Cryptography DA-4

QUESTION RSA :-

RSA cryptosystems:
(a) Generate Public and private keys of two communicating parties
(b) Encrypt a short text message of your choice with their RSA /ElGamal key and send them the encrypted message (as a number, or as a sequence of numbers if your message is longer than the block size for their n).
(c) Decrypt the encrypted message you receive from your partner.

Solution:-

Code:-

```
import random


def M_i(e, phi):
    d = 0
    x1 = 0
    x2 = 1
    y1 = 1
    temporary = phi

    while e > 0:
        t1 = temporary/e
        temp2 = temporary - t1 * e
        temporary = e
        e = temp2

        x = x2- t1* x1
        y = d - t1 * y1

        x2 = x1
        x1 = x
        d = y1
        y1 = y
```

```
    if temporary == 1:
        return d + phi

def caus(k, l):
    while l != 0:
        k, l = l, k % l
    return k


def is_prime(num):
    if num == 2:
        return True
    if num < 2 or num % 2 == 0:
        return False
    for n in xrange(3, int(num**0.5)+2, 2):
        if num % n == 0:
            return False
    return True

def gen_pair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be equal')

    n = p * q


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


    e = random.randrange(1, phi)


    g = caus(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = caus(e, phi)


    d = M_i(e, phi)
```

```
    return ((e, n), (d, n))

def encrypt(pk, plaintext):

    key, n = pk

    cipher = [(ord(char) ** key) % n for char in plaintext]

    return cipher

def decrypt(pk, ciphertext):

    key, n = pk

    plain = [chr((char ** key) % n) for char in ciphertext]

    return ''.join(plain)


if __name__ == '__main__':

    p = int(raw_input("Enter 1st prime number (17, 19, 23, ....): "))
    q = int(raw_input("Enter 2nd prime number : "))
    public, private = gen_pair(p, q)
    print "Public key  ", public ," Private key ", private
    message = raw_input("Message to encrypt with private key: ")
    encrypted_msg = encrypt(private, message)
    print "Encrypted message : "
    print ''.join(map(lambda x: str(x), encrypted_msg))
    print "Decrypting message with public key ", public ," . . ."
    print "Your message is:"
    print decrypt(public, encrypted_msg)
```

INPUT:-

Enter 1st prime number (17, 19, 23, ...): 17
Enter 2nd prime number : 19

OUTPUT:-

Public key (283, 323) and Private key (403, 323)

Message to encrypt with private key: Hello this is cryptography da-4

Encrypted message:
1323729293211621094212919116212919116217726617825210932110326614725242178162111147311307

Decrypting message with public key (283, 323) . . .
Your message is: Hello this is cryptography da-4

QUESTION ElGamal :-

ElGamal cryptosystems:
(a) Generate Public and private keys of two communicating parties
(b) Encrypt a short text message of your choice with their RSA /ElGamal key and send them the encrypted message (as a number, or as a sequence of numbers if your message is longer than the block size for their n).
(c) Decrypt the encrypted message you receive from your partner.

Solution:-

Code:-

```
import random
from math import pow

a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a % b == 0:
        return b;
    else:
        return gcd(b, a % b)


def gen_key(q):

    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)

    return key

def power(a, b, c):
    x = 1
```

```
    y = a

    while b > 0:
        if b % 2 == 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)

    return x % c


def encrypt(msg, q, h, g):

    en_msg = []

    k = gen_key(q)
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])

    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])

    return en_msg, p

def decrypt(en_msg, p, key, q):

    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))

    return dr_msg

def main():

    msg = 'Hello this is cryptography da-4'
    print("Original Message :", msg)
```

```
    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)

    key = gen_key(q)
    h = power(g, key, q)
    print("g used : ", g)
    print("g^a used : ", h)

    en_msg, p = encrypt(msg, q, h, g)
    dr_msg = decrypt(en_msg, p, key, q)
    dmsg = ''.join(dr_msg)
    print("Decrypted Message :", dmsg);


if __name__ == '__main__':
    main()
```

OUTPUT

```
('Original Message :', 'Hello this is cryptography da-4')
('g used : ', 128469188689566493180724902178455613808405391155 56L)
('g^a used : ', 211649721032226651394242874523301267887761082125 66L)
('g^k used : ', 463757103218526233459089028674131146383207896610 6L)
('g^ak used : ', 113209679255862906772625593046210922411304460689 11L)
('Decrypted Message :', 'Hello this is cryptography da-4')
```

ID - **********

Name - Hritish kumar