

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ  
Факультет физико-математических и естественных наук  
Кафедра теории вероятностей и кибербезопасности

## Лабораторная работа №2. Шифры перестановки

*Дисциплина: Математические основы защиты  
информации и информационной безопасности*

Студент: Банникова Екатерина Алексеевна  
Группа: НФИмд-02-23

Москва 2023

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Маршрутное шифрование . . . . .	9
4.2	Метод решеток . . . . .	11
4.3	Таблица Виженера . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>17</b>
<b>6</b>	<b>Список литературы</b>	<b>18</b>

# List of Figures

4.1	1 часть программного кода реализации маршрутного шифрования	9
4.2	2 часть программного кода реализации маршрутного шифрования	10
4.3	3 часть программного кода реализации маршрутного шифрования	10
4.4	Результат шифрования сообщений с использованием маршрутного шифрования . . . . .	11
4.5	1 часть программного кода реализации шифрования с помощью решеток . . . . .	11
4.6	2 часть программного кода реализации шифрования с помощью решеток . . . . .	12
4.7	3 часть программного кода реализации шифрования с помощью решеток . . . . .	12
4.8	4 часть программного кода реализации шифрования с помощью решеток . . . . .	13
4.9	5 часть программного кода реализации шифрования с помощью решеток . . . . .	13
4.10	Результат шифрования сообщений с использованием шифрования с помощью решеток . . . . .	14
4.11	1 часть программного кода реализации шифрования с помощью Таблица Виженера . . . . .	14
4.12	2 часть программного кода реализации шифрования с помощью Таблица Виженера . . . . .	15
4.13	3 часть программного кода реализации шифрования с помощью Таблица Виженера . . . . .	15
4.14	4 часть программного кода реализации шифрования с помощью Таблица Виженера . . . . .	16
4.15	Результат шифрования сообщений с использованием шифрования с помощью Таблица Виженера . . . . .	16

## List of Tables

# 1 Цель работы

Целью данной лабораторной работы является ознакомление с тремя методами шифрования: маршрутным шифрованием, шифрованием с помощью решеток, таблицей Виженера, – а так же их реализация на произвольном языке программирования.

## 2 Задание

1. Реализовать метод маршрутного шифрования.
2. Реализовать метод шифрования с помощью решеток.
3. Реализовать метод таблицы Вижера.

### 3 Теоретическое введение

Математическая часть подробно описана в задании к лабораторной работе. Я поставила перед собой задачу найти исторические сведения, факты о методах шифрования.

**Шифр перестановки** — это метод симметричного шифрования, в котором элементы исходного открытого текста меняют местами. Элементами текста могут быть отдельные символы, пары букв, тройки букв, комбинирование этих случаев и так далее. Типичными примерами перестановки являются анаграммы. В классической криптографии шифры перестановки можно разделить на два класса:

1. Шифры одинарной (простой) перестановки — при шифровании символы открытого текста перемещаются с исходных позиций в новые один раз.
2. Шифры множественной (сложной) перестановки — при шифровании символы открытого текста перемещаются с исходных позиций в новые несколько раз.

Метод **маршрутного шифрования** изобрел французский математик и криптограф Франсуа Виет. Этот способ относится к перестановочным шифрам. Шифр называется перестановочным, если все связанные с ним криптограммы получаются из соответствующих открытых текстов перестановкой букв. Способ, каким при шифровании переставляются буквы открытого текста, и является ключом шифра. Такой метод шифрования (столбцовая перестановка) в годы первой мировой войны использовала легендарная немецкая шпионка Мата Хари.

**Шифровальная решётка** — трафарет с прорезями-ячейками (из бумаги, картона или аналогичного материала), использовавшийся для шифрования открытого текста. Текст наносился на лист бумаги через такой трафарет по определённым правилам, и расшифровка текста была возможна только при наличии такого же трафарета.

**Вращающаяся решетка:** Прямоугольные решётки Кардано можно размещать в четырёх позициях. Шифр с сеткой в виде шахматной доски имеет только две позиции, но именно этот вариант вращающейся решётки послужил для разработки более сложной решётки с четырьмя позициями, которую можно вращать в двух направлениях.

**Шифр Виженера** является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джован Баттиста Беллазо (итал. Giovan Battista Bellaso) в книге *La cifra del. Sig. Giovan Battista Bellaso* в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.



## 4 Выполнение лабораторной работы

**Примечание:** комментарии по коду представлены на скриншотах к каждому из проделанных заданий.

### 4.1 Маршрутное шифрование

В соответствии с заданием, первой была написана программа для маршрутного шифрования. Программный код представлен ниже.

```
#задание 1
def marsh_shifr():
    m=5#длина блока
    n=6#количество блоков
    text="Нельзя недооценивать противника"#текст для шифрования
    text1=text.upper()#заглавными буквами
    result=list(text1)#сделали список из строки
    '''
    Исключаем пробелы, точки, запятые, тире и тп
    '''
    for symbol in result:
        if (symbol==' ' or (symbol=='.' or (symbol==',' or (symbol=='-' or (symbol=='?' or (symbol=='!' or (symbol==';' or (symbol==' ':
            index=result.index(symbol)
            element=result.pop(index)#вырезаем символ по заданному индексу
            result1=result.copy()
            result2=[]
            '''
            Создаем необходимую матрицу с шагом n
            '''
            for i in range (0,len(result1),n):
                result2.append(list(result1[i:n+i]))
            #недостающие элементы заполняем буквами A
            while (len(result2[m-1])<n):
                result2[m-1].append('A')
```

Figure 4.1: 1 часть программного кода реализации маршрутного шифрования

```

'''
Ввели пароль в корректной для работе формы
'''

text2="пароль"
text2=text2.upper()
password=list(text2)
###
result3=list(result2)
result3.append(password)#добавили к матрице пароль
alphabet="АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"#ввели алфавит
indices=[]#пустой список для индексов
'''

Смотрим на индексы пароля в алфавите
'''

for pas in password:
    for letter in alphabet:
        if pas==letter:
            ind=alphabet.find(letter)
            indices.append(ind)
result4=list(result3)
result4.append(indices)#добавили индексы в матрицу
result5=np.array(result4)
result6=result5[:,np.argsort(result5[-1,:])]#сортировка
result7=list(result6)

```

Figure 4.2: 2 часть программного кода реализации маршрутного шифрования

```

'''
убрали две последние строки в матрице
'''

del (result7[-1])
del (result7[-1])
result8=np.array(result7)
result9=result8.transpose()#транспонируем для того чтобы выписать шифр
result10=[]
'''

Начали работу над выписыванием шифра
'''

for i in range (n):
    result10.extend(result9[i])
print("".join(result10))#выписали строку шифра
marsh_shifr()

```

Figure 4.3: 3 часть программного кода реализации маршрутного шифрования

Результаты выполнения программы представлены ниже. В качестве параметров системы были взяты данные из описательной части лабораторной работы портала ТУИС.

## ЕЕНПНЗОАТАЪОВОКННЕЪВЛДИРИЯЦТИА

Figure 4.4: Результат шифрования сообщений с использованием маршрутного шифрования

### 4.2 Метод решеток

```
#задание2
def turning_grille():
    k=2#вводим k
    ...

    заполняем маленькую матрицу
    ...

    osnova=np.linspace(1,k**2,k**2)
    result=[]
    for i in range(0,len(osnova),k):
        result.append(list(osnova[i:i+k]))
    ...

    вводим функцию для поворота матрицы
    ...

    def rot90(matrix):
        return [list(reversed(col)) for col in zip(*matrix)]
    matrix=np.full((2*k,2*k),0)#создали и заполнили нулями матрицу 2k x 2k
    ...

    заполняем матрицу matrix по четвертям
    ...

    #1четверть
    matrix[:k,:k]=result
    #2четверть
    result2=rot90(result)
    matrix[k,k:2*k]=result2
    #3четверть
    result3=rot90(result2)
    matrix[k:2*k,k:2*k]=result3
    #4четверть
    result4=rot90(result3)
    matrix[k:2*k,:k]=result4
```

Figure 4.5: 1 часть программного кода реализации шифрования с помощью решеток

```

...
работа с отверстиями (определение координат)
...
holes=[]
for i in range (1,k**2+1):#прогонка по отдельному числу, например, по единицам
    indexes=[]
    for m in range(0,2*k):#прогонка по строкам
        for j in range(0,2*k):#прогонка по столбцам
            if matrix[m][j]==i:
                coords=tuple([m,j])
                indexes.append(coords)
        find=random.randint(0,3)#выбираем 1 из 4 координат
        holes.append(indexes[find])
...

работа с отверстиями (продолжение) визуализация поворотов и случаев размещений отверстий
...
template=np.full((2*k,2*k),0)
for d in range (k**2):
    template[holes[d][0],holes[d][1]]=1
#1поворот
templatel=rot90(template)
#2поворот
template2=rot90(templatel)
#3поворот
template3=rot90(template2)
text="ДОГОВОР ПОДПИСАЛИ"

```

Figure 4.6: 2 часть программного кода реализации шифрования с помощью решеток

```

...
прогоняем templates для нахождения координат для заполнения буквами
...
#1поворот
indexes1=[]
for m1 in range (0,2*k):
    for j1 in range (0,2*k):
        if templatel[m1][j1]==1:
            coords1=tuple([m1,j1])
            indexes1.append(coords1)
#2поворот
indexes2=[]
for m2 in range (0,2*k):
    for j2 in range (0,2*k):
        if template2[m2][j2]==1:
            coords2=tuple([m2,j2])
            indexes2.append(coords2)
#3поворот
indexes3=[]
for m3 in range (0,2*k):
    for j3 in range (0,2*k):
        if template3[m3][j3]==1:
            coords3=tuple([m3,j3])
            indexes3.append(coords3)
...

```

Figure 4.7: 3 часть программного кода реализации шифрования с помощью решеток

```

'''
Переходим к образованию матрицы с буквами
'''

letters_matrix=np.full((2*k,2*k),'O')
#0
for d in range (k**2):
    letters_matrix[holes[d][0],holes[d][1]]=text[d]
#1
for d in range (k**2):
    letters_matrix[indexes1[d][0],indexes1[d][1]]=text[d+k**2]
#2
for d in range (k**2):
    letters_matrix[indexes2[d][0],indexes2[d][1]]=text[d+2*(k**2)]
#3
for d in range (k**2):
    letters_matrix[indexes3[d][0],indexes3[d][1]]=text[d+3*(k**2)]
#####
letter_matrix=list(letters_matrix)
text2="шифр"
text2=text2.upper()
password=list(text2)
letter_matrix.append(password)
alphabet="АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"
indices=[]

```

Figure 4.8: 4 часть программного кода реализации шифрования с помощью решеток

```

'''
Смотрим на индексы пароля в алфавите
'''

for pas in password:
    for letter in alphabet:
        if pas==letter:
            ind=alphabet.find(letter)
            indices.append(ind)
letter_matrix.append(indices)
letter_matrix=np.array(letter_matrix)
letter_matrix=letter_matrix[:,np.argsort(letter_matrix[-1,:])]#упорядочили
letter_matrix=list(letter_matrix)
del (letter_matrix[-1])#убрали строку с индексами букв из пароля в алфавите
del (letter_matrix[-1])#убрали строку с индексами букв из пароля в алфавите
letter_matrix=np.array(letter_matrix)
letter_matrix=letter_matrix.transpose()
letter_matrix=list(letter_matrix)
#####
'''
Выводим ответ в виде строки
'''

result1=[]
for i in range (2*k):
    result1.extend(letter_matrix[i])
print("".join(result1))
turning_grille()

```

Figure 4.9: 5 часть программного кода реализации шифрования с помощью решеток

Результаты выполнения программы представлены ниже. В качестве параметров системы были взяты данные из описательной части лабораторной работы

портала ТУИС.

ДЛГПАВПОСДОИООИР

Figure 4.10: Результат шифрования сообщений с использованием шифрования с помощью решеток

### 4.3 Таблица Виженера

Ну и наконец мы перешли к шифрованию при помощи таблицы Виженера. Программный код представлен ниже.

```
#задание 3
def table_vigenera():
    text="криптография серьезная наука"
    password="математика"
    text1=text.upper()
    result=list(text1)
    ...
    Исключаем пробелы, точки, запятые, тире и тп
    ...

    for symbol in result:
        if (symbol==' ' or (symbol=='.' or (symbol==',' or (symbol=='-' or (symbol=='?' or (symbol=='!' or (symbol==';' or (symbol==' ':
            index=result.index(symbol)
            element=result.pop(index)#вырезаем символ по заданному индексу
            result1=result.copy()
            password_line=[]
            password=password.upper()
            password=list(password)
            ...

            Заполняем строку, которая будет использоваться для шифрования (с паролем)
            ...
            i=0
            while len(password_line)!=len(result1):
                if i==len(password):
                    i=0
                password_line.append(password[i])
                i+=1
            #####
```

Figure 4.11: 1 часть программного кода реализации шифрования с помощью Таблица Виженера

```

'''
создаем волшебную алфавитную матрицу
'''

alphabet_matrix=[]
alphabet="АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЭЮЯ"
alphabet_matrix.append(list(alphabet))
d=alphabet
i=0
while i<33:
    '''
    задаем, каким образом будет происходить смещение в строчке
    '''

    d=deque(d)
    d.rotate(-1)
    d=''.join(list(d))
    alphabet_matrix.append(list(d))
    i+=1
#####

```

Figure 4.12: 2 часть программного кода реализации шифрования с помощью Таблицы Виженера

```

'''
Задаем то, как будет осуществляться поиск по индексам в таблице алфавита (определяем индексы по поиску в таблице букв из строки для зашифровывания и пароля)
'''
indices1=[]
indices2=[]
for pas in password_line:
    for letter in alphabet:
        if pas==letter:
            ind1=alphabet.find(letter)
            indices1.append(ind1)
for res in result1:
    for letter in alphabet:
        if res==letter:
            ind2=alphabet.find(letter)
            indices2.append(ind2)
answer=[]

```

Figure 4.13: 3 часть программного кода реализации шифрования с помощью Таблицы Виженера

```

'''
    непосредственно поиск шифра
'''
j=0
while j<len(password_line):
    answer.append(list(alphabet_matrix[indices2[j]][indices1[j]]))
    j+=1
'''
Запись ответа
'''
answer1=[]
for i in range (len(answer)):
    answer1.extend(answer[i])
print("".join(answer1))
table_vigenera()

```

Figure 4.14: 4 часть программного кода реализации шифрования с помощью Таблица Виженера

Результаты выполнения программы представлены ниже. В качестве параметров системы были взяты данные из описательной части лабораторной работы портала ТУИС.

**ЦРЬФЯОХШКФФЯДКЭЪЧПЧАЛНТШЦА**

Figure 4.15: Результат шифрования сообщений с использованием шифрования с помощью Таблица Виженера



## 5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: я ознакомилась с тремя методами шифрования – маршрутным шифрованием, шифрованием с помощью решеток, таблицей Виженера, – а так же мне удалось реализовать их на языке программирования Python.

## 6 Список литературы

1. Википедия. Перестановочный шифр [Электронный ресурс]. Википедия, свободная энциклопедия. URL: [https://en.wikipedia.org/wiki/Transposition\\_cipher](https://en.wikipedia.org/wiki/Transposition_cipher).
2. VK. Метод маршрутного шифрования [Электронный ресурс]. VK, 2018. URL: [https://vk.com/\[cryptandcod-metod-marshrutnogo-shifrovaniya?\]](https://vk.com/[cryptandcod-metod-marshrutnogo-shifrovaniya?]).
3. Википедия. Шифровальная решетка [Электронный ресурс]. Википедия, свободная энциклопедия. URL: [https://en.wikipedia.org/wiki/Grille\\_\(cryptography\)](https://en.wikipedia.org/wiki/Grille_(cryptography)).
4. Wix. Криптография [Электронный ресурс]. Wixsite, 2021. URL: [https://en.wikipedia.org/wiki/Grille\\_\(cryptography\)](https://en.wikipedia.org/wiki/Grille_(cryptography)).
5. Википедия. Шифр Виженера [Электронный ресурс]. Википедия, свободная энциклопедия. URL: [https://en.wikipedia.org/wiki/Vigen%C3%A8re\\_cipher](https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher).