**Garvin Chanderia**

**2203132**

**LY-ISA1**

**Experiment No:3**

**Title:**  RSA Algorithm

**Problem statement:** Implement RSA public key cryptosystem for key generation and cipher verification.

**Aim:**

To understand,

1. Public key algorithm.
2. RSA algorithm
3. Concept of Public key and Private Key

**Theory:**

**Public Key Algorithm:**

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristics:

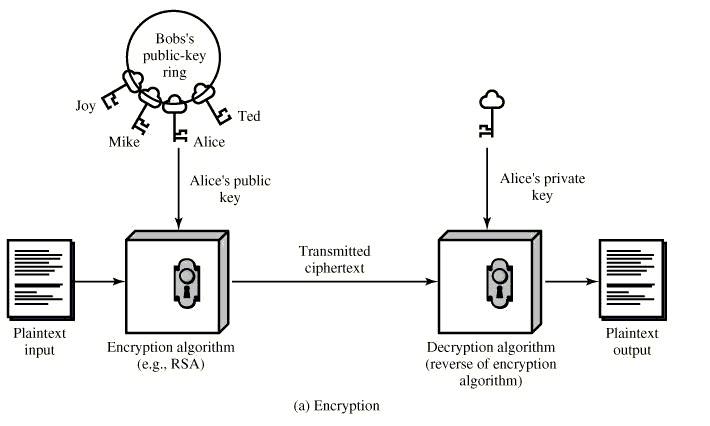
* It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristics:

* Either of the two related keys can be used for encryption, with the other used for decryption.

A public key encryption scheme has six ingredients:

* **Plaintext**: This is readable message or data that is fed into the algorithm as input.
* **Encryption algorithm**: The encryption algorithm performs various transformations on the plaintext.
* **Public and private key**: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
* **Cipher text**: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
* **Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.



**Figure:** Public key cryptography

The essential steps are as the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or the other accessible file. This is the public key. The companion key is kept private.  As  figure suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice’s public key.

When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice’s private key.

The RSA Algorithm:

The scheme developed by Rivest, Shamir and Adleman makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is the block size must be less than or equal to log2 (n); in practice the block size is I bits, where 2i<n<=2i+1. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

C = Me mod n M = Cd mod n

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}. For this

algorithm to be satisfactory for public key encryption, the following requirements must meet:

1. It is possible to find values of e, d, n.
2. It is relatively easy to calculate Me mod n and Cd mod n for all values of M<n.
3. It is feasible to determine d given e and n.

**Figure:** The RSA Algorithm

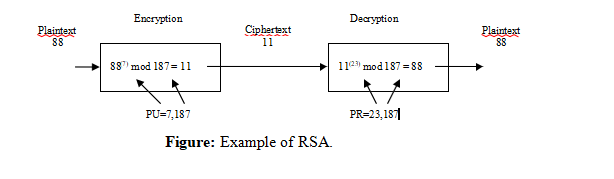
Example 1:

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17\*11 = 187.

3. Calculate Ø(n) = (p-1)(q-1) = 16\*10 = 160.

1. Select e such that relatively prime to Ø(n)=160 & less than Ø(n); we choose e = 7.
2. Determine d such that de ≡ 1 (mod 160) and d < 160. The correct value is d = 23; d can be calculated using the extended Euclid’s algorithm.

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}. The example shows the use of these keys for plaintext input of M=88.

Example 2:

**P = 7, Q = 13, M = 10.**

Step I: n = 3 \* 11 = 33

Step II: Ø(n) = 2 \* 10 = 20

Step III: Select e, such that gcd (Ø(n),e) = 1, gcd(20,3) = 1, So we can select e =3

Step IV: To calculate d, solve for gcd (Ø(n),e) using extended Euclid’s algorithm and pick up the value of t.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *q* | *r1* | *r2* | *r* | *t1* | *t2* | *t* |
| 6 | 20 | 3 | 2 | 0 | 1 | -6 |
| 1 | 3 | 2 | 1 | 1 | -6 | 7 |

d = 7

Step V: For Encryption,

C = Me mod n

= 2^3 mod 33

= 8 mod 33

= 8

Step VI: For Decryption,

M = Cd mod n

= 8^7 mod 33

= ((8^2 mod 33) (8^2 mod 33) (8^2 mod 33) (8^1 mod 33)) mod 33

= (31\* 31 \*31 \*8 ) mod 33

= (961 mod 33) (248 mod 33) mod 33

= 68 mod 33

= 2

Advantages:

1. Easy to implement.

Disadvantages:

1.  Anyone can announce the public key.

Algorithm:

1. Start
2. Input two prime numbers p and q.
3. Calculate n = pq.
4. Calculate Ø(n) = (p-1)(q-1).
5. Input value of e.
6. Determine d.
7. Determine PU and PR.
8. Take input plaintext.
9. Encrypt the plaintext and show the output.
10. Stop.

**Algorithm/Pseudocode:**

from decimal import Decimal

def gcd(m,n):

    if n==0:

        return a

    else:

        return gcd(n,m%n)

#input variables

p = input()

q = input()

no = input()

#calculate n

n = p\*q

#calculate totient

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

#calculate K

for k in range(2,totient):

    if gcd(k,totient)== 1:

        break

for i in range(1,10):

    x = 1 + i\*totient

    if x % k == 0:

        d = int(x/k)

        break

local\_cipher = Decimal(0)

local\_cipher =pow(message,k)

cipher\_text = ctt % n

decrypt\_t = Decimal(0)

decrypt\_t= pow(cipher\_text,d)

decrpyted\_text = decrypt\_t % n

print('n = '+str(n))

print(' k = '+str(k))

print(' totient = '+str(t))

print(' d = '+str(d))

print('cipher text = '+str(ct))

print(' decrypted text = '+str(dt))

Input : p = 79 , q = 89 , message = 44

Output :

p = 79 , q = 89

t = 1373

k = 5

cipher text = 4119

decypted text = 44

CONCLUSION:

We have studied and implemented the public key algorithm that is RSA algorithm.