Assignment 6

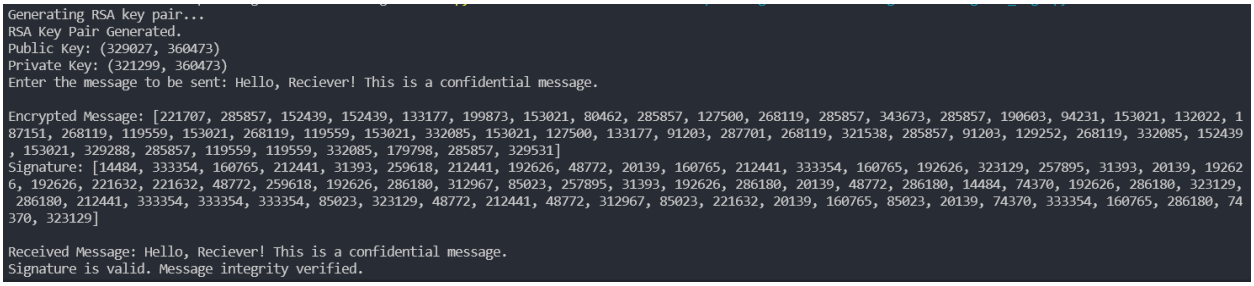
Digital Signature

Code:-

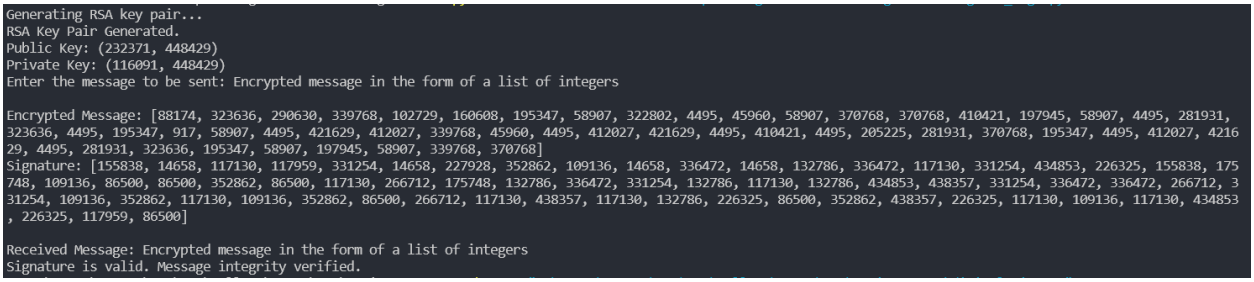
```
import random
import hashlib

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def mod_inverse(a, m):
    for i in range(1, m):
        if (a * i) % m == 1:
            return i
    return None

def generate_keypair():
    p = q = 1
    while not is_prime(p):
        p = random.randint(100, 1000)
    while not is_prime(q) or p == q:
        q = random.randint(100, 1000)
    n = p * q
    phi = (p - 1) * (q - 1)
    e = random.randint(1, phi)
    while gcd(e, phi) != 1:
        e = random.randint(1, phi)
```

```
    d = mod_inverse(e, phi)
    return ((e, n), (d, n))

def rsa_encrypt(message, public_key):
    e, n = public_key
    encrypted_message = [pow(ord(char), e, n) for char in message]
    return encrypted_message

def rsa_decrypt(encrypted_message, private_key):
    d, n = private_key
    decrypted_message = ''.join([chr(pow(char, d, n)) for char in encrypted_message])
    return decrypted_message

def generate_signature(message, private_key):
    hashed_message = hashlib.sha256(message.encode()).hexdigest()
    signature = rsa_encrypt(hashed_message, private_key)
    return signature

def verify_signature(message, signature, public_key):
    hashed_message = hashlib.sha256(message.encode()).hexdigest()
    decrypted_signature = rsa_decrypt(signature, public_key)
    return decrypted_signature == hashed_message

# Main program
print("Generating RSA key pair...")
public_key, private_key = generate_keypair()
print("RSA Key Pair Generated.")
print("Public Key:", public_key)
print("Private Key:", private_key)
message = input("Enter the message to be sent: ")

encrypted_message = rsa_encrypt(message, public_key)
signature = generate_signature(message, private_key)
print("\nEncrypted Message:", encrypted_message)
print("Signature:", signature)

received_message = rsa_decrypt(encrypted_message, private_key)
is_signature_valid = verify_signature(received_message, signature, public_key)

print("\nReceived Message:", received_message)
if is_signature_valid:
    print("Signature is valid. Message integrity verified.")
else:
    print("Invalid Signature. Message may have been tampered with.")
```

Output:-



```
Generating RSA key pair...
RSA Key Pair Generated.
Public Key: (329027, 360473)
Private Key: (321299, 360473)
Enter the message to be sent: Hello, Reciever! This is a confidential message.

Encrypted Message: [221707, 285857, 152439, 152439, 133177, 199873, 153021, 80462, 285857, 127500, 268119, 285857, 343673, 285857, 190603, 94231, 153021, 132022, 1
87151, 268119, 119559, 153021, 268119, 119559, 153021, 332085, 153021, 127500, 133177, 91203, 287701, 268119, 321538, 285857, 91203, 129252, 268119, 332085, 152439
, 153021, 329288, 285857, 119559, 119559, 332085, 179798, 285857, 329531]
Signature: [14484, 333354, 160765, 212441, 31393, 259618, 212441, 192626, 48772, 20139, 160765, 212441, 333354, 160765, 192626, 323129, 257895, 31393, 20139, 19262
6, 192626, 221632, 221632, 48772, 259618, 192626, 286180, 312967, 85023, 257895, 31393, 192626, 286180, 20139, 48772, 286180, 14484, 74370, 192626, 286180, 323129,
 286180, 212441, 333354, 333354, 333354, 85023, 323129, 48772, 212441, 48772, 312967, 85023, 221632, 20139, 160765, 85023, 20139, 74370, 333354, 160765, 286180, 74
370, 323129]

Received Message: Hello, Reciever! This is a confidential message.
Signature is valid. Message integrity verified.
```



```
Generating RSA key pair...
RSA Key Pair Generated.
Public Key: (232371, 448429)
Private Key: (116091, 448429)
Enter the message to be sent: Encrypted message in the form of a list of integers

Encrypted Message: [88174, 323636, 290630, 339768, 102729, 160608, 195347, 58907, 322802, 4495, 45960, 58907, 370768, 370768, 410421, 197945, 58907, 4495, 281931,
323636, 4495, 195347, 917, 58907, 4495, 421629, 412027, 339768, 45960, 4495, 412027, 421629, 4495, 410421, 4495, 205225, 281931, 370768, 195347, 4495, 412027, 4216
29, 4495, 281931, 323636, 195347, 58907, 197945, 58907, 339768, 370768]
Signature: [155838, 14658, 117130, 117959, 331254, 14658, 227928, 352862, 109136, 14658, 336472, 14658, 132786, 336472, 117130, 331254, 434853, 226325, 155838, 175
748, 109136, 86500, 86500, 352862, 86500, 117130, 266712, 175748, 132786, 336472, 331254, 132786, 117130, 132786, 434853, 438357, 331254, 336472, 336472, 266712, 3
31254, 109136, 352862, 117130, 109136, 352862, 86500, 266712, 117130, 438357, 117130, 132786, 226325, 86500, 352862, 438357, 226325, 117130, 109136, 117130, 434853
, 226325, 117959, 86500]

Received Message: Encrypted message in the form of a list of integers
Signature is valid. Message integrity verified.
```

Conclusion:-

The system effectively safeguards confidential messages through RSA encryption and digital signatures, ensuring unauthorized access, tampering, and message forgery, thereby ensuring their integrity.