

**Experiment No. 4**

**Title:** Digital Signature using RSA

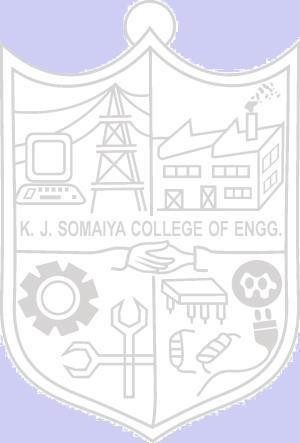
**Batch: B1 Roll No.: 1914078 Experiment No.: 4**

**Aim:** To implement digital signature using RSA

**Resources needed:** Windows/Linux, C or JAVA language

# Theory: Pre Lab/ Prior Concepts:

RSA is a public key algorithm named after its inventers Rivest, Shamir and Adleman. The characteristics of public key cryptography (which is also called as Asymmetric Cryptography) are as below:

* It has two keys. One key is called as private key and the other one is called as public key. Everyone who uses this cryptography has to have two keys each.
* Keys used for encryption and decryption should be different. If one is used for encryption, then other must be used for decryption. Any key can be used for encryption and then remaining key can be used for decryption.
* Public Key Cryptography is based on solid foundation of mathematics.
* It has large computational overheads. Hence ciphertext generated is of much larger size than the plaintext. Hence it is normally used for encrypting small size of data blocks. For example, passwords, symmetric keys, etc. It is not preferred to encrypt large files.
* RSA algorithm gets its security from the fact that it is extremely difficult to factorize large prime number.
* Security of the algorithm depends on the size of the key. Greater the size of the key, larger is the security. The key length is variable. The most commonly used key length is 512 bits.

# Procedure / Approach /Algorithm / Activity Diagram:

1. **Key generation Algorithm:**
   1. Choose large prime numbers p and q.
   2. Calculate product n= pq. The value of n can be revealed publicly, but n is large enough that even supercomputers cannot factor it in a reasonable amount of time (years or even centuries).
   3. Calculate phi = (p-1)(q-1)
   4. Select e < phi such that it is relatively prime to phi. The public key is (e, n).
   5. Determine d such that ed = 1 mod phi. The private key is (d, n). d, p ,q and phi are kept secret, only (e,n) is made public.

(Autonomous College Affiliated to University of Mumbai)

# Digital signature generation:

1. Suppose M is the message.
2. Encrypt this message using private key of the sender to generate digital signature, C = M^d mod n.

# Digital signature verification:

1. Decrypt the C generated in step 7 to verify the signature using public key of the sender, M = C^e mod n.

**Results:** (Program printout with output / Document printout as per the format)

**CODE:**

import math

p = int(input("Enter value of p: "))

q = int(input("Enter value of q: "))

def isPrime(n):

    if (n <= 1):

        return False

    for i in range(2, n):

        if (n % i == 0):

            return False

    return True

checkP = isPrime(p)

checkQ = isPrime(q)

while(((checkP == False) or (checkQ == False))):

    p = int(input("Enter a prime number for p: "))

    q = int(input("Enter a prime number for q: "))

    checkP = isPrime(p)

    checkQ = isPrime(q)

n = p\*q

print("\nRSA modulus, n is ", n)

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

e = int(input("\nEnter value of e: "))

while (e < phi):

    if(math.gcd(e, phi) == 1):

        break

    else:

        e += 1

public\_key = (e, n)

print(f"\nPublic key is {public\_key} ")

def egcd(a, b):

    if(a % b == 0):

        return(b, 0, 1)

    else:

        gcd, s, t = egcd(b, a % b)

        s = s-((a//b) \* t)

        print("%d: %d\*(%d) + (%d)\*(%d)" % (gcd, a, t, s, b))

        return(gcd, t, s)

def inverse(e, phi):

    gcd, s, \_ = egcd(e, phi)

    if(gcd != 1):

        return None

    else:

        if(s < 0):

            print("s=%d.\nSince %d is less than 0, s: s(modphi), s: %d." %

                  (s, s, s % phi))

        elif(s > 0):

            print("s=%d." % (s))

        return s % phi

d = inverse(e, phi)

private\_key = (d, n)

print(f"\nPrivate key is: {private\_key}\n")

def RsaEncrypt(priv, n\_text):

    d, n = priv

    x = []

    m = 0

    for i in n\_text:

        if(i.isupper()):

            m = ord(i)-65

            c = pow(m, d) % n

            x.append(c)

        elif(i.islower()):

            m = ord(i)-97

            c = pow(m, d) % n

            x.append(c)

        elif(i.isspace()):

            spc = 400

            x.append(400)

    return x

def RsaDecrypt(pub\_key, c\_text):

    e, n = pub\_key

    txt = list(c\_text)

    x = ''

    m = 0

    for i in txt:

        if(i == '400'):

            x += ' '

        else:

            m = pow(int(i), e) % n

            m += 65

            c = chr(m)

            x += c

    return x

message = input("Enter Messge: ")

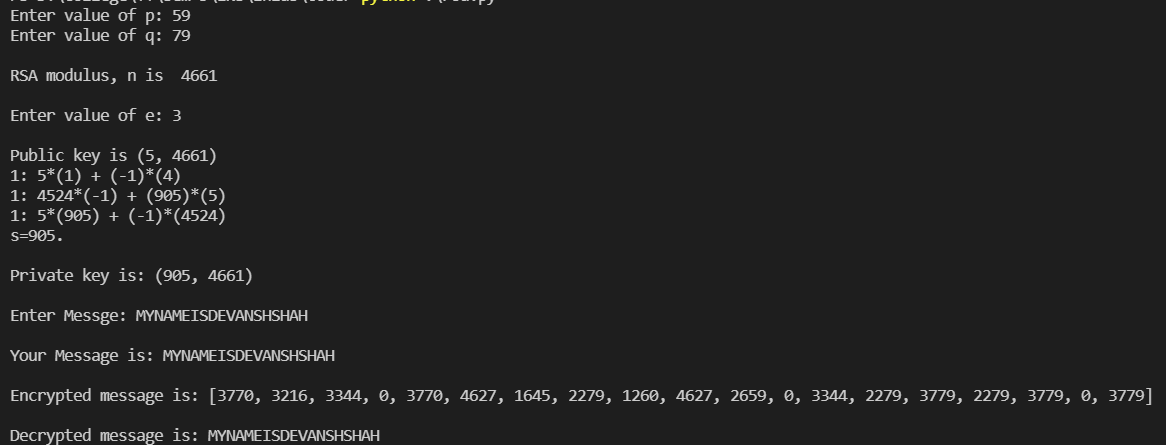
print(f"\nYour Message is: {message}")

print(f"\nEncrypted message is: {RsaEncrypt(private\_key,message)}")

print(

    f"\nDecrypted message is: {RsaDecrypt(public\_key,RsaEncrypt(private\_key,message))}")

**OUTPUT:**

****

# Questions:

1. In RSA cryptosystem each plaintext character is presented by the number between 00(A) and 25(Z). The number 26 represents the blank character. Bob wants to send Alice the message “Hello World”. So the plaintext is as below,

07 04 11 11 14 26 22 14 17 11 03 . Suppose p=11, q=3. Find out digital signature.

**Answer:**

p = 11

q = 3

n = p \* q = 11 \* 3 = 33

phi = (p-1) \* (q-1) = 10 \* 2 = 20

Now to find e such that e < phi and gcd(e, phi) = 1

For e = 2, gcd(2, 20) = 2

e = 3, gcd(3, 20) = 1

So, e = 3

Now, to find d such that e\*d = 1 mod(phi)

For d = 1, e\*d mod n = 3 \* 1 mod(20) = 3

d = 2, e\*d mod n = 3 \* 2 mod(20) = 6

d = 3, e\*d mod n = 3 \* 3 mod(20) = 9

d = 4, e\*d mod n = 3 \* 4 mod(20) = 12

d = 5, e\*d mod n = 3 \* 5 mod(20) = 15

d = 6, e\*d mod n = 3 \* 6 mod(20) = 18

d = 7, e\*d mod n = 3 \* 7 mod(20) = 1

So, d = 7

For digital signature, we encrypt using private key, i.e. (d, n)

Ciphertext = plaintext ^ d mod(n)

So, for message: 07 04 11 11 14 26 22 14 17 11 03

C1 = 07 ^ 7 mod(33) = 28

C2 = 04 ^ 7 mod(33) = 16

C3 = 11 ^ 7 mod(33) = 11

C4 = 11 ^ 7 mod(33) = 11

C5 = 14 ^ 7 mod(33) = 20

C6 = 26 ^ 7 mod(33) = 05

C7 = 22 ^ 7 mod(33) = 22

C8 = 14 ^ 7 mod(33) = 20

C9 = 17 ^ 7 mod(33) = 08

C10 = 11 ^ 7 mod(33) = 11

C11 = 03 ^ 7 mod(33) = 09

Thus, the digital signature generated is: 28 16 11 11 20 05 22 20 08 11 09

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Outcomes:**

**CO:** Illustrate different cryptographic algorithms for security

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conclusion:** We implemented Digital signature using RSA algorithm.

# Grade: AA / AB / BB / BC / CC / CD /DD Signature of faculty in-charge with date

**References: Books/ Journals/ Websites:**

1. Charles P. Pfleeger, “Security in Computing”, Pearson Education
2. Behrouz A. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
3. William Stalling, “Cryptography and Network Security”, Prentice Hall