# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

Practical No: 8

Date Of Performance: 01/09/2025

Aim: To implement the following algorithms:

* RSA
* Digital Signature Scheme with RSA

RSA CODE:

import math

p = 5

q = 13

n = p\*q

print("n =", n)

phi = (p-1)\*(q-1)

e = 7

while(e<phi):

    if (math.gcd(e, phi) == 1):

        break

    else:

        e += 1

print("e =", e)

k = 1

d = ((k\*phi)+1)/e

print("d =", d)

print(f'Public key: {e, n}')

print(f'Private key: {d, n}')

msg = 5

print(f'Original message:{msg}')

C = pow(msg, e)

C = math.fmod(C, n)

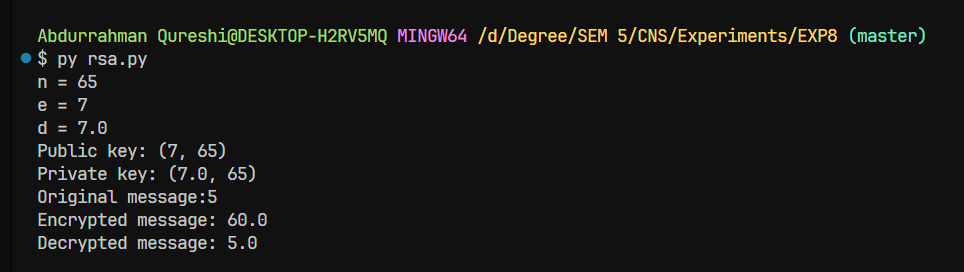
print(f'Encrypted message: {C}')

M = pow(C, d)

M = math.fmod(M, n)

print(f'Decrypted message: {M}')

OUPTUT:



Digital Signature Scheme with RSA CODE:

# rsa\_sign\_verify.py

from Crypto.PublicKey import RSA

from Crypto.Signature import pkcs1\_15

from Crypto.Hash import SHA256

def generate\_keys(bits=2048):

    key = RSA.generate(bits)

    private\_pem = key.export\_key()

    public\_pem = key.publickey().export\_key()

    return private\_pem, public\_pem

def sign\_message(private\_pem: bytes, message: bytes) -> bytes:

    key = RSA.import\_key(private\_pem)

    h = SHA256.new(message)

    signer = pkcs1\_15.new(key)

    signature = signer.sign(h)

    return signature

def verify\_signature(public\_pem: bytes, message: bytes, signature: bytes) -> bool:

    key = RSA.import\_key(public\_pem)

    h = SHA256.new(message)

    verifier = pkcs1\_15.new(key)

    try:

        verifier.verify(h, signature)

        return True

    except (ValueError, TypeError):

        return False

if \_\_name\_\_ == "\_\_main\_\_":

    message = b"Hello! This is a message to sign with RSA."

    # Generate keys

    priv, pub = generate\_keys(2048)

    print("Private key length:", len(priv), "bytes")

    print("Public key length:", len(pub), "bytes")

    # Sign

    sig = sign\_message(priv, message)

    print("Signature (hex):", sig.hex())

    # Verify

    ok = verify\_signature(pub, message, sig)

    print("Signature valid?", ok)

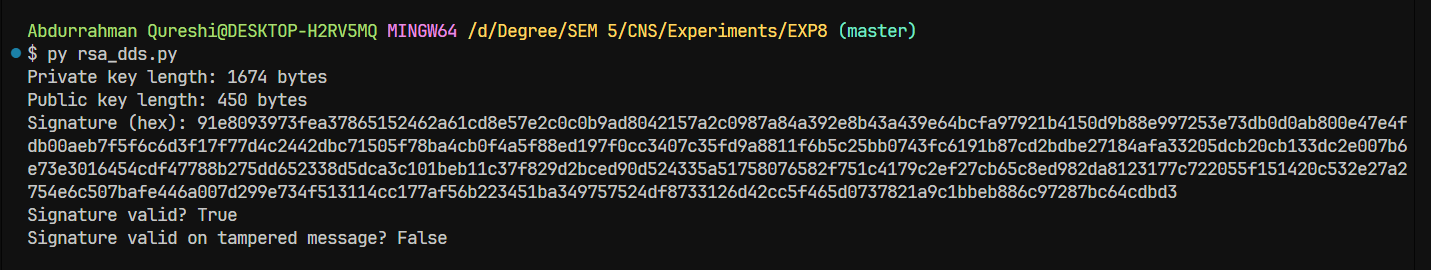
    # Tampering test

    tampered = b"Hello! This is a tampered message."

    ok2 = verify\_signature(pub, tampered, sig)

    print("Signature valid on tampered message?", ok2)

OUTPUT:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Performance  (7M) | Journal  (3M) | Lab Ethics  (2M) | Attendance  (3M) | Total  (15M) | Faculty Signature |
|  |  |  |  |  |  |