Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 9**

«ИССЛЕДОВАНИЕ АСИММЕТРИЧНЫХ ШИФРОВ»

Выполнил:

Студент: Лысков К.Е.

ФИТ 3 курс 4 группа

Преподаватель: Сазонова Д.В.

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования Python и позволяет провести шифрование асимметричным шифром.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, которое реализовывает генерацию сверхвозрастающей последовательности, вычисление нормальной последовательности, зашифрование сообщения и его расшифровка. На листинге 2.1 представлены функции, реализующие данную функциональность.

def ReadFile(name\_file):

    file\_for\_only\_read = open(name\_file, 'r', encoding="utf8")

    return file\_for\_only\_read.read()

def EGCD(a, b):

    if a == 0:

        return b, 0, 1

    else:

        g, y, x = EGCD(b % a, a)

        return g, x - (b // a) \* y, y

def GenerateSuperGrowingSequence(len\_sequence):

*# 2\_535\_301\_200\_456\_458\_802\_993\_406\_410\_752 - 100-битное число*

    sequence = [1, 2]

    while len(sequence) != len\_sequence:

        next\_value = sum(sequence) + 1

        sequence.append(next\_value)

    return sequence

def GetPublicKey(sequence, a, n):

    public\_key = []

    for elem in sequence:

        public\_key.append((elem \* a) % n)

    return public\_key

def GenerateMinimumValueA(sequence):

    n = sum(sequence) + 1

    counter = 0

    for i in range(2, n):

        if n % i != 0 and counter == 10:

            return i

        elif n % i != 0:

            counter += 1

def EncryptByPackingTheSatchel(public\_key, message):

    message\_2 = [(len(public\_key) - len(bin(ord(i))[2:]))

                 \* '0' + bin(ord(i))[2:] for i in message]

    encrypt\_message\_10 = []

    encrypt\_message = []

    for elem in message\_2:

        current\_encrypt\_number = 0

        for i in range(len(elem)):

            if elem[i] == '1':

                current\_encrypt\_number += public\_key[i]

        encrypt\_message\_10.append(current\_encrypt\_number)

        encrypt\_message.append(chr(current\_encrypt\_number))

    return encrypt\_message\_10, ''.join(encrypt\_message)

def DecryptByPackingTheSatchel(inverse\_a, encrypt\_message\_10, n, sequence):

    decrypt\_message\_10 = []

    decrypt\_message = []

    for i in encrypt\_message\_10:

        decrypt\_message\_10.append((i \* inverse\_a) % n)

    for i in decrypt\_message\_10:

        current\_symbol = ''

        for index, j in enumerate(sequence[::-1]):

            if j <= i:

                i -= j

                current\_symbol += '1'

            else:

                current\_symbol += '0'

        decrypt\_message.append(chr(int(current\_symbol[::-1], 2)))

    return decrypt\_message\_10, ''.join(decrypt\_message)

Листинг 2.1 –код программы, реализующие заданную ранее функциональность

**3. Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение. Рисунок 3.1, 3.2, 3.3 и 3.4 показывают требуемые в данной лабораторной работе результаты.

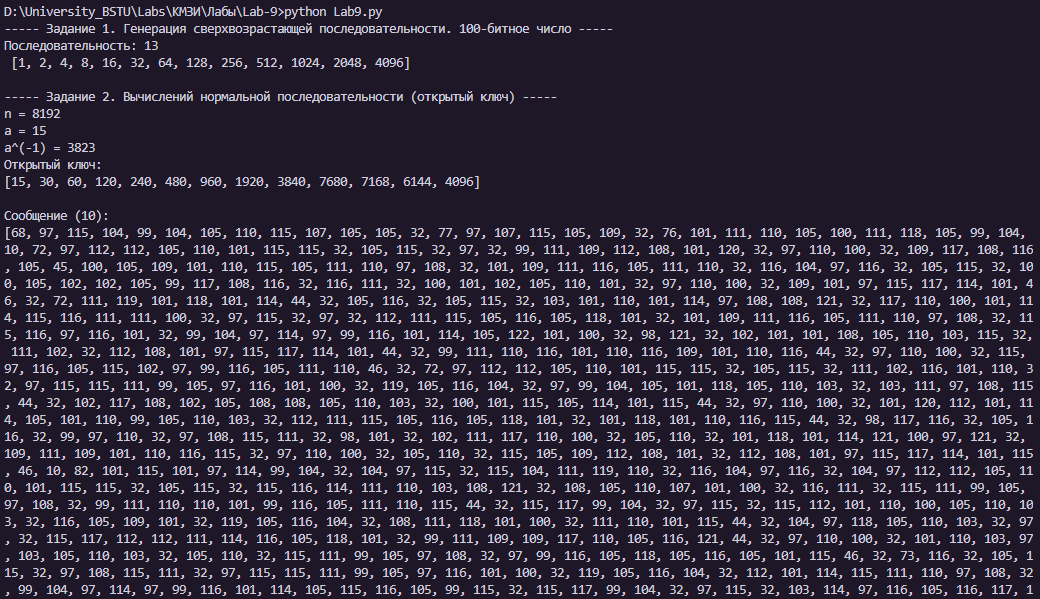


Рисунок 3.1 – Результат работы

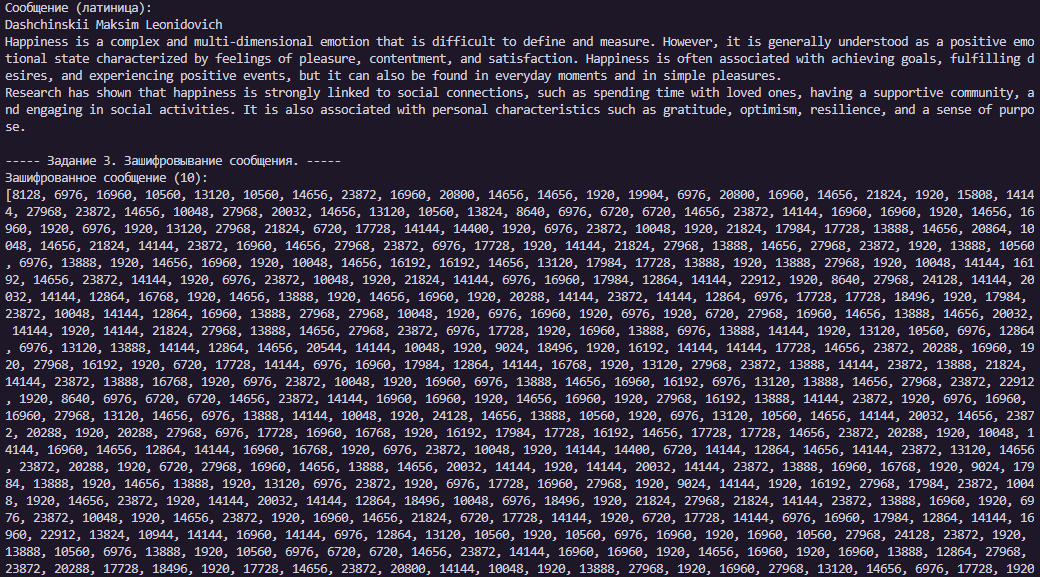


Рисунок 3.2 – Результат работы

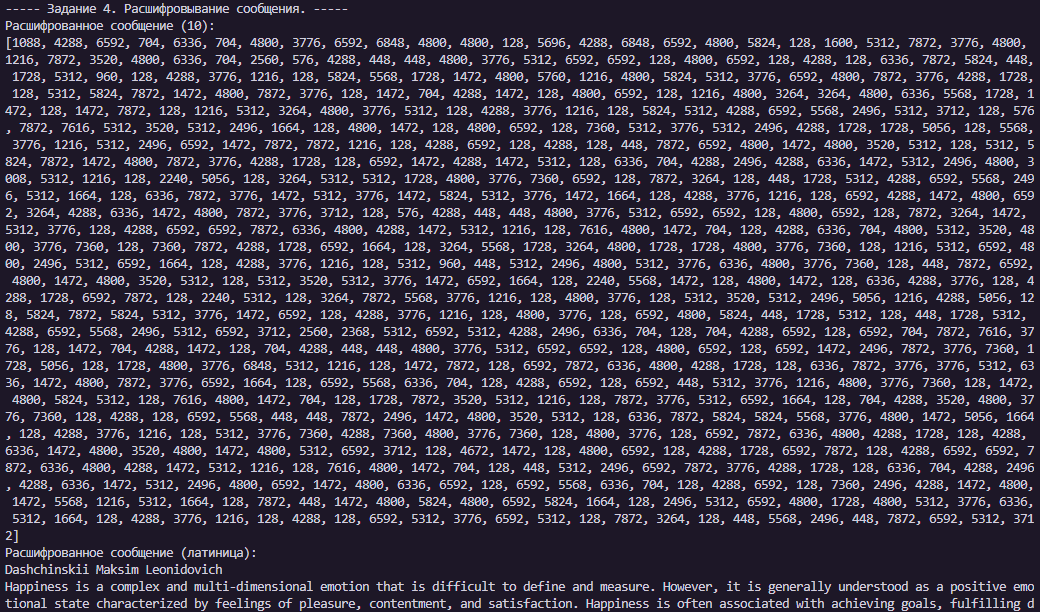


Рисунок 3.3 – Результат работы

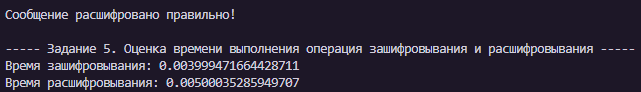


Рисунок 3.4 – Результат работы

**Вывод**

В ходе лабораторной работы было разработано приложение для генерации сверхвозрастающей последовательности, вычисление нормальной последовательности, зашифрование сообщения и его расшифровка.