EXPERIMENT NO 1

Aim : Implementation of RSA algorithm

Theory:

Encryption is the process of translating plaintext (i.e. normal data) into ciphertext (i.e. secret data). During encryption, plaintext information is translated to ciphertext using a key and an algorithm. To read the data, the ciphertext must be decrypted (i.e. translated back to plaintext) using a key and an algorithm.

An encryption algorithm is a series of mathematical operations applied to the numerical value(s) of the key and the numerical values of the characters in a string of plaintext. The results are the ciphertext. The larger the key, the more secure the ciphertext.

- **Plaintext(Cleartext)**: The intelligible message which will be converted into an unintelligible (encrypted) message
- **Ciphertext**: A message in encrypted form
- **Encryption**: The process of converting a plaintext message into a ciphertext message
- **Decryption**: The process of converting a ciphertext message into a plaintext message
- **Key**: A parameter used in the encryption and decryption process.
- **Cryptosystem**: A system to encrypt and decrypt information
- **Symmetric Cryptosystem**: A cryptosystem that uses the same key to encrypt and decrypt information
- **Asymmetric Cryptosystem**: A cryptosystem that uses one key to encrypt and a different key to decrypt
- **Cryptography**: The use of cryptosystems to maintain the confidentiality of information
- **Cryptoanalysis**: The study of breaking cryptosystems

There are two basic types of encryption algorithms:

1. Symmetric Algorithms that use the same key for both encryption and decryption.
2. Asymmetric Algorithms that use different keys for encryption and decryption.

If a symmetric algorithm is chosen, both the sender and the receiver must have the same key. If the sender and receiver are in different locations, the transmission of the key itself becomes a point of vulnerability.

If an asymmetric algorithm is chosen, there are two keys: a public key and a private key. Data encrypted with a public key can only be decrypted with the corresponding private key. The receiver first transmits their public key to the sender. The sender uses that public key to encrypt the message and then transmits the message to the receiver. The receiver decrypts the message with their private key.

With an asymmetric algorithm the receiver can transmit the public key to whomever they wish without fear of compromise because only the holder of the private key can decrypt the message.

Asymmetric algorithms solve the key distribution problem.

What is RSA ?

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977 . The RSA cryptosystem is the most widely-used public key cryptography algorithm in the world. RSA is an asymmetric cryptographic algorithm. It can be used to encrypt a message without the need to exchange a secret key separately. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Party A can send an encrypted message to party B without any prior exchange of secret keys. A just uses B's public key to encrypt the message and B decrypts it using the private key, which only he knows. RSA can also be used to sign a message, so A can sign a message using their private key and B can verify it using A's public key

Advantages and disadvantages of RSA Algorithm

Advantages

- RSA algorithm is safe and secure for its users through the use of complex mathematics.
- RSA algorithm is hard to crack since it involves factorization of prime numbers which are difficult to factorize.
- Moreover, RSA algorithm uses the public key to encrypt data and the key is known to everyone, therefore, it is easy to share the public key.

Disadvantages

- RSA algorithm can be very slow in cases where large data needs to be encrypted by the same computer.
- It requires a third party to verify the reliability of public keys.
- Data transferred through RSA algorithm could be compromised through middlemen who might temper with the public key system.

Algorithm:

RSA encrypts messages through the following algorithm, which is divided into 3 steps:

Step 1. Key Generation

- Choose two distinct prime numbers p and q.
- Find n such that n = p * q. n will be used as the modulus for both the public and private keys.

- Find the totient of n which is denoted by $\phi(n)$ such that $\phi(n)=(p-1)(q-1)$.
- Choose an e such that $1 < e < \phi(n)$, and such that e and $\phi(n)$ share no divisors other than 1 (e and $\phi(n)$ are relatively prime).e is kept as the public key exponent.
- Determine d (using modular arithmetic) which satisfies the congruence relation $de \equiv 1 (\text{mod } \phi(n))$.

  This is often computed using the Extended Euclidean Algorithm, since e and $\phi(n)$ are relatively prime and d is to be the modular multiplicative inverse of e. d is kept as the private key exponent.

  The public key has modulus n and the public (or encryption) exponent e. The private key has modulus n and the private (or decryption) exponent d, which is kept secret.

Step 2. Encryption

- Person A transmits his/her public key (modulus n and exponent e) to Person B, keeping his/her private key secret.
- When Person B wishes to send the message "M" to Person A, he first converts M to an integer such that $0 < m < n$ by using agreed upon reversible protocol known as a padding scheme.
- Person B computes, with Person A's public key information, the ciphertext c corresponding to $c \equiv m^e \pmod{n}$.

  Person B now sends message "M" in ciphertext, or c, to Person A.

Step 3. Decryption

- Person A recovers m from c by using his/her private key exponent, d, by the computation $m \equiv c^d \pmod{n}$.

Example:

Step 1 : Key Generation

Let us take two prime number as p=11 and q=13. The modulus is $n=p \times q=143$. The totient of n $\phi(n)=(p-1)x(q-1)=120$.

Calculating e such that e follows this condition

$1< e < \phi(n)$ and $\gcd(e, \phi(n))=1$

Take e=2 gcd(2,120)=2 condition not satisfied

Take e=3 gcd(3,120)=3 condition not satisfied

Take e=4 gcd(4,120)=4 condition not satisfied

Take e=5 gcd(5,120)=5 condition not satisfied

Take e=6 gcd(6,120)=6 condition not satisfied

Take e=7 gcd(2,120)=1 condition satisfied

Now we find d such that $de \equiv 1 \pmod{\phi(n)}$.
Which means we find d such that (((k* ϕ(n)) + 1)mod e) is 0 where k [0,1….]
k=0 (0*120+1)%7=1 and 1!=0
k=1 (1*120+1)%7=2 and 2!=0
k=2 (2*120+1)%7=3 and 3!=0
k=3 (3*120+1)%7=4 and 4!=0
k=4 (4*120+1)%7=5 and 5!=0
k=5 (5*120+1)%7=6 and 6!=0
k=6 (6*120+1)%7=0 and 0==0
Hence d= (((k* ϕ(n)) + 1)/e
=(6*120+1)/7
=103

7 is RSA public key e and RSA private key using the Extended Euclidean algorithm, which gives 103. RSA public key (n, e) which, in this example, is (143, 7) . Let plaintext message M be number 9 and is encrypted into ciphertext, C, as follows:

Step 2. Encryption

$M^e \bmod n = 9^7 \bmod 143 = 48 = C$

RSA private key (d, n) as follows:

Step 3. Decryption

$C^d \bmod n = 48^{103} \bmod 143 = 9 = M$

Code:

```
import random


def gcd(a , b):          #compute gcd
   while a != b :
      if a > b :
```

```
            a = a-b
        else :
            b = b-a
    return a

def isPrime(n):          #prime number checker
    if n==1 :
        return False
    elif n==2 :
        return True;
    else:
        for x in range(2,n):
            if(n % x==0):
                return False
        return True

primes = [i for i in range(227,811) if isPrime(i)]
first_prime_p = random.choice(primes)       #p
primes.remove(first_prime_p)
second_prime_q=random.choice(primes)    #q
print("Value of p : ",first_prime_p)
print("Value of q : ",second_prime_q)
n=first_prime_p*second_prime_q                  #n
print("Value of n : ",n)
phi_n=(first_prime_p-1)*(second_prime_q-1)       #phi(n)
```

```
print("Value of phi(n) : ",phi_n)
e=2                                  # 1<e<phi(n) && gcd(e,phi(n))==1 is the condition for e
while True:
    e = random.randint(2, phi_n)
    print("Random value of e : ",e,end="")
    if gcd(e, phi_n) == 1:
        print("  Condition Satisfied")
        break
    print("  Condition Not Satisfied")

print("Value of e : ",e)
k=0
d=0
while k<phi_n :
    d=0
    if(((k*phi_n) + 1)%e)== 0 :
        d=int(((k*phi_n) + 1)/e)
        print("Value of k : ",k)
        print("Value of d : ",d)
        break
    k=k+1
# [ 33(!)-122(z)]
# subtract 23 to make them in two digt
# [ 10 - 99 ]
```

```
text=""
text_num=0


while True :
    textnum = ""
    text = input("Enter the text : ").strip()
    for c in text:
        if c in [' ','{','}','|'] :
            text.replace(c,'')
            continue
        textnum = textnum + str(ord(c) - 23)
    text_num = int(textnum)
    if(text_num>n) :
        print("Message too large...")
        n_str=str(n-1)
        print("Enter msg less than or equal to : ",end='')
        i=0
        while i <= (len(n_str) - 2):
            print(chr(int(str(n_str[i]) + str(n_str[i + 1])) + 23), end="")
            i += 2
        print("\n")
    else :
        break
```

```
print("Text in numerical form will be : ",text_num)
print("Public Key : ( "+str(n)+", "+str(e)+")")
print("Public Key : ( "+str(n)+", "+str(d)+")")
encrypt=int((text_num** e) % n)
print("The encrypted message is : ",encrypt)
decrypt=int((encrypt**d)%n)
print("The decrypted message is : ",decrypt)
i=0
decrypt_str=str(decrypt)
while i<= (len(decrypt_str)-2) :
    print(chr(int(str(decrypt_str[i])+str(decrypt_str[i+1]))+23),end="")
    i+=2
```

Output:

```
#------------------------OUTPUT1-------------------------------------
# Value of p :  809
# Value of q :  811
# Value of n :  656099
# Value of phi(n) :  654480
# Random value of e :  14426  Condition Not Satisfied
# Random value of e :  517189  Condition Satisfied
# Value of e :  517189
# Value of k :  433258
# Value of d :  548269
# Enter the text : iMAC
```

Message too large...

Enter msg less than or equal to : XSy

#

Enter the text : MAC

Text in numerical form will be : 544244

Public Key : (656099, 517189)

Public Key : (656099, 548269)

The encrypted message is : 294490

The decrypted message is : 544244

MAC

Process finished with exit code 0

#------------------------OUTPUT2-----------------------------------

Value of p : 479

Value of q : 419

Value of n : 200701

Value of phi(n) : 199804

Random value of e : 158686 Condition Not Satisfied

Random value of e : 78333 Condition Satisfied

Value of e : 78333

Value of k : 61124

Value of d : 155909

Enter the text : R@

Text in numerical form will be : 5941

Public Key : (200701, 78333)

Public Key : (200701, 155909)

The encrypted message is : 20873

The decrypted message is : 5941

R@

Process finished with exit code 0

#------------------------OUTPUT3-------------------------------------

Value of p : 757

Value of q : 607

Value of n : 459499

Value of phi(n) : 458136

Random value of e : 248406 Condition Not Satisfied

Random value of e : 230373 Condition Not Satisfied

Random value of e : 324546 Condition Not Satisfied

Random value of e : 306916 Condition Not Satisfied

Random value of e : 68657 Condition Satisfied

Value of e : 68657

Value of k : 53442

Value of d : 356609

Enter the text : R @

Text in numerical form will be : 5941

Public Key : (459499, 68657)

Public Key : (459499, 356609)

The encrypted message is : 275740

The decrypted message is : 5941

R@

Process finished with exit code 0

Conclusion: RSA is the single most useful tool for building cryptographic protocols. We can implement this RSA algorithm in modern day applications to maintain the confidentiality of information. As programs and web services become more connected, secure data communication between remote processes should become a routine. Since asymmetric keys can be transmitted securely, they are preferable. However, they have a limited message size. A hybrid approach, using both symmetric and asymmetric algorithms can address both of these limitations.