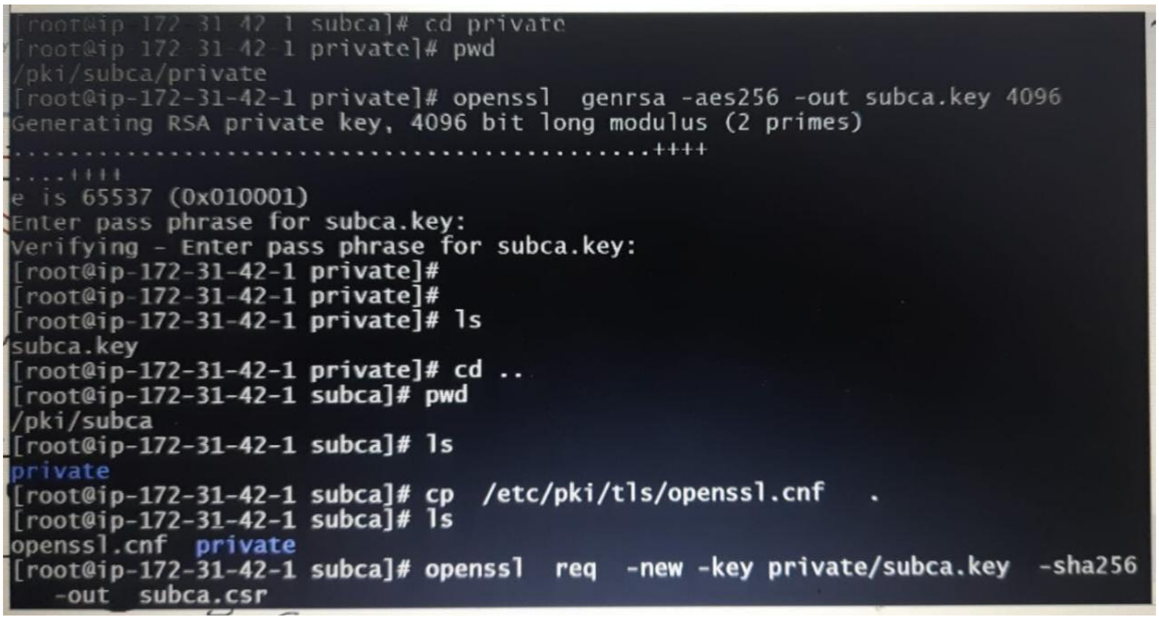
- Create sub-CA

- Create private key – unique identification of sub-CA



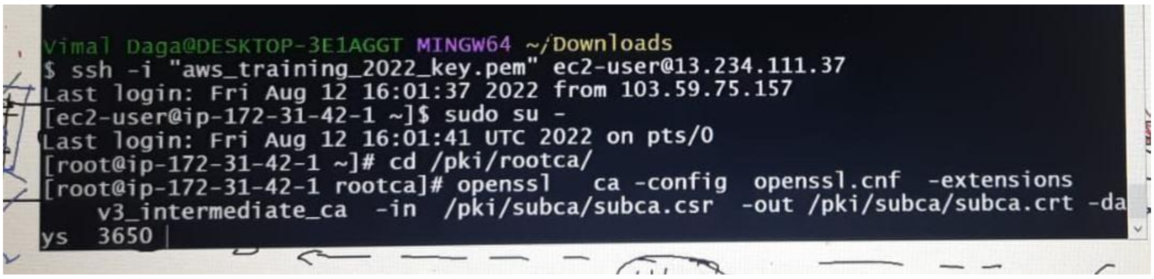
```
rootca
[root@ip-172-31-42-1 pki]# mkdir subca
[root@ip-172-31-42-1 pki]# cd subca
[root@ip-172-31-42-1 subca]# cd ..
[root@ip-172-31-42-1 pki]# ls
rootca  subca
[root@ip-172-31-42-1 pki]# cd subca/
[root@ip-172-31-42-1 subca]# ls
[root@ip-172-31-42-1 subca]# mkdir private
[root@ip-172-31-42-1 subca]# cd private
[root@ip-172-31-42-1 private]# pwd
/pki/subca/private
[root@ip-172-31-42-1 private]# openssl  genrsa -aes256 -out subca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.............................................++++
....++++
e is 65537 (0x010001)
Enter pass phrase for subca.key:
Verifying - Enter pass phrase for subca.key:
[root@ip-172-31-42-1 private]#
[root@ip-172-31-42-1 private]#
[root@ip-172-31-42-1 private]# ls
subca.key
[root@ip-172-31-42-1 private]# |
```

- Create a CSR



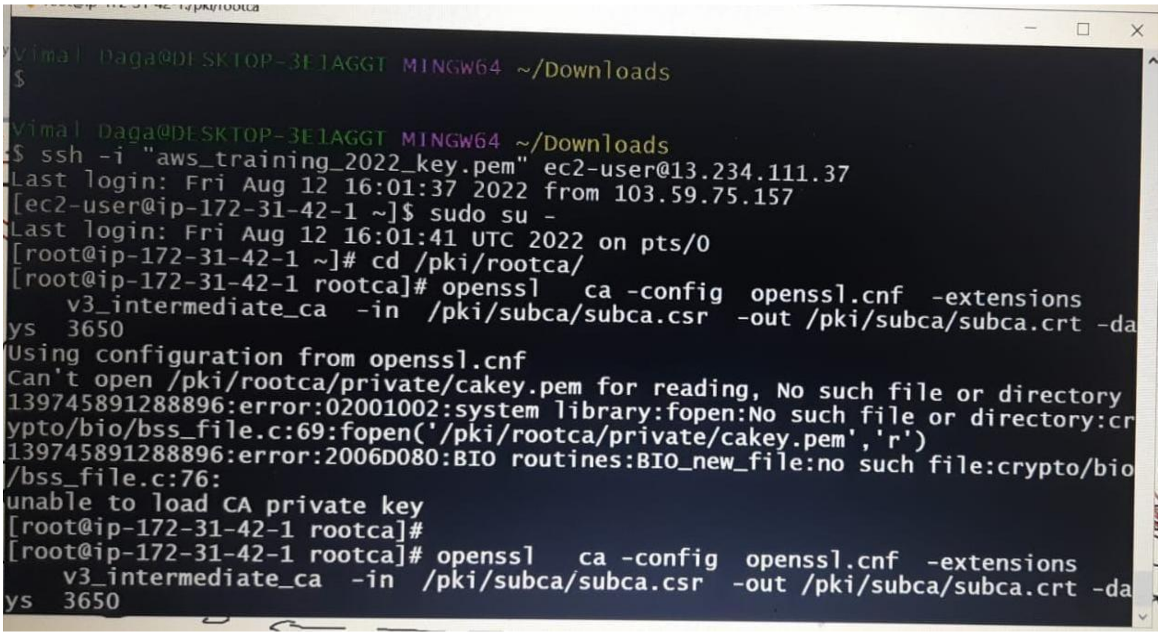
```
[root@ip-172-31-42-1 subca]# cd private
[root@ip-172-31-42-1 private]# pwd
/pki/subca/private
[root@ip-172-31-42-1 private]# openssl  genrsa -aes256 -out subca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.............................................++++
....++++
e is 65537 (0x010001)
Enter pass phrase for subca.key:
Verifying - Enter pass phrase for subca.key:
[root@ip-172-31-42-1 private]#
[root@ip-172-31-42-1 private]#
[root@ip-172-31-42-1 private]# ls
subca.key
[root@ip-172-31-42-1 private]# cd ..
[root@ip-172-31-42-1 subca]# pwd
/pki/subca
[root@ip-172-31-42-1 subca]# ls
private
[root@ip-172-31-42-1 subca]# cp  /etc/pki/tls/openssl.cnf   .
[root@ip-172-31-42-1 subca]# ls
openssl.cnf  private
[root@ip-172-31-42-1 subca]# openssl  req  -new -key private/subca.key  -sha256
   -out  subca.csr
```

- Create a CRT



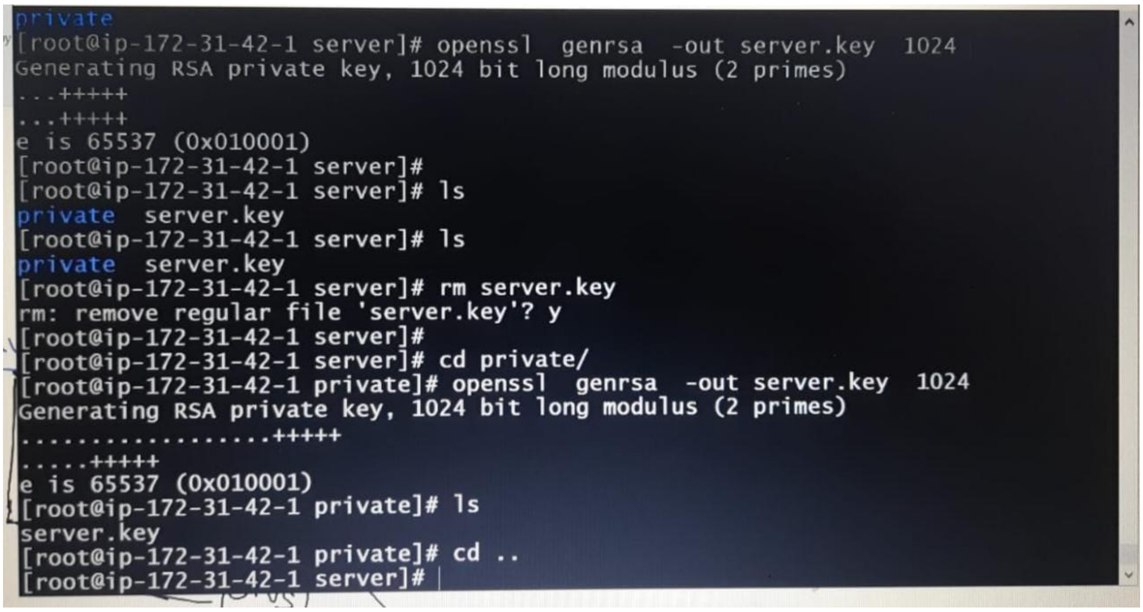
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Downloads
$ ssh -i "aws_training_2022_key.pem" ec2-user@13.234.111.37
Last login: Fri Aug 12 16:01:37 2022 from 103.59.75.157
[ec2-user@ip-172-31-42-1 ~]$ sudo su -
Last login: Fri Aug 12 16:01:41 UTC 2022 on pts/0
[root@ip-172-31-42-1 ~]# cd /pki/rootca/
[root@ip-172-31-42-1 rootca]# openssl   ca -config  openssl.cnf  -extensions
    v3_intermediate_ca  -in  /pki/subca/subca.csr  -out /pki/subca/subca.crt -da
ys  3650 |
```

- CA after signing a certificate becomes CRT
- CA serves
  - End users – for webservers
  - Sub CA- they work like root CA and further have authority to sign other CA – the chain continues
- Root CA decides how long the chain should be – that is called Path Length
- If Path Length = 0, the sub-CA servers only the webserver but cannot sign other CA
- Attributes of CA
  - CA True – sub-CA
  - CA False – server
- When root CA signs the certificate it adds the attributes in the configuration file
- We can create a customized extension in the configuration file to specify the attributes



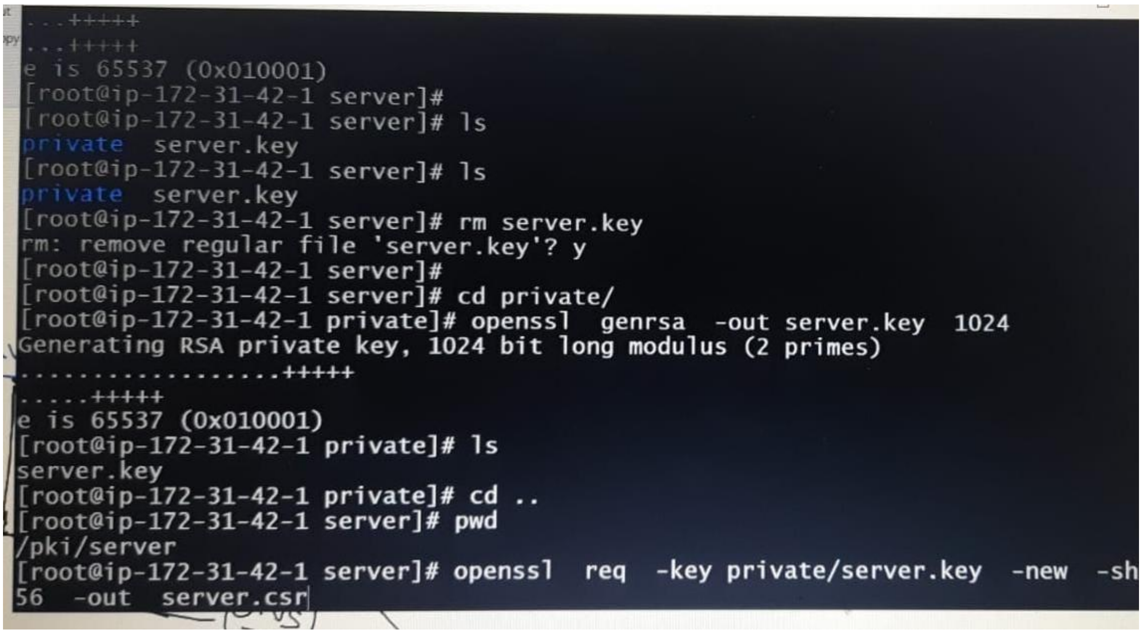
```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Downloads
$

Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Downloads
$ ssh -i "aws_training_2022_key.pem" ec2-user@13.234.111.37
Last login: Fri Aug 12 16:01:37 2022 from 103.59.75.157
[ec2-user@ip-172-31-42-1 ~]$ sudo su -
Last login: Fri Aug 12 16:01:41 UTC 2022 on pts/0
[root@ip-172-31-42-1 ~]# cd /pki/rootca/
[root@ip-172-31-42-1 rootca]# openssl   ca -config  openssl.cnf  -extensions
    v3_intermediate_ca  -in  /pki/subca/subca.csr  -out /pki/subca/subca.crt -da
ys  3650
Using configuration from openssl.cnf
Can't open /pki/rootca/private/cakey.pem for reading, No such file or directory
139745891288896:error:02001002:system library:fopen:No such file or directory:cr
ypto/bio/bss_file.c:69:fopen('/pki/rootca/private/cakey.pem','r')
139745891288896:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio
/bss_file.c:76:
unable to load CA private key
[root@ip-172-31-42-1 rootca]#
[root@ip-172-31-42-1 rootca]# openssl   ca -config  openssl.cnf  -extensions
    v3_intermediate_ca  -in  /pki/subca/subca.csr  -out /pki/subca/subca.crt -da
ys  3650
```

- Configure the webserver – first, we have to install httpd configure the server and start the service
- Identification of the certificate is by the domain name not by the IP Address
- We have to ask the client to access with domain name
- In OS the client checks the domain name in the host file
- Create a web server private key and CSR
- First, create the private key of the server

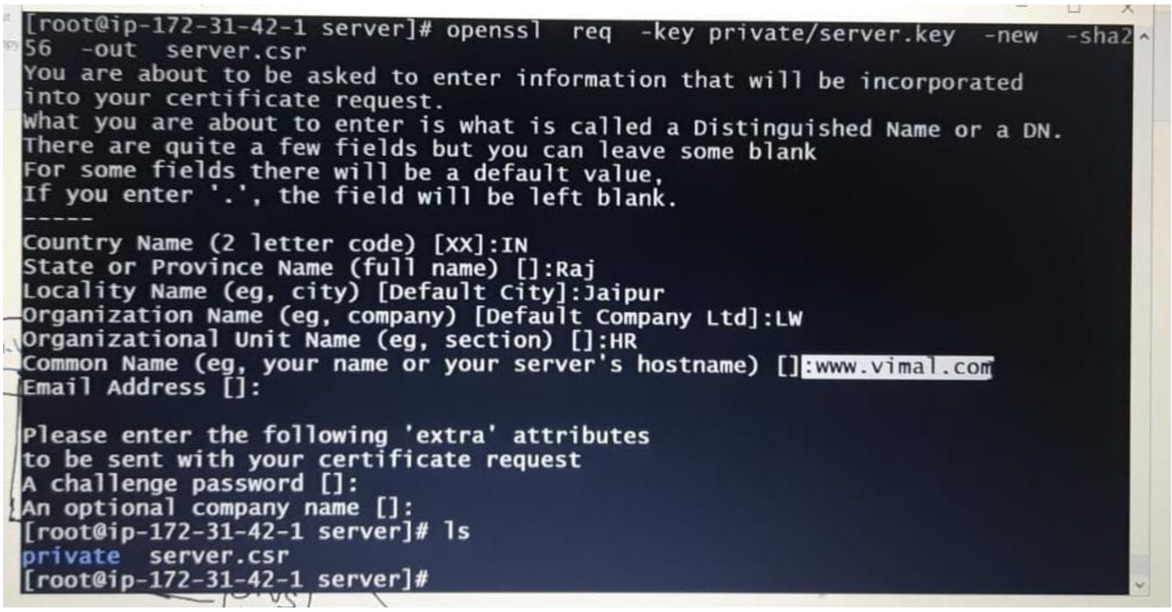


```
private
[root@ip-172-31-42-1 server]# openssl  genrsa  -out server.key  1024
Generating RSA private key, 1024 bit long modulus (2 primes)
...+++++
...+++++
e is 65537 (0x010001)
[root@ip-172-31-42-1 server]#
[root@ip-172-31-42-1 server]# ls
private  server.key
[root@ip-172-31-42-1 server]# ls
private  server.key
[root@ip-172-31-42-1 server]# rm server.key
rm: remove regular file 'server.key'? y
[root@ip-172-31-42-1 server]#
[root@ip-172-31-42-1 server]# cd private/
[root@ip-172-31-42-1 private]# openssl  genrsa  -out server.key  1024
Generating RSA private key, 1024 bit long modulus (2 primes)
..................+++++
.....+++++
e is 65537 (0x010001)
[root@ip-172-31-42-1 private]# ls
server.key
[root@ip-172-31-42-1 private]# cd ..
[root@ip-172-31-42-1 server]#
```

- Create server CSR

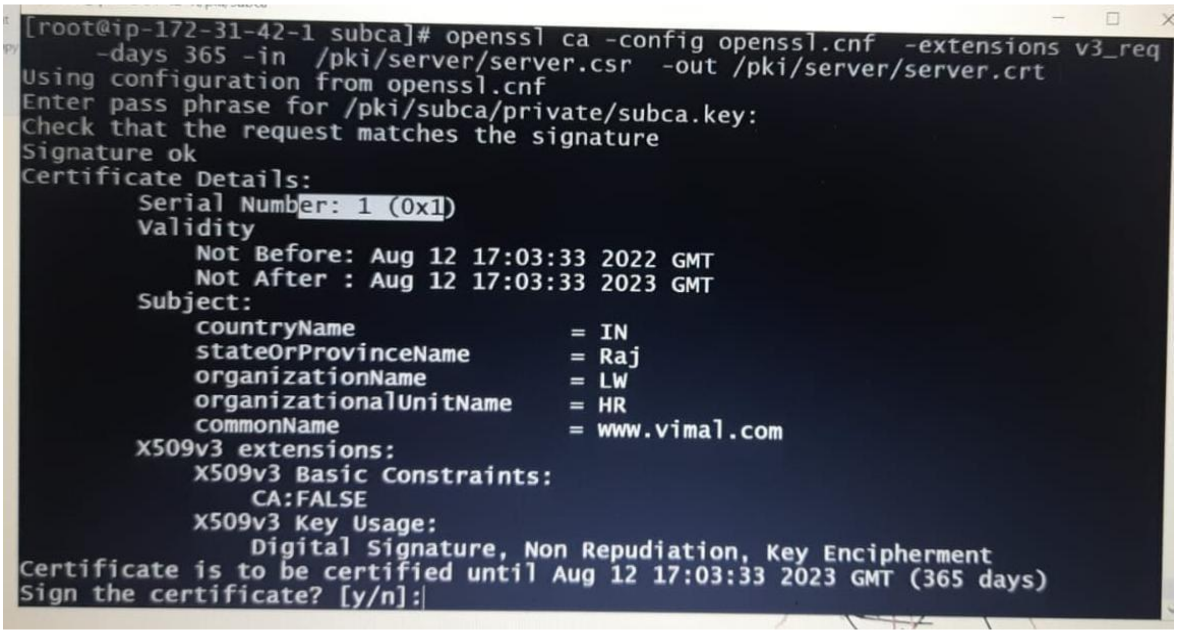


```
...+++++
...+++++
e is 65537 (0x010001)
[root@ip-172-31-42-1 server]#
[root@ip-172-31-42-1 server]# ls
private  server.key
[root@ip-172-31-42-1 server]# ls
private  server.key
[root@ip-172-31-42-1 server]# rm server.key
rm: remove regular file 'server.key'? y
[root@ip-172-31-42-1 server]#
[root@ip-172-31-42-1 server]# cd private/
[root@ip-172-31-42-1 private]# openssl  genrsa  -out server.key  1024
Generating RSA private key, 1024 bit long modulus (2 primes)
..................+++++
.....+++++
e is 65537 (0x010001)
[root@ip-172-31-42-1 private]# ls
server.key
[root@ip-172-31-42-1 private]# cd ..
[root@ip-172-31-42-1 server]# pwd
/pki/server
[root@ip-172-31-42-1 server]# openssl  req  -key private/server.key  -new  -sh
56  -out  server.csr
```



```
[root@ip-172-31-42-1 server]# openssl  req  -key private/server.key  -new  -sha2
56  -out  server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:Raj
Locality Name (eg, city) [Default City]:Jaipur
Organization Name (eg, company) [Default Company Ltd]:LW
Organizational Unit Name (eg, section) []:HR
Common Name (eg, your name or your server's hostname) []:www.vimal.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@ip-172-31-42-1 server]# ls
private  server.csr
[root@ip-172-31-42-1 server]#
```

- ➢ Create server CRT –certificate signed



```
[root@ip-172-31-42-1 subca]# openssl ca -config openssl.cnf  -extensions v3_req
    -days 365 -in  /pki/server/server.csr  -out /pki/server/server.crt
Using configuration from openssl.cnf
Enter pass phrase for /pki/subca/private/subca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Aug 12 17:03:33 2022 GMT
            Not After : Aug 12 17:03:33 2023 GMT
        Subject:
            countryName               = IN
            stateOrProvinceName       = Raj
            organizationName          = LW
            organizationalUnitName    = HR
            commonName                = www.vimal.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
Certificate is to be certified until Aug 12 17:03:33 2023 GMT (365 days)
Sign the certificate? [y/n]:
```

- ➢ Demonstration of the server-client set up – create a dummy server and from client access the server using the domain name
- ➢ A detailed description of the implementation of HTTPS – while transmitting, data is encrypted
- ➢ Brief on TLS 1.2 – how hackers can record the past packets but cannot see them – but once they get the private key they can view the packets
- ➢ Brief on TLS1.3 ECDHE protocol – every time they keep generating new keys –by this Perfect Forward Secrecy can be achieved

Important Links –

Hash13 link for Extra Sessions and session recording - https://learning.hash13.com/

Community Link to post Query, Doubts, and share your blogs - https://hash13-community.circle.so/home