**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з комп’ютерного практикуму №2 з дисципліни

«Математичні основи захисту інформації»

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Прищепа В. С.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Марковський О. П.*

*-18 Лаб on-line.*

Київ 2024

**Варіант 19**

19 mod 17 = 2

|  |  |
| --- | --- |
| Номер | Завдання |
| 2 | Згенерувати пару ключів (відкриваючий та закриваючий) розрядністю 16 для алгоритму RSA і потім імітувати злам алгоритму шляхом факторизації модуля |

**Ціль роботи:** Отримати практичні навички в шифруванні повідомлень алгоритмом RSA та взламу цього алгоритму шляхом факторизації модуля.

**Хід роботи:**

**Лістинг:**

import random

import math

# GCD

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def modinv(a, m):

m0, x0, x1 = m, 0, 1

if m == 1:

return 0

while a > 1:

q = a // m

m, a = a % m, m

x0, x1 = x1 - q \* x0, x0

if x1 < 0:

x1 += m0

return x1

# prime generation

def generate\_prime(bits):

while True:

num = random.getrandbits(bits)

if num % 2 == 0:

num += 1

for \_ in range(100):

if all(num % p != 0 for p in range(3, int(math.sqrt(num)) + 1, 2)):

return num

num += 2

#rsa keys generation

def generate\_rsa\_keys(bits):

p = generate\_prime(bits // 2)

q = generate\_prime(bits // 2)

n = p \* q

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

e = 3

while gcd(e, phi) != 1:

e += 2

d = modinv(e, phi)

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

# encryption and decryption

def rsa\_encrypt(message, pubkey):

e, n = pubkey

encrypted\_text = 1

while e > 0:

encrypted\_text \*= message

encrypted\_text %= n

e -= 1

return encrypted\_text

def rsa\_decrypt(ciphertext, privkey):

d, n = privkey

decrypted = 1

while d > 0:

decrypted \*= ciphertext

decrypted %= n

d -= 1

return decrypted

# facrorization

def factorize(n):

for i in range(2, int(math.sqrt(n)) + 1):

if n % i == 0:

return i, n // i

return n, 1

def calculate\_private\_key(e, p, q):

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

d = modinv(e, phi)

return d

def encoder(message, pubkey):

encoded = []

# Calling the encrypting function in encoding function

for letter in message:

encoded.append(rsa\_encrypt(ord(letter), pubkey))

return encoded

def decoder(encoded, privkey):

s = ''

# Calling the decrypting function decoding function

for num in encoded:

s += chr(rsa\_decrypt(num, privkey))

return s

# 16-bit RSA key pair generation

pubkey, privkey = generate\_rsa\_keys(16)

print("Public key (e, n):", pubkey)

print("Private key (d, n):", privkey)

# Verification of text message encryption and decryption

# Uncomment below for manual input

message = input("Enter the message\n")

# Calling the encoding function

coded = encoder(message, pubkey)

print("\nThe encoded message(encrypted by public key)")

print(''.join(str(p) for p in coded))

print("The decoded message(decrypted by private key)")

print(''.join(str(p) for p in decoder(coded, privkey)))

# Simulating RSA cracking by modulus factorization

n = pubkey[1]

p, q = factorize(n)

print("\nCracking RSA by factoring:")

print("Module n:", n)

print("Factored into prime numbers p and q:", p, q)

print("Check: p \* q =", p \* q)

# Calculation of the private key

e = pubkey[0]

d = calculate\_private\_key(e, p, q)

cracked\_privkey = (d, n)

print("Calculated private key (d, n):", cracked\_privkey)

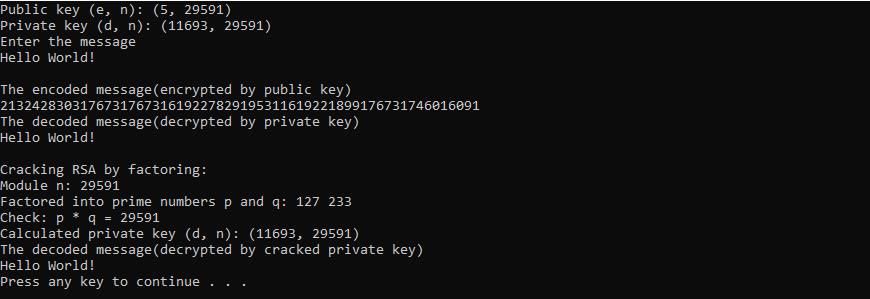
# Using the calculated private key for decryption

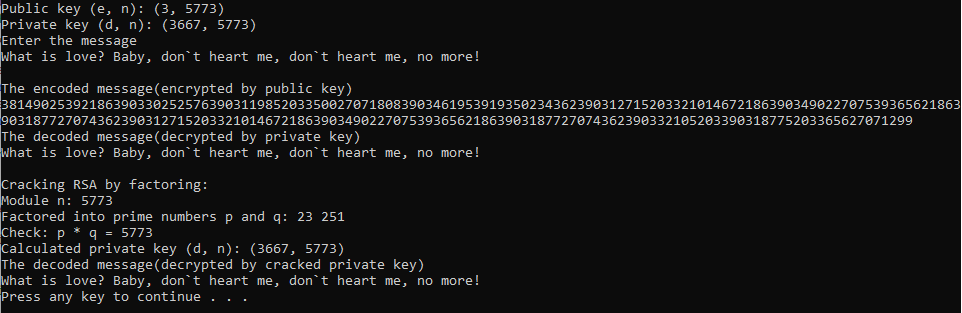
print("The decoded message(decrypted by cracked private key)")

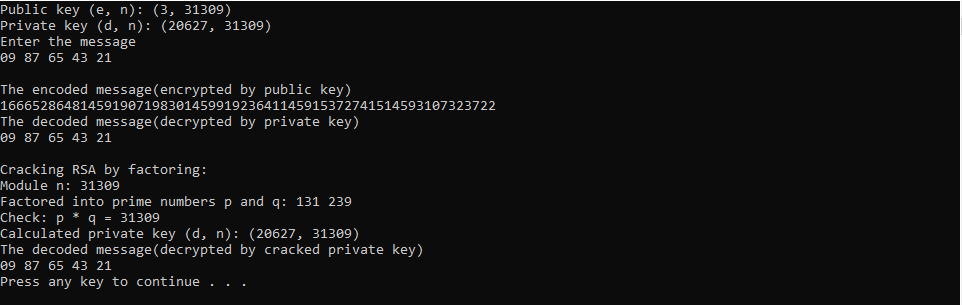
print(''.join(str(p) for p in decoder(coded, cracked\_privkey)))

**Результат виконання:**

Message = “Hello World!”

Message = “What is love? Baby, don`t heart me, don`t heart me, no more!”

Message = “09 87 65 43 21”

**Висновок:**

У ході виконання лабораторної роботи №2 я був ознайомлений з алгоритмом шифрування повідомлень RSA. В процесі розробки програмного забезпечення для реалізації алгоритму була використана мова програмування Python. Під час роботи було згенеровано відкриваючий та закриваючий ключі розрядністю 16, зчитане повідомлення користувача, потім зашифроване і розшифроване ключами, а резульати шифрування та розшифрування виведені на екран.

Потім було зімітовано злам алгоритму шляхом факторизації модуля. Було вирахувано закриваючий ключ, і за допомогою нього розшифровано зашифроване повідомлення. Результати розшифрування співпали, отже злам був успішний.

У результаті виконання лабораторної роботи було отримано практичні навички з розробки алгоритмів шифрування та перевірки їх правильності, що є важливим кроком у вивченні криптографії та інформаційної безпеки.