SEM - VII - 2022-23

CNS Lab

B3 - 2019BTECS00094 - Sweety Shrawan Gupta

Assignment 10

RSA

Theory:

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.

Code:

```
# Iterative Function to calculate (x^n)%p in O(logy)
def power(x, y, p):
  res = 1 # Initialize result

  x = x % p # Update x if it is more than or equal to p

  while (y > 0):
    # If y is odd, multiply x with result
    if (y & 1):
      res = (res * x) % p

    # y must be even now
    y = y >> 1 # y = y/2
    x = (x * x) % p
```

```
  return res

def encrypt(plaintext, e, n):
  return power(plaintext, e, n)

def decrypt(ciphertext, d, n):
  return power(ciphertext, d, n)

def main():

  print('1. Encrypt')
  print('2. Decrypt')

  choice = int(input('Enter your choice: '))

  if choice == 1:
    plaintext = int(input('Enter plaintext: '))

    print('Enter public key:')
    e = int(input('e = '))
    n = int(input('n = '))

    ciphertext = encrypt(plaintext, e, n)
    print('Ciphertext :',ciphertext)
  else:
    ciphertext = int(input('Enter ciphertext: '))

    print('Enter private key:')
    d = int(input('d = '))
    n = int(input('n = '))

    plaintext = decrypt(ciphertext, d, n)
    print("Plaintext: ",plaintext)

main()
```

Output:

```
In [4]: runfile('D:/CNS Lab/rsa.py', wdir='D:/CNS Lab')
1. Encrypt
2. Decrypt

Enter your choice: 1

Enter plaintext: 2
Enter public key:

e = 7

n = 527
Ciphertext : 128

In [5]: runfile('D:/CNS Lab/rsa.py', wdir='D:/CNS Lab')
1. Encrypt
2. Decrypt

Enter your choice: 2

Enter ciphertext: 128
Enter private key:

d = 343

n = 527
Plaintext:  2
```

For english character:

```
#Iterative Function to calculate(x ^ n) % p in O(logy)
```

```
def power(x, y, p):
  res = 1 # Initialize result

  x = x % p # Update x if it is more than or equal to p

  while (y > 0):
#If y is odd, multiply x with result
    if (y & 1):
      res = (res * x) % p

#y must be even now
    y = y >> 1 # y = y/2
    x = (x * x) % p

  return res

def encrypt(plaintext, e, n):
  return power(plaintext, e, n)

def decrypt(ciphertext, d, n):
  return power(ciphertext, d, n)

def main():

  print('1. Encrypt')
  print('2. Decrypt')

  choice = int(input('Enter your choice: '))

  if choice == 1:

    text = input('Enter plaintext: ')
    print('Enter public key:')
    e = int(input('e = '))
    n = int(input('n = '))
    print('Ciphertext :')
    for i in range(0, len(text)):
        plaintext= ord(text[i])
        ciphertext = encrypt(plaintext, e, n)
        print(ciphertext,end="")
```

```
  else:
    ciphertext = int(input('Enter ciphertext: '))

    print('Enter private key:')
    d = int(input('d = '))
    n = int(input('n = '))

    plaintext = decrypt(ciphertext, d, n)
    print("Plaintext: ",plaintext)

main()
```

```
In [5]: runfile('D:/CNS Lab/rsa.py', wdir='D:/CNS Lab')
1. Encrypt
2. Decrypt

Enter your choice: 1

Enter plaintext: sweety
Enter public key:

e = 7

n = 527
Ciphertext :
21119333391417
```

RSA Key Generator:-

```
def isPrime(n):
  if n < 2:
    return False
```

```
  if n != 2 and n % 2 == 0:
    return False

  for i in range(3, int(n ** 0.5) + 1, 2):
    if n % i == 0:
      return False

  return True

def gcd(a, b):
  r1 = a
  r2 = b

  while r2 != 0:
    q = r1 // r2
    r = r1 - q * r2

    r1 = r2
    r2 = r

  return r1

def multiplicativeInverse(a, b):
  r1 = a
  r2 = b
  t1 = 0
  t2 = 1

  while r2 != 0:
    q = r1 // r2
    r = r1 - q * r2
    t = t1 - q * t2

    r1 = r2
    r2 = r
    t1 = t2
    t2 = t

  if t1 > 0:
```

```
    return t1
  else:
    return a + t1

def generateKeys():
  p = 0
  q = 0

  while True:
    p = int(input('Enter large prime p: '))

    if isPrime(p):
      break
    else:
      print('Given number is not prime')

  while True:
    q = int(input('Enter large prime q: '))

    if p == q:
      print('q can not be equal to p')
    elif isPrime(q):
      break
    else:
      print('Given number is not prime')

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

  e = 0
  d = 0

  while True:
    e = int(input(f'Enter number e coprime to {phi}: '))

    if gcd(e, phi) == 1:
      d = multiplicativeInverse(phi, e)

      if e != d:
        break
```

```
      else:
        print('e == d not allowed; Try some other value for e')
    else:
      print('Given number not coprime')

  print(f'Public key: {{{e}, {n}}}')
  print(f'Private key: {{{d}, {n}}}')




generateKeys()
```

```
In [1]: runfile('D:/CNS Lab/rsa_key_generator.py', wdir='D:/CNS Lab')

Enter large prime p: 23

Enter large prime q: 13

Enter number e coprime to 264: 243
Given number not coprime

Enter number e coprime to 264: 245
Public key: {245, 299}
Private key: {125, 299}
```