

### 1. Use openssl to generate RSA public/private key:

```

root@ip-172-31-26-78:~/Desktop/lab5# ls
root@ip-172-31-26-78:~/Desktop/lab5# openssl genrsa -aes128 -out private.pem 1024 ←
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
root@ip-172-31-26-78:~/Desktop/lab5# ls
private.pem
root@ip-172-31-26-78:~/Desktop/lab5# openssl rsa -in private.pem -pubout > public.pem ←
Enter pass phrase for private.pem:
writing RSA key
root@ip-172-31-26-78:~/Desktop/lab5# ls
private.pem public.pem
root@ip-172-31-26-78:~/Desktop/lab5# openssl rsa -in private.pem -text -noout ←
Enter pass phrase for private.pem:
Private-Key: (1024 bit, 2 primes)
modulus:
00:db:14:e1:e1:46:87:44:b1:ea:8f:c1:84:ab:0e:
a1:38:40:43:0c:cd:0f:c1:93:af:7e:88:e8:fb:a9:
bd:04:11:04:fe:ab:19:6d:98:2c:27:5b:99:9e:52:
d4:af:69:c1:a7:16:04:84:06:15:c8:66:8f:02:4f:
e0:b6:28:04:0f:4f:48:03:7c:b9:18:69:43:cd:f6:
a0:73:3a:81:f6:3a:c5:12:04:77:df:f4:80:01:6c:
13:51:9e:25:9f:74:16:75:4c:fd:1c:68:67:54:29:
e3:e5:26:eb:90:a5:69:84:61:d6:77:27:aa:e2:58:
82:f6:2a:87:32:95:34:13:97
publicExponent: 65537 (0x10001)
privateExponent:
00:d9:10:d9:b5:41:12:29:88:36:a5:f4:d1:a4:42:
bc:8e:65:6d:89:b3:6b:d7:1a:a3:19:36:41:d1:88:
1e:55:77:ed:97:de:a3:35:29:3f:26:47:4e:ef:2e:
96:bb:83:6a:47:28:03:94:94:0a:05:22:68:b3:9b:
bd:43:3e:65:f5:87:29:09:10:7f:ba:21:c3:7c:bc:
a3:d5:55:19:06:63:8a:da:e6:a9:00:09:03:b9:2c:
15:42:19:34:2e:57:3e:a6:ef:f1:68:ec:42:eb:8c:
9f:5c:ea:9f:3f:5b:f9:59:10:6f:ef:7d:31:9e:04:
bf:17:c0:f8:64:c1:24:b1
prime1:
00:fb:a1:57:97:49:16:6c:30:d6:5d:37:1a:bd:f0:
c5:1e:20:c8:84:6e:41:f9:99:60:c7:5b:cd:d0:8b:
f7:0c:1f:3a:39:ac:2a:3e:36:34:56:87:c1:8b:bd:
94:4a:69:52:80:46:02:96:ef:4a:4c:11:6b:6f:12:
18:23:16:b0:35
prime2:
00:de:e2:d7:28:f3:f0:b4:c1:f6:a3:e1:e3:fb:ab:
ba:06:07:df:3c:a3:38:7e:3b:b4:85:a8:0b:3d:a0:
d1:b3:a6:36:5f:a9:a1:d9:26:b5:85:d6:ee:d3:13:
43:88:aa:0b:40:f1:bf:6d:a5:fc:05:2b:53:1c:99:
64:03:3d:46:1b
exponent1:
62:83:a6:41:4a:92:06:c5:90:97:6a:9d:83:a3:91:
e8:db:88:e5:70:20:50:45:26:48:a6:be:80:59:60:
dd:54:14:7d:a3:d3:de:0b:66:ae:41:f4:1a:7b:9c:
65:3f:b7:d7:14:91:ca:f9:e0:42:ab:c7:3d:f2:fd:
94:2f:3b:c9
exponent2:
03:1e:5f:6e:56:43:9e:5b:0c:9c:86:a2:9f:01:df:
1f:78:7a:ba:29:7a:7f:el:ab:fb:f6:c6:f9:7c:7:
92:60:ef:ff:d4:aa:cc:14:0b:94:2c:d6:b8:a3:5b:
5d:80:2b:8d:24:14:ce:9e:f4:f0:c9:93:67:22:00:
51:59:1b:23
coefficient:
18:b6:0f:7e:be:2c:48:60:70:82:08:e3:88:04:d0:
61:f4:4d:a0:c0:c1:f9:7e:06:1c:95:fb:b3:6a:61:

```

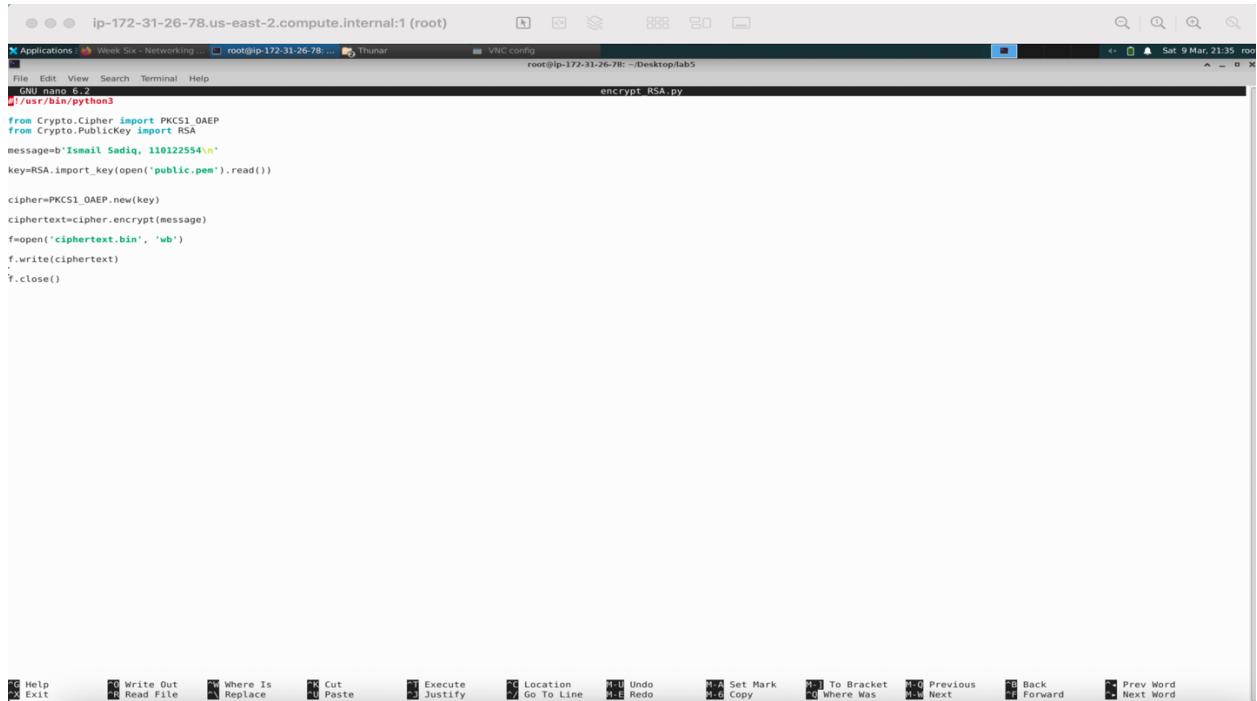
  

```

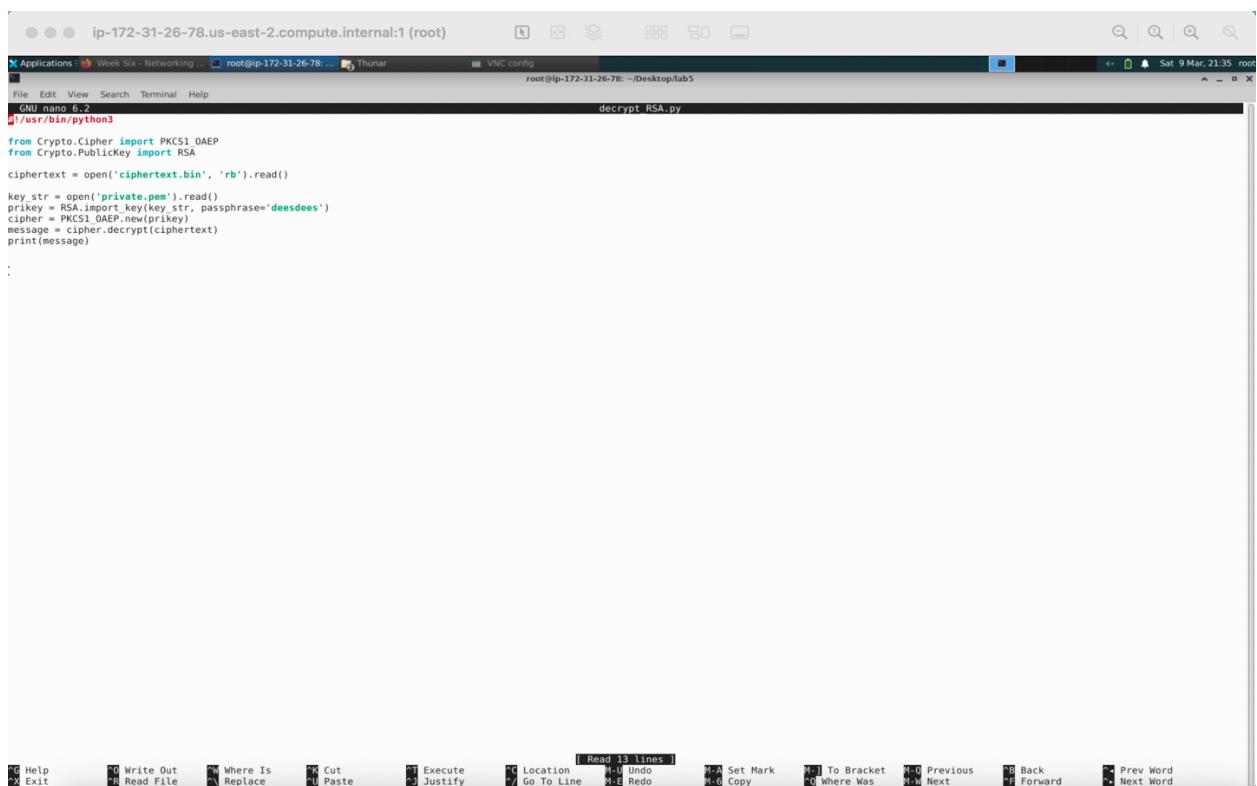
root@ip-172-31-26-78:~/Desktop/lab5# openssl rsa -in public.pem -text -noout ←
Public-Key: (1024 bit)
Modulus:
00:db:14:e1:e1:46:87:44:b1:ea:8f:c1:84:ab:0e:
a1:38:40:43:0c:cd:0f:c1:93:af:7e:88:e8:fb:a9:
bd:04:11:04:fe:ab:19:6d:98:2c:27:5b:99:9e:52:
d4:af:69:c1:a7:16:04:84:06:15:c8:66:8f:02:4f:
e0:b6:28:04:0f:4f:48:03:7c:b9:18:69:43:cd:f6:
a0:73:3a:81:f6:3a:c5:12:04:77:df:f4:80:01:6c:
13:51:9e:25:9f:74:16:75:4c:fd:1c:68:67:54:29:
e3:e5:26:eb:90:a5:69:84:61:d6:77:27:aa:e2:58:
82:f6:2a:87:32:95:34:13:97
Exponent: 65537 (0x10001)
root@ip-172-31-26-78:~/Desktop/lab5#

```

**2.** In this problem, you need to practice RSA encryption and decryption:



```
GNU nano 6.2
#!/usr/bin/python3
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
message=b'Ismail Sadiq, 110222554\n'
key=RSA.import_key(open('public.pem').read())
cipher=PKCS1_OAEP.new(key)
ciphertext=cipher.encrypt(message)
f=open('ciphertext.bin', 'wb')
f.write(ciphertext)
f.close()
```



```
GNU nano 6.2
#!/usr/bin/python3
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
ciphertext = open('ciphertext.bin', 'rb').read()
key_str = open('private.pem').read()
prikey = RSA.import_key(key_str, passphrase='deesdees')
cipher = PKCS1_OAEP.new(prikey)
message = cipher.decrypt(ciphertext)
print(message)
```

```

root@ip-172-31-26-78:/root# ls
crypto decrypt_RSA.py encrypt_RSA.py ender_AES.py hash_comp.py private.pem public.pem sign_RSA.py verify_RSA.py
root@ip-172-31-26-78:/root# nano encrypt_RSA.py
root@ip-172-31-26-78:/root# nano decrypt_RSA.py
root@ip-172-31-26-78:/root# python3 encrypt_RSA.py
root@ip-172-31-26-78:/root# ls
cipher.py crypto.py ender_AES.py hash_comp.py private.pem public.pem sign_RSA.py verify_RSA.py
root@ip-172-31-26-78:/root# hexdump -C ciphertext.bin
00000000 95 02 a1 5f 07 8e c3 b6 4b 1a 7c f4 a1 44 89 [....K.|.D.|.
00000010 31 19 ff ad db 58 be 93 6e 14 c9 e8 c7 14 59 ae [....X..n....Y.|.
00000020 31 19 ff ad db 58 be 93 6e 14 c9 e8 c7 14 59 ae [....X..n....Y.|.
00000030 a4 47 85 11 78 3e 32 c2 20 8e 98 94 79 58 6e 9a [..G..x2.....yXn.|.
00000040 6d 77 7f 39 e4 95 8e dd 1f 03 46 b2 e7 e7 ba 14 [..W..g3.....Wh|.
00000050 eb 77 a7 39 e4 95 8e dd 1f 03 46 b2 e7 e7 ba 14 [..W..g3.....Wh|.
00000060 37 b8 5f 70 e3 1f 5e 5b b3 a6 60 b9 68 12 bc [7..p..p...'.h.|.
00000070 08 d6 fd 17 00 a6 f6 57 b7 f4 8a 91 a4 50 88 e2 [.....W.....P..|.
00000080
root@ip-172-31-26-78:/root# python3 decrypt_RSA.py
b'Ismail Sadiq, 11012254\n'
root@ip-172-31-26-78:/root# 

```

### 3. (optional)

4. In this problem, you will use Diffie-Hellman with authentication to protect the client-server communication. Implement the following functionalities:

#### Server.py:

```

import socket
from time import sleep
from Crypto.Random.random import getrandbits
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256, MD5
from Crypto.Util.Padding import pad, unpad

def diffie_hellman():
    p = 25822498780869085896559191720030118743297057928292235128306593565406476220168411946296453532801378314
    35903171972747559779
    g = 2
    x = getrandbits(400)
    X = pow(g, x, p)
    return p, g, x, X

def key_exchange(sock):
    p, g, x, X = diffie_hellman()
    sock.send(p.to_bytes(256, 'big'))
    sock.send(g.to_bytes(256, 'big'))
    sock.send(X.to_bytes(256, 'big'))
    Y = int.from_bytes(sock.recv(256), 'big')
    shared_key = pow(Y, x, p)
    shared_key = shared_key.to_bytes(128, 'big')
    shared_key = MD5.new(shared_key).digest()
    return shared_key

"server.py" 79L, 2450C

```

```

def aes_encrypt(message, key):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(pad(message.encode('utf-8'), 16))
    tag = SHA256.new(ciphertext).digest()
    return ciphertext, tag, iv

def aes_decrypt(ciphertext, key, tag, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted = unpad(cipher.decrypt(ciphertext), 16)
    if SHA256.new(ciphertext).digest() != tag:
        raise Exception("Message integrity verification failed!")
    return decrypted.decode('utf-8')

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 12345))
    server.listen(1)

    print("Server listening on port 12345...")
    connection, address = server.accept()
    print(f"Connection from {address}")
    shared_key = key_exchange(connection)

```

47,0-1 50%

```

while True:
    message = input("You: ")
    if message.lower() == 'exit':
        break

    ciphertext, tag, iv = aes_encrypt(message, shared_key)
    connection.send(ciphertext)
    sleep(1)
    connection.send(tag)
    sleep(1)
    connection.send(iv)

    received_ciphertext = connection.recv(1024)
    received_tag = connection.recv(32)
    received_iv = connection.recv(32)
    try:
        received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
        print(f"Client: {received_message}")
    except Exception as e:
        print(f"Error: {e}")

    connection.close()
    server.close()

if __name__ == "__main__":
    start_server()

```

77,0-1 Bot

## Client.py:

```
import socket
from time import sleep
from Crypto.Random.random import getrandbits
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES
from Crypto.Hash import SHA256, MD5
from Crypto.Util.Padding import pad, unpad

def diffie_hellman(sock):
    p = int.from_bytes(sock.recv(256), 'big')
    g = int.from_bytes(sock.recv(256), 'big')
    Y = int.from_bytes(sock.recv(256), 'big')
    x = getrandbits(400)
    X = pow(g, x, p)
    sock.send(X.to_bytes(256, 'big'))
    shared_key = pow(Y, x, p)
    shared_key = shared_key.to_bytes(128, 'big')
    shared_key = MD5.new(shared_key).digest()
    return shared_key

def aes_encrypt(message, key):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(pad(message.encode('utf-8'), 16))
    tag = SHA256.new(ciphertext).digest()
    return ciphertext, tag, iv

def aes_decrypt(ciphertext, key, tag, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted = unpad(cipher.decrypt(ciphertext), 16)
    if SHA256.new(ciphertext).digest() != tag:
        raise Exception("Message integrity verification failed!")

    return decrypted.decode('utf-8')

def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 12345))
    shared_key = diffie_hellman(client)
    while True:
        received_ciphertext = client.recv(1024)
        received_tag = client.recv(32)
        received_iv = client.recv(32)

        received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
        print(f"Ciphertext: {received_ciphertext}\nTag: {received_tag}\nShared key: {shared_key}\nDecrypted text: {received_message}")

        print(f"Server: {received_message}")

        message = input("You: ")

        if message.lower() == 'exit':
            break
```

1,1

Top

47,0-1

69%

```

while True:
    received_ciphertext = client.recv(1024)
    received_tag = client.recv(32)
    received_iv = client.recv(32)

    received_message = aes_decrypt(received_ciphertext, shared_key, received_tag, received_iv)
    print(f"Ciphertext: {received_ciphertext}\nTag: {received_tag}\nShared key: {shared_key}\nDecrypted text: {received_message}")

    print(f"Server: {received_message}")

    message = input("You: ")

    if message.lower() == 'exit':
        break

    ciphertext, tag, iv = aes_encrypt(message, shared_key)
    client.send(ciphertext)
    sleep(1)
    client.send(tag)
    sleep(1)
    client.send(iv)

client.close()

if __name__ == "__main__":
    start_client()

```

65,1

Bot

## Output:

```

[03/08/24]seed@VM:~/.../lab5-doc$ python3 server.py
Server listening on port 12345...
Connection from ('127.0.0.1', 38964)
You: hi there, how are you
Client: oh wow, long time, I'm good
You: this is an encrpyted text
■

```

```

[03/08/24]seed@VM:~/.../lab5-doc$ python3 client.py
Ciphertext: b'\xc0Z\xd8\xal\xc5\x12h.\xe8B\x94\x8eY+\x9a(\x86|\xb2b\x95\x18\xbc\x8cN\xedl\x19I\xc7!('
Tag: b'\x0cK\xcdm[\xd5\x80.\xd04e\x8f\xbf?j\xdfL\xb3,\x8e\x08 a\x04\x1e\xf3\xfd@x\xc9\xb1\n'
Shared key: b'rW\x9z\x0cH\xf5p\x85Nh\x1bx\xfd\xb7y'
Decrypted text: hi there, how are you
Server: hi there, how are you
You: oh wow, long time, I'm good
Ciphertext: b'\x9b\x85\x1aT\x89\x8b\x06\x83;Qd\xf4y\xbb\x96\xd3\xd10\x06\n"\x82za<"]
Tag: b'\t\xe1J2w\xd9\xf5l\x88\xad\x08\x91%\xbao\r\xe2\x9f|\xf0\x87\x8f\xc9^\'\xe6\xcc\xdd{\x0fR\xc6'
Shared key: b'rW\x9z\x0cH\xf5p\x85Nh\x1bx\xfd\xb7y'
Decrypted text: this is an encrpyted text
Server: this is an encrpyted text
You: ■

```