

# **Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №2**

**Шифры перестановки**

Лубышева Ярослава Михайловна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	11
5	Список литературы	12

# List of Figures

3.1	Первая часть программной реализации шифров . . . . .	7
3.2	Программная реализация маршрутного шифрования . . . . .	8
3.3	Программная реализация шифрования с помощью решеток . . .	9
3.4	Программная реализация шифра по талице Виженера . . . . .	9
3.5	Результат работы программы для маршрутного шифрования . . .	10
3.6	Результат работы программы для шифрования с помощью решеток	10
3.7	Результат работы программы для шифра по талице Виженера . .	10

## List of Tables

# **1 Цель работы**

Выполнить задание к лабораторной работе № 2 [1].

## 2 Задание

- 1) Изучить шифры перестановки: маршрутное шифрование, шифрование с помощью решеток, шифр по таблице Виженера.
- 2) Реализовать программно каждый шифр.

### 3 Выполнение лабораторной работы

Для реализации каждого вида шифрования была написана программа на языке программирования Python (fig. 3.1 - fig. 3.4).

```
from itertools import islice
import numpy as np

# алфавит
alphabet = ['a', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п',
            'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']

# функция для транспонирования матрицы
def transpose(l1):
    l2 = []
    for i in range(len(l1[0])):
        row = []
        for item in l1:
            row.append(item[i])
        l2.append(row)
    return l2
```

Figure 3.1: Первая часть программной реализации шифров

```

# Маршрутное шифрование
# n - длина блоков
# m - количество блоков
# password - пароль
# mes - сообщение для шифрования
def route_encryption(password, mes):
    mes = list(mes.lower())
    password = list(password.lower())
    n = len(password)
    m = len(mes)//n if len(mes)%n == 0 else len(mes)//n+1
    print("Пароль: ", password, "m: ", m, "n: ", n)

    # массив шифрования
    # преобразовываем сообщение в матрицу размера n*m для дальнейшего шифрования
    it = iter(mes)
    encrypt = [list(islice(it, i)) for i in [n]*m]

    # припишем к последнему блоку буквы из алфавита,
    # если он получился меньше остальных
    encrypt[-1] += alphabet[0:n-len(encrypt[-1])]

    # транспонируем матрицу-сообщение для упрощения вывода криптограммы
    encrypt_2 = transpose(encrypt)

    # согласно алфавитному порядку пароля выпишем криптограмму
    print("Криптограмма: ", end='')
    for let in alphabet:
        if let in password:
            # выписываем строку из encrypt_2 по индексу буквы в пароле
            print(''.join(encrypt_2[password.index(let)]), end='')

```

Figure 3.2: Программная реализация маршрутного шифрования



```

# шифрование с помощью решеток
def grid_encryption(password, mes, grid):
    mes = mes.lower()
    password = list(password.lower())
    # если в сообщении недостаточно символов для шифрования, допишем буквы в конец сообщения
    mes += "".join(alphabet[0:len(grid)**2-len(mes)])
    # пустой массив шифрования
    data = [[0, 0, 0, 0],
            [0, 0, 0, 0],
            [0, 0, 0, 0],
            [0, 0, 0, 0]]
    # счетчик для количества букв в сообщении
    let_i = 0
    # заполняем массив шифрования согласно сетке, затем поворачиваем ее по часовой стрелке
    for _ in range(4):
        # проходим по каждому элементу сетки и для каждого значения "1" ставим букву сообщения в соответствующую клетку массива data
        for i in range(len(grid)):
            for j in range(len(grid[i])):
                if grid[i][j] == 1:
                    data[i][j] = mes[let_i]
                    let_i += 1
        # отрицательный знак второго аргумента - поворот по часовой стрелке
        grid = np.rot90(grid, -1)
    print("Итоговая таблица криптограммы:\n", data)
    # транспонируем матрицу-криптограмму для упрощения вывода криптограммы
    encrypt_2 = transpose(data)
    # согласно алфавитному порядку пароля выпишем криптограмму
    print("Криптограмма: ", end='')
    for let in alphabet:
        if let in password:
            # выписываем строку из encrypt_2 по индексу буквы в пароле
            print(''.join(encrypt_2[password.index(let)]), end='')

```

Figure 3.3: Программная реализация шифрования с помощью решеток

```

def vishener_encryption(password, mes):
    mes = list(mes.lower())
    password = list(password.lower())

    # сделаем, чтобы пароль password дублировался на всю длину сообщения mes
    password *= len(mes)//len(password) + 1
    password = password[:len(mes)]

    print(password)
    print(mes)
    print()

    cryptogram = ""

    # находим букву пароля в вертикальном алфавите,
    # а букву сообщения в горизонтальном - на пересечении буква криптограммы
    for let_password, let_mes in zip(password, mes):
        # если обе буквы находятся в алфавите
        if let_password in alphabet and let_mes in alphabet:
            # находим, какой индекс буква на пересечении имеет в алфавите
            num = ( alphabet.index(let_password)+1 + alphabet.index(let_mes)+1 ) % (len(alphabet)) - 1
            cryptogram += ''.join(alphabet[num])

    print("Криптограмма: ", cryptogram)

```

Figure 3.4: Программная реализация шифра по таблице Виженера

Результаты работы маршрутного шифрования представлены на следующем рисунке (fig. 3.5).

```
password = "пароль" #input("пароль ")
mes = "НельзяНедооцениватьПротивника" #input("сообщение ")
route_encryption(password, mes)
```

```
Пароль: ['п', 'а', 'р', 'о', 'л', 'ь'] m: 5 n: 6
Криптограмма: еенпнзоатаьовокннеевлдиряцтиа
```

Figure 3.5: Результат работы программы для маршрутного шифрования

Результаты работы шифрования с помощью решеток представлены на следующем рисунке (fig. 3.6).

```
password = "шифр"
mes = "ДоговорПодписали"

grid = [[0, 0, 0, 1],
        [0, 0, 0, 0],
        [0, 1, 0, 1],
        [0, 0, 1, 0]]
```

```
grid_encryption(password, mes, grid)
```

```
Итоговая таблица криптограмма:
[['с', 'о', 'а', 'д'], ['д', 'в', 'п', 'л'], ['о', 'о', 'и', 'г'], ['и', 'р', 'о', 'п']]
Криптограмма: овордлгпапиосдои
```

Figure 3.6: Результат работы программы для шифрования с помощью решеток

Результаты работы шифра по талице Виженера представлены на следующем рисунке (fig. 3.7).

```
password = "математика"
mes = "КриптографияСерьёзнаяНаука"
vishener_encryption(password, mes)
```

```
['м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а', 'т', 'и', 'к', 'а', 'м', 'а', 'т', 'е', 'м', 'а']
['к', 'р', 'и', 'п', 'т', 'о', 'г', 'р', 'а', 'ф', 'и', 'я', 'с', 'ь', 'р', 'ь', 'ё', 'з', 'н', 'а', 'я', 'н', 'а', 'у', 'к', 'а']

Криптограмма: шсьхапцълхцаекюэщсбмоушб
```

Figure 3.7: Результат работы программы для шифра по талице Виженера

## **4 Выводы**

Выполнено задание к лабораторной работе № 2.

## **5 Список литературы**

1. Методические материалы курса