Course Name:	Information Security (116U01L602)	Semester:	VI
Date of Performance:	03 / 02 / 2025	DIV/ Batch No:	A-4
Student Name:	Hyder Presswala	Roll No:	16010122151

Title: Application of RSA Algorithm for various security services like confidentiality, authentication, signature, non-repudiation and integrity

Objectives:

To write a program to convert plain text into cipher text using Caesar cipher and Transposition cipher

Expected Outcome of Experiment:

CO1 **:-** Explain various security goals, threats, vulnerabilities and controls
CO2 **:-** Apply various cryptographic algorithms for software security

Books/ Journals/ Websites referred:

1. Security in Computing
2. Cryptography and Network Security
3. Cryptography and Network Security: Principles and Practice

Pre Lab/ Prior Concepts:

New Concepts to be learned:

Abstract:

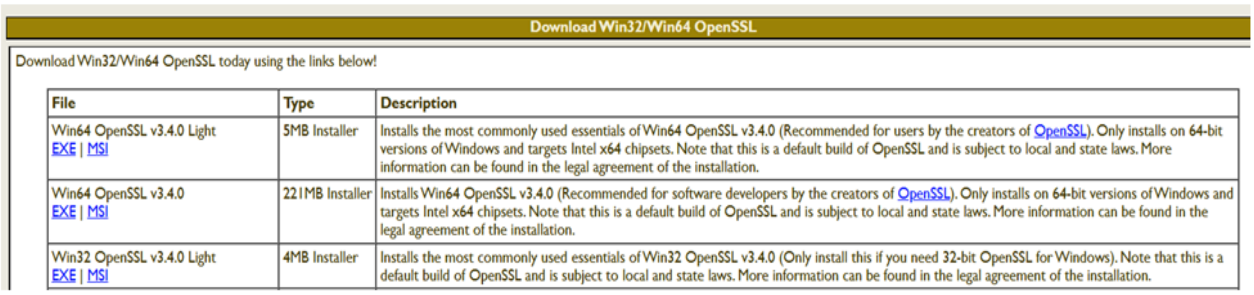
Related Theory:

Implementation Details with Output :

1. Enlist all the Steps followed and various options explored

I. Download and Install OpenSSL

1. Download the Download Win64 OpenSSL v3.0.8 (EXE) from the website.

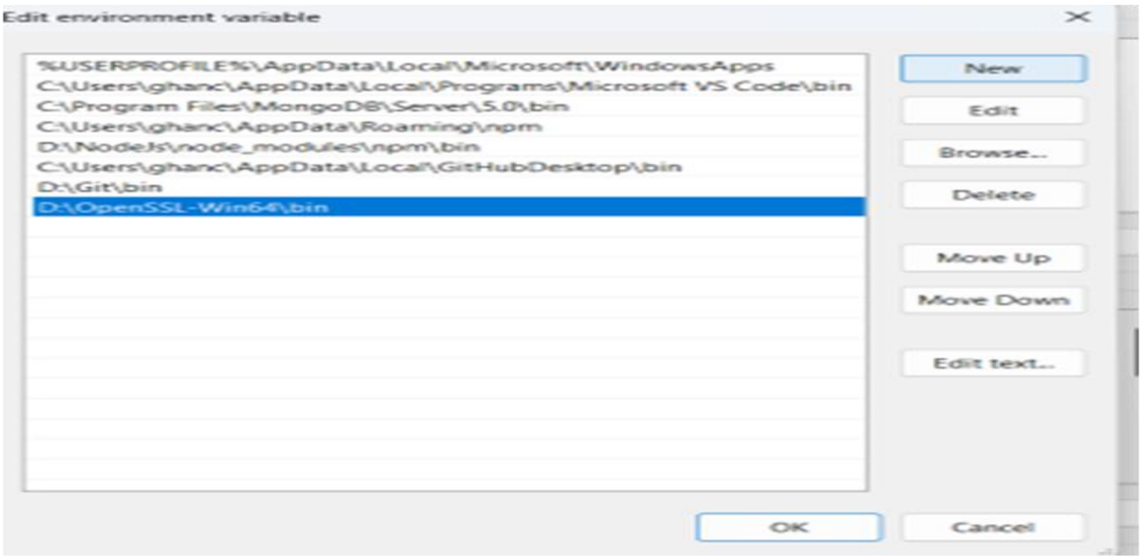


Download Win32/Win64 OpenSSL

Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.4.0 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.4.0 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.4.0 EXE | MSI	221MB Installer	Installs Win64 OpenSSL v3.4.0 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.4.0 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.4.0 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

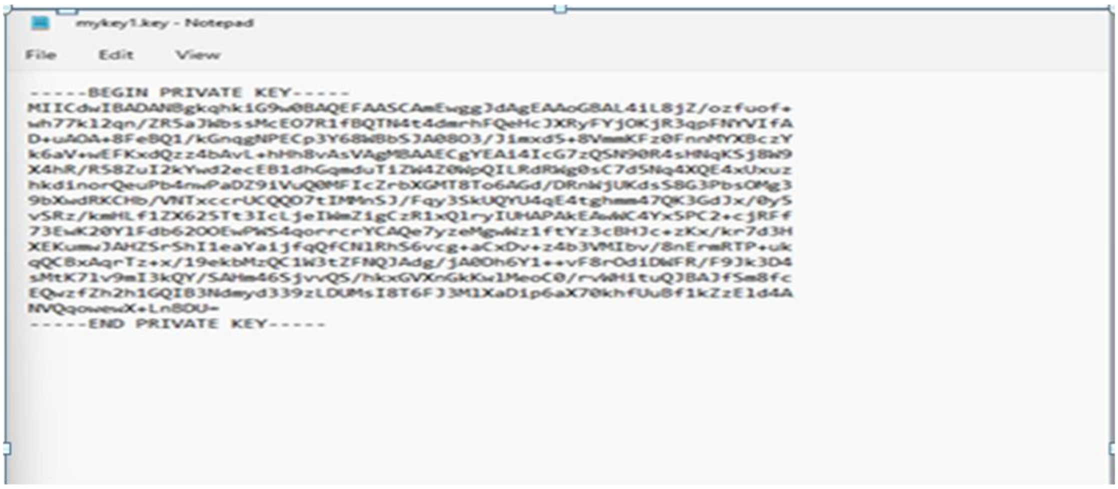
2. Install and update the Path variable in Environment Variables.



II. Generating RSA private /public key pair

1. Generate key pair using: opensslgenrsa -out mykey1.key 1024

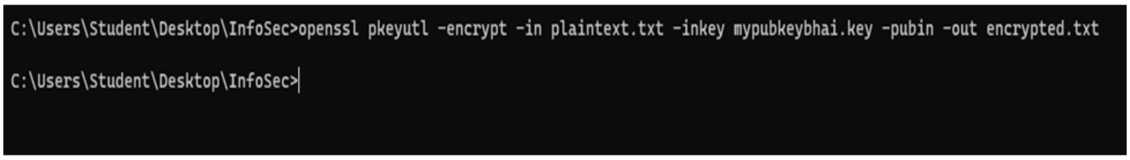
2. Extract public key from key pair using: opensslrsa -in mykey1.key - pubout -out mypublickey.key

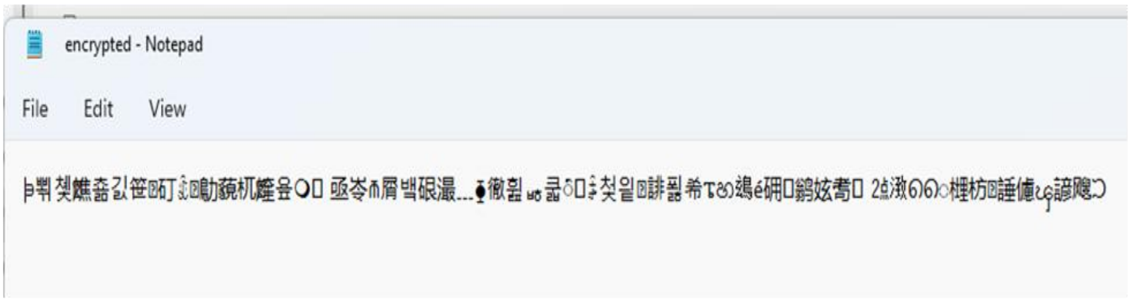


III. Public Key Encryption

Using the public key we can encrypt the text in plaintext.txt using:

opensslpkeyutl - encrypt -in plaintext.txt -inkeymypublickey.key -pubin -out encrypted.txt



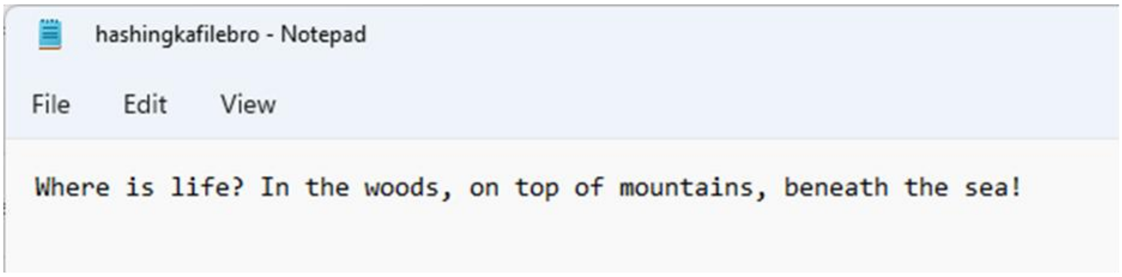


IV. Hash Functions

Create a file to be hashed.

Use the command: openssldgst -sha256 hashfile.txt

```
C:\Users\Student\Desktop\InfoSec>openssl dgst -sha256 hashingkafilebro.txt
SHA2-256(hashingkafilebro.txt)= e9d188f5fb9c7dcae9be7d5209025a3a454835807a1afc411f3b90dee0985263

C:\Users\Student\Desktop\InfoSec>
```

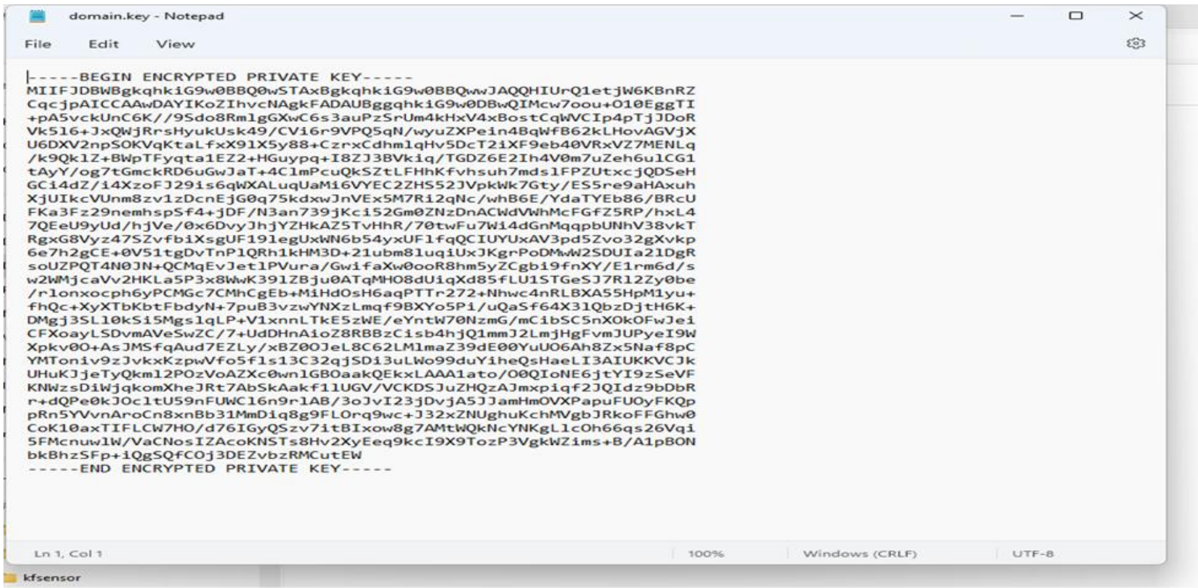


V. Certificate Creation

1. Create a private key using:

   opensslgenrsa -des3 -out domain.key 2048

```
C:\Users\Student\Desktop\InfoSec>openssl genrsa -des3 -out domain.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

C:\Users\Student\Desktop\InfoSec>
```



domain.key - Notepad

File Edit View


```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFJDBWBgkqhkiG9w0BBQ0wSTAxBgkqhkiG9w0BBQwwJAQQHIUrQ1etjW6KBnRZ
CqcjpAICCAAwDAYIKoZIhvcNAgkFADAUBggqhkiG9w0DBwQIMcw7oou+O10EggTI
+pA5vckUnC6K//9Sdo8RmlgGXwC6s3auPzSrUm4kHxV4xBostCqWVCIp4pTjJDoR
Vk516+JxQWjRrsHyukUsk49/CVi6r9VPQ5qN/wyuZXPein4BqWfB62kLHovAGVjX
U6DXV2npSOKVqKtaLfxX91X5y88+CzrxCdhmlqHv5DcT2iXF9eb40VRxVZ7MENLq
/k9Qk1Z+BWpTFyqta1EZ2+HGuypq+I8ZJ3BVkiq/TGDZ6E2Ih4V0m7uZeh6ulCG1
tAyY/og7tGmckRD6uGwJaT+4ClmPcuQkSZtLFHhKfvhsuh7mds1FPZUtxcjQDSeH
GCi4dZ/i4XzoFJ29is6qWXALuqUaMi6VYEC2ZHS52JVpkWk7Gty/ES5re9aHAxuh
XjUIkcVUnm8zv1zDcnEjG0q75kdxwJnVEx5M7Ri2qNc/whB6E/YdaTYEb86/BRcU
FKa3Fz29nemhspSf4+jDF/N3an739jKci52Gm0ZNzDnACWdVWhMcFGfZ5RP/hxL4
7QEeU9yUd/hjVe/0x6DvyJhjYZHkAZ5TvHhR/70twFu7Wi4dGnMqqpbUNhV38vkT
RgxG8Vyz47SZvfbiXsgUF19legUxWN6b54yxUF1fqQCIUYUxAV3pd5Zvo32gXvkp
6e7h2gCE+0V51tgDvTnP1QRh1kHM3D+21ubm8luqiUxJKgrPoDMwW2SDUIa21DgR
soUZPQT4N0JN+QCMqEvJet1PVura/GwifaXw0ooR8hm5yZCgbi9fnXY/E1rm6d/s
w2WMjcaVv2HKLa5P3x8WwK391ZBju0ATqMHO8dUiqXd85fLU1STGeSJ7R12Zy0be
/r1onxocph6yPCMGc7CMhCgEb+MiHdOsH6aqPTTr272+Nhwc4nRLBXA55HpM1yu+
fhQc+XyXTbKbtFbdyN+7puB3vzwYNXzLmqf9BXYo5Pi/uQaSf64X31QbzDjtH6K+
DMgj3SL10kSi5Mgs1qLP+V1xnnLTkE5zWE/eYntW70NzmG/mCibSC5nXOkOFwJei
CFXoayLSDvmAVeSwZC/7+UdDHnAioZ8RBBzCisb4hjQ1mmJ2LmjHgFvmJUPyeI9W
Xpkv0O+AsJMSfqAud7EZLy/xBZ0OJeL8C62LMlmaZ39dE00YuUO6Ah8Zx5Naf8pC
YMToniv9zJvkxKzpwVfo5f1s13C32qjSDi3uLWo99duYiheQsHaeLI3AIUKKVCJk
UHuKJjeTyQkml2POzVoAZXc0wnlGBOaakQEkxLAAA1ato/O0QIoNE6jtYI9zSeVF
KNWzsDiWjqkomXheJRt7AbSkAakf11UGV/VCKDSJuZHQzAJmxpiqf2JQIdz9bDbR
r+dQPe0kJOc1tU59nFUWCl6n9r1AB/3oJvI23jDvjA5JJamHmOVXPapuFUOyFKQp
pRn5YVvnAroCn8xnBb31MmDiq8g9FLOrq9wc+J32xZNUghuKchMVgbJRkoFFGhw0
CoK10axTIFLCW7HO/d76IGyQSzv7itBIxow8g7AMtWQkNcYNKgL1cOh66qs26Vqi
5FMcnuw1W/VaCNosIZAcoKNSTs8Hv2XyEeq9kcI9X9TozP3VgkWZims+B/A1pBON
bkBhzSFp+iQgSQfCOj3DEZvbzRMCutEW
-----END ENCRYPTED PRIVATE KEY-----
```

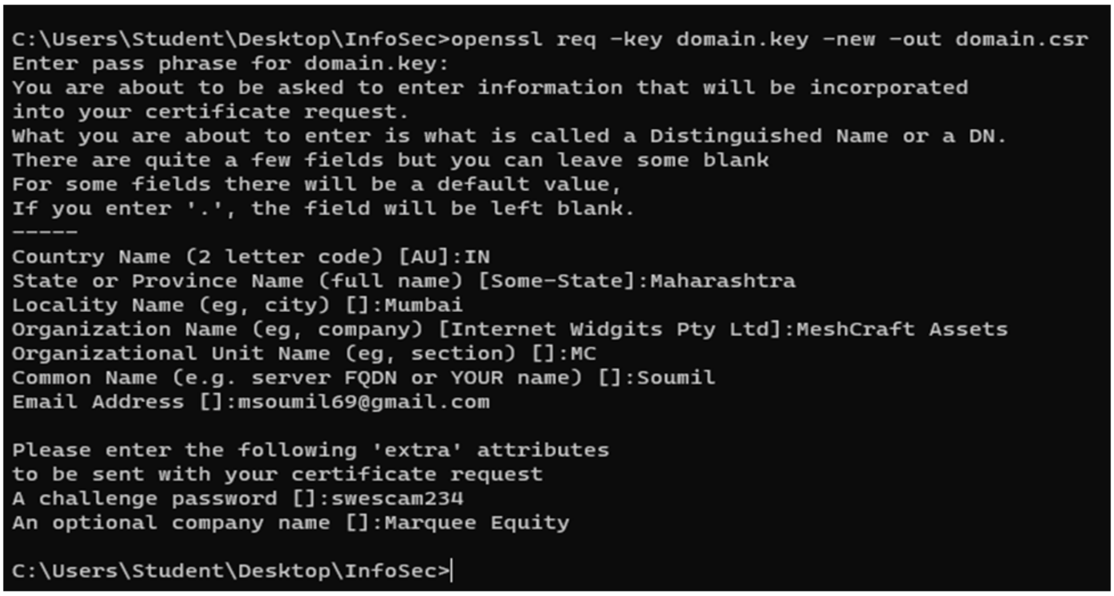

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

kfsensor

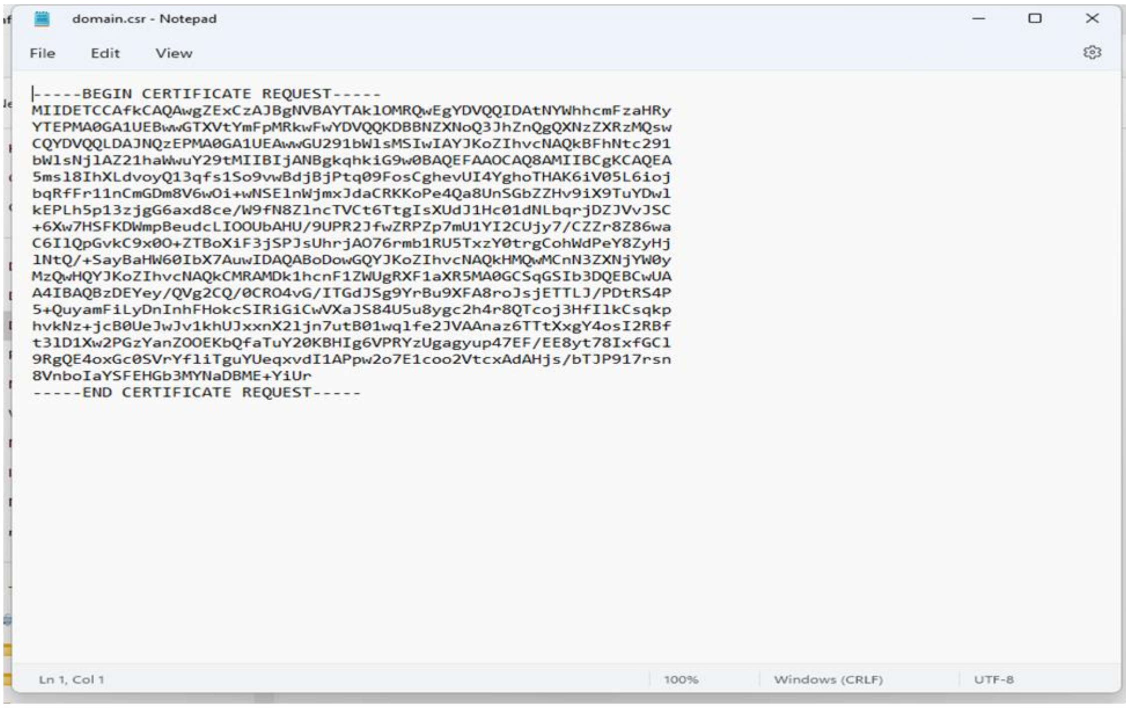
2. Create a certificate using the key using:

opensslreq -key domain.key -new -out domain.csr

Enter all the necessary information for the certificate

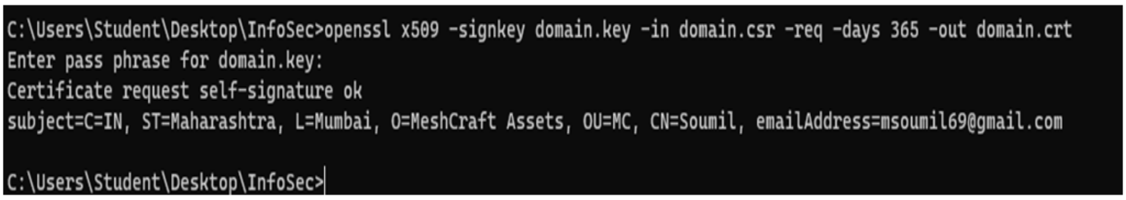


```
C:\Users\Student\Desktop\InfoSec>openssl req -key domain.key -new -out domain.csr
Enter pass phrase for domain.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There 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]:MeshCraft Assets
Organizational Unit Name (eg, section) []:MC
Common Name (e.g. server FQDN or YOUR name) []:Soumil
Email Address []:msoumil69@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:swescam234
An optional company name []:Marquee Equity

C:\Users\Student\Desktop\InfoSec>
```



3. Create a self signed certificate used:

openssl x509 -signkeydomain.key -in domain.csr -req -days 365 -out domain.crt

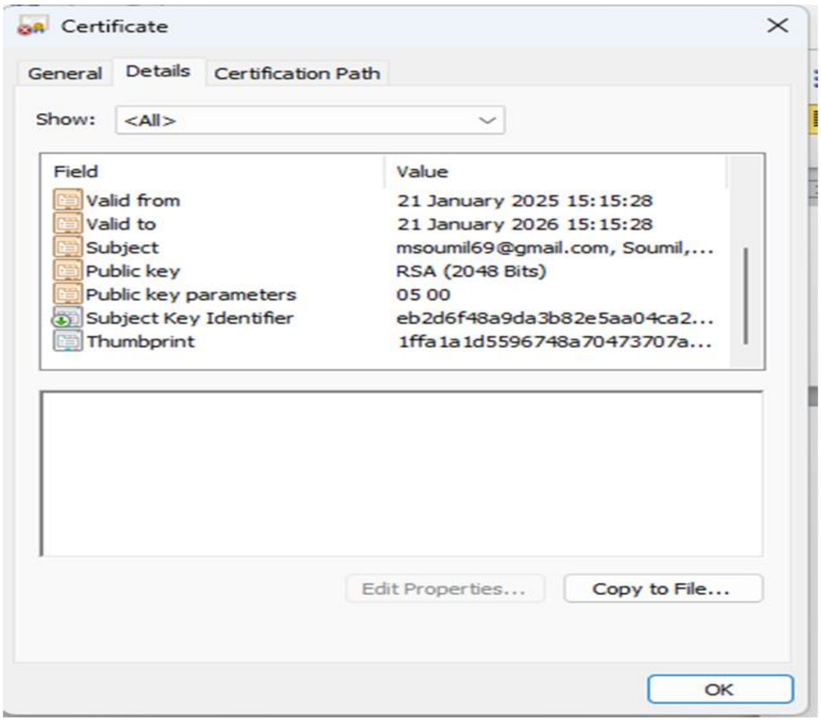


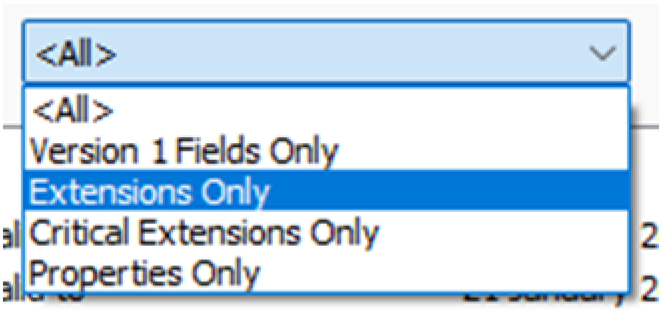
4. View the certificate using:

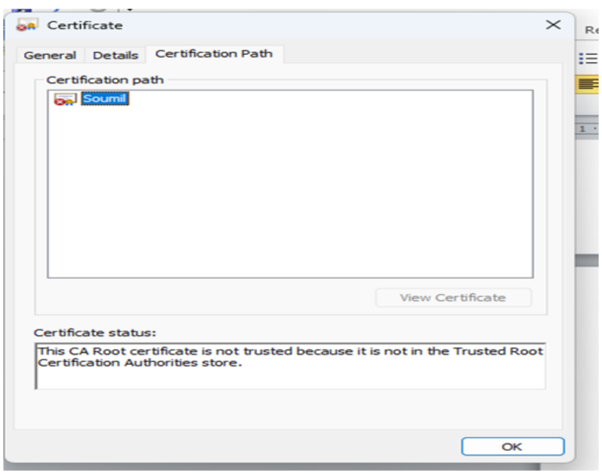
openssl x509 -text -noout -in domain.crt

```
        66:9b:23:16:d7:09:a6:7d:32:4e:94:46:d6:b2:c9:c6:b2:e7:
        43:fc:d9:93:25:2f:d5:7c:f6:e2:1e:a6:2e:4a:0d:3d:61:d8:
        c7:76:bd:a8:9e:42:41:c4:08:7a:ae:29:71:37:8a:b6:be:bf:
        b9:89:86:02:60:84:15:4f:6d:63:a4:89:8a:1c:a0:02:89:6d:
        07:25:6c:99:5c:f5:a5:c5:6b:1e:8d:6e:40:0e:98:00:29:d0:
        90:e3:dc:a5:1a:1d:cc:26:05:49:33:5e:71:72:b8:d3:f3:ea:
        2f:e6:77:40:09:dc:89:ea:3f:4e:29:92:45:2c:94:6e:81:f5:
        c4:fd:3d:33:8f:2c:95:71:0d:54:f3:bd:84:76:4f:e8:20:d0:
        b5:44:ea:eb:5e:ea:d7:0c:fd:a4:b9:cd:b8:06:c0:5b:c5:f9:
        61:73:08:67:f6:61:66:c0:80:97:5d:fa:f6:34:a7:e6:c3:b7:
        05:fe:e4:30:f6:17:53:f2:43:dd:82:75:ed:52:35:5b:88:7b:
        a9:db:19:82:e1:0f:cb:30:06:ea:50:cb:1e:45:82:17:e4:fc:
        ca:b5:a1:fd

C:\Users\Student\Desktop\InfoSec>
```

```
C:\Users\Student\Desktop\InfoSec>openssl x509 -text -noout -in domain.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            14:d0:47:70:5c:ac:d7:da:e7:b4:d6:3d:1f:04:75:aa:f0:6b:bd:79
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IN, ST=Maharashtra, L=Mumbai, O=MeshCraft Assets, OU=MC, CN=Soumil, emailAddress=msoumil69@gmail.com
        Validity
            Not Before: Jan 21 09:45:28 2025 GMT
            Not After : Jan 21 09:45:28 2026 GMT
        Subject: C=IN, ST=Maharashtra, L=Mumbai, O=MeshCraft Assets, OU=MC, CN=Soumil, emailAddress=msoumil69@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:e6:6b:25:f0:88:57:2d:db:e8:c9:0d:77:a9:fb:
                    35:4a:8f:6f:c0:17:63:06:33:ed:ab:4f:45:a2:c0:
                    a0:85:eb:d4:23:86:20:86:84:c7:00:ae:a2:57:4e:
                    4b:ea:2a:23:6e:a4:5f:16:bd:75:9c:29:86:0e:6f:
                    15:eb:03:a2:fb:03:52:12:59:d6:8e:6c:49:75:a0:
                    91:28:aa:0f:7b:84:1a:f1:49:d2:19:b6:59:1e:ff:
                    62:5f:d4:ee:60:3c:25:90:43:cb:87:9a:75:df:38:
                    e0:1b:a6:b1:77:c7:1e:fd:6f:5f:37:c6:65:9d:c4:
                    d5:0a:de:93:b6:02:2c:5d:47:49:d4:77:34:d5:d3:
                    4b:6e:aa:e3:0d:92:55:bc:94:82:fb:a5:f0:ec:74:
                    85:28:35:a6:a4:17:ae:75:c2:c8:38:e5:1b:00:75:
                    3f:f5:43:d1:d8:97:f0:65:13:d9:a7:b9:94:d5:82:
                    36:09:48:f2:ef:f0:99:66:bf:19:f3:ac:1a:0b:a2:
                    25:42:91:af:90:2f:71:d0:ef:99:4c:1a:17:88:5d:
                    e3:48:f2:6c:52:1a:e3:00:ee:fa:ae:66:f5:45:4e:
                    53:c7:36:34:b6:b8:02:a2:15:9d:3d:e6:3c:67:21:
                    e3:94:db:50:ff:e4:9a:c8:16:87:5b:ad:08:6d:7e:
                    c0:bb
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                EB:2D:6F:48:A9:DA:3B:82:E5:AA:04:CA:2D:D1:7E:36:64:4C:16:EA
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        8b:f2:ea:3a:8d:4b:27:18:75:4a:6a:5e:eb:a0:cf:d1:82:85:
        b3:c7:95:b4:85:98:71:2f:e0:2e:b5:ed:39:ee:f2:31:10:01:
```


Certificate
General
Details
Certification Path
Show:
<All>
Field
Value
Valid from
21 January 2025 15:15:28
Valid to
21 January 2026 15:15:28
Subject
msoumil69@gmail.com, Soumil,...
Public key
RSA (2048 Bits)
Public key parameters
05 00
Subject Key Identifier
eb2d6f48a9da3b82e5aa04ca2...
Thumbprint
1ffa1a1d5596748a70473707a...
Edit Properties...
Copy to File...
OK


<All>
<All>
Version 1 Fields Only
Extensions Only
Critical Extensions Only
Properties Only


Certificate
General
Details
Certification Path
Certification path
Soumil
View Certificate
Certificate status:
This CA Root certificate is not trusted because it is not in the Trusted Root Certification Authorities store.
OK

VI. Digital signature

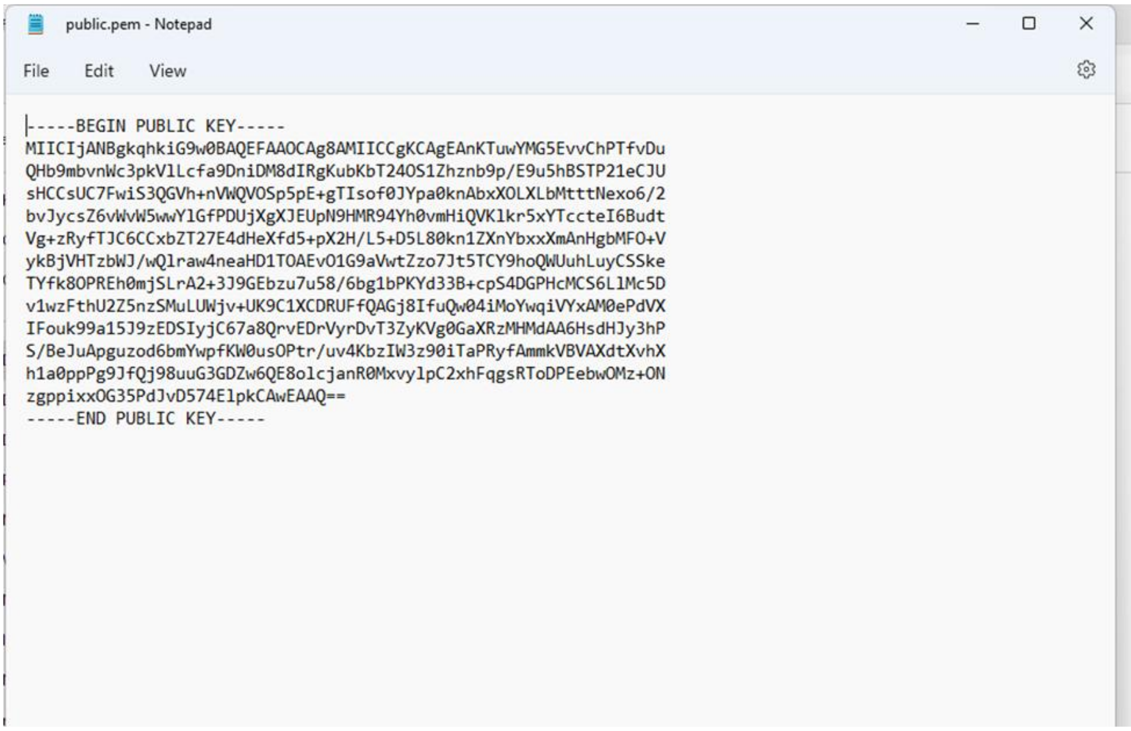
1. Create private and public key using:

   -opensslgenrsa -aes128 -passout pass<phrase>: -out private.pem 4096

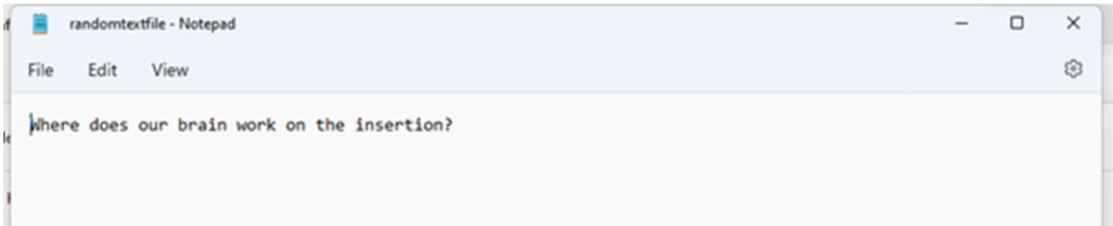
   -opensslrsa -in private.pem -passin pass:<phrase> -pubout -out public.pem

```
C:\Users\Student\Desktop\InfoSec>openssl genrsa -aes128 -passout pass:soumaiya -out private.pem 4096

C:\Users\Student\Desktop\InfoSec>
```


```
C:\Users\Student\Desktop\InfoSec>openssl rsa -in private.pem -passin pass:soumaiya -pubout -out public.pem
writing RSA key

C:\Users\Student\Desktop\InfoSec>
```

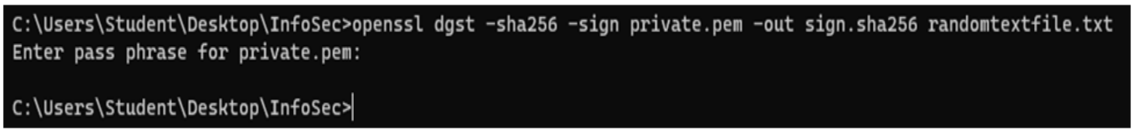


2. Create a text file



3. Generate the signature of a file using:

-openssldgst -sha256 -sign<private key> -out /tmp/sign.sha256<file> -openssl base64 -in /tmp/sign.sha256 -out <signature>



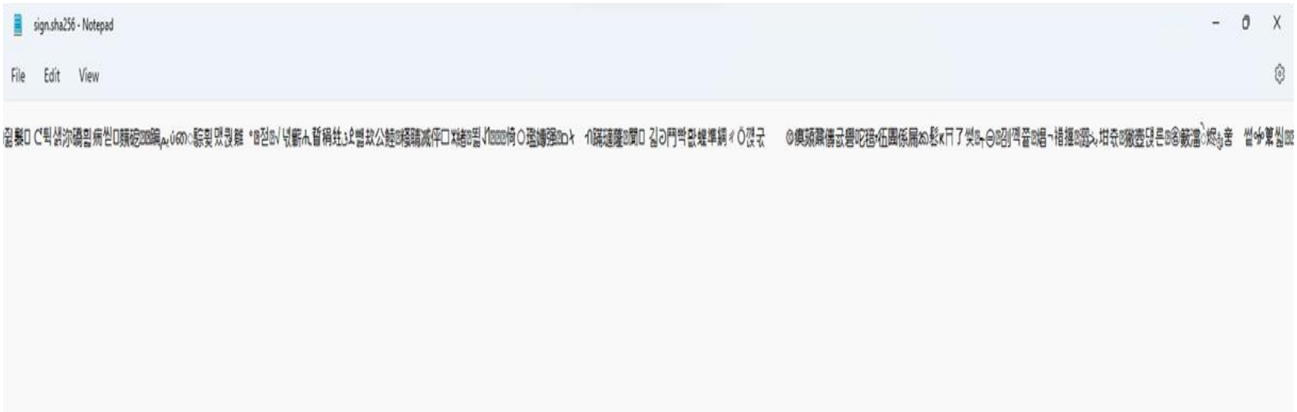
4. Verify the signature using:

-openssl base64 -d -in <signature> -out /tmp/sign.sha256 <file>

```
C:\Users\Student\Desktop\InfoSec>openssl base64 -d -in randomtextfile.txt

C:\Users\Student\Desktop\InfoSec>
```

-openssldgst -sha256 -verify <pub-key> -signature /tmp/sign.sha256

```
C:\Users\Student\Desktop\InfoSec>openssl dgst -sha256 -verify public.pem -signature sign.sha256 randomtextfile.txt
Verified OK

C:\Users\Student\Desktop\InfoSec>
```



Conclusion:

Thus, in this experiment the concept of RSA algorithm for various security services like confidentiality, authentication, signature, non-repudiation and integrity was understood and applied by developing a website.

Post-Lab Questions:

2.1 In the RSA algorithm, p= 7, q=11 and e= 13, then what will be the value of d?

In the **RSA algorithm**, the value of **d** is calculated as follows:

Given:

- p=7p = 7p=7
- q=11q = 11q=11
- e=13e = 13e=13

Step 1: Compute n

$$n=p\times q=7\times 11=77$$

Step 2: Compute ϕ(n)

$$\phi(n)=(p-1)\times(q-1)=(7-1)\times(11-1)=6\times 10=60$$

Step 3: Compute d

d is the modular multiplicative inverse of e modulo ϕ(n), meaning:

$$d\times e\equiv 1 \bmod \phi(n)$$

$$d\times 13\equiv 1 \bmod 60$$

To find d, we need to solve:

$$d=e^{-1} \bmod 60$$

This means finding d such that:

$$(13\times d) \bmod 60=1$$

Using the **Extended Euclidean Algorithm**:
We solve for d in:

$$13^{-1} \bmod 60$$

Using the **Euclidean Algorithm**:

1. $60 \div 13 = 4$ remainder $60 - (13 \times 4) = 8$
2. $13 \div 8 = 1$ remainder $13 - (8 \times 1) = 5$
3. $8 \div 5 = 1$ remainder $8 - (5 \times 1) = 3$
4. $5 \div 3 = 1$ remainder $5 - (3 \times 1) = 2$
5. $3 \div 2 = 1$ remainder $3 - (2 \times 1) = 1$
6. $2 \div 1 = 2$ remainder $2 - (1 \times 2) = 0$

Since the GCD is **1**, we can backtrack:

$$1 = 3 - (1 \times 2)$$

$$1 = 3 - (1 \times (5 - 1 \times 3)) = 2 \times 3 - 1 \times 5$$

$$1 = 2 \times (8 - 1 \times 5) - 1 \times 5 = 2 \times 8 - 3 \times 5$$

$$1 = 2 \times 8 - 3 \times (13 - 1 \times 8) = 5 \times 8 - 3 \times 13$$

$$1 = 5 \times (60 - 4 \times 13) - 3 \times 13 = 5 \times 60 - 23 \times 13$$

$$-23 \times 13 \equiv 1 \mod 60$$

Since d must be positive:

$$d=60-23=37$$

Theirfore d=37

2.2 Discuss various cryptanalysis attacks possible to be carried out on RSA

- **Integer Factorization Attack**: This is the most well-known attack on RSA. It involves factoring the large composite number N (which is the product of two large primes) to retrieve the private key. The difficulty of this attack depends on the size of NN.
- **Timing Attacks**: These attacks exploit the time taken by the RSA algorithm to perform certain operations. By analyzing the time variations, an attacker can deduce information about the private key2.
- **Side-Channel Attacks**: These attacks exploit information leaked during the physical implementation of the RSA algorithm, such as power consumption, electromagnetic radiation, or even sound.
- **Chosen Ciphertext Attacks**: In this attack, the attacker chooses a ciphertext and attempts to decrypt it to gain information about the private key or plaintext.
- **Quantum Computing Attacks**: Although still theoretical, quantum computers could potentially break RSA by efficiently solving problems that are currently intractable, such as integer factorization.

2.3 Comment on drawbacks of RSA. Discuss solution(s) over the same.

Drawbacks of RSA

1. **Computational Complexity**:
   - RSA requires significant computational resources, especially for key generation, encryption, and decryption. This can be a drawback for devices with limited processing power.
2. **Large Key Sizes**:
   - To ensure security, RSA requires large key sizes (2048 bits or more), resulting in slower encryption and decryption processes, which can be inefficient for some applications.
3. **Vulnerability to Quantum Computing**:
   - Future quantum computers could potentially break RSA by efficiently solving integer factorization problems, posing a significant risk to the long-term security of RSA.

Solutions

1. **Hybrid Cryptosystems**:
   - Combine RSA with symmetric key algorithms (e.g., AES) to mitigate computational complexity. RSA can be used to securely exchange a symmetric key, which is then used for efficient data encryption and decryption.
2. **Elliptic Curve Cryptography (ECC)**:
   - ECC offers similar security to RSA but with much smaller key sizes, resulting in faster computations and reduced resource requirements. ECC is a potential alternative to RSA for many applications.
3. **Post-Quantum Cryptography**:
   - Research into post-quantum cryptographic algorithms is ongoing. These algorithms are designed to be secure against quantum computing attacks. Implementing post-quantum cryptography can future-proof systems against quantum threats.