Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчёт

по лабораторной работе №4

**Асимметричная криптография.**

**Алгоритм Эль-Гамаля**

|  |  |
| --- | --- |
| Выполнил:  студент группы 653502  Куликов А.Д. | Проверил:  Артемьев В.С. |

Минск 2019

# Постановка задачи

Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма Эль-Гамаля.

# Теоретическая справка

Схема Эль-Гамаля (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94).

Схема была предложена Тахером Эль-Гамалем в 1985 году. Эль-Гамаль разработал один из вариантов алгоритма Диффи-Хеллмана. Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA алгоритм Эль-Гамаля не был запатентован и, поэтому, стал более дешевой альтернативой, так как не требовалась оплата взносов за лицензию. Считается, что алгоритм попадает под действие патента Диффи-Хеллмана.

Генерация ключей:

1. Генерируется случайное простое число p.
2. Выбирается целое число g — первообразный корень p.
3. Выбирается случайное целое число x такое, что 1<x<p-1.
4. Вычисляется y = gx mod p.
5. Открытым ключом является y, закрытым ключом — число x.

Шифросистема Эль-Гамаля является фактически одним из способов выработки открытых ключей Диффи — Хеллмана. Шифрование по схеме Эль-Гамаля не следует путать с алгоритмом цифровой подписи по схеме Эль-Гамаля.

Шифрование

Сообщение M должно быть меньше числа p. Сообщение шифруется следующим образом:

1. Выбирается сессионный ключ — случайное целое число k такое, что 1<k<p-1.
2. Вычисляются числа a=gk mod p и b=yk \* M mod p.
3. Пара чисел (a,b) является шифротекстом.

Нетрудно увидеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

Расшифрование

Зная закрытый ключ x, исходное сообщение можно вычислить из шифротекста (a,b) по формуле: M=b\*(ax)-1 mod p

При этом нетрудно проверить, что (ax)-1 = g –kx mod p

и поэтому b\*(ax)-1 = (yk\*M) \* g –kx = (gxk \* M) \* g –kx = M (mod p).

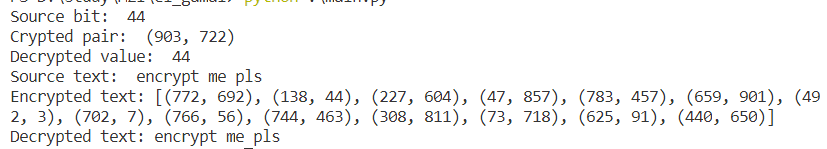
Для практических вычислений больше подходит следующая формула: M=b (ax)-1 = ba p-1-x (mod p).

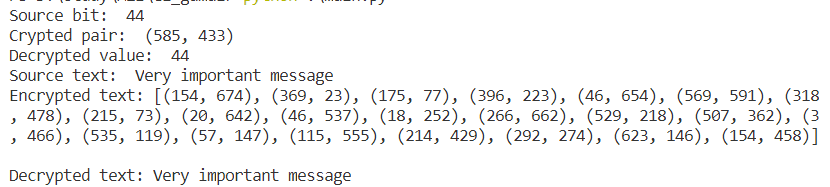
# Блок-схема алгоритма

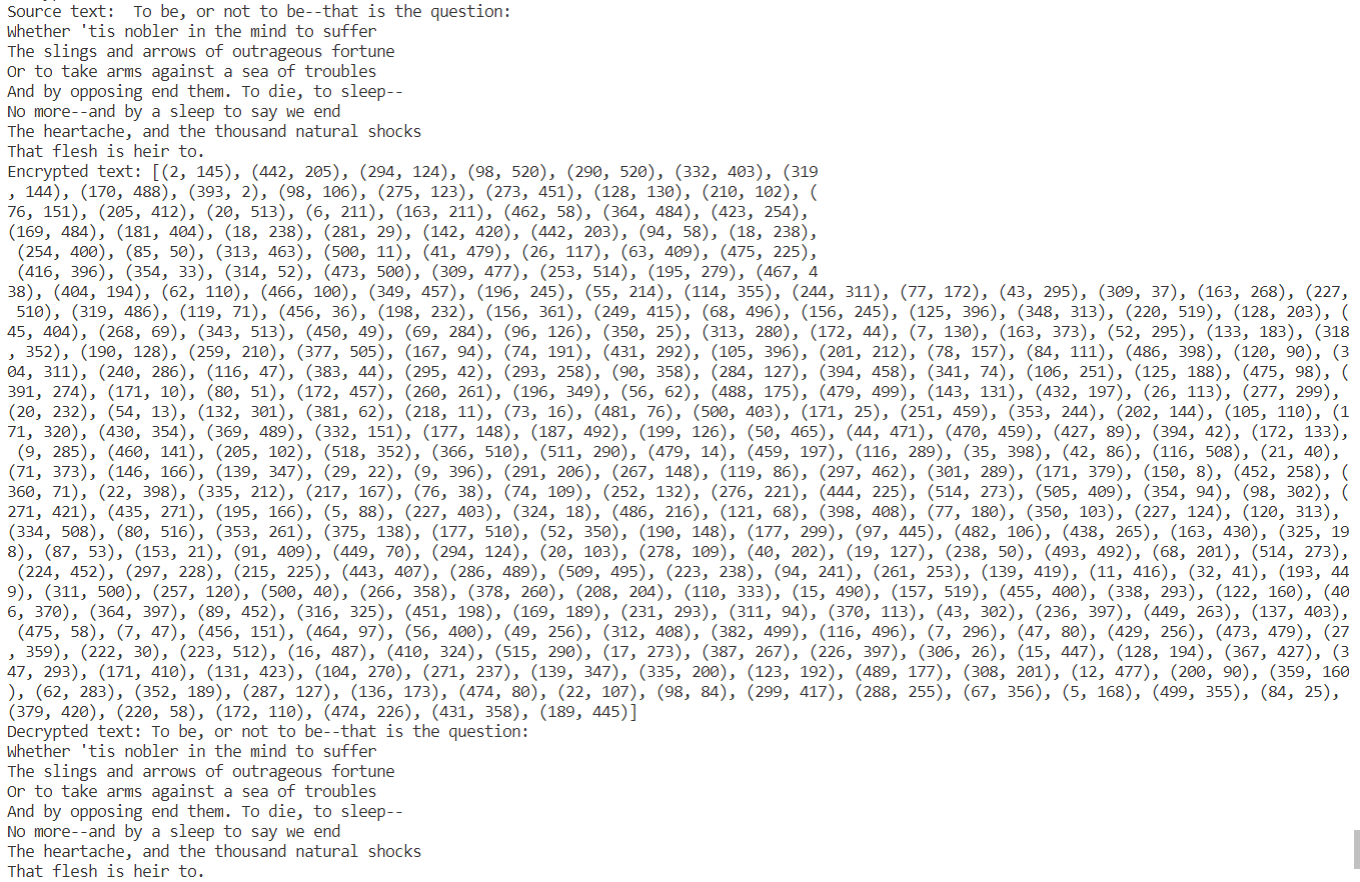
A screenshot of a cell phone

Description automatically generated

# Результат выполнения программы







# Приложение

**crypto.py**

import random

def encrypt\_bit(number, p, g, y):

    k = random.randint(2, p - 2)

    a = pow(g, k, p)

    b = number \* pow(y, k, p) % p

    return a, b  # crypto pair

def decrypt\_bit(crypto\_pair, public\_key, private\_key):

    p, \_, \_ = public\_key

    x = private\_key

    a, b = crypto\_pair

    decrypted = (b \* a \*\* (p - 1 - x)) % p  # wiki формула для практических вычислений

    return decrypted

def encrypt(text, public\_key):

    p, g, y = public\_key

    encrypted = []

    for char in text:

        encrypted.append(encrypt\_bit(ord(char), p, g, y))

    return encrypted

def decrypt(crypto\_pairs, public\_key, private\_key):

    decrypted = []

    for crypto\_pair in crypto\_pairs:

        decrypted.append(chr(decrypt\_bit(crypto\_pair, public\_key, private\_key)))

    return decrypted

**math\_utils.py**

import random

MAX\_NUM = 1000

def gcd(a, b):

    while b != 0:

        a, b = b, a % b

    return a

def is\_prime(n):

    if n <= 1:

        return False

    for i in range(2, n):

        if n % i == 0:

            return False

    return True

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

def generate\_prime\_number():

    num = random.randint(500, MAX\_NUM)

    while not is\_prime(num):

        num = random.randint(500, MAX\_NUM)

    return num

**main.py**

import random

from math\_utils import generate\_prime\_number, primitive\_root

from crypto import encrypt\_bit, encrypt, decrypt\_bit, decrypt

def main():

    p = generate\_prime\_number()

    g = primitive\_root(p)

    x = random.randint(2, p - 2)  # private key

    y = (g \*\* x) % p

    public\_key = (p, g, y)

    private\_key = x

    source = 44

    crypto\_pair = encrypt\_bit(source, p, g, y)

    decrypted\_value = decrypt\_bit(crypto\_pair, public\_key, private\_key)

    print('Source bit: ', source)

    print('Crypted pair: ', crypto\_pair)

    print('Decrypted value: ', decrypted\_value)

    source = """To be, or not to be--that is the question:

Whether 'tis nobler in the mind to suffer

The slings and arrows of outrageous fortune

Or to take arms against a sea of troubles

And by opposing end them. To die, to sleep--

No more--and by a sleep to say we end

The heartache, and the thousand natural shocks

That flesh is heir to."""

    encrypted\_text = encrypt(source, public\_key)

    decrypted\_text = decrypt(encrypted\_text, public\_key, private\_key)

    print('Source text: ', source)

    print('Encrypted text:', encrypted\_text)

    print('Decrypted text:', ''.join(decrypted\_text))

if \_\_name\_\_ == '\_\_main\_\_':

    main()