

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

**Шифры перестановки**

Терентьев Егор Дмитриевич 1132236902

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Маршрутное шифрование . . . . .	6
2.2	Шифрование с помощью решеток . . . . .	8
2.3	Таблица Виженера . . . . .	11
<b>3</b>	<b>Выводы</b>	<b>13</b>
<b>4</b>	<b>Список литературы</b>	<b>14</b>

# List of Figures

2.1	route_funcs . . . . .	7
2.2	route_main_func . . . . .	8
2.3	route_output . . . . .	8
2.4	grid_funcs . . . . .	9
2.5	grid_main_func . . . . .	10
2.6	grid_output . . . . .	10
2.7	viginere_funcs . . . . .	11
2.8	viginere_main_func . . . . .	12
2.9	viginere_output . . . . .	12

## List of Tables

# 1 Цель работы

Освоить на практике шифры перестановки.

## 2 Выполнение лабораторной работы

Требуется реализовать:

1. Маршрутное шифрование.
2. Шифрование с помощью решеток.
3. Табоица Виженера

### 2.1 Маршрутное шифрование

Текст разбивается на равные блоки  $N$  длиной  $M$ . Если в конце не хватает букв, то они добавляются в конец. Блоки записываются построчно в таблицу. Затем буквы выписываются по столбцам, которые упорядиваются согласно паролю: внизу таблицы приписывается слово из  $n$  неповторяющихся букв и столбы нумеруются по алфавитному порядку букв пароля

Чтобы реализовать программу был написан след. код на python:

1. Функции проверки правильности пароля, значения  $k$
2. Функция берущая столбцы матрицы в виде ключа буквы пароля в алфавитном порядке (был использован словарь для удобства) fig. 2.1.

```

import math
import re

def is_pass_valid(password, columns):
    if len(password) != columns:
        print("pass len should be equal to row len")
        raise ValueError
    alphabeticValues = re.findall(r'[a-zA-Zа-яА-Я]', password)
    if len(alphabeticValues) != len(password):
        print("pass should contain only eng and ru letters")
        raise ValueError

def is_len_columns_valid(columns, inputString):
    if columns > len(inputString):
        print("len columns should be <= len of Input String")
        raise ValueError
    if columns <= 1:
        raise ValueError

def to_dict(inputString, password, columns, row):
    list_of_slices = []
    for i in range(row):
        list_of_slices.append(inputString[columns*i:columns*(i+1)])

    while len(list_of_slices[-1]) != columns:
        list_of_slices[-1] += 'a'

    routeDict = {}

    for i in range(columns):
        stringToAppend = ""
        for j in range(row):
            stringToAppend += list_of_slices[j][i]
        routeDict[password[i]] = stringToAppend

    sorted_dict = dict(sorted(routeDict.items()))

    return sorted_dict

```

Figure 2.1: route\_funcs

Main функция в которой вводятся начальные значения, запускаются вышепоказанные функции fig. 2.2.

```

inputedString = input("Input string to encrypt: ")
columns = int(input("Input int value to determine the number columns: "))
password = input("Password: ").lower()

inputString = inputedString.replace(" ", "") # remove spaces
password = password.replace(" ", "") # remove spaces
password = ''.join([password[i] for i in range(len(password)-1) if password[i+1] != password[i]+[password[-1]]]) #remove duplicates
is_len_columns_valid(columns, inputString)
is_pass_valid(password, columns)
row = math.ceil(len(inputString) / int(columns))

routeDict = to_dict(inputString, password, columns, row)

cryptogram = ""
for key in routeDict:
    cryptogram += routeDict[key]

print("начальная фраза: ", inputString)
print("криптограмма: ", cryptogram)

```

Figure 2.2: route\_main\_func

Вывод программы (пример как в методических материалах) fig. 2.3.

```

Input string to encrypt: нельзя недооценивать противника
Input int value to determine the number columns: 6
Password: пароль
начальная фраза:  нельзя недооценивать противника
криптограмма:  еенпнзоатаьовокннеьвлдирияцтиа
PS F:\учеба 5 курс\информационная безопасность>

```

Figure 2.3: route\_output

## 2.2 Шифрование с помощью решеток

Строится квадрат из  $k$  чисел. Затем к нему добавляются еще 3 квадрата, которые поворачиваются на 90 градусов и получается большой квадрат  $2k$  размерностью. Дальше из большого квадрата вырезаются клетки и прорези записываются буквы. Когда заполнятся все прорези решето поворачивается на 90 градусов. И так продолжается пока не заполнится вся таблица. И буквы выписываются по алфавитному порядку пароля.

Чтобы реализовать программу был написан след. код на python:



1. Функция генерирующая сетку (матрицу) (использована библиотека numpy Для удобства) и ее заполнение
2. Функция заполняющая сетку значениями букв из текста и переворачивающая матрицу
3. Функция выбирающая столбцы в алфавитном порядке пароля
4. Функция объединяющая все вышепоказанные функции и проверки правильности введенных данных fig. 2.4

```
1 import numpy as np
2
3 def generate_grid(k):
4     grid = np.zeros((k, k))
5     for i in range(k):
6         for j in range(k):
7             grid[i][j] = i * k + j + 1
8     return grid
9
10 def encrypt_with_grid(grid, text):
11     k = len(grid)
12     encrypted = np.chararray((2*k, 2*k), unicode=True)
13     encrypted[:] = ''
14
15     idx = 0
16     for _ in range(4):
17         for i in range(k):
18             for j in range(k):
19                 if grid[i][j] and idx < len(text):
20                     encrypted[i][j] = text[idx]
21                     idx += 1
22
23     grid = np.rot90(grid)
24
25     return encrypted
26
27 def extract_by_password(grid, password):
28     order = sorted(range(len(password)), key=lambda x: password[x])
29     return ''.join([''.join(grid[:, i]) for i in order])
30
31 def encrypt(text, password):
32     k = int(len(text)**0.5)
33     if k*k != len(text):
34         raise ValueError("Length of the text should be a perfect square.")
35
36     if len(password) != k:
37         raise ValueError(f"Length of the password should be {k}.")
38
39     grid = generate_grid(k)
40     encrypted_grid = encrypt_with_grid(grid, text)
41     result = extract_by_password(encrypted_grid, password)
42
43     return result
```

Figure 2.4: grid\_funcs

Main функция запуска программы fig. 2.5

```

def encrypt(text, password):
    k = int(len(text)**0.5)
    if k*k != len(text):
        raise ValueError("Length of the text should be a perfect square.")

    if len(password) != k:
        raise ValueError(f"Length of the password should be {k}.")

    grid = generate_grid(k)
    encrypted_grid = encrypt_with_grid(grid, text)
    result = extract_by_password(encrypted_grid, password)

    return result

# Пример использования
text = "нужно подписать новый указ"
password = "шифр"
print(encrypt(text, password))

```

Figure 2.5: grid\_main\_func

Пример работы программы fig. 2.6

```

44
45  # Пример использования
46  text = "нужно подписать новый указ"
47  password = "шифр"
48  print(encrypt(text, password))
49

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\учеба 5 курс\информационная безопасность> & C:  
○ уоаванпйдоинуаждтызнпсок  
PS F:\учеба 5 курс\информационная безопасность>

Figure 2.6: grid\_output

## 2.3 Таблица Виженера

В таблице записаны буквы русского алфавита. При переходе от одной строке к другой происходит циклический сдвиг на одну позицию. Пароль записывается с повторениями над буквами сообщения. В горизонтальном алфавите ищем букву нашего текста, а в вертикальном букву пароля и на их пересечении будет нужная нам буква.

Чтобы реализовать программу был написан след. код на python:

1. Функция шифрования (построение таблицы Вижинера)
2. Функция дешифровки fig. 2.7

```
1 def vigenere_encrypt(text, key):
2     alph = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
3     encrypted_text = ''
4
5     #размерность пароля = размерности текста
6     key_expanded = ''
7     while len(key_expanded) < len(text):
8         key_expanded += key
9     key_expanded = key_expanded[:len(text)]
10
11     #сравниваем по таблице и получаем на пересечении букву
12     for t, k in zip(text, key_expanded):
13         if t in alph:
14             new_pos = (alph.index(t) + alph.index(k)) % len(alph)
15             encrypted_text += alph[new_pos]
16         else:
17             encrypted_text += t
18
19     return encrypted_text
20
21 def vigenere_decrypt(encrypted_text, key):
22     alph = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
23     decrypted_text = ''
24
25     key_expanded = ''
26     while len(key_expanded) < len(encrypted_text):
27         key_expanded += key
28     key_expanded = key_expanded[:len(encrypted_text)]
29
30     for e, k in zip(encrypted_text, key_expanded):
31         if e in alph:
32             new_pos = (alph.index(e) - alph.index(k)) % len(alph)
33             decrypted_text += alph[new_pos]
34         else:
35             decrypted_text += e
36
37     return decrypted_text
38
```

Figure 2.7: viginere\_funcs

Main функция запуска программы fig. 2.8

```

# Пример использования
text = "криптографиясерьезнаянаука"
password = "математика"
encrypted_text = vigenere_encrypt(text, password)
print(f"Encrypted: {encrypted_text}")
print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")

```

Figure 2.8: vigenere\_main\_func

Пример работы программы (как в методических материалах) fig. 2.9

```

38
39 # Пример использования
40 text = "криптографиясерьезнаянаука"
41 password = "математика"
42 encrypted_text = vigenere_encrypt(text, password)
43 print(f"Encrypted: {encrypted_text}")
44 print(f"Decrypted: {vigenere_decrypt(encrypted_text, password)}")
45

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS F:\учеба 5 курс\информационная безопасность> & C:/Python311/python.exe "f:/учеба
Encrypted: црѣфюохшкфягкьчпчалнтща
Decrypted: криптографиясерьезнаянаука
PS F:\учеба 5 курс\информационная безопасность>

```

Figure 2.9: vigenere\_output

## **3 Выводы**

В результате выполнения работы я освоил на практике применение шифров перестановки.

## **4 Список литературы**

1. Методические материалы курса