Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 4**

«ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПОДСТАНОВКИ (ЗАМЕНЫ) СИМВОЛОВ»

Выполнил:

Студент: Дащинский М.Л..

ФИТ 3 курс 4 группа

Преподаватель: Сазонова Д.В.

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования Python и позволяет провести шифрование Трисемусом и по формулам y ≡ x + k mod N и х ≡ у – k mod N при k = 7.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, позволяющее произвести шифрование Трисемусом и по формулам y ≡ x + k mod N и х ≡ у – k mod N при k = 7. На листинге 2.1 и 2.2 представлены функции, реализующие данную функциональность.

from matplotlib import pyplot as plt

import time

with open('text.txt', 'r', encoding='utf-8') as file:

    text = file.read().upper()

alphabet\_de = u"ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜß"

def Trisemus(key\_word, strvalue, action):

    keyword, (height, width) = key\_word

    doEncode = True if action == "encd" else False

    doDecode = True if action == "decd" else False

    start\_pos = 0

    table = [['.' for x in range(width)] for y in range(height)]

    temp\_char = {}

    for i in keyword + alphabet\_de:

        if temp\_char.get(i) is None:

            temp\_char[i] = start\_pos

            table[int(start\_pos / width)][int(start\_pos % width)] = i

            start\_pos += 1

            if start\_pos >= width \* height:

                break

    print(table)

    result = ""

    for i in strvalue:

        start\_pos = temp\_char.get(i)

        if start\_pos is not None:

            x = start\_pos % width

            if doEncode:

                y = (start\_pos // width + 1) % height

            elif doDecode:

                y = (start\_pos // width - 1 + height) % height

            else:

                y = start\_pos // width % height

            result += table[int(y)][int(x)]

        else:

            result += i

    return result

def BuildHistogram(alphabet, message, encrypted\_message):

    list\_alphabet = [i for i in alphabet]

    np\_probability\_original\_message = [round(message.count(i) / len(message), 4) for i in alphabet]

    np\_probability\_encrypted\_message = [round(encrypted\_message.count(i) / len(encrypted\_message), 4) for i in alphabet]

    fig, ax = plt.subplots(2, 1)

    ax[0].bar(list\_alphabet, np\_probability\_original\_message)

    ax[1].bar(list\_alphabet, np\_probability\_encrypted\_message)

    plt.show()

keyword = 'ENIGMA'

tablesize = (5, 6)

key = (keyword, tablesize)

print("Key = " + str(key))

start\_time = time.time()

encrypted\_text = Trisemus(key, text, "encd")

encryption\_time = round(time.time() - start\_time, 7)

with open('encrypted\_text.txt', 'w', encoding='utf-8') as f:

    f.write(encrypted\_text)

start\_time = time.time()

decrypted\_text = Trisemus(key, encrypted\_text, "decd")

decryption\_time = round(time.time() - start\_time, 7)

with open('decrypted\_text.txt', 'w', encoding='utf-8') as f:

    f.write(decrypted\_text)

print("Encryption time: " + str(encryption\_time) + " seconds")

print("Decryption time: " + str(decryption\_time) + " seconds")

BuildHistogram(alphabet\_de, text, encrypted\_text)

Листинг 2.1 – функция, реализующая шифрование Трисемуса

from matplotlib import pyplot as plt

import time

alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜ '

def encrypt(text):

    N = len(alphabet)  *# number of characters in the alphabet*

    k = 7              *# shift key*

    result = ''

    for char in text:

        if char.upper() in alphabet:

            x = alphabet.index(char.upper())

            y = (x + k) % N

            encrypted = alphabet[y]

        else:

            encrypted = char

        result += encrypted

    return result

def decrypt(text):

    N = len(alphabet)  *# number of characters in the alphabet*

    k = 7              *# shift key*

    result = ''

    for char in text:

        if char.upper() in alphabet:

            y = alphabet.index(char.upper())

            x = (y - k) % N

            decrypted = alphabet[x]

        else:

            decrypted = char

        result += decrypted

    return result

def BuildHistogram(alphabet, message, encrypted\_message):

    list\_alphabet = [i for i in alphabet]

    np\_probability\_original\_message = [round(message.count(i) / len(message), 4) for i in alphabet]

    np\_probability\_encrypted\_message = [round(encrypted\_message.count(i) / len(encrypted\_message), 4) for i in alphabet]

    fig, ax = plt.subplots(2, 1)

    ax[0].bar(list\_alphabet, np\_probability\_original\_message)

    ax[1].bar(list\_alphabet, np\_probability\_encrypted\_message)

    plt.show()

*# read the input file*

with open('text.txt', 'r', encoding='utf-8') as file:

    text = file.read()

start\_time = time.time()

*# encrypt the text*

encrypted\_text = encrypt(text)

encryption\_time = round(time.time() - start\_time, 7)

*# write the encrypted text to a file*

with open('encrypted.txt', 'w', encoding='utf-8') as file:

    file.write(encrypted\_text)

*# read the encrypted file*

with open('encrypted.txt', 'r', encoding='utf-8') as file:

    encrypted\_text = file.read()

start\_time = time.time()

*# decrypt the text*

decrypted\_text = decrypt(encrypted\_text)

decryption\_time = round(time.time() - start\_time, 7)

*# write the decrypted text to a file*

with open('decrypted.txt', 'w', encoding='utf-8') as file:

    file.write(decrypted\_text)

print("Encryption time: " + str(encryption\_time) + " seconds")

print("Decryption time: " + str(decryption\_time) + " seconds")

BuildHistogram(alphabet, text, encrypted\_text)

Листинг 2.2 – функция, реализующая шифрование по формулам y ≡ x + k mod N и х ≡ у – k mod N при k = 7

В листинге 2.1 функция Trisemus используется для шифрование текста при помощи таблицы Трисемуса немецкого алфавита. Через входные параметры будет шифровать или расшифровать текст из файла.

В листинге 2.2 функция encrypt используется для шифрование текста по формуле y ≡ x + k mod N при k = 7 и по немецкому алфавиту. Функция decrypt используется для расшифрование текста по формуле у – k mod N при k = 7 и по немецкому алфавиту.

**3. Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение и создает файлы с шифрованным текстом и расшифрованным текстом. Рисунок 3.1, 3.2, 3.3 и 3.4 показывают требуемые в данной лабораторной работе результаты, а рисунки 3.5 и 3.6 показывают гистограммы частот.

Время шифрования и расшифрования по Трисемусу:

Encryption time: 0.0050032 seconds

Decryption time: 0.0079992 seconds

Время шифрования и расшифрования по формуле:

Encryption time: 0.0050013 seconds

Decryption time: 0.0049951 seconds

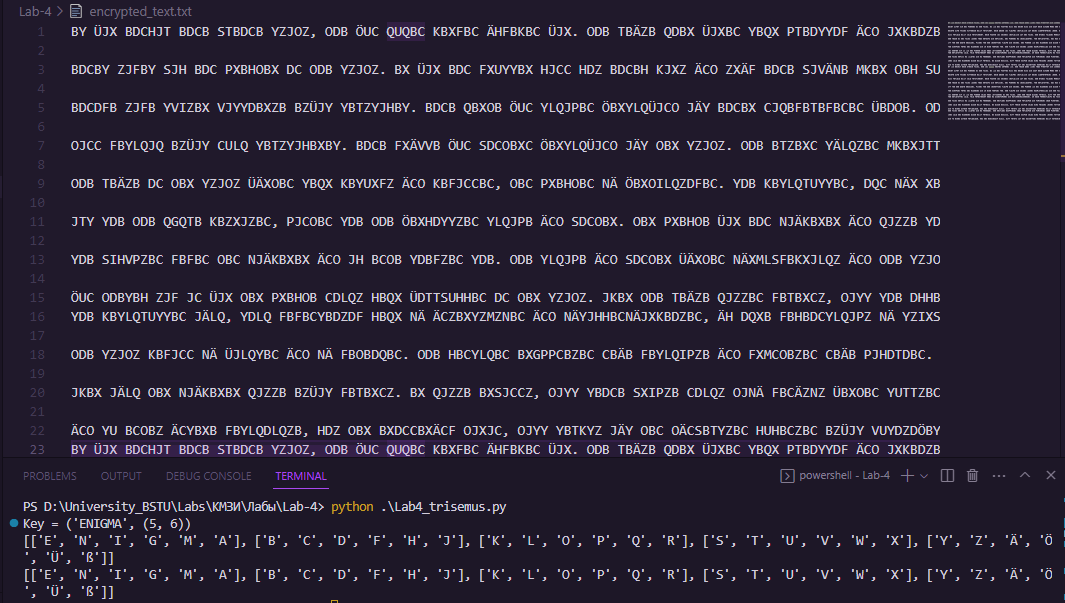


Рисунок 3.1 – Результат работы по Трисемусу, шифрованный текст

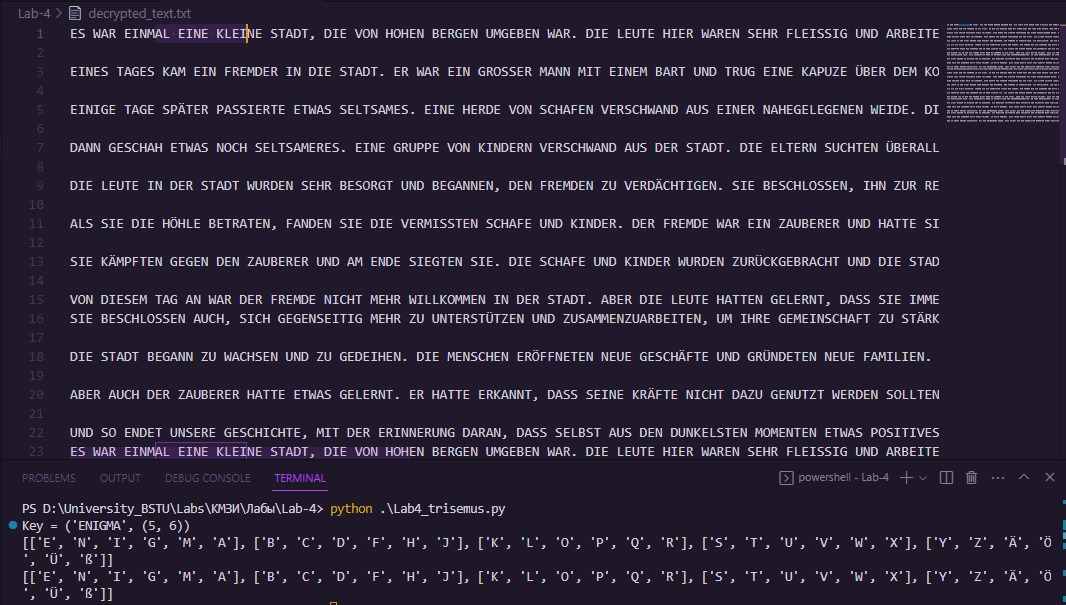


Рисунок 3.2 – Результат работы по Трисемусу, расшифрованный текст

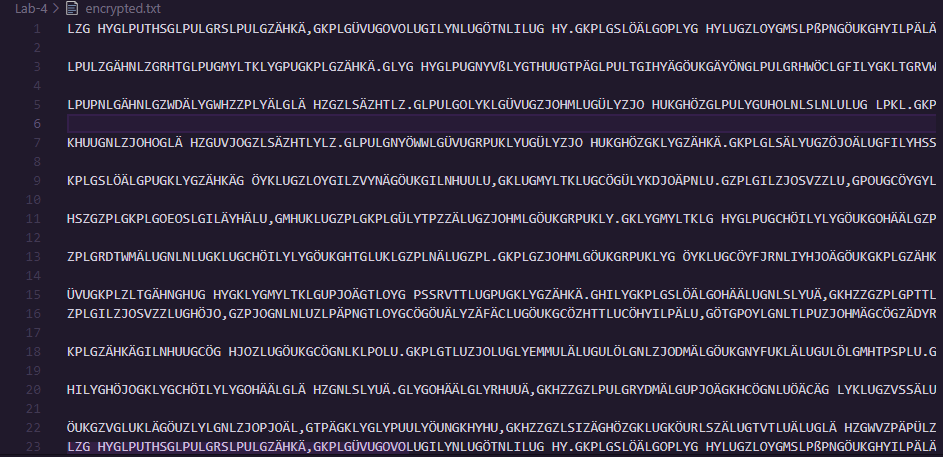


Рисунок 3.3 – Результат работы по формуле, шифрованный текст

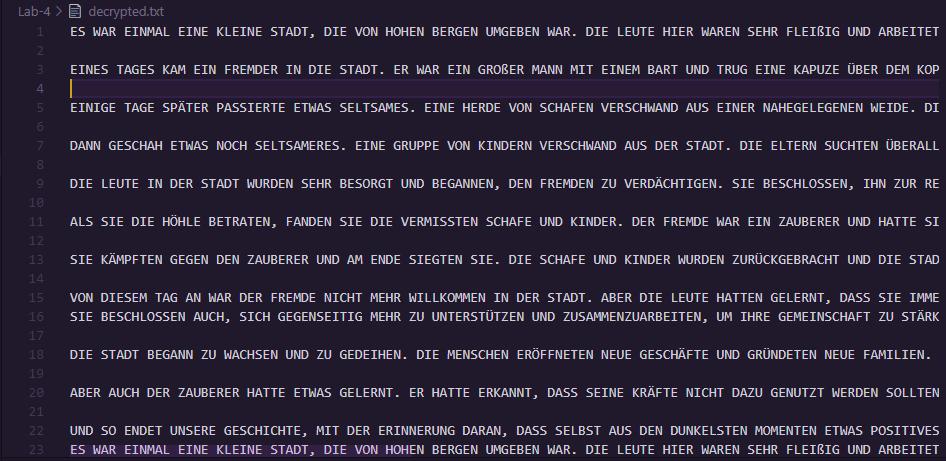


Рисунок 3.4 – Результат работы по формуле, расшифрованный текст

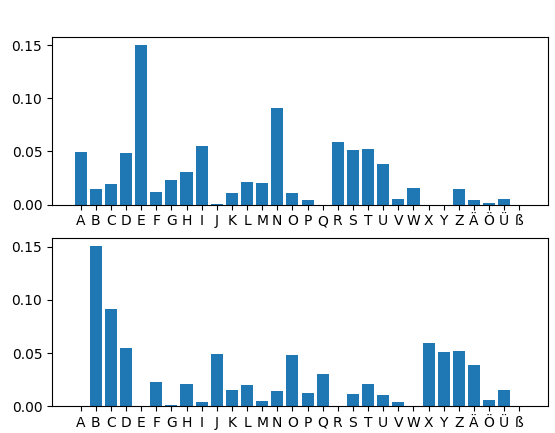


Рисунок 3.5 – Результат работы по Трисемусу, гистограмма частот

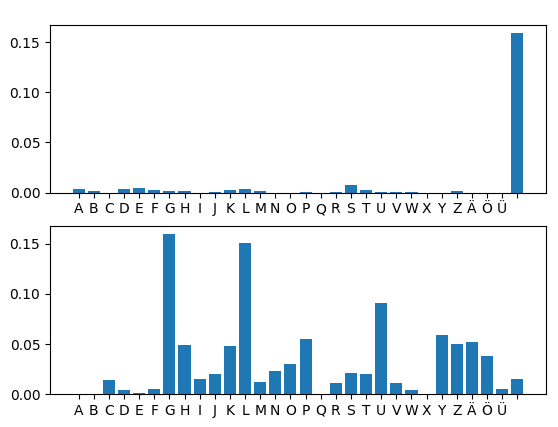


Рисунок 3.6 – Результат работы по формуле, гистограмма частот

**Вывод**

В ходе изучения теоретических материалов лабораторной работы и выполнения её практической части были изучены и приобретены практические навыки разработки и использования приложений для реализации подстановочных шифров. На практике полученные знания были закреплены через создание собственного приложения, реализующего функционал шифрование Трисемусом и через формулу.