Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра экономической математики, информатики и статистики (ЭМИС)

РАЗРАБОТКА ПРОГРАММЫ ШИФРОВАНИЯ НА ОСНОВЕ МЕТОДА УМНОЖЕНИЯ МАТРИЦ

Отчет по практической работе по дисциплине «Защита информации»

Студент гр. 590-1

\_\_\_\_\_\_\_\_\_\_\_\_/Г.К. Петров

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Доктор технических наук

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_/ В.Г. Спицын

оценка подпись

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Томск 2023

**Цель работы:** изучение метода шифрования с помощью умножения матриц, а также его применение для шифрования и расшифровки фраз.

**Задание:**

1. Заполнить вектор, в котором нужно отобразить все буквы русского алфавита от а до я и от А до Я, а также символы: пробел, точка, двоеточие, восклицательный знак, вопросительный знак и запятая (всего 72 символа);

2. Зашифровать любую фразу, введенную с клавиатуры, методом произведения матриц;

3. Определить, какой должна быть матрица, чтобы зашифрованную фразу можно было расшифровать;

4. Расшифровать полученную в пункте 2 зашифрованную строку.

**Результат выполнения задания.**

Пример выполнения заданий 1, 2 и 4 представлен на рисунках 1-3 соответственно. Полный код на языке Python представлен в приложении А.

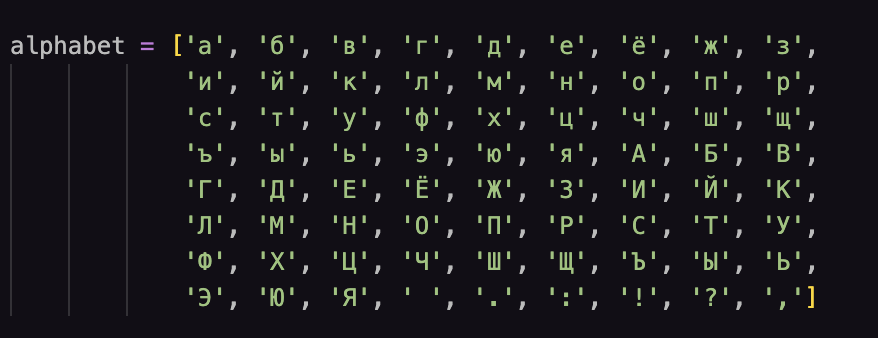


Рисунок 1 – Заполненная матрица с алфавитом

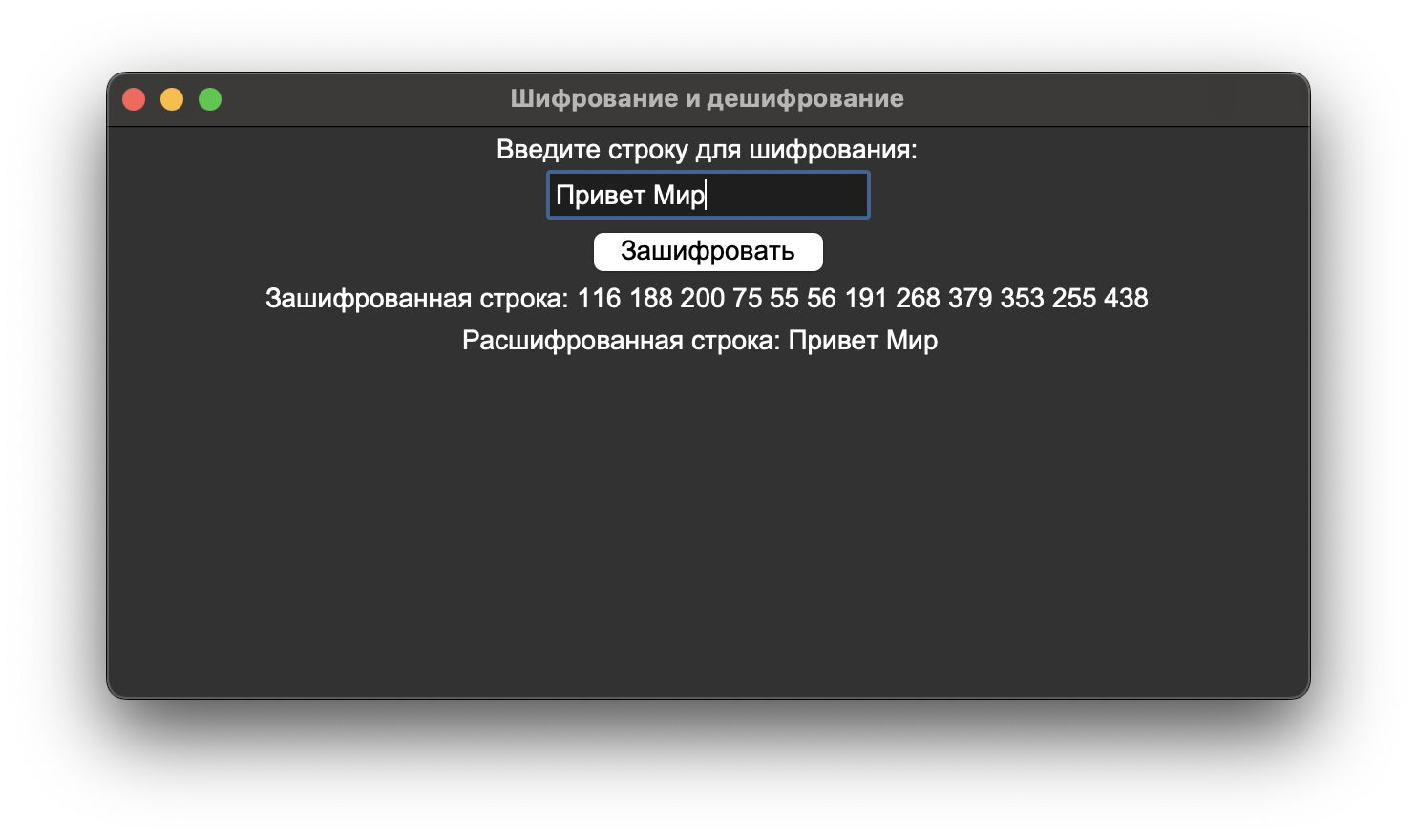


Рисунок 2 – Зашифрованное и расшифрованное сообщения

Для того, чтобы зашифрованную фразу можно было зашифровать, матрица ключа должна иметь столько же столбцов, сколько строк имеет матрица зашифрованной фразы.

**Вывод:** в процессе работы был изучен и применён метод шифрования с умножением матриц.

**Приложение А**

(обязательное)

Код на языке Python

import PySimpleGUI as psg

import numpy as np

def decrypt(message, key, alphabet):

    msg=[]

    word = ""

    inv = np.linalg.inv(key)

    message = np.dot(inv, message)

    rows = message.shape[0]

    cols = message.shape[1]

    for i in range(rows):

        for j in range(cols):

            msg.append(int(message[i][j]))

    print(msg)

    for num in range(len(msg)):

        for i in range(8):

            for j in range(9):

                if msg[num]==int(str(i)+str(j)):

                    word += alphabet[i][j]

    return(word)

def incrypt(message, start\_key, alphabet):

    #Определяем массив, который будет хранить наше сообщение

    msg=[]

    key=[]

    for ltr in range(len(message)):

        for i in range(8):

            for j in range(9):

                if message[ltr]==alphabet[i][j]:

                    msg.append(int(str(i)+str(j)))#Соединяем два элемента матрицы воедино и преобразуем в число

    for ltr in range(len(start\_key)):

        for i in range(8):

            for j in range(9):

                if start\_key[ltr]==alphabet[i][j]:

                    key.append(int(str(i)+str(j)))#Делаем то же самое для ключа

    #Сейчас мы имеем одномерный массив, но я преобразую его в двумерный

    #Выясним, на какое количество строк можно поделить наш массив

    max\_cols=1

    max\_rows=1

    for i in range(2,len(msg)):

        if (len(msg)%i == 0):

            max\_cols=int(len(msg)/i)

            max\_rows=int(len(msg)/max\_cols)

            break

    #Создадим нужную нам матрицу

    word = np.empty([max\_rows,max\_cols])

    ltr=0

    for i in range(max\_rows):

        for j in range(max\_cols):

            word[i][j]=int(msg[ltr])

            ltr+=1

    #Теперь мы имеем матрицу с наименьшим числом строк. Зашифруем её. Для этого создадим матрицу-ключ

    ltr=0

    key\_matrix=np.empty([max\_rows, max\_rows])

    for i in range(max\_rows):

        for j in range(max\_rows):

            key\_matrix[i][j]=key[ltr]

            if ltr<(len(key)-1): ltr+=1

            else: ltr=0

    word = np.dot(key\_matrix, word)

    return(word, key\_matrix)

#Искомый алфавит для шифрования

matrix=[

    ["а","б","в","г","д","е","ё","ж","з"],

    ["и","й","к","л","м","н","о","п","р"],

    ["с","т","у","ф","х","ц","ч","ш","щ"],

    ["ъ","ы","ь","э","ю","я","А","Б","В"],

    ["Г","Д","Е","Ё","Ж","З","И","Й","К"],

    ["Л","М","Н","О","П","Р","С","Т","У"],

    ["Ф","Х","Ц","Ч","Ш","Щ","Ъ","Ы","Ь"],

    ["Э","Ю","Я"," ",".",":","!","?",","]

    ]

#Для работы окна

layout = [

        [psg.Button('Зашифровать',enable\_events=True, key='-INCRYPT-')],

        [psg.Button('Расшифровать',enable\_events=True, key='-DECRYPT-')],

        [psg.Text('Слово:', font=(20)), psg.InputText(key='-TEXT-', size=(60,1), do\_not\_clear=False)],

        [psg.Text('Ключ матрицы:', font=(20)), psg.InputText(key='-KEY-', size=(60,1), do\_not\_clear=False)],

        [psg.Text("Результат: ", font=(20)), psg.Text(key="-RESULT-")]

          ]

window = psg.Window('Лабораторная 3',layout, size=(500,300))

while(True):

    #Для работы окна

    event, values = window.read()

    #Закрытие окна

    if event in (psg.WIN\_CLOSED, 'Exit'):

        break

    #Если мы хотим зашифровать сообщение

    if event == "-INCRYPT-":

        if str(values["-TEXT-"])<str(values["-KEY-"]):

            psg.popup(f"Введённый ключ не должен быть длиннее слова")

            #А потому что тогда матрица ключа будет больше матрицы слова, а так нельзя по правилам перемножения матриц

        else:

        #Создаём массив, состоящий из переведённого в числа сообщения, которое ввёл пользователь

            word, key\_matrix = incrypt(str(values["-TEXT-"]),

                        str(values["-KEY-"]),

                        matrix)

            window["-RESULT-"].update(word)

            print("Ключ\n", key\_matrix)

    #Если мы хотим расшифровать сообщение

    if event == "-DECRYPT-":

        word = decrypt(word, key\_matrix, matrix)

        window["-RESULT-"].update(word)

        print("Ключ\n", key\_matrix)