Date of submission: 31st January 2025
Batch: C1 Roll No.: 16010122323
Student Name: Vedansh Savla
Experiment No: 2
Staff In-charge : Shivani D

TITLE: Application of RSA Algorithm for various security services like confidentiality, authentication, signature, non-repudiation and integrity
AIM: To implement Application using RSA Algorithm.
OUTCOME: Student will be able to
CO1: Explain various security goals, threats, vulnerabilities and controls
CO2: Apply various cryptographic algorithms for software security

---

Theory about SSL:

SSL/TLS is a cryptographic protocol designed to provide secure communication over a computer network, most commonly the internet. It ensures confidentiality, integrity, and authentication between two communicating applications (e.g., a web browser and a server). SSL is the predecessor to TLS, but the term "SSL" is often used interchangeably with TLS.

Block Diagram of Encryption, Decryption, Digital Signature and Digital Certificate for the implementation steps:

Implementation Details with outputs:

1. Generating RSA private /public key pair

Select Win64 OpenSSL Command Prompt

```
C:\Users\student\Desktop>openssl genrsa -out mykey1.key 1024
```

mykey1 - Notepad

File Edit Format View Help


```
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAMzO9ksc99z3AS0b
I/jA1GBhkbEYQEea4k6mGe4Gc/IsCtevOkU4HGytfIzn910fc6emjocxTkRnfoCj
mP1ac9kkb633kkq+b/OaeRK6K7SZFj/26XcX9GZKFj7Z+myY7s3gYXJOKVymXIE+
7XilRJUvbBwzoxKAq1t/qH0xk3J7AgMBAAECgYB1M/q+AVXWnYOOzH1JYRAoF63I
KpieYYQgfe1YyViQvaN2ylAOTI33o1AjUpspr1YAGtB1jUG8430BIJ3sXlbsZcIE
zye55DP9kzAMrB2KJ61NspiZY0q0OQ8tRvIQVBq4id/Y+SWFIJ9FfnXFnSw/vda9
8cPQyzk3O7wQgF6kAQJBAPT39yLUYbIsjRvVhrcQcTaFScIlZuK/m6aWcIlmEnz1
cwBlTpfsL4TIaJCEX/glG+5aouuiJeaRN1tv8lMOKVcCQQDWCAY46miHlNzVsw5w
6DeN36tJCXE4BJORkJTvYfRHXb8aW5Lwydtleu9p2xbMrGHQNHOLRJ68+4tQ3gdm
D/V9AkBSRTzwq5cr+Z5ibLHTpWgAQX9l/p2xJVym8/tTib5HzT8UMPHWP/ItjAji
EDMS6ZOZ10LEFC7aYobO5KfjkOv9AkEAuREfMD+gOvwBJ9c3+XER+F8a4ICFoYZz
gz6dSZBNhRgOkl/0YrhoOQtf0QxcXoX/3eQ8tIOVLpvjDv2tMBdEkQJAJ3cPew0o
G3eOmsYeAerfMy8ljIdLXsiyL7/+iTFw4SWrDzK+r70vDInFuK4VriYRKWMKTqFO
60wOyDG75Zyxcw==
-----END PRIVATE KEY-----
```

2. Public Key Encryption

```
C:\Users\student\Desktop>openssl rsa -in mykey1.key -pubout -out mypublickey.key
writing RSA key

C:\Users\student\Desktop>
```

mypublickey - Notepad

File Edit Format View Help

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDMzvZLHPfc9wEtGyP4wNRgYZGx
GEBHmuJOphnuBnPyLArXrzpFOBxsrXyM5/ZdH3Onpo6HMU5EZ36Ao5j9WnPZJG+t
95JKvm/zmnkSuiu0mRY/9ul3F/RmShY+2fpsm07N4GFyTilcplyBPu14pUSVL2wc
M6MSgKtbf6h9MZNyewIDAQAB
-----END PUBLIC KEY-----
```

3. Public Key Encryption

Win64 OpenSSL Command Prompt

```
C:\Users\student\Desktop\IS>openSSL pkeyutl -encrypt -in plaintext.txt -inkey mypublickey.key -pubin -out encrypted.txt
C:\Users\student\Desktop\IS>
```

*plaintext - Notepad

File Edit Format View Help

```
vedansh
```

```
C:\Users\student\Desktop\IS>openSSL pkeyutl -encrypt -in plaintext.txt -inkey mypublickey.key -
```

plaintext - Notepad

File Edit Format View Help

```
vedansh
```

encrypted - Notepad

File Edit Format View Help

4. hash function

```
C:\Users\student\Desktop\IS>
C:\Users\student\Desktop\IS>openSSL dgst -sha256 hashfile.txt
SHA2-256(hashfile.txt)= 3384cb59fe42654d48b3c9e62ed3f6b2400c393b078eb27167ca4bb4040ed179
```

*hashfile - Notepad

File Edit Format View Help

```
hash this file
```

5. Certificate Creation

```
C:\Users\student\Desktop\IS>openSSL genrsa -des3 -out domain.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

C:\Users\student\Desktop\IS>_
```

6. Certificate Creation using key

```
Win64 OpenSSL Command Prompt
C:\Users\student\Desktop\IS>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-State
Locality Name (eg, city) []:Mumbai
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:Comps
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:vedansh.savla@somaiya.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:vedansh
```

7. Create a self signed certificate

```
C:\Users\student\Desktop\IS>openssl x509 -signkey domain.key -in domain.csr -req -days 365 -out domain.csr
Enter pass phrase for domain.key:
Certificate request self-signature ok
subject=C=IN, ST=Maharashtra-State, L=Mumbai, O=Internet Widgits Pty Ltd, OU=Comps, emailAddress=vedansh.savla@somaiya.edu
```

8. View the certificate

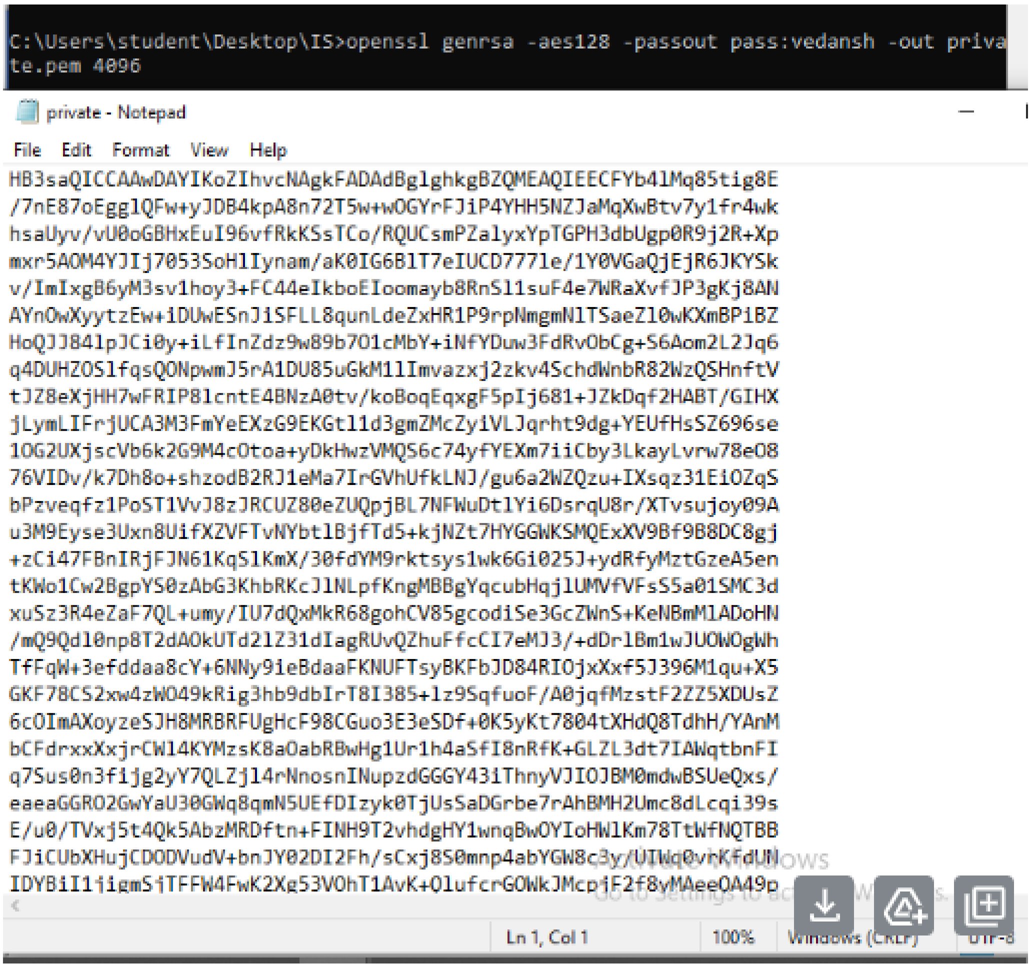
Win64 OpenSSL Command Prompt

```
C:\Users\student\Desktop\IS>openssl x509 -text -noout -in domain.csr
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            3a:e6:e2:ed:c7:2d:75:12:31:0c:8c:c7:8c:3f:54:a9:a8:49:e1:44
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IN, ST=Maharashtra-State, L=Mumbai, O=Internet Widgits Pty Ltd, OU=Comps, emailAddress=vedansh.savla@somaiya.edu
        Validity
            Not Before: Jan 31 09:37:41 2025 GMT
            Not After : Jan 31 09:37:41 2026 GMT
        Subject: C=IN, ST=Maharashtra-State, L=Mumbai, O=Internet Widgits Pty Ltd, OU=Comps, emailAddress=vedansh.savla@somaiya.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b9:59:0f:df:f1:7b:1e:f1:8d:3e:d1:dd:2a:07:
                    01:ee:74:f9:f0:51:f8:da:ed:1c:66:34:36:31:80:
                    d3:88:58:27:de:fe:62:31:58:e7:95:c4:81:8c:70:
                    c9:bf:35:9b:f4:fb:84:45:03:5b:0a:ae:16:b8:72:
                    83:c0:d6:37:71:5a:7e:ca:3d:a7:0c:02:3f:23:76:
                    fe:80:a9:da:d7:03:8d:73:fb:08:32:70:cc:7e:c4:
                    01:43:03:1a:2d:8a:44:bf:48:4e:1b:84:55:16:ba:
                    7e:cd:7f:61:6c:47:21:2b:f1:46:30:b0:1e:42:35:
                    d8:2a:76:c6:e8:14:21:6d:c0:16:d9:40:9c:b7:db:
                    01:99:34:7a:0d:bc:19:d6:5d:e4:3d:fb:e9:77:cf:
                    d9:bd:15:b0:d0:25:cd:63:b9:bc:b2:de:0b:55:90:
                    51:42:19:0f:1b:21:c9:03:0a:bd:78:41:8e:ca:30:
                    59:67:3f:bc:34:41:1f:7a:da:66:79:46:d2:a3:fc:
                    c5:35:bc:4f:f0:1f:e5:42:95:cb:94:f5:f9:f8:25:
                    dd:40:92:64:ae:6e:30:0e:57:df:d4:3d:cd:01:c7:
                    b6:9c:8e:d7:e0:8a:bb:04:ca:13:80:ef:9c:84:ea:
                    0a:8b:18:b0:a9:d4:91:47:15:7d:07:9a:7d:ed:d4:
                    03:d1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                3D:D0:16:B5:F5:DA:6F:F3:42:D0:E7:42:A2:EE:69:7D:12:DA:6A:7E
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        4d:2e:30:57:46:72:9b:33:d0:fa:fc:2b:bb:9f:69:aa:4a:13:
        95:d2:8b:65:1e:03:e4:55:f4:5a:12:c3:c0:90:33:69:5e:a6:
        91:b1:06:93:68:9f:cb:48:95:3f:23:7c:cf:88:fa:a2:12:6f:
```

```
                    0a:8b:18:b0:a9:d4:91:47:15:7d:07:9a:7d:ed:d4:
                    03:d1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                3D:D0:16:B5:F5:DA:6F:F3:42:D0:E7:42:A2:EE:69:7D:12:DA:6A:7E
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        4d:2e:30:57:46:72:9b:33:d0:fa:fc:2b:bb:9f:69:aa:4a:13:
        95:d2:8b:65:1e:03:e4:55:f4:5a:12:c3:c0:90:33:69:5e:a6:
        91:b1:06:93:68:9f:cb:48:95:3f:23:7c:cf:88:fa:a2:12:6f:
        31:6d:28:2c:fc:91:80:af:ee:eb:15:a0:c3:94:90:b4:13:3c:
        95:5f:f9:3f:8c:35:36:7c:16:8d:5f:7d:7a:b6:1f:e9:3c:81:
        86:82:3d:6d:7a:16:80:c9:95:a1:f6:20:1d:0b:74:05:65:39:
        3c:b6:20:de:e8:92:00:43:15:6c:25:3a:ba:5b:6b:91:75:bf:
        dc:e1:af:14:d9:d1:4d:c0:4a:e4:32:b2:d2:3f:af:3d:20:79:
        19:45:35:40:09:56:ab:b7:66:4c:2b:a9:2c:d6:18:b3:79:b5:
        1d:7c:eb:a0:5c:96:ba:76:34:f4:31:50:ab:c0:82:a9:df:87:
        11:3e:94:c2:8f:44:44:36:12:2c:3d:71:53:99:ea:52:20:94:
        17:df:4e:e0:c3:fb:d6:dd:1d:1c:21:72:ba:e6:dd:7c:f4:a3:
        01:19:23:5d:43:55:0c:6f:2a:c1:3f:b2:a9:5b:4e:f5:50:e0:
        fe:e4:d1:b1:f5:07:63:7c:60:7c:01:8b:f4:01:d0:73:ff:07:
        b0:30:a9:52

:\Users\student\Desktop\IS>
:\Users\student\Desktop\IS>_
```

9. Digital signature



```
C:\Users\student\Desktop\IS>openssl genrsa -aes128 -passout pass:vedansh -out private.pem 4096
```

private - Notepad

File Edit Format View Help

HB3saQICCAAwDAYIKoZIhvcNAgkFADAdBglghkgBZQMEAQIEECFYb41Mq85tig8E
/7nE87oEgglQFw+yJDB4kpA8n72T5w+wOGYrFJiP4YHH5NZJaMqXwBtv7y1fr4wk
hsaUyv/vU0oGBHxEuI96vfRkKSsTCo/RQUCsmPZalyxYpTGPH3dbUgp0R9j2R+Xp
mxr5AOM4YJIj7053SoHlIynam/aK0IG6B1T7eIUCD7771e/1Y0VGaQjEjR6JKYSk
v/ImIxgB6yM3sv1hoy3+FC44eIkboEIoomayb8RnSl1suF4e7WRaXvfJP3gKj8AN
AYnOwXyytzEw+iDUwESnJiSFLL8qunLdeZxHR1P9rpNmgmNlTSaeZl0wKXmBPiBZ
HoQJJ84lpJCi0y+iLfInZdz9w89b7O1cMbY+iNfYDuw3FdRvObCg+S6Aom2L2Jq6
q4DUHZOSlfqsQONpwmJ5rA1DU85uGkM1lImvazxj2zkv4SchdWnbR82WzQSHnftV
tJZ8eXjHH7wFRIP8lcntE4BNzA0tv/koBoqEqxgF5pIj681+JZkDqf2HABT/GIHX
jLymLIFrjUCA3M3FmYeEXzG9EKGtl1d3gmZMcZyiVLJqrht9dg+YEUfHsSZ696se
1OG2UXjscVb6k2G9M4cOtoa+yDkHwzVMQS6c74yfYEXm7iiCby3LkayLvrw78eO8
76VIDv/k7Dh8o+shzodB2RJ1eMa7IrGVhUfkLNJ/gu6a2WZQzu+IXsqz31EiOZqS
bPzveqfz1PoST1VvJ8zJRCUZ80eZUQpjBL7NFWuDtlYi6DsrqU8r/XTvsujoy09A
u3M9Eyse3Uxn8UifXZVFTvNYbtlBjfTd5+kjNZt7HYGGWKSMQExXV9Bf9B8DC8gj
+zCi47FBnIRjFJN61KqSlKmX/30fdYM9rktsys1wk6Gi025J+ydRfyMztGzeA5en
tKWo1Cw2BgpYS0zAbG3KhbRKcJlNLpfKngMBBgYqcubHqjlUMVfVFsS5a01SMC3d
xuSz3R4eZaF7QL+umy/IU7dQxMkR68gohCV85gcodiSe3GcZWnS+KeNBmM1ADoHN
/mQ9Qdl0np8T2dAOkUTd21Z31dIagRUvQZhuFfcCI7eMJ3/+dDr1Bm1wJUOWOgWh
TfFqW+3efddaa8cY+6NNy9ieBdaaFKNUFTsyBKFbJD84RIOjxXxf5J396M1qu+X5
GKF78CS2xw4zWO49kRig3hb9dbIrT8I385+1z9SqfuoF/A0jqfMzstF2ZZ5XDUsZ
6cOImAXoyzeSJH8MRBRFUgHcF98CGuo3E3eSDf+0K5yKt7804tXHdQ8TdhH/YAnM
bCFdrxxXxjrCWl4KYMzsK8aOabRBwHg1Ur1h4aSfI8nRfK+GLZL3dt7IAWqtbnFI
q7Sus0n3fijg2yY7QLZjl4rNnosnINupzdGGGY43iThnyVJIOJBM0mdwBSUeQxs/
eaeaGGRO2GwYaU30GWq8qmN5UEfDIzyk0TjUsSaDGrbe7rAhBMH2Umc8dLcqi39s
E/u0/TVxj5t4Qk5AbzMRDftn+FINH9T2vhdgHY1wnqBwOYIoHWlKm78TtWfNQTBB
FJiCUbXHujCDODVudV+bnJY02DI2Fh/sCxj8S0mnp4abYGW8c3y/UIWq0vrKfdUN
IDYBiI1jigmSjTFFW4FwK2Xg53VOhT1AvK+OlufcrGOWkJMcpjF2f8yMAeeOA49p

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

10. Digital signature

```
C:\Users\student\Desktop\IS>openssl rsa -in private.pem -passin pass:vedansh -pubou
t -out public.pem
writing RSA key
```

public - Notepad

File Edit Format View Help


```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAxTj8U4X4nQnRWVEIz6pS
pJ9arxz47uy/IEdP1q4NORfMbuw1w0fBjyeGaQit57jksHsi9RIyuReV3ouCZVzf
CqAC6svWgva1vsAxweM8NQOOgqRQ/r+Ci6fmGyvefQSO2xQV82bqYcI97aA9gM59
fqyob6Mn1kAgFrQ64IKApd5N7Xpbg4GjwiAdAzx19gA2dgF4aXW6DAUgx/0YAFXu
cmP/VcBPUX4bJWdLJeRausC2jDN5hHfULIFzUn6leOQM8MNaYNLpR6UZNfEx4p4x
rDcqE4sgYT8keq36qXObC2UqGnBvkad+6sinb4mwX6H/7RAZRs4oU6fuQmsIertA
EBqVgTnN9XxXNWr712JyLwwQecQEaa/hTraYz8EVRSuUl22SLH6skejSshwYAhtw
y0LRycatqaJ7pZ53byPzPZugiSKy54rG0Ysokinx1glxCluM3sYYLY7Qut6g+xcV
qQG3I1SUopguGloN2K1S71N3+//HGPmpTAcIM0kaxNtFKEWkzx2PspgMZOElrKj0
C0PDhVRpQDsLi1jHDdzI6jAyOI0BKuDOde4ErDvXHPFwgMIcde1u8RpW/D2gIazf
be/wRAsodyFJVU8LoTFjO27n/3o/Te7ONRfqI2RO7su3x3+qsCFFUy35sDMVn4sc
sM5wTItRxxd28boTWg8F7tUCAwEAAQ==
-----END PUBLIC KEY-----
```


11. Create a text file

```
C:\Users\student\Desktop\IS>openssl dgst -sha256 -sign private.pem -out sign.sha256 text.txt
Enter pass phrase for private.pem:
```

text - Notepad

File Edit Format View Help

```
hello i am vedansh
```

12. Generate the signature of a file

```
C:\Users\student\Desktop\IS>openssl base64 -in sign.sha256 -out sign.bin

C:\Users\student\Desktop\IS>
```

13. Verify the signature

Win64 OpenSSL Command Prompt

```
:\Users\student\Desktop\IS>openssl base64 -d -in sign.bin -out sign.sha256

:\Users\student\Desktop\IS>openssl dgst -sha256 -verify public.pem -signature sign.sha256 tex
t.txt
Verified OK

:\Users\student\Desktop\IS>
```

RSA Implementation details:

```
import math
import random

def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

def generate_prime_number():
    while True:
        num = random.randint(100, 1000)
        if is_prime(num):
            return num

# Generate two distinct primes
p = generate_prime_number()
q = generate_prime_number()
while p == q:
    q = generate_prime_number()

print(f"First prime (p): {p}")
print(f"Second prime (q): {q}")

n = p * q
print(f"\nModulus (n = p * q): {n}")

phi = (p - 1) * (q - 1)
print(f"\nEuler's totient ϕ(n): {phi}")

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def choose_e(phi):
    e = 65537
    if e >= phi:
        for i in range(3, phi, 2):
            if gcd(i, phi) == 1:
```

```
                return i
    return e

e = choose_e(phi)
print(f"\nPublic exponent (e): {e}")

def mod_inverse(a, m):
    old_r, r = a, m
    old_s, s = 1, 0
    old_t, t = 0, 1

    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t

    if old_r != 1:
        return None  # Inverse doesn't exist
    return old_s % m

d = mod_inverse(e, phi)
print(f"Private exponent (d): {d}")

def encrypt(message_str, e, n):
    # Convert message to bytes
    bytes_message = message_str.encode('utf-8')
    m = int.from_bytes(bytes_message, byteorder='big')
    if m >= n:
        raise ValueError("Message is too long to encrypt with the current
modulus.")
    c = pow(m, e, n)
    return c

def decrypt(c, d, n):
    m = pow(c, d, n)
    # Convert integer back to bytes
    byte_length = (m.bit_length() + 7) // 8
    bytes_message = m.to_bytes(byte_length, byteorder='big')
    return bytes_message.decode('utf-8')

# Example usage
try:
    message = input("\nEnter a message to encrypt: ")
```

```
    ciphertext = encrypt(message, e, n)
    print(f"\nEncrypted ciphertext (c = m^e mod n): {ciphertext}")

    decrypted_message = decrypt(ciphertext, d, n)
    print(f"Decrypted message: {decrypted_message}")
except ValueError as e:
    print(f"Error: {e}")
```

Output:

```
Vedansh@Vedansh MINGW64 ~/OneDrive/Desktop/KJSSE,
$ C:/Users/mites/AppData/Local/Programs/Python/Py
First prime (p): 491
Second prime (q): 929

Modulus (n = p * q): 456139

Euler's totient ϕ(n): 454720

Public exponent (e): 65537
Private exponent (d): 287873

Enter a message to encrypt: f

Encrypted ciphertext (c = m^e mod n): 176699
Decrypted message: f
```

Post Lab Questions:

1. Explain and implement RSA Algorithm.

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

$$\phi(n) = (7-1) \times (11-1) = 6 \times 10$$
$$= 60$$
$$(e \times d) \bmod \phi(n) = 1$$
$$(13 \times d) \bmod 60 = 1$$
$$d = 37$$
$$\therefore \text{key} = 37$$

2. List the algorithms used in this experiment for OpenSSL application.

The algorithms used in this experiment for the OpenSSL application include:

1. **RSA (Rivest-Shamir-Adleman)** – Used for key pair generation, encryption, and digital signatures.

2. **SHA (Secure Hash Algorithm)** – Used for creating cryptographic hash functions in digital signatures.
3. **AES (Advanced Encryption Standard)** – Often used for encrypting data in SSL/TLS communications.
4. **MD5/SHA-256** – Used for message integrity verification.
5. **HMAC (Hash-based Message Authentication Code)** – Used for ensuring message authenticity.
6. **X.509 Certificate Standard** – Used for creating and verifying digital certificates.
7. **Diffie-Hellman (Optional in SSL/TLS)** – Used for secure key exchange.
8. **Digital Signature Algorithm (DSA)** – Used as an alternative for digital signatures

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.