Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 5**

«ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПЕРЕСТАНОВКИ 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 маршрутной перестановкой (маршрут - змейкой) и множественной перестановкой.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, позволяющее произвести маршрутной перестановкой (маршрут - змейкой) и множественной перестановкой. На листинге 2.1 и 2.2 представлены функции, реализующие данную функциональность.

import math

import time

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

    text = f.read().lower()

def enc(text):

    if math.sqrt(len(text)) <= 5:

        n = 5

    else:

        n = math.ceil(math.sqrt(len(text)))

    a = ['\_'] \* n

    for i in range(n):

        a[i] = ['\_'] \* n

    text = text.replace(' ', '\_')

    t = 0

    for i in range(0, n):

        for j in range(n \* (i % 2) - i % 2,  n \* ((i + 1) % 2)  - i % 2, 1 - (i % 2) \* 2):

            if t < len(text):

                a[j][i] = text[t]

                t += 1

    enctext = ''

    for i in range(n):

        for j in range(n):

            enctext += a[i][j]

    return enctext

def dec(text):

    if math.sqrt(len(text)) <= 5:

        n = 5

    else:

        n = math.ceil(math.sqrt(len(text)))

    a = ['\_'] \* n

    for i in range(n):

        a[i] = ['\_'] \* n

    t = 0

    for i in range(n):

        for j in range(n):

            if t < len(text):

                a[i][j] = text[t]

                t += 1

    dectext = ''

    for i in range(0, n):

        for j in range(n \* (i % 2) - i % 2,  n \* ((i + 1) % 2)  - i % 2, 1 - (i % 2) \* 2):

            dectext += a[j][i]

    dectext = dectext.replace('\_', ' ').strip()

    return dectext

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()

start\_time = time.time()

print("Cipher text: " + enc(text))

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

print("-----------------------------")

start\_time = time.time()

print("Deciphered text: " + dec(enc(text)))

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

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

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

BuildHistogram(alphabet\_de, plaintext, ciphertext)

Листинг 2.1 –код программы, реализующая шифрование маршрутной перестановкой (маршрут змейкой)

import numpy as np

import time

*# Define the keys as name and surname*

name = "Maksim"

surname = "Dashchinskii"

*# Define the encryption function using a stream cipher*

def encrypt(plaintext, key\_matrix):

    ciphertext = ""

    prev\_char = "A"

    for i in range(len(plaintext)):

        j = i % key\_matrix.shape[1]

        k = (i // key\_matrix.shape[1]) % key\_matrix.shape[0]

        row = (ord(prev\_char) + key\_matrix[k][j]) % 26

        col = j

        curr\_char = chr((ord(plaintext[i]) - ord("A") + key\_matrix[k][j]) % 26 + ord("A"))

        ciphertext += curr\_char

        prev\_char = curr\_char

    return ciphertext

*# Define the decryption function using the same key matrix as the encryption function*

def decrypt(ciphertext, key\_matrix):

    plaintext = ""

    prev\_char = "A"

    for i in range(len(ciphertext)):

        j = i % key\_matrix.shape[1]

        k = (i // key\_matrix.shape[1]) % key\_matrix.shape[0]

        row = (ord(prev\_char) + key\_matrix[k][j]) % 26

        col = j

        curr\_char = chr((ord(ciphertext[i]) - ord("A") - key\_matrix[k][j]) % 26 + ord("A"))

        plaintext += curr\_char

        prev\_char = ciphertext[i]

    return plaintext

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 plaintext from a file*

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

    plaintext = f.read().upper()

*# Remove any whitespace and special characters from the plaintext*

plaintext = "".join(c for c in plaintext if c.isalpha())

*# Generate the key matrix using the keys*

num\_rows = len(surname)

num\_cols = len(name)

key\_matrix = np.zeros((num\_rows, num\_cols), dtype=int)

for i in range(num\_rows):

    for j in range(num\_cols):

        key\_matrix[i][j] = ord(surname[i]) + ord(name[j])

start\_time = time.time()

*# Encrypt the plaintext using the stream cipher*

ciphertext = encrypt(plaintext, key\_matrix)

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

start\_time = time.time()

*# Decrypt the ciphertext using the same key matrix as the encryption function*

decrypted\_plaintext = decrypt(ciphertext, key\_matrix)

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

*# Print the original plaintext, encrypted ciphertext, and decrypted plaintext*

print("Original plaintext: " + plaintext)

print("Encrypted ciphertext: " + ciphertext)

print("Decrypted plaintext: " + decrypted\_plaintext)

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

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

BuildHistogram(alphabet\_de, plaintext, ciphertext)

Листинг 2.2 – код программы, реализующая шифрование множественной перестановкой

В листинге 2.1 функция enc используется для шифрование текста при помощи маршрутной перестановкой (маршрут змейкой). Через входные параметры передается текст. Функция dec используется для расшифрования текста при помощи маршрутной перестановкой (маршрут змейкой). Через входные параметры передается текст.

В листинге 2.2 функция encrypt используется для шифрование текста при помощи множественной перестановки. Через входные параметры передается текст и матрица. Функция decrypt используется для расшифрование текста при помощи множественной перестановки. Через входные параметры передается текст и матрица.

**3. Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение и создает файлы с шифрованным текстом и расшифрованным текстом. Рисунок 3.1 и 3.3 показывают требуемые в данной лабораторной работе результаты, а рисунки 3.2 и 3.4 показывают гистограммы частот появления символов для исходного и зашифрованного текста, которые строются в программе.

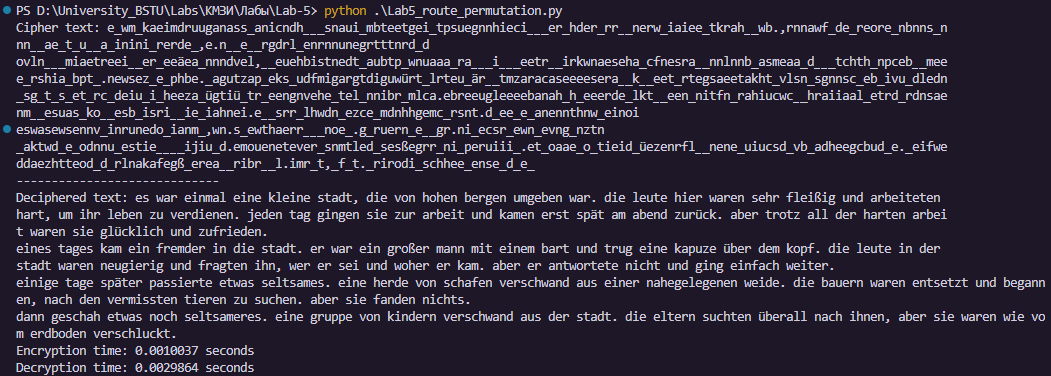


Рисунок 3.1 – Результат работы по маршрутной перестановкой (маршрут змейкой), шифрованный и расшифрованный текст

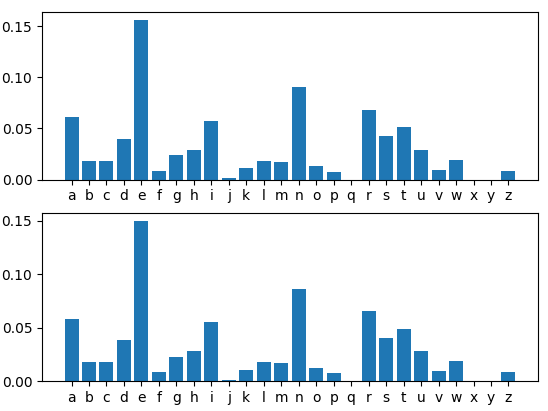


Рисунок 3.2 – Гистограмма частот по маршрутной перестановкой (маршрут змейкой), оригинальный и шифрованный текст

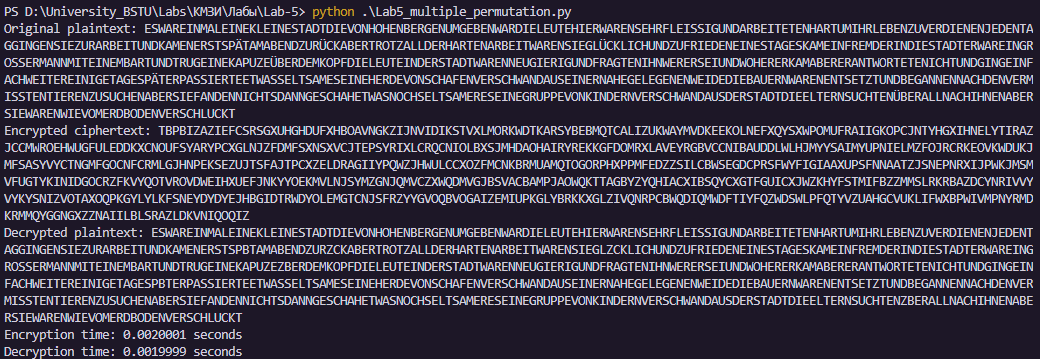


Рисунок 3.3 – Результат работы по множественной перестановки, шифрованный и расшифрованный текст

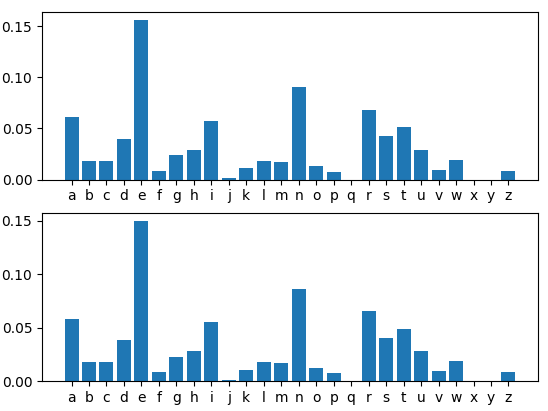


Рисунок 3.4 – Гистограмма частот по множественной перестановки (маршрут змейкой), оригинальный и шифрованный текст

**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.