Lab Assignment 4

AP21110010302
Krish Srivastava
CSE – E
Network Security – CSE 315L

1. Write a program to ensure sender authentication, integrity and confidentiality in client server communication using asymmetric key based mechanism.

Hint:

a. Generate public private key pair

b. Use protocol in slide 19 (include mac in it)

SERVER.py

```
import socket
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256

def create_key_pair():
    private_key = RSA.generate(2048)
    public_key = private_key.publickey()
    return private_key, public_key

def load_public_key_from_pem(pem):
    return RSA.import_key(pem)

def serialize_public_key(public_key):
    return public_key.export_key().decode()

def encrypt_message(public_key, message):
    cipher = PKCS1_OAEP.new(public_key)
    return cipher.encrypt(message.encode()).hex()

def decrypt_message(private_key, encrypted_message):
    cipher = PKCS1_OAEP.new(private_key)
    return cipher.decrypt(bytes.fromhex(encrypted_message)).decode()

def sign_message(private_key, message):
    h = SHA256.new(message.encode())
```

```
    signer = PKCS1_v1_5.new(private_key)
    return signer.sign(h).hex()

def verify_signature(public_key, message, signature):
    h = SHA256.new(message.encode())
    verifier = PKCS1_v1_5.new(public_key)
    return verifier.verify(h, bytes.fromhex(signature))

def send_data(data, connection):
    connection.send(data.encode())

def receive_data(connection):
    return connection.recv(1024).decode()

def exchange_messages(connection, private_key, other_public_key):
    while True:
        message = receive_data(connection)
        print("Received (Encrypted):", message, "\n")
        signature = receive_data(connection)
        print("Received (Signature):", signature, "\n")

        if verify_signature(other_public_key, message, signature):
            message = decrypt_message(private_key, message)
            print("Decrypted:", message, "\n")
            print("Signature Verification: Successful", "\n")
        else:
            print("Signature Verification: Failed", "\n")
            continue

        outgoing_message = input("You: ")
        outgoing_message = encrypt_message(other_public_key, outgoing_message)
        print("Sent (Encrypted):", outgoing_message, "\n")
        signature = sign_message(private_key, outgoing_message)
        print("Sent (Signature):", signature, "\n")
        send_data(outgoing_message, connection)
        send_data(signature, connection)

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 12345))
server_socket.listen(1)
print("Server is listening...")

connection, address = server_socket.accept()
print("Connection from:", address)

private_key, public_key = create_key_pair()

send_data(serialize_public_key(public_key), connection)
```

```
other_public_key = load_public_key_from_pem(receive_data(connection))

exchange_messages(connection, private_key, other_public_key)
```

CLIENT.py

```
import socket
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256

def create_key_pair():
    private_key = RSA.generate(2048)
    public_key = private_key.publickey()
    return private_key, public_key

def load_public_key_from_pem(pem):
    return RSA.import_key(pem)

def serialize_public_key(public_key):
    return public_key.export_key().decode()

def encrypt_message(public_key, message):
    cipher = PKCS1_OAEP.new(public_key)
    return cipher.encrypt(message.encode()).hex()

def decrypt_message(private_key, encrypted_message):
    cipher = PKCS1_OAEP.new(private_key)
    return cipher.decrypt(bytes.fromhex(encrypted_message)).decode()

def sign_message(private_key, message):
    h = SHA256.new(message.encode())
    signer = PKCS1_v1_5.new(private_key)
    return signer.sign(h).hex()

def verify_signature(public_key, message, signature):
    h = SHA256.new(message.encode())
    verifier = PKCS1_v1_5.new(public_key)
    return verifier.verify(h, bytes.fromhex(signature))

def send_data(data, connection):
    connection.send(data.encode())

def receive_data(connection):
    return connection.recv(1024).decode()
```

```
def exchange_messages(connection, private_key, other_public_key):
    while True:
        outgoing_message = input("You: ")
        outgoing_message = encrypt_message(other_public_key, outgoing_message)
        print("Sent (Encrypted):", outgoing_message, "\n")
        signature = sign_message(private_key, outgoing_message)
        print("Sent (Signature):", signature, "\n")
        send_data(outgoing_message, connection)
        send_data(signature, connection)

        message = receive_data(connection)
        print("Received (Encrypted):", message, "\n")
        signature = receive_data(connection)
        print("Received (Signature):", signature, "\n")

        if verify_signature(other_public_key, message, signature):
            message = decrypt_message(private_key, message)
            print("Decrypted:", message, "\n")
            print("Signature Verification: Successful", "\n")
        else:
            print("Signature Verification: Failed", "\n")
            continue

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 12345))

server_public_key_pem = receive_data(client_socket)
server_public_key = load_public_key_from_pem(server_public_key_pem)

client_private_key, client_public_key = create_key_pair()

send_data(serialize_public_key(client_public_key), client_socket)

exchange_messages(client_socket, client_private_key, server_public_key)
```

OUTPUT

```
PS C:\Users\krish\OneDrive\Desktop\6th Sem\Network Security CSE 315L\Lab 4> & C:/Users/krish/AppData/L
ocal/Programs/Python/Python312/python.exe "c:/Users/krish/OneDrive/Desktop/6th Sem/Network Security CS
E 315L/Lab 4/Server2.py"
Server is listening...
Connection from: ('127.0.0.1', 55126)
Received (Encrypted): cefb486e2bf8192d42d846b14716ae9ef3f321d617994c7ca6145edaa949d3e320a3bf7c09d5b2a0
6ef35b31748fd813d4994846a3e60b12f0bbc2ba424bad7520438f8a92aecfb4ec43e01ef92067a9e4722cd152ff7f17b53e18
2e92b0799de8cf349300f3a94eb05bcf4814d7cfc3f65a3e9c3d27c345684a5ce8f883438226ba5a3091a2505903b779dfb398
d0c4f610ff25ec13c9e10de49a9af36c4374f7fc6b2618674166d740fe3b0832cc8b5ce0eec3dbabfab71e098ec49c8041acdd
410b13ea4784c123833906000e0d0fd3430318e21107f610be313796a1dede64cb7b7d3bfcf510b4f636f5f097f4bf4e6e5d52
869d3c575885c7b9de7e9196

Received (Signature): 8ce370686f5a9c237f5c3fb422e32c705fcb87e9a0a7fb5a5dbab60a02507b8dd51ad42dcdd0799b
8b2d70faece0df79aa23579fd4c0bb5c9327998774c36cfda087f9aa4f0a27f0f68dd6ce9fe0dcd31df9681a977655f06225e1
e31d1593e83887e0786c1ed6b46eac08ecab99a33debee50dddd42d5f7a0fe64f0b17000d2a465806fc31403c28a5cd79c9120
42cfe5c2290038b617285e2211a378f924b639d4da7d33660cf6fdbaefe5127cba7cd9f61635c0852c761cf1b9ca83bbad79ce
f19c9c9abb2d78173685fbe054f0d01717f04a221fb0b4358020d7984ce831702047766c32733422c5b0dfaec91391854b881b
07f0cf59c6ae6a2cbf64e788

Decrypted: hello krish this side

Signature Verification: Successful

You: hello server here
Sent (Encrypted): 6ca1ef27a5fb028849674b98e9801cf5fd77d3e5a36886d464646363b2ab8cbc8a6d547a9dbece821ce0
29dcd1ea4758583fea44ea445bcbf67a491668faf1faae612dd8c52e3a27897051b14bf17acd7ffee2fe1b495f80fa0ac96f14
d10af40bc9a97403b32f1e99e9b9a105e002cd8c85b70caf47c72b476b50cc5f1b3a9a03b22c78705d18e82469cef80a63c751
b02954006f7c24c6dc4b276b5b3f9e8a027d7c6ba8d47a6435b89766b46358621f45534dafdd1b4748970d9941a7def9e264a6
ea1673e968cc0c1377f06129558a087e91de95dab083acc4ba92d22d2d22bd42593473d4cfc30540858f8dee4675d88ac0b9f4
f0c1beee484334dad9a2

Sent (Signature): 4f43c23ec9543f3540befd229655f02bf518920e357eeb8df16746525d7114acb77c5c5c7afdc1d5ab97
823b0fcf9c6c114b83ec7efa1fceb2d2e2785f130102604d6a2120724ad8821ffb883ca16e0e475534c43be382e9df7144d5cf
641b3ad62d7c9e8da1f630a4afcbc5650177d4970e12492fdff8ed7b31864e1434b51021d8bae84f23ca3ed0cf8e2d0c87d848
ba531bcb57311a5b6d1753549738f55485b258d188f37aae687c7c3ab36e5acd3f149b1bebf2f0d69f9bedb07921f7f0a98ac4
55ccb1bca7a8ccabbac4a40cf0018cc45135f0daeff107fb585f3bae7bfc8ccca0278fe47759af6e3ea1ec3f89876c9ae90462
d8ad780a6537db153437
```

```
PS C:\Users\krish\OneDrive\Desktop\6th Sem\Network Security CSE 315L\Lab 4> & C:/Users/krish/AppData/
Local/Programs/Python/Python312/python.exe "c:/Users/krish/OneDrive/Desktop/6th Sem/Network Security
CSE 315L/Lab 4/Client2.py"
You: hello krish this side
Sent (Encrypted): cefb486e2bf8192d42d846b14716ae9ef3f321d617994c7ca6145edaa949d3e320a3bf7c09d5b2a06ef
35b31748fd813d4994846a3e60b12f0bbc2ba424bad7520438f8a92aecfb4ec43e01ef92067a9e4722cd152ff7f17b53e182e
92b0799de8cf349300f3a94eb05bcf4814d7cfc3f65a3e9c3d27c345684a5ce8f883438226ba5a3091a2505903b779dfb398d
0c4f610ff25ec13c9e10de49a9af36c4374f7fc6b2618674166d740fe3b0832cc8b5ce0eec3dbabfab71e098ec49c8041acdd
410b13ea4784c123833906000e0d0fd3430318e21107f610be313796a1dede64cb7b7d3bfcf510b4f636f5f097f4bf4e6e5d5
2869d3c575885c7b9de7e9196

Sent (Signature): 8ce370686f5a9c237f5c3fb422e32c705fcb87e9a0a7fb5a5dbab60a02507b8dd51ad42dcdd0799b8b2
d70faece0df79aa23579fd4c0bb5c9327998774c36cfda087f9aa4f0a27f0f68dd6ce9fe0dcd31df9681a977655f06225e1e3
1d1593e83887e0786c1ed6b46eac08ecab99a33debee50dddd42d5f7a0fe64f0b17000d2a465806fc31403c28a5cd79c91204
2cfe5c2290038b617285e2211a378f924b639d4da7d33660cf6fdbaefe5127cba7cd9f61635c0852c761cf1b9ca83bbad79ce
f19c9c9abb2d78173685fbe054f0d01717f04a221fb0b4358020d7984ce831702047766c32733422c5b0dfaec91391854b881
b07f0cf59c6ae6a2cbf64e788

Received (Encrypted): 6ca1ef27a5fb028849674b98e9801cf5fd77d3e5a36886d464646363b2ab8cbc8a6d547a9dbece8
21ce029dcd1ea4758583fea44ea445bcbf67a491668faf1faae612dd8c52e3a27897051b14bf17acd7ffee2fe1b495f80fa0a
c96f14d10af40bc9a97403b32f1e99e9b9a105e002cd8c85b70caf47c72b476b50cc5f1b3a9a03b22c78705d18e82469cef80
a63c751b02954006f7c24c6dc4b276b5b3f9e8a027d7c6ba8d47a6435b89766b46358621f45534dafdd1b4748970d9941a7de
f9e264a6ea1673e968cc0c1377f06129558a087e91de95dab083acc4ba92d22d2d22bd42593473d4cfc30540858f8dee4675d
88ac0b9f4f0c1beee484334dad9a2

Received (Signature): 4f43c23ec9543f3540befd229655f02bf518920e357eeb8df16746525d7114acb77c5c5c7afdc1d
5ab97823b0fcf9c6c114b83ec7efa1fceb2d2e2785f130102604d6a2120724ad8821ffb883ca16e0e475534c43be382e9df71
44d5cf641b3ad62d7c9e8da1f630a4afcbc5650177d4970e12492fdff8ed7b31864e1434b51021d8bae84f23ca3ed0cf8e2d0
c87d848ba531bcb57311a5b6d1753549738f55485b258d188f37aae687c7c3ab36e5acd3f149b1bebf2f0d69f9bedb07921f7
f0a98ac455ccb1bca7a8ccabbac4a40cf0018cc45135f0daeff107fb585f3bae7bfc8ccca0278fe47759af6e3ea1ec3f89876
c9ae90462d8ad780a6537db153437

Decrypted: hello server here

Signature Verification: Successful

You:
```