# Информационная безопасность. Лабораторная работа №1. из сборника А. А. Ожиганова Криптографические системы с секретным ключом.


| | |
|---|---|
| Группа: | P34121 |
| Студент: | Гиниятуллин Арслан Рафаилович |
| Преподаватель: | Маркина Татьяна Анатольевна |
| Вариант: | 4 |



Санкт-Петербург
2024

# 1 Цель работы

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

# 2 Порядок выполнения работы

1. Ознакомьтесь с теоретическими основами шифрования данных, которые приведены в [1] и [2].
2. Получите вариант задания у преподавателя.
3. Напишите программу согласно варианту задания.
4. Отладьте разработанную программу и покажите результаты программы преподавателю.
5. Составьте отчет по лабораторной работе.

# 3 Содержание отчета

Отчет по лабораторной работе должен содержать следующие сведения:

- название и цель работы;
- вариант задания;
- листинг разработанной программы с комментариями;
- результаты работы программы.

# 4 Текст задания

Текст задания для моего варианта – 4.
Реализовать в программе шифрование и дешифрацию файла с использованием квадрата Полибия, обеспечив его случайное заполнение.

# 5 Этапы выполнения работы

## 5.1 Программирование алгоритма де/шифрования

```
from abc import ABC, abstractmethod
import random
import math

class Cipher(ABC):
    @abstractmethod
    def encrypt(self, text):
        pass

    @abstractmethod
    def decrypt(self, text):
        pass

    def process_file(self, input_file, output_file, mode='encrypt'):
        with open(input_file, 'r') as f:
            content = f.read()

        if mode == 'encrypt':
            processed_content = self.encrypt(content)
```

```
        elif mode == 'decrypt':
            processed_content = self.decrypt(content)
        else:
            raise ValueError("Mode should be either 'encrypt' or 'decrypt'.")

        with open(output_file, 'w') as f:
            f.write(processed_content)

class PolybiusSquareCipher(Cipher):
    def __init__(self, alphabet):
        self.alphabet = alphabet.upper()
        self.square_size = math.ceil(math.sqrt(len(self.alphabet)))
        self._validate_alphabet()
        self.square = self._generate_random_square()

    def _validate_alphabet(self):
        if len(set(self.alphabet)) != len(self.alphabet):
            raise ValueError("Alphabet must contain unique characters.")

    def _generate_random_square(self):
        shuffled_alphabet = list(self.alphabet)
        random.shuffle(shuffled_alphabet)
        square = []
        index = 0
        for i in range(self.square_size):
            row = []
            for j in range(self.square_size):
                if index < len(shuffled_alphabet):
                    row.append(shuffled_alphabet[index])
                    index += 1
                else:
                    row.append('')
            square.append(row)
        return square

    def encrypt(self, text):
        text = text.upper()
        encrypted_text = ''
        for char in text:
            if char in self.alphabet:
                for i, row in enumerate(self.square):
                    if char in row:
                        j = row.index(char)
                        encrypted_text += str(i + 1) + str(j + 1)
        return encrypted_text

    def decrypt(self, code):
        decrypted_text = ''
        if len(code) % 2 != 0:
            raise ValueError("The code length must be even.")

        for i in range(0, len(code), 2):
            row, col = int(code[i]) - 1, int(code[i + 1]) - 1
            if 0 <= row < self.square_size and 0 <= col < self.square_size:
                decrypted_text += self.square[row][col]
        return decrypted_text

en_alph = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

```
rus_alph = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЭЮЯ'
rus_alph_extended = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЭЮЯ- :;\n'
cipher = PolybiusSquareCipher(alphabet=rus_alph_extended)

cipher.process_file('input.txt', 'encrypted.txt', mode='encrypt')

cipher.process_file('encrypted.txt', 'decrypted.txt', mode='decrypt')
```

## 5.2 Проверка алгоритма де/шифрования

Проверим реализованный алгоритм на следующем отрывке А. С. Пушкина из поэмы «Руслан и Людмила».

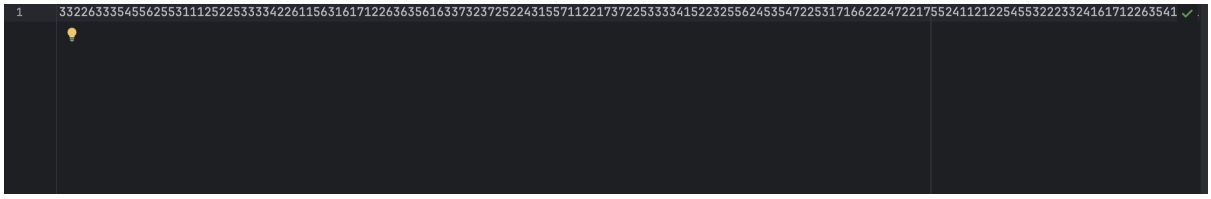
```
У лукоморья дуб зелёный;
Златая цепь на дубе том:
И днём и ночью кот учёный
Всё ходит по цепи кругом;
Идёт направо - песнь заводит,
Налево - сказку говорит.
Там чудеса: там леший бродит,
Русалка на ветвях сидит;
```

Зададим следующий алфавит:

```
rus_alph_extended = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЭЮЯ- :;\n'
```

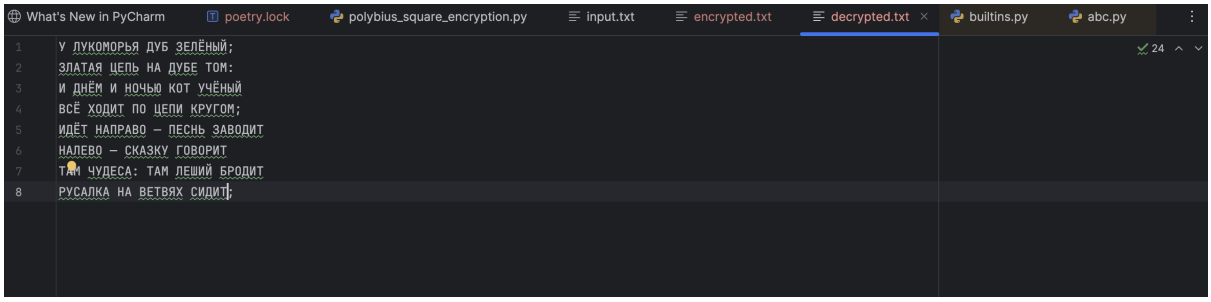
Посмотрим на зашифрованное сообщение *ecnrypted.txt*:

```
33226333545562553111252253333422611563161712263635616337323725224315571122173722533334152232556245354722531716622247221755241121225455322233241617122635414
431557112217372253333415223255624535472253171662224722175524112125455322333241617122635414
254553222332416171226354156162213555347322257552243155747225431334
445562363547531632221737573137415522522257155617112261374155534733
235173763154155225222565437615433224455415531473235323762222433533
155637452232376222631527472622343155534732353133563763543722173722
411532412513225647534732364
```




А теперь это же сообщение дешифруем в *decrypted.txt*:

```
У ЛУКОМОРЬЯ ДУБ ЗЕЛЁНЫЙ;
ЗЛАТАЯ ЦЕПЬ НА ДУБЕ ТОМ:
И ДНЁМ И НОЧЬЮ КОТ УЧЁНЫЙ
ВСЁ ХОДИТ ПО ЦЕПИ КРУГОМ;
ИДЁТ НАПРАВО - ПЕСНЬ ЗАВОДИТ
НАЛЕВО - СКАЗКУ ГОВОРИТ
ТАМ ЧУДЕСА: ТАМ ЛЕШИЙ БРОДИТ
РУСАЛКА НА ВЕТВЯХ СИДИТ;
```




# 6 Заключение

В ходе выполнения данной лабораторной работы я освоил программную реализацию шифрования и дешифрации с использованием квадрата Полибия.

# Список литературы

[1] Учебное пособие по дисциплине «Криптография». Электронный ресурс находится по адресу: http://isu.ifmo.ru (Вход через личный кабинет).

[2] Учебно-методическое пособие к выполнению лабораторных работ по дисциплине «Криптография». Электронный ресурс находится по адресу: http://isu.ifmo.ru (Вход через личный кабинет).