МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №1.1**

по дисциплине

«Информационная безопасность»

Криптография. Основы шифрования данных.

Вариант №6

**Студент:**

Воробьев К.О.

            Группа P34302

**Преподаватель:**

Фамилия И.О.

A black and white logo

Description automatically generated

Санкт-Петербург, 2023

**Цель работы**

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

**Вариант задания №6:**

Реализовать в программе шифрование и дешифрацию файла методом биграмм с двойным квадратом. Квадраты генерировать динамически для каждого шифрования.

**Листинг разработанной программы**

import random  
from enum import Enum  
  
ALPHABET\_EN = "ABCDEFGHIJKLMNOPQRSTUVWXYZ " # Алфавит для составления матрицы на латинице  
ALPHABET\_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ " # Алфавит для составления матрицы на кириллице  
  
  
# Класс-перечисление, созданный для удобства (используется для задания-определения языка)  
class Language(Enum):  
 NULL = 0  
 EN = 1  
 RU = 2  
  
  
# Функция которая читает файл и определяет на каком языке находится текст внутри  
def read\_file\_and\_detect\_language(filename):  
 with open(filename, 'r', encoding='utf-8') as file:  
 text = file.read()  
 if text[0].upper() in ALPHABET\_EN:  
 language = Language.EN  
 else:  
 if text[0].upper() in ALPHABET\_RU:  
 language = Language.RU  
 else:  
 language = Language.NULL  
 return text, language  
  
  
# Функция создания матрицы Кириллицы (размером 7x5)  
def generate\_ru\_matrix():  
 alphabet\_list = list(ALPHABET\_RU)  
 random.shuffle(alphabet\_list)  
 shuffled\_alphabet = ''.join(alphabet\_list)  
 matrix = []  
 i = 0  
 while len(matrix) < 7:  
 row = []  
 while len(row) < 5:  
 row.append(shuffled\_alphabet[i])  
 i += 1  
 matrix.append(row)  
 return matrix  
  
  
# Функция создания матрицы Латиницы (размером 7x4)  
def generate\_en\_matrix():  
 alphabet\_list = list(ALPHABET\_EN)  
 random.shuffle(alphabet\_list)  
 shuffled\_alphabet = ''.join(alphabet\_list)  
 matrix = []  
 i = 0  
 while len(matrix) < 7:  
 row = []  
 while len(row) < 4:  
 row.append(shuffled\_alphabet[i])  
 i += 1  
 matrix.append(row)  
 return matrix  
  
  
# Функция шифрования  
def encrypt(text, matrix1, matrix2):  
 if len(text) % 2 == 1:  
 text = text + " " # Если количество символов нечетное - добавляется пробел в конце  
 i = 0  
 encrypted\_text = ""  
 for row in matrix1: # Вывод первой сгенерированной матрицы в консоль  
 print(row)  
 print()  
 for row2 in matrix2: # Вывод второй сгенерированной матрицы в консоль  
 print(row2)  
 while i < len(text):  
 letter1 = text[i].upper()  
 letter2 = text[i + 1].upper() # В каждом цикле берется пара символов  
 for j in range(len(matrix1)):  
 for k in range(len(matrix1[j])): # Для первого символа находится совпадение в первой матрице  
 if matrix1[j][k] == letter1:  
 letter1\_xy = [j, k]  
 break  
 for m in range(len(matrix2)):  
 for n in range(len(matrix2[m])): # Для второго символа находится совпадение во второй матрице  
 if matrix2[m][n] == letter2:  
 letter2\_xy = [m, n]  
 break  
 # К результату шифрования добавляются два других символа, образующих с выбранными прямоугольник  
 encrypted\_text += matrix2[letter1\_xy[0]][letter2\_xy[1]] + matrix1[letter2\_xy[0]][letter1\_xy[1]]  
 i = i + 2  
 return encrypted\_text  
  
  
# Функция дешифрации  
def decrypt(text, matrix1, matrix2):  
 decrypted\_text = ""  
 i = 0  
 while i < len(text):  
 letter1 = text[i]  
 letter2 = text[i + 1] # В каждом цикле берется пара символов  
 for j in range(len(matrix2)):  
 for k in range(len(matrix2[j])): # Для первого символа находится совпадение в первой матрице  
 if matrix2[j][k] == letter1:  
 letter1\_xy = [j, k]  
 break  
 for m in range(len(matrix1)):  
 for n in range(len(matrix1[m])): # Для второго символа находится совпадение во второй матрице  
 if matrix1[m][n] == letter2:  
 letter2\_xy = [m, n]  
 break  
 # К результату дещифрации добавляются два других символа, образующих с выбранными прямоугольник  
 decrypted\_text += matrix1[letter1\_xy[0]][letter2\_xy[1]] + matrix2[letter2\_xy[0]][letter1\_xy[1]]  
 i = i + 2  
 return decrypted\_text  
  
  
# Функция сохранения текста в файл  
def save\_text\_to\_file(filename, encrypted\_file):  
 with open(filename, 'w', encoding='utf-8') as output\_file:  
 output\_file.write(encrypted\_file)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Шифрование  
 res = read\_file\_and\_detect\_language('res/input.txt')  
 INPUT\_TEXT = res[0]  
 LANG = res[1]  
 if LANG == Language.RU:  
 matrix1 = generate\_ru\_matrix()  
 matrix2 = generate\_ru\_matrix()  
 if LANG == Language.EN:  
 matrix1 = generate\_en\_matrix()  
 matrix2 = generate\_en\_matrix()  
 encrypted = encrypt(INPUT\_TEXT, matrix1, matrix2)  
 save\_text\_to\_file('res/encrypted.txt', encrypted)  
  
 # Дешифрация  
 res = read\_file\_and\_detect\_language('res/encrypted.txt')  
 ENCRYPTED\_INPUT = res[0]  
 LANG = res[1]  
 decrypted = decrypt(ENCRYPTED\_INPUT, matrix1, matrix2, )  
 save\_text\_to\_file('res/decrypted.txt', decrypted)

**Результаты выполнения**

A screenshot of a computer

Description automatically generated

*input.txt*

Hello

*encrypted.txt*

ITYICM

*decrypted.txt*

HELLO

*matrix 1*

*['F', 'P', 'Y', 'L']*

*[' ', 'X', 'T', 'B']*

*['C', 'Q', 'H', 'O']*

*['S', 'D', 'E', 'I']*

*['W', 'V', ' ', 'M']*

*['J', 'R', 'G', 'U']*

*['Z', 'K', 'A', 'N']*

*matrix 2*

**['Y', 'S', 'O', 'N']**

**['J', 'E', 'U', 'R']**

**['M', 'I', 'C', 'T']**

**['L', 'A', ' ', 'D']**

**['G', 'K', ' ', 'Z']**

**['Q', 'X', 'F', 'H']**

**['V', 'B', 'P', 'W']**

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с таким методом шифрования, как метод биграмм с двойным квадратом. Также, мне удалось реализовать программу шифрования и дешифрации файла с использованием этого метода.