Лабораторная работа №2 «Традиционные методы шифрования»

Задание 1.

Реализация программы шифрования Плейфейра с паролем (вариант 6):

Ключ: «**Повадится овца не хуже козы.**»

Код программы:

import re

#создание таблицы по ключу

def generate\_playfair\_table(key):

    alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЫЬЭЮЯ"

    key = "".join(dict.fromkeys(key.upper().replace("Ё", "Е")))

    table = key + "".join(c for c in alphabet if c not in key)

    return [list(table[i:i+5]) for i in range(0, 25, 5)]

#поиск позиции

def find\_position(table, letter):

    for row in range(5):

        for col in range(5):

            if table[row][col] == letter:

                return row, col

    return None

#шифр по варианту 6

def playfair\_encrypt(text, table):

    text = re.sub(r'[^А-ЯЁ]', '', text.upper().replace("Ё", "Е"))

    text\_pairs = []

    i = 0

    while i < len(text):

        a = text[i]

        b = text[i+1] if i+1 < len(text) and text[i] != text[i+1] else 'Х'

        if find\_position(table, a) is None or find\_position(table, b) is None:

            i += 1

            continue

        text\_pairs.append((a, b))

        i += 2 if text[i] != b else 1

    encrypted\_text = ""

    for a, b in text\_pairs:

        row1, col1 = find\_position(table, a)

        row2, col2 = find\_position(table, b)

        if row1 is None or row2 is None:

            continue

        if row1 == row2:

            encrypted\_text += table[row1][(col1+1)%5] + table[row2][(col2+1)%5]

        elif col1 == col2:

            encrypted\_text += table[(row1+1)%5][col1] + table[(row2+1)%5][col2]

        else:

            encrypted\_text += table[row1][col2] + table[row2][col1]

    return encrypted\_text

#главная функция для вызова алгоритма

def main():

    key = input("Введите ключ: ")

    file\_path = input("Введите путь к файлу (.txt): ")

    try:

        with open(file\_path, 'r', encoding='utf-8') as file:

            text = file.read()

    except FileNotFoundError:

        print("Файл не найден")

        return

    table = generate\_playfair\_table(key)

    encrypted\_text = playfair\_encrypt(text, table)

    print("Зашифрованный текст:", encrypted\_text)

    output\_file = file\_path.replace(".txt", "\_encrypted.txt")

    with open(output\_file, 'w', encoding='utf-8') as file:

        file.write(encrypted\_text)

    print(f"Результат сохранен в {output\_file}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результат тестирования загружается в файл.

Задание 2.

Реализация программы шифрования тремя способами:

Код программы:

import re

import random

import matplotlib.pyplot as plt

# шифр сдвигом

def shift\_cipher(text, shift, alphabet):

    encrypted\_text = ""

    for char in text:

        if char in alphabet:

            new\_index = (alphabet.index(char) + shift) % len(alphabet)

            encrypted\_text += alphabet[new\_index]

        else:

            encrypted\_text += char

    return encrypted\_text

# шифр виженером

def vigenere\_cipher(text, key, alphabet):

    key = (key \* (len(text) // len(key) + 1))[:len(text)]

    encrypted\_text = ""

    for char, key\_char in zip(text, key):

        if char in alphabet:

            shift = alphabet.index(key\_char)

            new\_index = (alphabet.index(char) + shift) % len(alphabet)

            encrypted\_text += alphabet[new\_index]

        else:

            encrypted\_text += char

    return encrypted\_text

# шифр lfsr

def lfsr\_cipher(text, seed, polynomial, alphabet):

    state = seed[:]

    encrypted\_text = ""

    for char in text:

        if char in alphabet:

            bit = state[0]

            new\_index = (alphabet.index(char) + bit) % len(alphabet)

            encrypted\_text += alphabet[new\_index]

            feedback = sum(state[i] for i in polynomial) % 2

            state = state[1:] + [feedback]

        else:

            encrypted\_text += char

    return encrypted\_text

# функция для построения гистограммы

def plot\_histogram(text, alphabet):

    frequencies = {char: text.count(char) for char in alphabet}

    plt.bar(frequencies.keys(), frequencies.values())

    plt.xlabel("Символы")

    plt.ylabel("Частота")

    plt.title("Гистограмма частотности символов")

    plt.show()

# главная функция для вызова функций

def main():

    alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЫЬЭЮЯ"

    file\_path = input("Введите путь к файлу (.txt): ")

    method = input("Выберите метод (1 - сдвиг, 2 - Виженер, 3 - LFSR): ")

    try:

        with open(file\_path, 'r', encoding='utf-8') as file:

            text = re.sub(r'[^А-Я]', '', file.read().upper().replace("Ё", "Е"))

    except FileNotFoundError:

        print("Файл не найден")

        return

    if method == "1":

        shift = int(input("Введите константу (номер студента): ")) % len(alphabet)

        encrypted\_text = shift\_cipher(text, shift, alphabet)

    elif method == "2":

        key = input("Введите ключевое слово (поговорку из таблицы 4): ").upper().replace("Ё", "Е")

        key = re.sub(r'[^А-Я]', '', key)

        encrypted\_text = vigenere\_cipher(text, key, alphabet)

    elif method == "3":

        seed = [random.randint(0, 1) for \_ in range(5)]

        polynomial = [0, 2]

        encrypted\_text = lfsr\_cipher(text, seed, polynomial, alphabet)

        plot\_histogram(encrypted\_text, alphabet)  # вызов гистограммы для LFSR

    else:

        print("Некорректный выбор метода")

        return

    output\_file = file\_path.replace(".txt", "\_encrypted.txt")

    with open(output\_file, 'w', encoding='utf-8') as file:

        file.write(encrypted\_text)

    print(f"Зашифрованный текст сохранен в {output\_file}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результат тестирования загружается в файл.