Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 10**

«Исследование асимметричных шифров RSA и Эль-Гамаля»

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 RSA и Эль-Гамалем.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, которое реализовывает шифровние при помощи RSA и Эль-Гамаля. 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 BuildHistogram(list\_time, list\_value\_parameter):

    fig, ax = plt.subplots(1, 1)

    ax[0].bar(list\_time, list\_value\_parameter)

    plt.show()

def SearchForPrimeNumbersOnTheIntervalUsingRecursion(listNumber=None):

    if listNumber is None:

        listNumber = []

    if len(listNumber) == 2:

        listPrimeNumber = []

        traversableArray = np.arange(listNumber[0], listNumber[1] + 1, 1)

        if listNumber != [2, 3]:

            primeNumbersUpToUpperBound = \

                SearchForPrimeNumbersOnTheIntervalUsingRecursion(

                    [2, math.ceil(math.pow(listNumber[1], 0.5))])[0]

        else:

            primeNumbersUpToUpperBound = [2, 3]

        for i in range(len(traversableArray)):

            for j in range(len(primeNumbersUpToUpperBound)):

                if (traversableArray[i] % primeNumbersUpToUpperBound[j] == 0 and traversableArray[i] !=

                        primeNumbersUpToUpperBound[j]):

                    break

            else:

                listPrimeNumber.append(traversableArray[i])

    return listPrimeNumber, len(listPrimeNumber)

def InverseNumber(a, N):

    if N == 0:

        return a, 1, 0

    else:

        d, x, y = InverseNumber(N, a % N)

        return d, y, x - y \* (a // N)

def IsPrimitiveRoot(number, p):

    remains = set([(number \*\* i) % p for i in range(1, p)])

    return len(remains) == p - 1

def IsPrimitiveRoot(number, p):

    remains = set([(number \*\* i) % p for i in range(1, p)])

    return len(remains) == p – 1

def GenerateMainComponents(p):

    for i in range(2, p):

        if IsPrimitiveRoot(i, p):

            g = i

            break

    x = int(random.random() \* (p - 1))

    y = (g \*\* x) % p

    public\_key = p, g, y

    private\_key = p, g, x

    k = int(random.random() \* (p - 1))

    return public\_key, private\_key, k

def EncryptEG(message, public\_key, k):

    p, g, y = public\_key

    M = [ord(i) for i in message]

    encrypt\_message = []

    for i in M:

        encrypt\_message.append(((g \*\* k) % p, (y \*\* k \* i) % p))

    return encrypt\_message

def DecryptEG(encrypt\_message, private\_key, k):

    p, g, x = private\_key

    decrypt\_message = []

    for a\_b in encrypt\_message:

        decrypt\_message.append(chr((a\_b[1] \* (a\_b[0] \*\* (p - x - 1))) % p))

    return ''.join(decrypt\_message)

def ElGamal(message, p=None):

    if p is None or p > 200:

        p = 191

    message\_10 = [ord(i) for i in message]

    public\_key, private\_key, k = GenerateMainComponents(p)

    encrypt\_time = time.time()

    encrypt\_message = EncryptEG(message, public\_key, k)

    encrypt\_time = time.time() - encrypt\_time

    decrypt\_time = time.time()

    decrypt\_message = DecryptEG(encrypt\_message, private\_key, k)

    decrypt\_time = time.time() - decrypt\_time

    print(f'----- Входная информация -----\n'

          f'Сообщение: {message}\n'

          f'Сообщение(10): {message\_10}\n'

          f'p = {p}\n'

          f'----- Ключи -----\n'

          f'Открытый ключ: {public\_key}\n'

          f'Закрытый ключ: {private\_key}\n'

          f'----- Зашифровывание -----\n'

          f'Открытый ключ: {public\_key}\n'

          f'k = {k}\n'

          f'Зашифрованное сообщение: {encrypt\_message}\n'

          f'----- Расшифровывание -----\n'

          f'Закрытый ключ: {private\_key}\n'

          f'Расшифрованное сообщение: {decrypt\_message}\n'

          f'----- Успешность расшифровывания -----\n'

          f'Сообщение расшифровано {"правильно" if message == decrypt\_message else "неправильно"}\n'

          f'Время шифрования: {encrypt\_time}\n'

          f'Время расшифрования: {decrypt\_time}\n')

def RSA(message):

*# p = max(SearchForPrimeNumbersOnTheIntervalUsingRecursion([2, 25])[0])*

*# q = max(SearchForPrimeNumbersOnTheIntervalUsingRecursion([2, 6])[0])*

    p = 101

    q = 439

    n = p \* q

    e = 17

    function\_euler\_n = (p - 1) \* (q - 1)

    e\_inverse = InverseNumber(e, function\_euler\_n)[1]

    d = e\_inverse % function\_euler\_n

    public\_key = e, n

    secret\_key = d, n

    message\_10 = [ord(i) for i in message]

    encrypt\_message = []

    decrypt\_message = []

    encrypt\_time = time.time()

    for i in message\_10:

        encrypt\_message.append((i \*\* e) % n)

    encrypt\_time = time.time() - encrypt\_time

    decrypt\_time = time.time()

    for i in encrypt\_message:

        decrypt\_message.append((i \*\* d) % n)

    decrypt\_time = time.time() - decrypt\_time

    print(f'----- Основные компоненты -----\n'

          f'Число p: {p}\n'

          f'Число q: {q}\n'

          f'Число n: {n}\n'

          f'Число e: {e}\n'

          f'Число e^(-1): {e\_inverse}\n'

          f'Число d: {d}\n'

          f'Функция Эйлера для n: {function\_euler\_n}\n'

          f'----- Ключи -----\n'

          f'Публичный ключ: {public\_key}\n'

          f'Тайный ключ: {secret\_key}\n'

          f'----- Зашифровывание -----\n'

          f'Сообщение: {message\_10}\n'

          f'Зашифрованное сообщение: {encrypt\_message}\n'

          f'----- Расшифровывание -----\n'

          f'Расшифрованное сообщение: {decrypt\_message}\n'

          f'----- Успешность расшифровывания -----\n'

          f'Сообщение расшифровано {"правильно" if decrypt\_message == message\_10 else "неправильно"}\n'

          f'Время шифрования: {encrypt\_time}\n'

          f'Время расшифрования: {decrypt\_time}\n')

Листинг 2.1 –код программы, реализующие заданную ранее функциональность

**3. Результаты работы приложения**

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

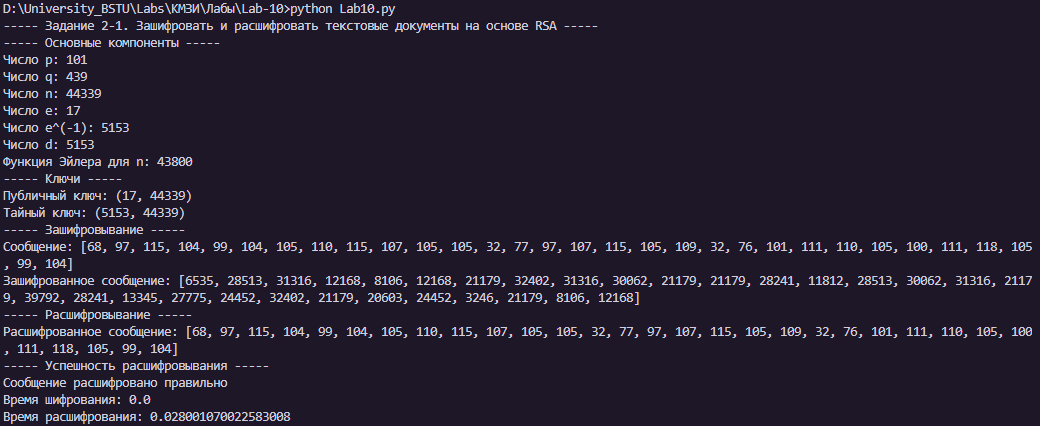


Рисунок 3.1 – Результат работы

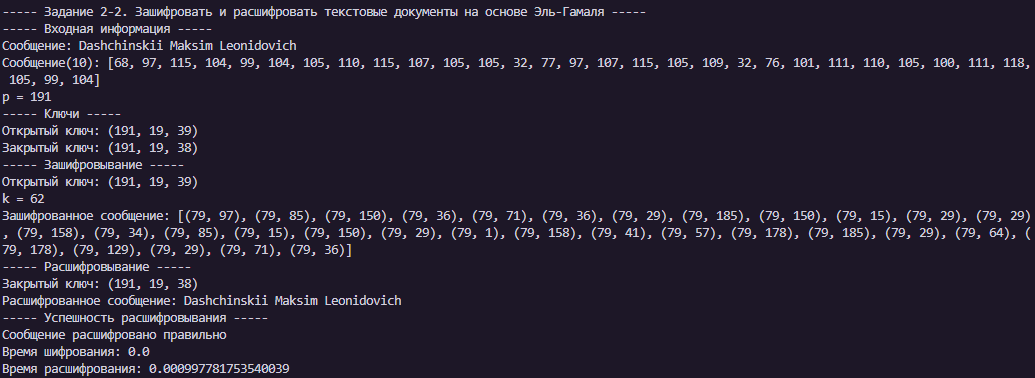


Рисунок 3.2 – Результат работы

**Вывод**

В ходе лабораторной работы было разработано приложение для шифрование при помощи RSA и Эль-Гамалем.