**GAYATHRI K**

**CB.EN.P2CSE19008**

**BIG DATA SECURITY**

1. Code for Finding a large prime number (min 20-digit).

import time

def SieveOfEratosthenes(n):

prime = [True for i in range(n+1)]

p = 2

while(p \* p <= n):

if (prime[p] == True):

for i in range(p \* p, n + 1, p):

prime[i] = False

p += 1

c = 0

for p in range(2, n):

if prime[p]:

c += 1

return c,i

t0 = time.time()

c,i = SieveOfEratosthenes(100000000)

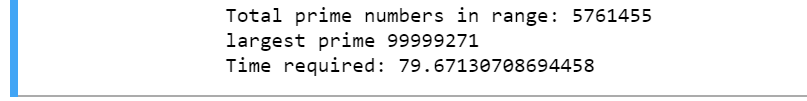
print("Total prime numbers in range:", c )

print("largest prime", i)

t1 = time.time()

print("Time required:", t1 - t0)

**Output**:



1. Code for finding Primitive root (min 10 digits)

from math import gcd as bltin\_gcd

def primRoots(modulo):

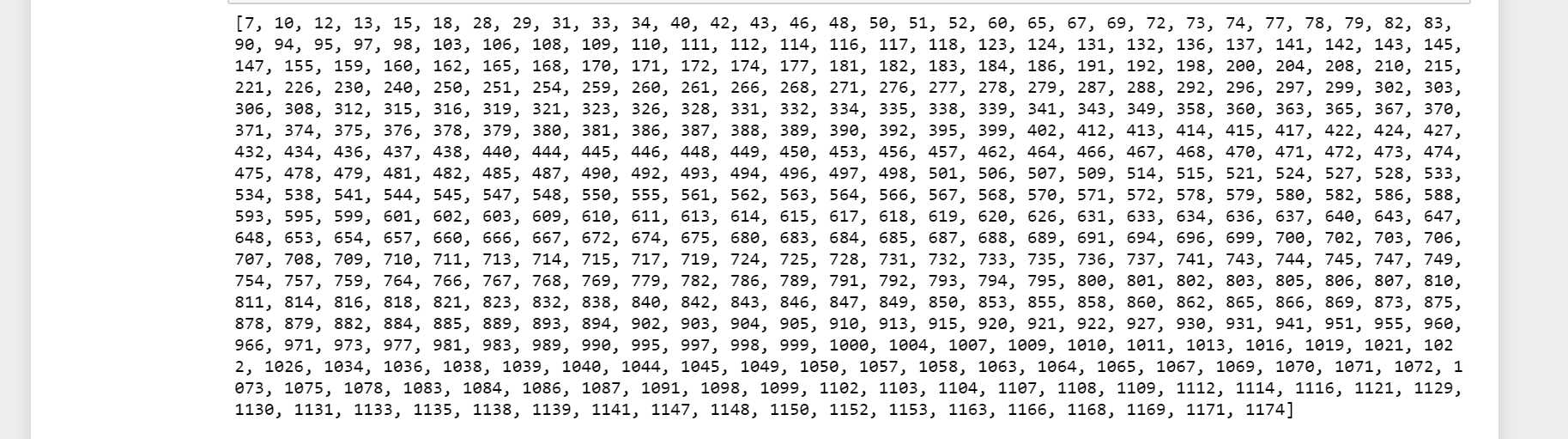
required\_set = {num for num in range(1, modulo) if bltin\_gcd(num, modulo) }

return [g for g in range(1, modulo) if required\_set == {pow(g, powers, modulo)

for powers in range(1, modulo)}]

print(primRoots(1181))

**Output**:



1. Code for DH (Taking random private keys)

from random import getrandbits

from random import randint

import sys

def is\_prime\_calc(num):

return all(num % i for i in range(2, num))

def is\_prime(num):

return is\_prime\_calc(num)

def get\_random\_prime():

while True:

n = getrandbits(12) + 3;

if is\_prime(n):

return n

def gcd(a,b):

while a != b:

if a > b:

a = a - b

else:

b = b - a

return a

def primitive\_root(modulo):

required\_set = set(num for num in range (1, modulo) if gcd(num, modulo) == 1)

for g in range(1, modulo):

actual\_set = set(pow(g, powers) % modulo for powers in range (1, modulo))

if required\_set == actual\_set:

return g

# Generating private keys

alice\_private = randint(999, 999999)

print ('Alice private key is %d' % alice\_private)

bob\_private = randint(999, 999999)

print ('Bob private key is %d' % bob\_private)

# Generating p-g parameters

p = get\_random\_prime()

g = primitive\_root(p)

print ('\n p parameter is %d, g parameter is %d \n' % (p, g))

# Generating public keys

alice\_public = pow(g, alice\_private) % p

bob\_public = pow(g, bob\_private) % p

print ('Alice public key is %d' % alice\_public)

print ('Bob public key is %d' % bob\_public)

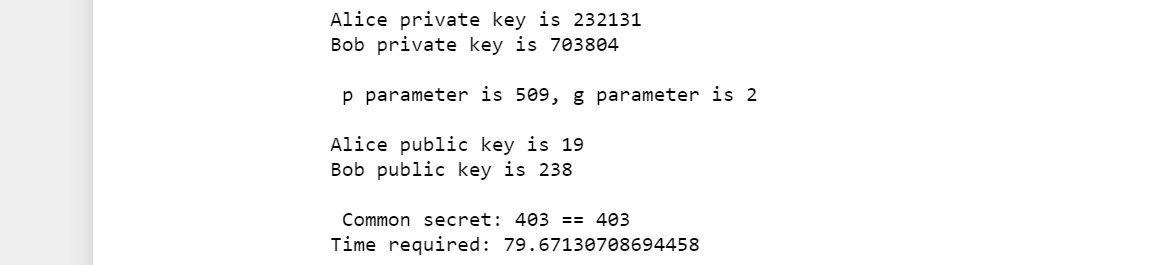
alice\_key = (pow(bob\_public, alice\_private)) % p

bob\_key = (pow(alice\_public, bob\_private)) % p

print ('\n Common secret: %d == %d' % (alice\_key, bob\_key))

print("Time required:", t1 - t0)

**Output**:



1. Code to Find an integer **k** such that  where a and m are relatively prime.

import math;

def discreteLogarithm(a, b, m):

n = int(math.sqrt (m) + 1);

# Calculate a ^ n

an = 1;

for i in range(n):

an = (an \* a) % m;

value = [0] \* m;

# Store all values of a^(n\*i) of LHS

cur = an;

for i in range(1, n + 1):

if (value[ cur ] == 0):

value[ cur ] = i;

cur = (cur \* an) % m;

cur = b;

for i in range(n + 1):

# Calculate (a ^ j) \* b and check

# for collision

if (value[cur] > 0):

ans = value[cur] \* n - i;

if (ans < m):

return ans;

cur = (cur \* a) % m;

return -1;

# Driver code

a = 200;

b = 3;

m = 5;

print(discreteLogarithm(a, b, m));

a = 350;

b = 7;

m = 11;

print(discreteLogarithm(a, b, m));

**Output**:



1. Code

from decimal import Decimal

def gcd(a,b):

if b==0:

return a

else:

return gcd(b,a%b)

p = int(input('Enter the value of p = '))

q = int(input('Enter the value of q = '))

no = int(input('Enter the value of text = '))

n = p\*q

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

for e in range(2,t):

if gcd(e,t)== 1:

break

for i in range(1,10):

x = 1 + i\*t

if x % e == 0:

d = int(x/e)

break

ctt = Decimal(0)

ctt =pow(no,e)

ct = ctt % n

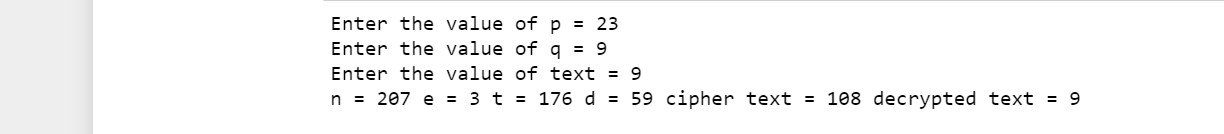
dtt = Decimal(0)

dtt = pow(ct,d)

dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))

**Output**:



from decimal import Decimal

def gcd(a,b):

if b==0:

return a

else:

return gcd(b,a%b)

p = int(input('Enter the value of p = '))

q = int(input('Enter the value of q = '))

no = int(input('Enter the value of text = '))

n = p\*q

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

for e in range(2,t):

if gcd(e,t)== 1:

break

for i in range(1,10):

x = 1 + i\*t

if x % e == 0:

d = int(x/e)

break

ctt = Decimal(0)

ctt =pow(no,e)

ct = ctt % n

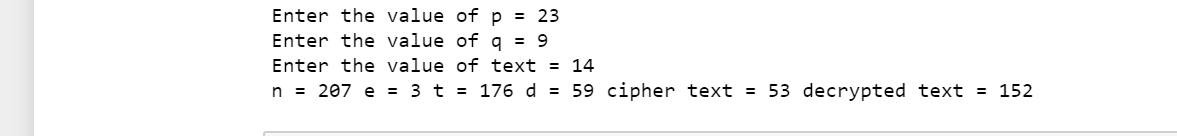
dtt = Decimal(0)

dtt = pow(ct,d)

dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))

**Output**:



from decimal import Decimal

def gcd(a,b):

if b==0:

return a

else:

return gcd(b,a%b)

p = int(input('Enter the value of p = '))

q = int(input('Enter the value of q = '))

no = int(input('Enter the value of text = '))

n = p\*q

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

for e in range(2,t):

if gcd(e,t)== 1:

break

for i in range(1,10):

x = 1 + i\*t

if x % e == 0:

d = int(x/e)

break

ctt = Decimal(0)

ctt =pow(no,e)

ct = ctt % n

dtt = Decimal(0)

dtt = pow(ct,d)

dt = dtt % n

print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))

**Output**:

