ЛАБОРАТОРНАЯ РАБОТА №1

“Основы шифрования данных”

по дисциплине

‘Информационная безопасность’

Вариант 5

Выполнил:
Соболев Иван
Александрович
Группа: P34312

Преподаватель:
Маркина Татьяна
Анатольевна

Санкт-Петербург, 2024

Цель работы

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

Программные и аппаратные средства

Для выполнения лабораторной работы был использован компьютер со следующими характеристиками:

- Процессор: Apple M2
- Видеокарта: Apple M2
- Объем оперативной памяти: 8GB
- Использована операционная система: macOS 14.4.1
- Версия Python: 3.12

Задание

Реализовать в программе шифрование и дешифрацию файла с использованием квадрата Кардано размером 4x4.

Листинг разработанной программы

main.py

```
from encrypt import encrypt, decrypt
from io_utils import read_from_file, read_key_from_console, decide_action, write_to_file,
DECRYPT_ACTION, ENCRYPT_ACTION

if __name__ == '__main__':
    action = decide_action()
    if action == ENCRYPT_ACTION:
        s = read_from_file(action)
        if s is not None:
            encrypted, key = encrypt(s)
            write_to_file('encrypted.txt', encrypted)
            print(f'Результат: "{encrypted}"')
            print(f'Ключ: "{key}"')
    elif action == DECRYPT_ACTION:
        s = read_from_file(action)
        if s is not None:
            k = read_key_from_console()
            if k is not None:
                decrypted = decrypt(s, k)
                print(f'Результат: "{decrypted}"')
```

io_utils.py

```
import os
from typing import Optional, Any

ENCRYPT_ACTION = "enc"
DECRYPT_ACTION = "dec"
```

```
"""Функция для вывода ошибок"""
def print_red(message: str) -> None:
  print(f"\033[91m{message}\033[0m")

"""Функция для вывода ключа"""
def print_green(message: str) -> None:
  print(f"\033[92m{message}\033[0m")
```


```
"""Функция для вывода ошибок"""
def print_red(message: str) -> None:
  print(f"\033[91m{message}\033[0m")

"""Функция для вывода ключа"""
def print_green(message: str) -> None:
  print(f"\033[92m{message}\033[0m")
```


```
"""Функция записи в файл"""
def write_to_file(filename, content):
  try:
    with open(filename, 'w', encoding='utf-8') as file:
      file.write(content)
    print_green(f"Данные успешно записаны в файл: {filename}")
  except IOError as e:
    print_red(f"Ошибка записи в файл {filename}: {e}")
  except Exception as e:
    print_red(f"Произошла ошибка: {e}")

"""Функция чтения из файла"""
def read_from_file(action) -> Optional[str]:
  try:
    if action == 'enc':
      filename = input("Введите названия файла с текстом для шифрования: ")
    else:
      filename = input("Введите названия файла с текстом для дешифрования: ")

    if not filename.strip():
      print_red("Ошибка: имя файла не должно быть пустым.")
      return None

    if not os.path.isfile(filename):
      print_red(f"Ошибка: файл '{filename}' не найден.")
      return None
    with open(filename, "r") as f:
      buf = f.read().split('\n')[0]
      if not buf.strip():
        print_red("Ошибка: файл пуст.")
        return None
      return buf
  except Exception as e:
    print_red(f'Ошибка чтения из файла: {str(e)}')
    return None
```

```
"""Функция чтения ключа из консоли"""
def read_key_from_console() -> Optional[str]:
    k = input("Введите ключ шифрования: ")
    if not k.strip():
        print_red("Ошибка: ключ не должен быть пустым.")
        return None
    return k

"""Функция выбора действия"""
def decide_action() -> Any | None:
    print('Выберите действие: \n'
          '1. Зашифровать текст с помощью ключа\n'
          '2. Расшифровать текст с помощью ключа')
    try:
        option = int(input("Введите номер: "))
        if option not in [1, 2]:
            print_red(f'Ошибка: опция {option} недоступна!')
            return None
        if option == 1:
            return ENCRYPT_ACTION
        elif option == 2:
            return DECRYPT_ACTION
        else:
            return None
    except Exception:
        print_red(f'Ошибка: неверный ввод. Необходимо ввести целое число.')
        return None
```

transformer.py

```
from typing import Tuple

"""Функция поворота квадрата на 90 градусов"""
def rotate_90(row: int, col: int, n: int) -> Tuple[int, int]:
    return col, n - 1 - row

"""Функция поворота квадрата на 180 градусов"""
def rotate_180(row: int, col: int, n: int) -> Tuple[int, int]:
    return n - 1 - row, n - 1 - col

"""Функция поворота квадрата на 270 градусов"""
def rotate_270(row: int, col: int, n: int) -> Tuple[int, int]:
    return n - 1 - col, row

"""Функция поворота ключа на 90 градусов"""
def rotate_key_90(key: list[int], side_len: int) -> list[int]:
    return [get_position(*rotate_90(*get_coordinates(i, side_len), side_len), side_len) for i in key]

"""Функция поворота ключа на 180 градусов"""
def rotate_key_180(key: list[int], side_len: int) -> list[int]:
    return [get_position(*rotate_180(*get_coordinates(i, side_len), side_len), side_len) for i in key]
```

```
"""Функция поворота ключа на 270 градусов"""
def rotate_key_270(key: list[int], side_len: int) -> list[int]:
    return [get_position(*rotate_270(*get_coordinates(i, side_len), side_len), side_len) for i in key]

"""Функция вычисления позиции ячейки в квадрате по ее координатам"""
def get_position(row: int, col: int, n: int) -> int:
    return row * n + col

"""Функция вычисления координат ячейки в квадрате по ее позиции"""
def get_coordinates(pos: int, n: int) -> Tuple[int, int]:
    return pos // n, pos % n

"""Функция вычисления позиций поворотов ячейки"""
def find_opposites(side_len: int, window: int) -> list[int]:
    opposites = [window]
    row, col = get_coordinates(window, side_len)
    w1 = get_position(*rotate_90(row, col, side_len), side_len)
    w2 = get_position(*rotate_180(row, col, side_len), side_len)
    w3 = get_position(*rotate_270(row, col, side_len), side_len)
    # Если повороты одинаковые, нет смысла их добавлять (выбранное окно - центр квадрата,
размер квадрата нечетный).
    if window != w1:
        opposites += [w1, w2, w3]
    return opposites
```

encrypt.py

```
import random
from typing import Tuple
from transformer import find_opposites, rotate_key_90, rotate_key_180, rotate_key_270


def encrypt_message(text: str) -> Tuple[str, str]:
    square_len = nearest_square_greater_than(len(text))
    padded_text = text.ljust(square_len ** 2)
    square = [i for i in range(square_len ** 2)]
    key = []
    # Генерация квадрата Кардано и ключа
    while len(square) != 0:
        empty_cell = random.choice(square)
        key.append(empty_cell)
        windows = find_opposites(square_len, empty_cell)
        for w in windows:
            square.remove(w)

    encrypted = []
    for si in key:
        encrypted.append(padded_text[si])
    for si in rotate_key_90(key, square_len):
        encrypted.append(padded_text[si])
    for si in rotate_key_180(key, square_len):
        encrypted.append(padded_text[si])
```

```
    for si in rotate_key_270(key, square_len):
        encrypted.append(padded_text[si])
    return ''.join(encrypted), '-'.join(map(str, key))


def decrypt_message(encrypted: str, key: str) -> str:
    size = int(len(encrypted) ** 0.5)
    key = list(map(int, key.split('-')))
    decrypted = [' '] * (size ** 2)

    quarter = len(encrypted) // 4
    rotations = [
        key,
        rotate_key_90(key, size),
        rotate_key_180(key, size),
        rotate_key_270(key, size)
    ]

    for i, rotation in enumerate(rotations):
        for j, pos in enumerate(rotation):
            decrypted[pos] = encrypted[i * quarter + j]

    return ''.join(decrypted)


def nearest_square_greater_than(size: int) -> int:
    n = 1
    while n ** 2 < size:
        n += 1
    return n
```

Результаты работы программы

Исходный текст:
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покривившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.

Шифрование текста:

```
Выберите действие:
1. Зашифровать текст с помощью ключа
2. Расшифровать текст с помощью ключа
Введите номер: 1
Введите названия файла с текстом для шифрования: input.txt
Данные успешно записаны в файл: encrypted.txt
Результат: "ыюе сив лвйла  еека хсвтапулно нлю ж кдкмершдблн д,куибнядн.ас мжн имос ослин,янагмвирчокдьорор реМ енат  оноониоео,воамовм ы дкринсниызииы бл йаоенц,аоВмяо,
Ключ: "19-157-111-125-92-55-129-7-97-88-6-152-141-10-117-46-38-98-119-167-20-151-86-12-14-81-22-15-140-44-42-69-52-77-67-137-104-112-115-134-41-138-84"

Process finished with exit code 0
```

Результат шифрования:
ыюе сив лвйла еека хсвтапулно нлю ж кдкмершдблн д,куибнядн.ас мжн имос ослин,янагмвирчокдьорор реМ енат оноониоео,воамовм ы дкринсниызииы бл йаоенц,аоВмяо, р ворлео гонр

Ключ: 19-157-111-125-92-55-129-7-97-88-6-152-141-10-117-46-38-98-119-167-20-151-86-12-14-81-22-15-140-44-42-69-52-77-67-137-104-112-115-134-41-138-84

Дешифрование текста:

```
Выберите действие:
1. Зашифровать текст с помощью ключа
2. Расшифровать текст с помощью ключа
Введите номер: 2
Введите названия файла с текстом для дешифрования: encrypted.txt
Введите ключ шифрования: 19-157-111-125-92-55-129-7-97-88-6-152-141-10-117-46-38-98-119-167-20-151-86-12-14-81-22-15-140-44-42-69-52-77-67-137-104-112-115-134-41-138-84
Результат: "В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покрившимся балконом, жила некогда барыня, вдова, окруженная многочисленною д
```

Дешифрованный текст:
В одной из отдаленных улиц Москвы, в сером доме с белыми колоннами, антресолью и покрившимся балконом, жила некогда барыня, вдова, окруженная многочисленною дворней.

Отличие – два пробела в конце, так как длина фразы 167, ближайший больший квадрат 169.