МИНОБРНАУКИ РОССИИ

Изображение выглядит как текст

Автоматически созданное описаниеФедеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Отчет

По лабораторной работе №5

«Ассиметричные алгоритмы шифрования»»

По дисциплине

«Защита информации»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капранов С.Н.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Игнаков К. М.

(подпись) (фамилия, и.,о.)

19-ВМ

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2022

**Задание №8.1.**

Реализовать алгоритм шифрования данных «RSA».

**RSA** — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

1. Вычисление пары ключей (КB, kB) получателем В, на основе начального условия должно быть простым.

2. Отправитель А, зная открытый ключ КB и сообщение М, может легко вычислить криптограмму

С = EKB(М) = ЕB(М)

3. Получатель В, используя секретный ключ kB и криптограмму С, может легко восстановить исходное сообщение

M = DkB(C) = DB(C) = DB[EB(M)].

4. Противник, зная открытый ключ КB при попытке вычислить секретный ключ kB наталкивается на непреодолимую вычислительную проблему.

5. Противник, зная пару (КB, С), при попытке вычислить исходное сообщение М наталкивается на непреодолимую вычислительную проблему.

**Алгоритм.**

1. Вводим простые числа P и Q.

2. Затем вычисляем N, KB и kb:

* N = P \* Q
* f(N) = (P-1)\*(Q-1)
* KB = [2;f] и НОД(KB,f) = 1
* kb = (k\*f+1)modKB, где k = 1,2,3…

3. Вводим сообщение.

4. Каждый символ переводим в число и кодируем: MiKBmodN=Ci.

5. Выводим закодированное сообщение.

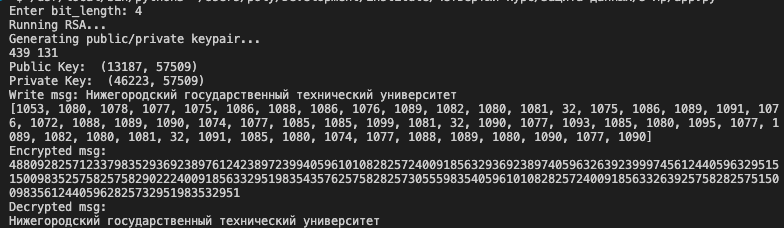
6. Каждый код мы декодируем: CikbmodN=Mi.

7. Затем каждое полученное число переводим в символ.

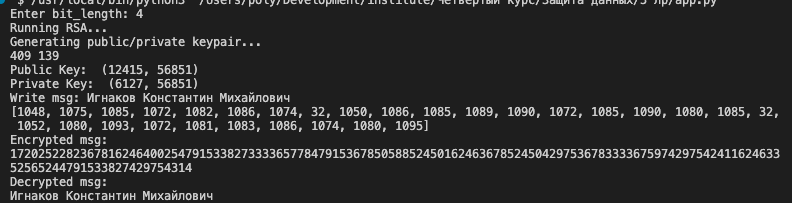
8. Выводим расшифрованное сообщение.

**Примеры шифрования/расшифровывания.**

1. Нижегородский государственный технический университет:



2.Игнаков Константин Михайлович:



**Текст программы.**

# required for the sqrt() function, if you want to avoid doing \*\*0.5

import random

from math import sqrt

# required for randrange

from random import randint as rand

# just to use the well known keyword rand() from C++

def gcd(a, b):

if b == 0:

return a

else:

return gcd(b, a % b)

def mod\_inverse(a, m):

for x in range(1, m):

if (a \* x) % m == 1:

return x

return -1

def isprime(n):

if n < 2:

return False

elif n == 2:

return True

else:

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

if n % i == 0:

return False

return True

# initial two random numbers p,q

p = rand(1, 1000)

q = rand(1, 1000)

def generate\_keypair(p, q, keysize):

# keysize is the bit length of n so it must be in range(nMin,nMax+1).

# << is bitwise operator

# x << y is same as multiplying x by 2\*\*y

# i am doing this so that p and q values have similar bit-length.

# this will generate an n value that's hard to factorize into p and q.

nMin = 1 << (keysize - 1)

nMax = (1 << keysize) - 1

primes = [2]

# we choose two prime numbers in range(start, stop) so that the difference of bit lengths is at most 2.

start = 1 << (keysize // 2 - 1)

stop = 1 << (keysize // 2 + 1)

if start >= stop:

return []

for i in range(3, stop + 1, 2):

for p in primes:

if i % p == 0:

break

else:

primes.append(i)

while (primes and primes[0] < start):

del primes[0]

# choosing p and q from the generated prime numbers.

while primes:

p = random.choice(primes)

primes.remove(p)

q\_values = [q for q in primes if nMin <= p \* q <= nMax]

if q\_values:

q = random.choice(q\_values)

break

print(p, q)

n = p \* q

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

# generate public key 1<e<phi(n)

e = random.randrange(1, phi)

g = gcd(e, phi)

while True:

# as long as gcd(1,phi(n)) is not 1, keep generating e

e = random.randrange(1, phi)

g = gcd(e, phi)

# generate private key

d = mod\_inverse(e, phi)

if g == 1 and e != d:

break

# public key (e,n)

# private key (d,n)

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

def encrypt(msg\_plaintext, package):

# unpack key value pair

e, n = package

msg\_ciphertext = [pow(ord(c), e, n) for c in msg\_plaintext]

return msg\_ciphertext

def decrypt(msg\_ciphertext, package):

d, n = package

msg\_plaintext = [chr(pow(c, d, n)) for c in msg\_ciphertext]

# No need to use ord() since c is now a number

# After decryption, we cast it back to character

# to be joined in a string for the final result

return (''.join(msg\_plaintext))

# -------------------------------------------------------------

# driver program

if \_\_name\_\_ == "\_\_main\_\_":

bit\_length = int(input("Enter bit\_length: "))

print("Running RSA...")

print("Generating public/private keypair...")

public, private = generate\_keypair(

p, q, 2 \*\* bit\_length) # 8 is the keysize (bit-length) value.

print("Public Key: ", public)

print("Private Key: ", private)

msg = input("Write msg: ")

print([ord(c) for c in msg])

encrypted\_msg = encrypt(msg, public)

print("Encrypted msg: ")

print(''.join(map(lambda x: str(x), encrypted\_msg)))

print("Decrypted msg: ")

print(decrypt(encrypted\_msg, private))