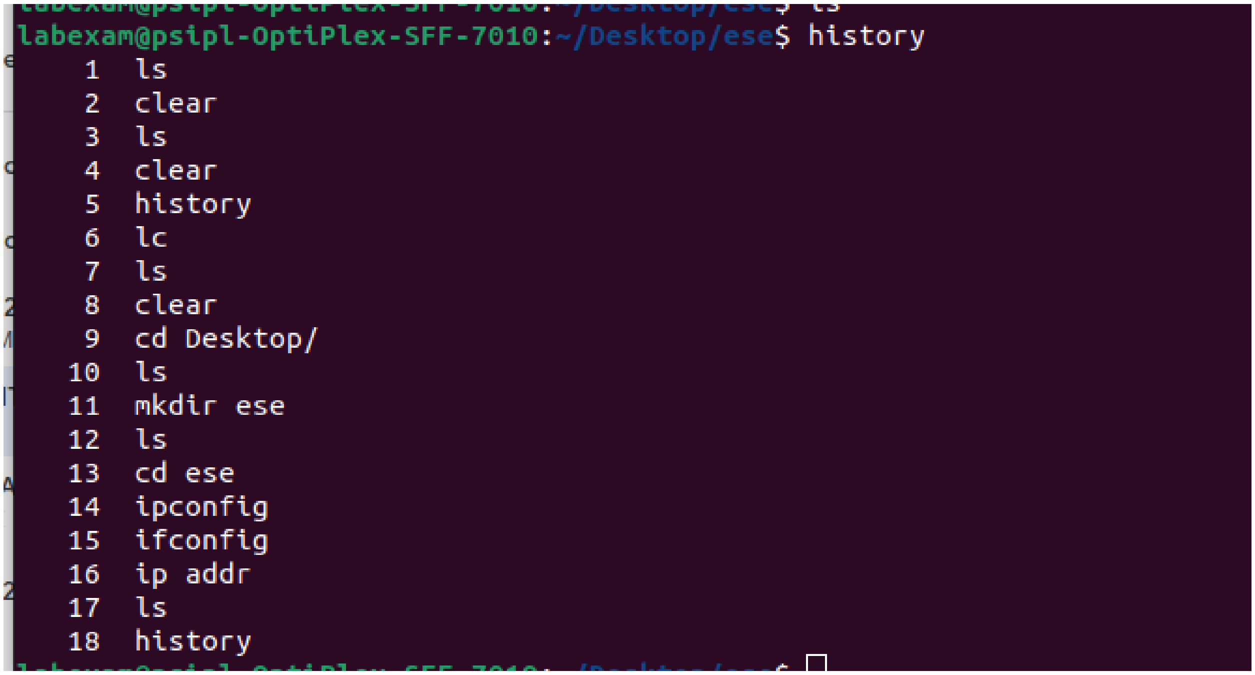
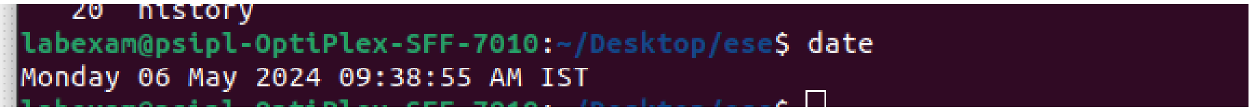
OEIT1- BCT (2023-2024)
Lab ESE

Date: 6-5-24
Student Name: Manan Hemant Chhajed
UCID: 2021600011
Branch: CS (AI - ML)
Sem: 6

Lab title: Blockchain primitives - cryptography
Brief description of each task with screenshots (caption)



```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ history
    1  ls
    2  clear
    3  ls
    4  clear
    5  history
    6  lc
    7  ls
    8  clear
    9  cd Desktop/
   10  ls
   11  mkdir ese
   12  ls
   13  cd ese
   14  ipconfig
   15  ifconfig
   16  ip addr
   17  ls
   18  history
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```



```
   20  history
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ date
Monday 06 May 2024 09:38:55 AM IST
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

Task-1:
Perform the symmetric encryption using AES-256-CBC using OpenSSL or Pyhton-Cryptography

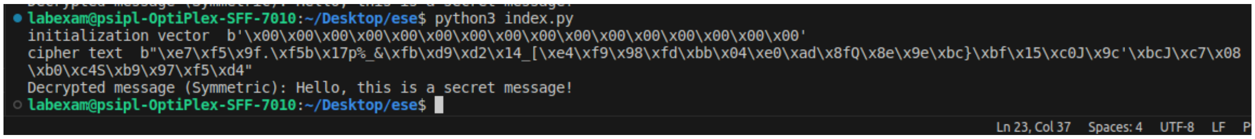
```
from cryptography.hazmat.primitives import padding, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding as
asymmetric_padding

# aes advanced encryption standard
def encrypt_symmetric(message, key):
   padder = padding.PKCS7(128).padder()
   padded_message = padder.update(message) + padder.finalize()

   iv = b'\x00' * 16

   # Create AES cipher object in CBC mode
   cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())

   # Encrypt the padded message
   encryptor = cipher.encryptor()
   ciphertext = encryptor.update(padded_message) + encryptor.finalize()

   print("initialization vector ", iv)
   print("cipher text ", ciphertext)
   return iv, ciphertext

def decrypt_symmetric(iv, ciphertext, key):
   # Create AES cipher object in CBC mode
   cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())

   decryptor = cipher.decryptor()
   decrypted_message = decryptor.update(ciphertext) + decryptor.finalize()

   unpadder = padding.PKCS7(128).unpadder()
   original_message = unpadder.update(decrypted_message) +
unpadder.finalize()

   return original_message
```

```
message = b"Hello, this is a secret message!"

key = b'\x00' * 32  # 256-bit key
iv, ciphertext = encrypt_symmetric(message, key)

decrypted_message = decrypt_symmetric(iv, ciphertext, key)
print("Decrypted message (Symmetric):", decrypted_message.decode())
```

Cipher text refers to the encrypted form of a message or data. When data is encrypted using an encryption algorithm like AES it is transformed into an unintelligible form called cipher text.

The Initialization Vector (IV) is a random value used along with a key to encrypt data securely, particularly when using CBC (Cipher Block Chaining). The IV adds randomness and ensures that the same plaintext encrypted with the same key results in different cipher texts, enhancing security.

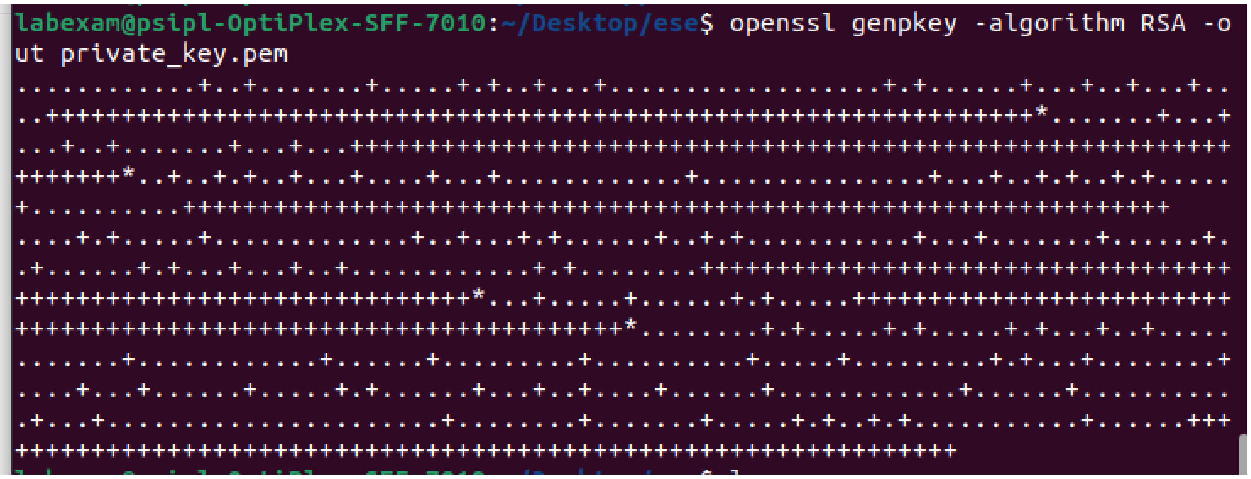
A Cipher object is created using AES (Advanced Encryption Standard) algorithm with the provided key and iv in CBC mode.



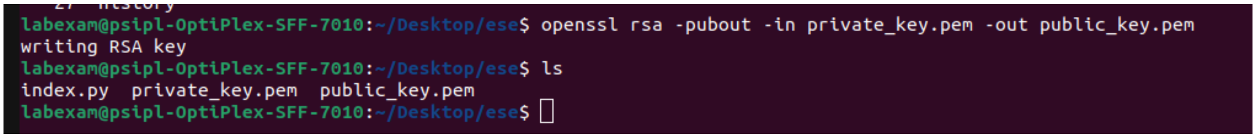
```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ python3 index.py
initialization vector  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
cipher text  b"\xe7\xf5\x9f.\xf5b\x17p%_&\xfb\xd9\xd2\x14_[\xe4\xf9\x98\xfd\xbb\x04\xe0\xad\x8fQ\x8e\x9e\xbc}\xbf\x15\xc0J\x9c'\xbcJ\xc7\x08
\xb0\xc4S\xb9\x97\xf5\xd4"
Decrypted message (Symmetric): Hello, this is a secret message!
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

Task-3:

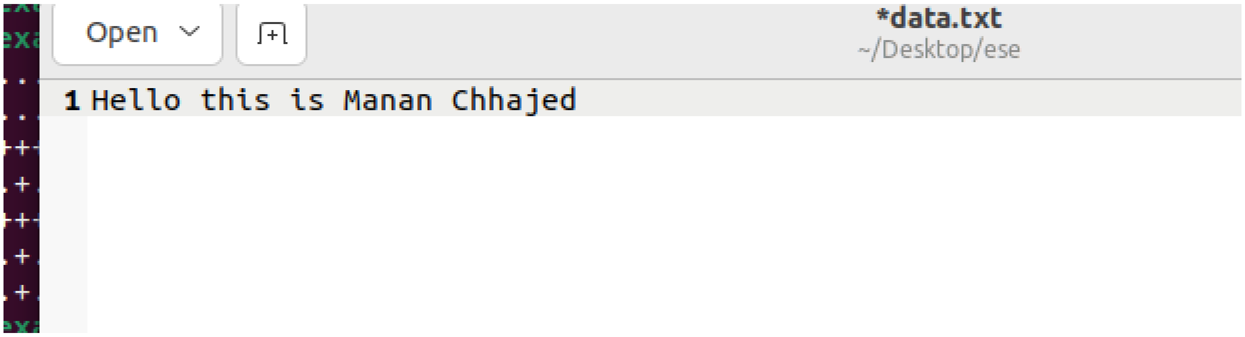
Create and verify the digital signature.



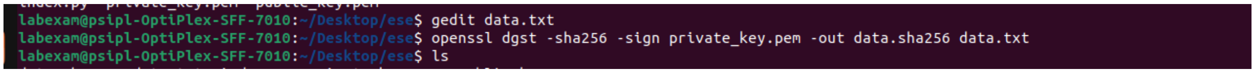
```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl genpkey -algorithm RSA -o
ut private_key.pem
............+..+.......+.....+.+..+...+..................+.+......+...+..+...+..
..+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+...+
...+..+.......+...+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++*..+..+.+..+...+....+...+............+...............+...+..+.+..+.+.....
+..........+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
....+.+.....+.............+..+...+.+......+..+.+...........+...+.......+......+.
.+......+.+...+...+..+............+.+........+++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++*...+.....+......+.+.....+++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++*........+.+.....+.+.....+.+...+..+.....
.......+............+......+.........+..........+.....+.........+.+...+........+
....+...+......+.....+.+......+...+..+....+......+............+......+..........
.+...+......................+........+.......+.....+.+..+.+...........+......+++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```



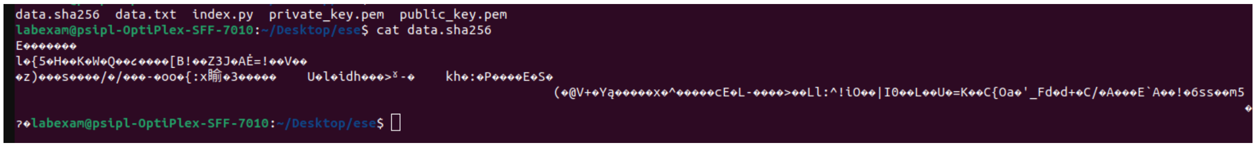
```
   27  history
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl rsa -pubout -in private_key.pem -out public_key.pem
writing RSA key
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
index.py  private_key.pem  public_key.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```



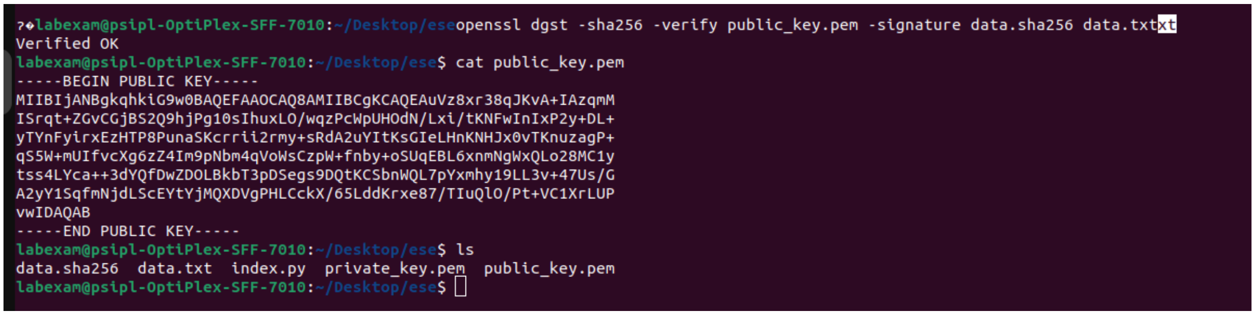
```
1 Hello this is Manan Chhajed
```



```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ gedit data.txt
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl dgst -sha256 -sign private_key.pem -out data.sha256 data.txt
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
```



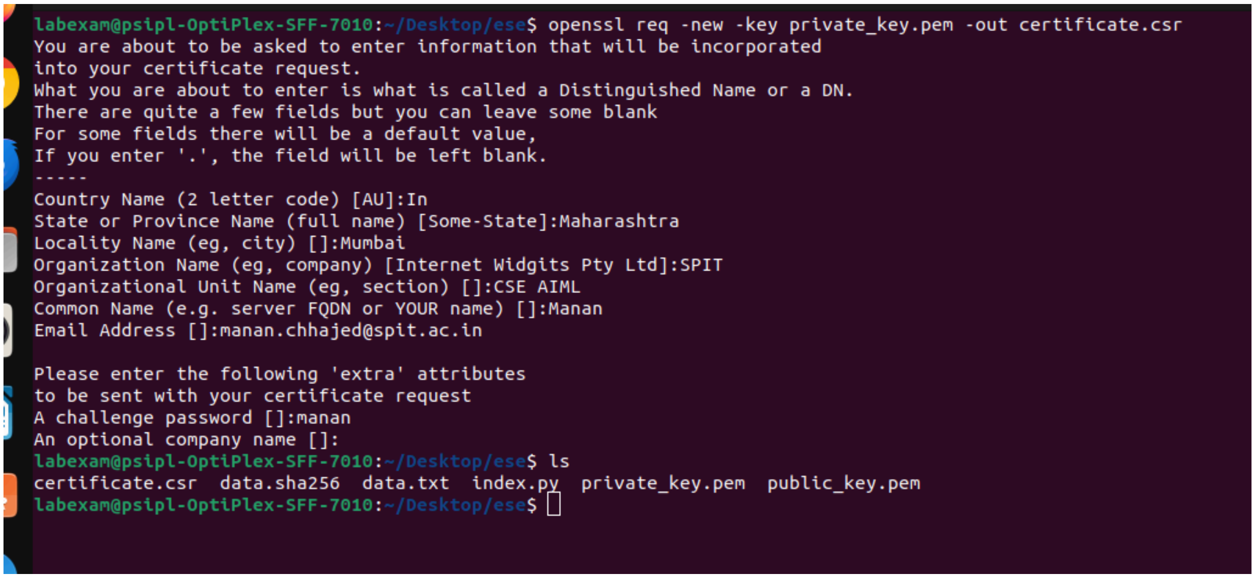
```
data.sha256  data.txt  index.py  private_key.pem  public_key.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ cat data.sha256
E�������
l�{5�H��K�W�Q��¿����[B!��Z3J�AĖ=!��V��
�z)���s����/�/���-�oo�{:x睮�3�����    U�l�idh���>ᵡ-�    kh�:�P����E�S�
                                                        (�@V+�Yą�����x�^�����cE�L-����>��Ll:^!iO��|I0��L��U�=K��C{Oa�'_Fd�d+�C/�A���E`A��!�6ss��m5
                                                                                                                                                  �
?�labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```



```
?�labexam@psipl-OptiPlex-SFF-7010:~/Desktop/eseopenssl dgst -sha256 -verify public_key.pem -signature data.sha256 data.txtxt
Verified OK
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ cat public_key.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuVz8xr38qJKvA+IAzqmM
ISrqt+ZGvCGjBS2Q9hjPg10sIhuxLO/wqzPcWpUHOdN/Lxi/tKNFwInIxP2y+DL+
yTYnFyirxEzHTP8PunaSKcrrii2rmy+sRdA2uYItKsGIeLHnKNHJx0vTKnuzagP+
qS5W+mUIfvcXg6zZ4Im9pNbm4qVoWsCzpW+fnby+oSUqEBL6xnmNgWxQLo28MC1y
tss4LYca++3dYQfDwZDOLBkbT3pDSegs9DQtKCSbnWQL7pYxmhy19LL3v+47Us/G
A2yY1SqfmNjdLScEYtYjMQXDVgPHLCckX/65LddKrxe87/TIuQlO/Pt+VC1XrLUP
vwIDAQAB
-----END PUBLIC KEY-----
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
data.sha256  data.txt  index.py  private_key.pem  public_key.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

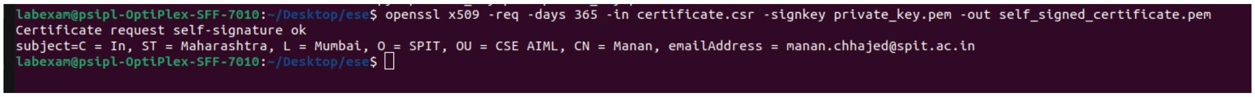
Task-4:

Generate Digital Certificate (self-signed)

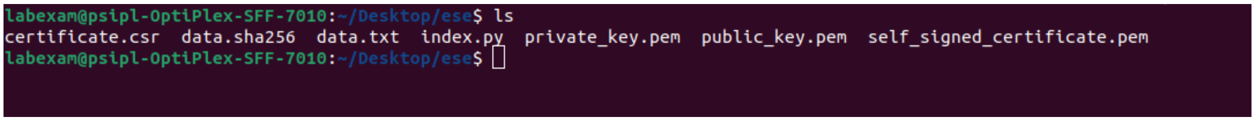


```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl req -new -key private_key.pem -out certificate.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite 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) [AU]:In
State or Province Name (full name) [Some-State]:Maharashtra
Locality Name (eg, city) []:Mumbai
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SPIT
Organizational Unit Name (eg, section) []:CSE AIML
Common Name (e.g. server FQDN or YOUR name) []:Manan
Email Address []:manan.chhajed@spit.ac.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:manan
An optional company name []:
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
certificate.csr  data.sha256  data.txt  index.py  private_key.pem  public_key.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

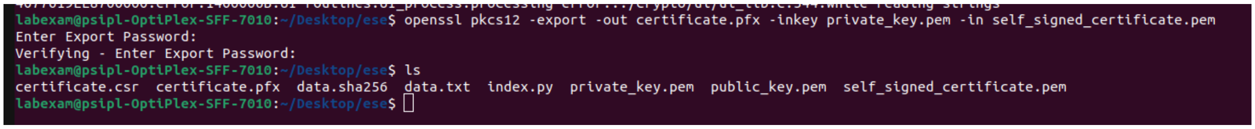
Now, let’s generate the self signed certificate



```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl x509 -req -days 365 -in certificate.csr -signkey private_key.pem -out self_signed_certificate.pem
Certificate request self-signature ok
subject=C = In, ST = Maharashtra, L = Mumbai, O = SPIT, OU = CSE AIML, CN = Manan, emailAddress = manan.chhajed@spit.ac.in
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

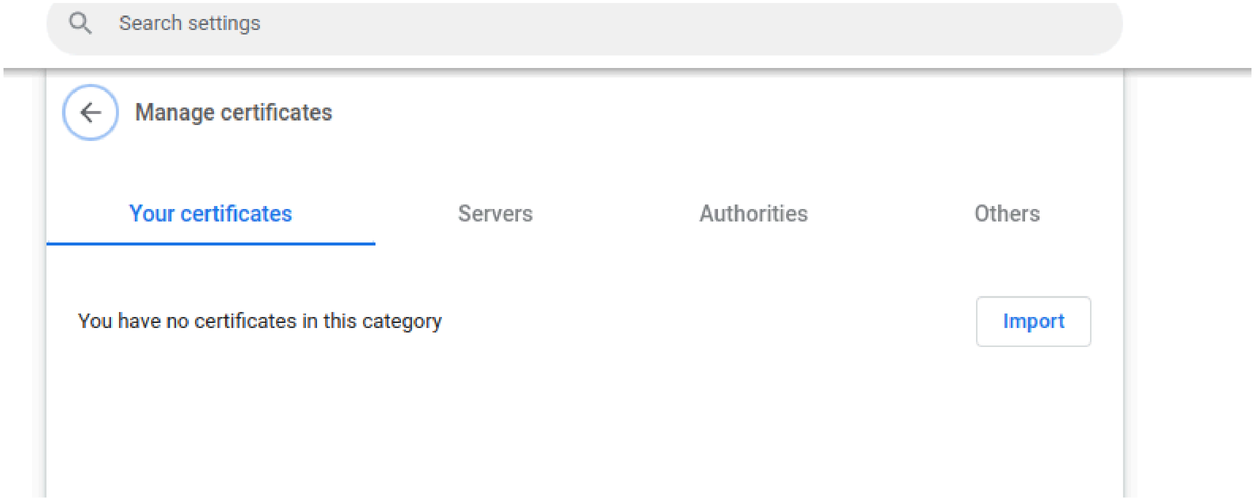


```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
certificate.csr  data.sha256  data.txt  index.py  private_key.pem  public_key.pem  self_signed_certificate.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

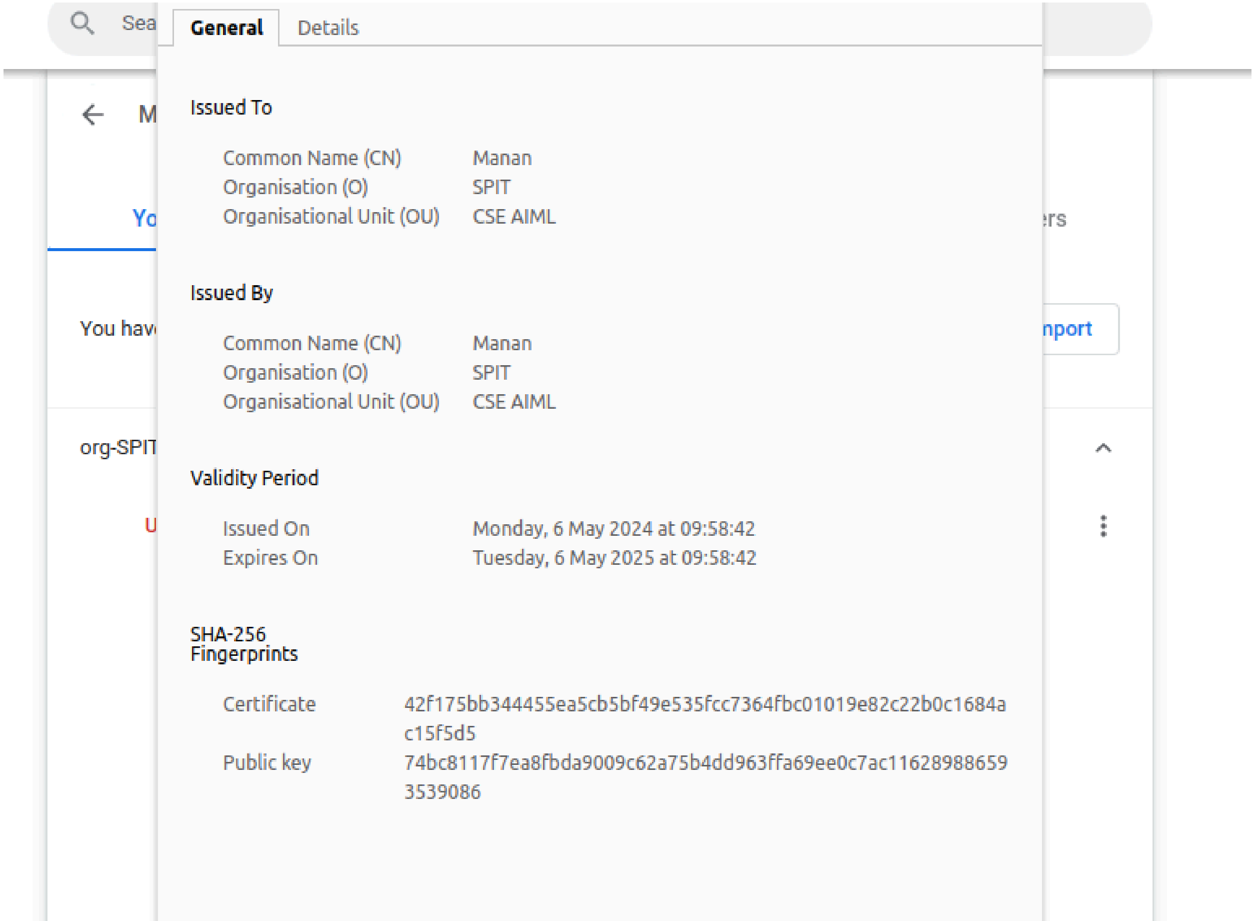


```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ openssl pkcs12 -export -out certificate.pfx -inkey private_key.pem -in self_signed_certificate.pem
Enter Export Password:
Verifying - Enter Export Password:
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ ls
certificate.csr  certificate.pfx  data.sha256  data.txt  index.py  private_key.pem  public_key.pem  self_signed_certificate.pem
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```

Now .pfx file contains both the certificate and private key



You have no certificates in this category



Issued To

Common Name (CN)	Manan
Organisation (O)	SPIT
Organisational Unit (OU)	CSE AIML

Issued By

Common Name (CN)	Manan
Organisation (O)	SPIT
Organisational Unit (OU)	CSE AIML

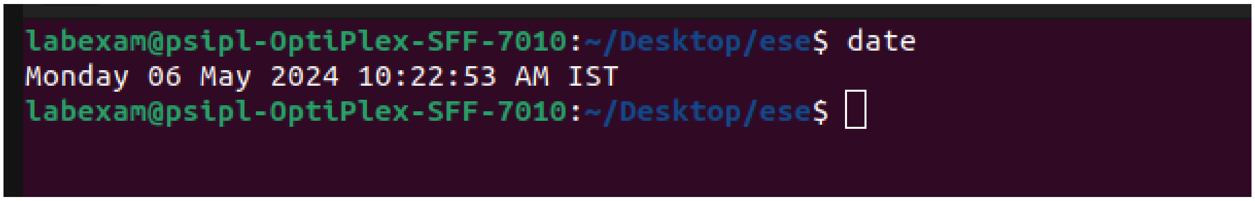
Validity Period

Issued On	Monday, 6 May 2024 at 09:58:42
Expires On	Tuesday, 6 May 2025 at 09:58:42

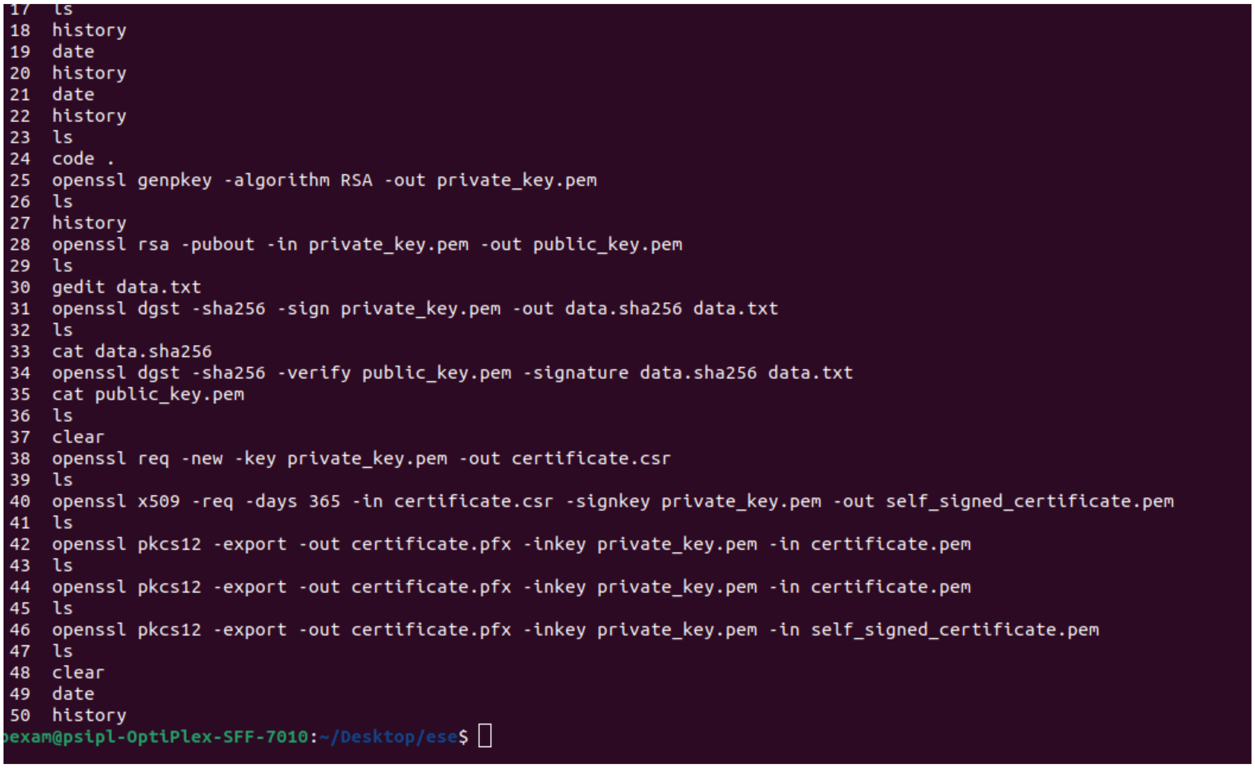
SHA-256 Fingerprints

Certificate	42f175bb344455ea5cb5bf49e535fcc7364fbc01019e82c22b0c1684ac15f5d5
Public key	74bc8117f7ea8fbda9009c62a75b4dd963ffa69ee0c7ac116289886593539086

After importing the certificate, we can go to the website using SSL / TLS and see our certificate being used for secure communication



```
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ date
Monday 06 May 2024 10:22:53 AM IST
labexam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$ 
```



```
 17  ls
 18  history
 19  date
 20  history
 21  date
 22  history
 23  ls
 24  code .
 25  openssl genpkey -algorithm RSA -out private_key.pem
 26  ls
 27  history
 28  openssl rsa -pubout -in private_key.pem -out public_key.pem
 29  ls
 30  gedit data.txt
 31  openssl dgst -sha256 -sign private_key.pem -out data.sha256 data.txt
 32  ls
 33  cat data.sha256
 34  openssl dgst -sha256 -verify public_key.pem -signature data.sha256 data.txt
 35  cat public_key.pem
 36  ls
 37  clear
 38  openssl req -new -key private_key.pem -out certificate.csr
 39  ls
 40  openssl x509 -req -days 365 -in certificate.csr -signkey private_key.pem -out self_signed_certificate.pem
 41  ls
 42  openssl pkcs12 -export -out certificate.pfx -inkey private_key.pem -in certificate.pem
 43  ls
 44  openssl pkcs12 -export -out certificate.pfx -inkey private_key.pem -in certificate.pem
 45  ls
 46  openssl pkcs12 -export -out certificate.pfx -inkey private_key.pem -in self_signed_certificate.pem
 47  ls
 48  clear
 49  date
 50  history
exam@psipl-OptiPlex-SFF-7010:~/Desktop/ese$
```