**Practical No : 5**

**Title :** RSA cryptosystem and Digital signature scheme.

**Aim :** Implementation and analysis of RSA cryptosystem and Digital signature scheme using Python.

**Objectives:**

* To understand the concept of public key cryptosystem.
* To understand the RSA cryptosystem.
* To understand the possible attacks on RSA cryptosystem.
* To understand the principles of signature scheme

**Outcomes:** CO2

**Theory :**

RSA cryptosystem

RSA was invented by Ron Rivest, Adi Shamir, and Len Adleman and hence, it is termed as

RSA cryptosystem.

Two aspects of the RSA cryptosystem:

1. first generation of key pair
2. second encryption-decryption algorithms.

* Generation of RSA Key Pair:

Each person or a party who desires to participate in communication using encryption needs to

generate a pair of keys, namely public key and private key.

The process followed in the generation of keys is described below:

* Generate the RSA modulus (n)
* Select two large primes, p and q.
* Calculate n=p\*q. For strong unbreakable encryption, let n be a large number
* Calculate ø(n) = ø(p\*q)= ø(p)\* ø(q)= (p-1)\*(q-1)
* Derived Number (e)

Number e must be greater than 1 and less than ((p − 1)(q −1)) or ø(n) .

There must be no common factor for e and (p − 1)(q − 1) except for 1.

In other words two numbers e and (p – 1)(q – 1) are coprime.

* Form the public key

The pair of numbers (e, n ) form the RSA public key and is made public.

Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

* Generate the private key

Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.

Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q- 1).

This relationship is written mathematically as follows −

ed = 1 mod (p − 1)(q − 1)

or

d = ((ø(n) \*i ) +1))/e

assume i = 0, 1, 2... until d value is not a decimal.

assume i = 0, then calculate d if it is a decimal increase the i value, don&#39;t bother

finding the exact value of the decimal).

* Form the private key ={d , n}

Encryption - Find out Cipher text using

C= P^e mod n where P<n

Where C= Cipher text, P=plaintext , e= Encryption Key and n= Block size.

Decryption -Plaintext can be obtained using formula P= C d mod n

Where d= Decryption Key

RSA Digital signature

• Digital signature scheme changes the role of the private and public keys.

• Private and public keys of the sender are used not the receiver.

• Sender uses her/his own private key to sign the document and the receiver uses the

sender’s public key to verify it.

• The signing and verifying sets use the same function, but with different parameters.

The verifier compares the message and the output of the function if the result is true

the message is accepted.

RSA Digital Signature Scheme

The algorithm works as follows:

Key Generation:

1. Select two Prime numbers p & q where p ≠ q

2. Calculate n =p\*q

3. Calculate ø(n) = (p-1)\*(q-1)

4. Select e such that , e is relatively prime to ø(n) i.e. gcd (e, ø(n)) = 1 and 1&lt;e&lt; ø(n)

5. Calculate d= e -1 mod ø(n) or e d mod ø(n)=1

6. Public key = {e , n} , Private key ={d , n}

Signature Generation:

Use private key ={d,n}

Calculate signature S= M d mod n

where M&lt;n , S = Generated Signature , M= plaintext, d=encryption key & n=Block size

Signature verification:

Use signers public key ={e, n}compute V m

V m =S e mod n

verify if V m (Verified message ) = M (Original message)

**Python Code:**

import random

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def generate\_keypair(p, q):

n = p \* q

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

while True:

e = random.randrange(1, phi)

if gcd(e, phi) == 1:

break

d = 0

while True:

d += 1

if (d \* e) % phi == 1:

break

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

def encrypt(pk, plaintext):

key, n = pk

cipher = [pow(ord(char), key, n) for char in plaintext]

return cipher

def decrypt(pk, ciphertext):

key, n = pk

plain = [chr(pow(char, key, n)) for char in ciphertext]

return ''.join(plain)

if \_\_name\_\_ == '\_\_main\_\_':

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

q = int(input("Enter another prime number (q): "))

public\_key, private\_key = generate\_keypair(p, q)

print(f"\nPublic Key (e, n): {public\_key}")

print(f"Private Key (d, n): {private\_key}")

message = input("\nEnter a message to encrypt: ")

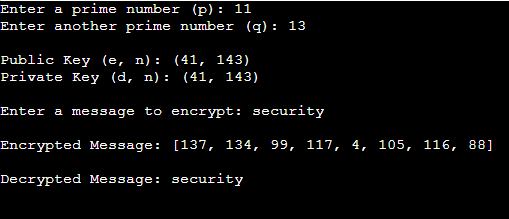
encrypted\_message = encrypt(public\_key, message)

print(f"\nEncrypted Message: {encrypted\_message}")

decrypted\_message = decrypt(private\_key, encrypted\_message)

print(f"\nDecrypted Message: {decrypted\_message}")

**Output:**



**Conclusion:** RSA is a strong encryption algorithm. RSA implements a public-key cryptosystem that allows secure communications and its security is due to cost of factoring large numbers. Also RSA digital signature scheme implemented and digital signature is verified by the receiver in scheme.