**DAY-5**

Q.No.16 Write a high level code for DES algorithm for decryption, the 16 keys (K1, K2,... K16) are used in reverse order. Design a key-generation scheme with the appropriate shift schedule for the decryption process

PROGRAM:

def printString(S, N):

plaintext = [None] \* 5

freq = [0] \* 26

freqSorted = [None] \* 26

used = [0] \* 26

for i in range(N):

if S[i] != ' ':

freq[ord(S[i]) - 65] += 1

for i in range(26):

freqSorted[i] = freq[i]

T = "ETAOINSHRDLCUMWFGYPBVKJXQZ"

freqSorted.sort(reverse = True)

for i in range(5):

ch = -1

for j in range(26):

if freqSorted[i] == freq[j] and used[j] == 0:

used[j] = 1

ch = j

break

if ch == -1:

break

x = ord(T[i]) - 65

x = x - ch

curr = ""

for k in range(N):

if S[k] == ' ':

curr += " "

continue

y = ord(S[k]) - 65

y += x

if y < 0:

y += 26

if y > 25:

y -= 26

curr += chr(y + 65)

plaintext[i] = curr

for i in range(5):

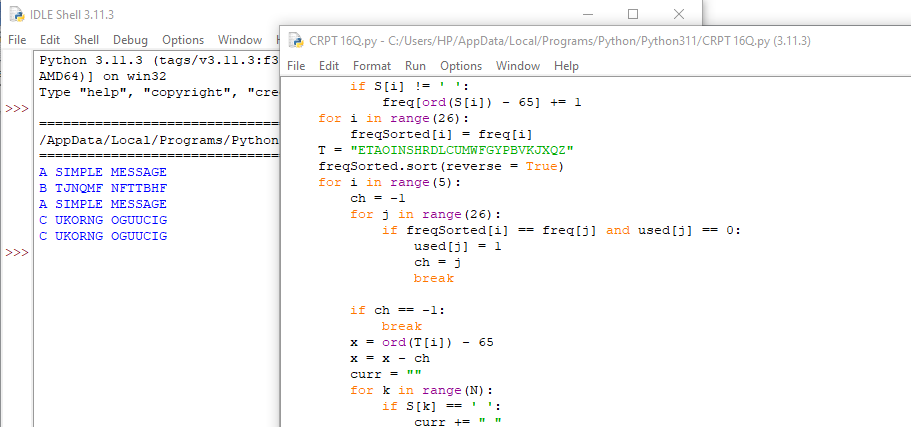
print(plaintext[i])

S = "B TJNQMF NFTTBHF"

N = len(S)

printString(S, N)

OUTPUT:



Q.No.17 Write a high-level code for RSA public-key encryption scheme, each user has a public key, e, and a private key, d. Suppose Bob leaks his private key. Rather than generating a new modulus, he decides to generate a new public and a new private key. Is this safe?

PROGRAM:

import java.math.BigInteger;

import java.security.SecureRandom;

public class RSA {

private final static BigInteger one = new BigInteger("1");

private final static SecureRandom random = new SecureRandom();

private BigInteger privateKey;

private BigInteger publicKey;

private BigInteger modulus;

public RSA(int bitLength) {

BigInteger p = BigInteger.probablePrime(bitLength / 2, random);

BigInteger q = BigInteger.probablePrime(bitLength / 2, random);

modulus = p.multiply(q);

BigInteger phi = (p.subtract(one)).multiply(q.subtract(one));

publicKey = new BigInteger("65537"); // Common public exponent

privateKey = publicKey.modInverse(phi);

}

public BigInteger encrypt(BigInteger message) {

return message.modPow(publicKey, modulus);

}

public BigInteger decrypt(BigInteger encryptedMessage) {

return encryptedMessage.modPow(privateKey, modulus);

}

public static void main(String[] args) {

RSA rsa = new RSA(1024);

BigInteger message = new BigInteger("1234567890");

BigInteger encryptedMessage = rsa.encrypt(message);

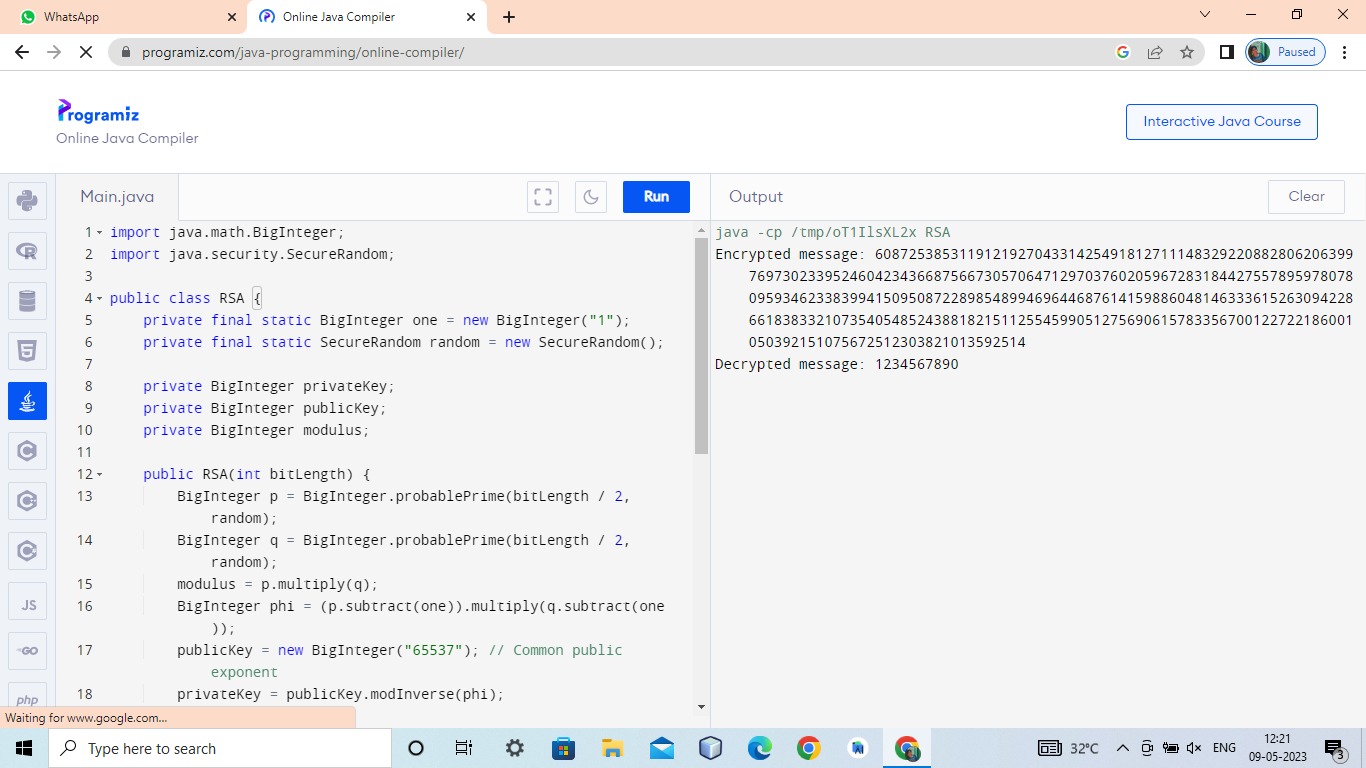
System.out.println("Encrypted message: " + encryptedMessage);

BigInteger decryptedMessage = rsa.decrypt(encryptedMessage);

System.out.println("Decrypted message: " + decryptedMessage);

}}

OUTPUT:



Q.No. 18. Write a high-level code for Bob using the RSA cryptosystem with a very large modulus n for which the factorization cannot be found in a reasonable amount of time. Suppose Alice sends a message to Bob by representing each alphabetic character as an integer between 0 and 25 (A S 0, c, Z S 25) and then encrypting each number separately using RSA with large e and large n. Is this method secure? If not, describe the most efficient attack against this encryption method.

PROGRAM:

import random

import math

# Function to compute modular exponentiation (base^exponent mod modulus)

def mod\_exp(base, exponent, modulus):

result = 1

while exponent > 0:

if exponent % 2 == 1:

result = (result \* base) % modulus

base = (base \* base) % modulus

exponent = exponent // 2

return result

# Function to generate a random prime number within a given range

def generate\_prime(min\_value, max\_value):

while True:

prime\_candidate = random.randint(min\_value, max\_value)

if is\_prime(prime\_candidate):

return prime\_candidate

# Function to check if a number is prime

def is\_prime(number):

if number < 2:

return False

for i in range(2, int(number \*\* 0.5) + 1):

if number % i == 0:

return False

return True

# Function to generate RSA keys

def generate\_rsa\_keys():

# Choose two large prime numbers p and q

p = generate\_prime(10\*10, 10\*11)

q = generate\_prime(10\*10, 10\*11)

# Compute modulus n

n = p \* q

# Compute Euler's totient function value (phi(n))

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

# Choose a large public exponent e

e = random.randint(2, phi - 1)

while math.gcd(e, phi) != 1:

e = random.randint(2, phi - 1)

# Compute the modular multiplicative inverse of e (private exponent d)

d = pow(e, -1, phi)

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

# Function to encrypt a plaintext using RSA public key

def encrypt(plaintext, public\_key):

e, n = public\_key

ciphertext = [mod\_exp(char, e, n) for char in plaintext]

return ciphertext

# Function to decrypt a ciphertext using RSA private key

def decrypt(ciphertext, private\_key):

d, n = private\_key

plaintext = [mod\_exp(char, d, n) for char in ciphertext]

return plaintext

# Generate RSA keys

public\_key, private\_key = generate\_rsa\_keys()

# Example usage

plaintext = [0, 18, 2, 14] # Representing "A R C O"

ciphertext = encrypt(plaintext, public\_key)

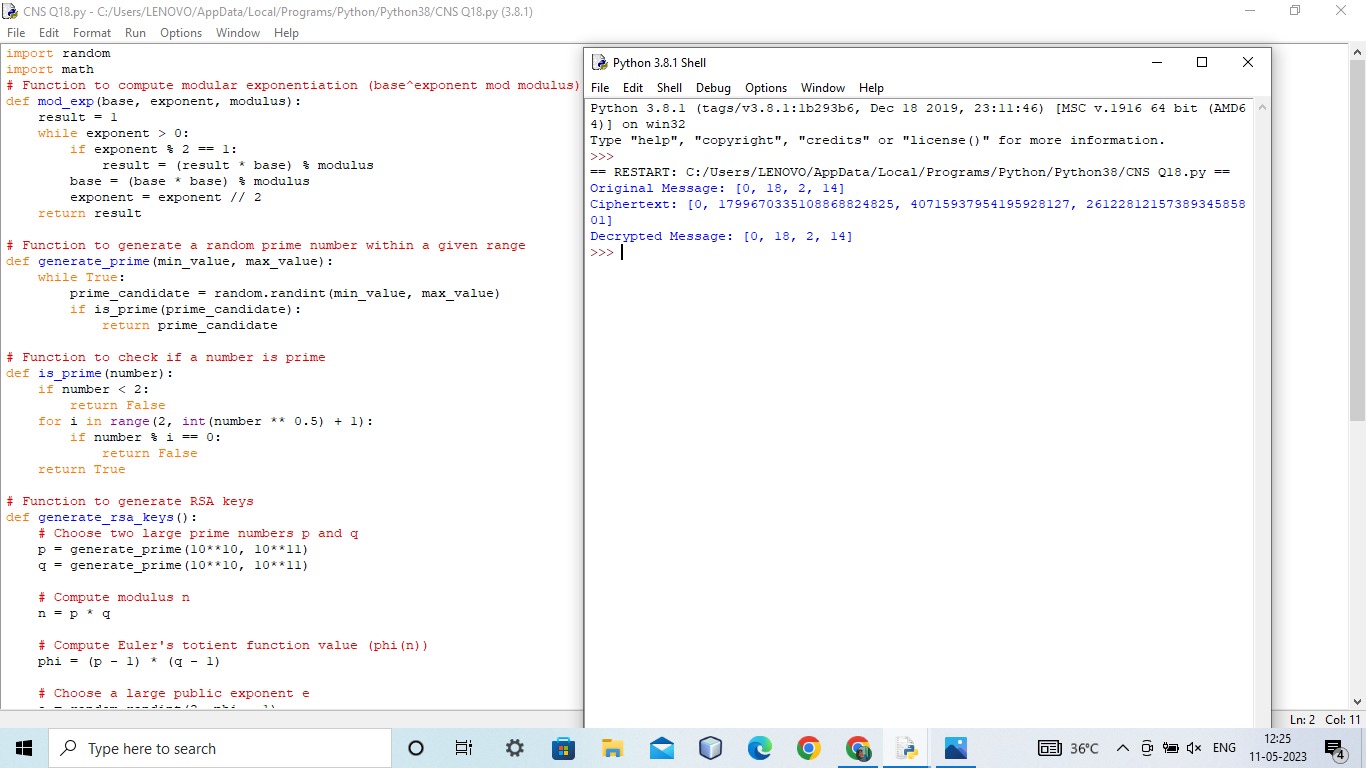
decrypted\_text = decrypt(ciphertext, private\_key)

print("Original Message:", plaintext)

print("Ciphertext:", ciphertext)

print("Decrypted Message:", decrypted\_text)

OUTPUT:



Q.No. 19 Write a High-level code for the Diffie-Hellman protocol, each participant selects a secret number x and sends the other participant ax mod q for some public number a. What would happen if the participants sent each other xa for some public number an instead? Give at least one method Alice and Bob could use to agree on a key. Can Eve break your system without finding the secret numbers? Can Eve find the secret numbers?

PROGRAM:

#include <math.h>

#include <stdio.h>

long long int power(long long int a, long long int b,

long long int P)

{

if (b == 1)

return a;

else

return (((long long int)pow(a, b)) % P);

}

int main()

{

long long int P, G, x, a, y, b, ka, kb;

P = 23;

printf("The value of P : %lld\n", P);

G = 9;

printf("The value of G : %lld\n\n", G);

a = 4;

printf("The private key a for Alice : %lld\n", a);

x = power(G, a, P);

b = 3;

printf("The private key b for Bob : %lld\n\n", b);

y = power(G, b, P);

ka = power(y, a, P);

kb = power(x, b, P);

printf("Secret key for the Alice is : %lld\n", ka);

printf("Secret Key for the Bob is : %lld\n", kb);

return 0;

}

OUTPUT:

