

Математические основы защиты информации и информационной безопасности. Лабораторная работа №2

Шифры перестановки

Студент: Лесков Данила Валерьевич НФИмд-02-21

Преподаватель: Кулябов Дмитрий Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Маршрутное шифрование	7
3.2	Шифрование с помощью решеток	8
3.3	Таблица Виженера	9
4	Выполнение лабораторной работы	11
4.1	Описание реализации методов	11
4.1.1	Описание реализации маршрутного шифрования	11
4.1.2	Описание реализации шифрования с помощью решеток	11
4.1.3	Описание реализации шифрования с помощью таблицы Виженера	12
4.2	Листинг	12
4.2.1	Маршрутное шифрование	12
4.2.2	Шифрование с помощью решеток	14
4.2.3	Таблица виженера	18
4.3	Полученные результаты	20
4.3.1	Маршрутное шифрование	20
4.3.2	Шифрование с помощью решеток	20
4.3.3	Таблица Виженера	21
5	Выводы	23
	Список литературы	24

List of Figures

3.1	Маршрутное шифрование	8
3.2	Шифрование с помощью решеток	9
3.3	Таблица Виженера	10
4.1	Результаты маршрутного шифрования	20
4.2	Результаты шифрования с помощью решеток	21
4.3	Результаты шифрования с помощью таблица Виженера	22

List of Tables

1 Цель работы

Ознакомиться с шифрами перестановки на примере маршрутного шифрования, шифрования с помощью решеток и таблицы Виженера.

2 Задание

1. Реализовать маршрутное шифрование;
2. Реализовать шифрование с помощью решеток;
3. Реализовать шифрование с помощью таблицы Виженера.

3 Теоретическое введение

Шифр перестановки — это метод симметричного шифрования, в котором элементы исходного открытого текста меняют местами. Элементами текста могут быть отдельные символы (самый распространённый случай), 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:

3.1 Маршрутное шифрование

При шифровании в такую таблицу вписывают исходное сообщение по определённому маршруту, а выписывают (получают шифрограмму) – по другому. Для данного шифра маршруты вписывания и выписывания, а также размеры таблицы являются ключом. [1]

В рамках работы данного алгоритма шифрования задаются две переменные: m - количество столбцов таблицы, которое равно длине ключа и n - количество строк в таблице.

Для случая, когда в сообщении недостаточно букв для того, чтобы заполнить всю таблицу, предусмотрено добавление случайных букв в конец сообщения.

н	е	л	ь	з	я
н	е	д	о	о	ц
е	н	и	в	а	т
ь	п	р	о	т	и
в	н	и	к	а	а
<hr/>					
п	а	р	о	л	ь

Figure 3.1: Маршрутное шифрование

В результате отработки алгоритма возвращаются отсортированные столбцы таблицы по алфавитному порядку букв ключа. На рис. 3.1 ключом является пароль, соответственно в результирующее сообщение сначала записывается столбец под буквой а ключа, и заканчивается столбцом под ь.

3.2 Шифрование с помощью решеток

Поворотная решетка — это прямоугольная или квадратная карточка с четным числом строк и столбцов $2k \times 2k$. В ней проделаны отверстия таким образом, что при последовательном отражении или поворачивании и заполнении открытых клеток карточки постепенно будут заполнены все клетки листа.

Карточку сначала отражают относительно вертикальной оси симметрии, затем - относительно горизонтальной оси, и снова - относительно вертикальной [2]. На рисунке 3.2 изображена последовательность поворота решетки для заполнения её буквами сообщения:

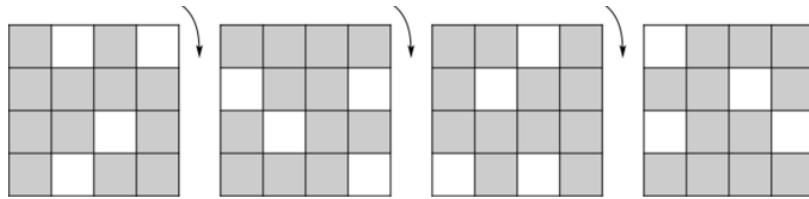


Figure 3.2: Шифрование с помощью решеток

По итогу, когда таблица заполнена, как и в предыдущем алгоритме столбцы решетки сортируются по алфавитному порядку букв ключа.

3.3 Таблица Виженера

Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая *tabula recta* или таблица Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет такой вид:

ATTACKATDAWN

Человек, посылающий сообщение, записывает ключевое слово («LEMON») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

LEMONLEMONLE

Если n — количество букв в алфавите, m_j — номер буквы открытого текста, k_j — номер буквы ключа в алфавите, то шифрование Виженера можно записать

следующим образом:

$$c_j = (m_j + k_j) \mod n$$

Пример таблицы виженера для латинского алфавита изображен на рис. 3.3:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 3.3: Таблица Виженера

Более подробно о шифре Вижинера: [3]

4 Выполнение лабораторной работы

В рамках данной лабораторной работы были описаны алгоритмы трех типов шифрования.

4.1 Описание реализации методов

4.1.1 Описание реализации маршрутного шифрования

Для реализации данного шифрования были описаны 4 метода: `get_nm(message, password)` для расчета размеров `n` и `m` матрицы, `message_to_dict(message, m, n)` для записи сообщения в таблицу, представленную python словарем с ключами букв пароля, `sort_dict(dict)` для сортировки словаря по ключу и метод преобразования словаря в зашифрованное сообщение `dict_to_string(dict)`.

4.1.2 Описание реализации шифрования с помощью решеток

Для реализации данного шифрования были описаны пять новых методов: `count_k(message)` для расчета размерности матрицы `k`, `init_list(k)` для создания первичной таблицы размерности `k`, `rotate_list(list)` для поворота таблицы, `init_big_table(list)` для инициализации большой таблицы и метод освобождения и заполнения свободных ячеек буквами сообщения `open_spaces(list, message)`.

4.1.3 Описание реализации шифрования с помощью таблицы

Виженера

Для реализации данного шифрования были описаны три метода: `create_table(alphabet)` для создания таблицы виженера, `make_password(message, password)` для формирования пароля длиной равной длине сообщения и `visioner(message, password, table)` для шифрования сообщения.

4.2 Листинг

Код приведенных ниже программ реализован на языке python.

4.2.1 Маршрутное шифрование

```
from _collections import OrderedDict
import random

def get_nm(message, password):
    n = len(password)
    result = len(message) / len(password)
    while True:
        if int(result) != result:
            message += random.choice('йцукенгшщзхъфыварлжэячсмитьбю')
            result = len(message) / len(password)
        else:
            break
    m = int(result)
    return n, m, message
```

```

def message_to_dict(message, m, n):
    message_dict = {}
    for i in range(n):
        temp_message = []
        for j in range(m):
            temp_message.append(message[i + j * n])
        message_dict[password[i]] = temp_message
    return message_dict

def sort_dict(dict):
    return OrderedDict(sorted(dict.items()))

def dict_to_string(dict):
    new_message = ''
    for keys in dict:
        for key in keys:
            new_message += "".join(dict[key])
    return new_message

if __name__ == '__main__':
    message = input("Введите сообщение: ").replace(" ", "")
    password = input("Введите пароль: ")
    n, m, message = get_nm(message, password)
    dict_message = message_to_dict(message, m, n)
    ordered_dict_message = sort_dict(dict_message)

```

```
print("Зашифрованное сообщение: ")
print(dict_to_string(ordered_dict_message))
```

4.2.2 Шифрование с помощью решеток

```
import random
import math
from _collections import OrderedDict

def count_k(message):
    k = int(math.ceil((math.sqrt(len(message)) / 2)))
    while True:
        if len(message) == ((2 * k) ** 2):
            break
        else:
            message += random.choice('йцукенгшщзхъфыварлжэячсмитьбю')
    return k, message

def init_list(k):
    list_to_init = []
    counter = 0
    for i in range(k):
        temp_list = []
        for j in range(k):
            counter += 1
            temp_list.append(counter)
        list_to_init.append(temp_list)
    return list_to_init
```

```

def rotate_list(list):
    new_list = []
    for i, row in enumerate(list):
        temp_row_list = []
        for j, col in enumerate(row):
            temp_row_list.append(list[len(list) - j - 1][i])
        new_list.append(temp_row_list)
    return new_list

```

```

def init_big_table(list):
    not_rotated_list = list
    result_list = []
    rotated_list1 = rotate_list(not_rotated_list)
    rotated_list2 = rotate_list(rotated_list1)
    rotated_list3 = rotate_list(rotated_list2)
    for ix, item in enumerate(rotated_list1):
        temp_row = not_rotated_list[ix] + rotated_list1[ix]
        result_list.append(temp_row)
    for ix, item in enumerate(rotated_list2):
        temp_row = rotated_list3[ix] + rotated_list2[ix]
        result_list.append(temp_row)
    return result_list

```

```

def open_spaces(list, message):
    message_letters_left = message

```

```

spaces = k ** 2
i = 1
while True:
    if i == spaces + 1:
        break
    rand_index_i = random.randint(0, (k * 2) - 1)
    rand_index_j = random.randint(0, (k * 2) - 1)
    if list[rand_index_i][rand_index_j] == i:
        list[rand_index_i][rand_index_j] = message_letters_left[0]
        message_letters_left = message_letters_left[1:]
        i += 1
return message_letters_left

```

```

def sorted_to_string(res, password):
    res_dict = dict(zip(password, res))
    print("Зашифрованное сообщение в виде словаря до сортировки: ")
    print(res_dict)
    sorted_dict = sort_dict(res_dict)
    print("Зашифрованное сообщение в виде словаря после сортировки: ")
    print(sorted_dict)
    string_message = dict_to_string(sorted_dict)
    return string_message

```

```

def sort_dict(dict):
    return OrderedDict(sorted(dict.items()))

```



```

def dict_to_string(dict):
    new_message = ''
    for keys in dict:
        for key in keys:
            new_message += "".join(dict[key])
    return new_message

if __name__ == '__main__':
    message = input("Введите сообщение: ").replace(' ', '')

    k, message = count_k(message)

    print("Сообщение с учетом добавления произвольных символов: ")
    print(message)

    inited = init_list(k)

    print("Исходная матрица: ")
    print(*inited, sep="\n")

    res = init_big_table(inited)
    print("Образованная большая таблица k*2: ")
    print(*res, sep="\n")
    sliced_message = open_spaces(res, message)

    res = rotate_list(res)
    sliced_message = open_spaces(res, sliced_message)

```

```

res = rotate_list(res)
sliced_message = open_spaces(res, sliced_message)

res = rotate_list(res)
sliced_message = open_spaces(res, sliced_message)
print("Зашифрованное сообщение в списковом представлении: ")
print(*res, sep="\n", end="\n\n")

password = input("Введите ключ (длина ключа = {}): ".format(len(res)))
result = sorted_to_string(res, password)
print("\n\nЗашифрованное сообщение: ")
print(result)

```

4.2.3 Таблица виженера

```

message = 'криптографиясерьезнаянаука'
password = 'математикаматематикаматема'
alphabet = 'абвгдежзийклмнопрстуфхцчщъыьэюя'

```

```

def create_table(alphabet):
    alphabet_list = []
    current_row = alphabet
    alphabet_list.append(current_row)
    for item in alphabet:
        current_row = current_row[1:] + current_row[0]
        alphabet_list.append(current_row)
    alphabet_list.pop()
    return alphabet_list

```

```

def visioner(message, password, table):
    indexes_i = []
    res_string = ''
    for ix, letter in enumerate(message):
        index_i = table[0].find(letter)
        indexes_i.append(index_i)
    indexes_j = []
    for ix, letter in enumerate(password):
        index_j = table[0].find(letter)
        indexes_j.append(index_j)
    for i, row in enumerate(indexes_i):
        res_string += table[indexes_i[i]][indexes_j[i]]
    return res_string

def make_password(message, password):
    new_password = ''
    for ix, item in enumerate(message):
        new_password += password[ix % len(password)]
    print(new_password)
    return new_password

if __name__ == '__main__':
    message = input("Введите сообщение: ").replace(' ', '')
    print("Форматированное сообщение: ")
    print(message)
    print()
    password = input("Введите пароль (не превышающий длину сообщения): ")
    print("Дополненный ключ до длины сообщения: ")

```

```

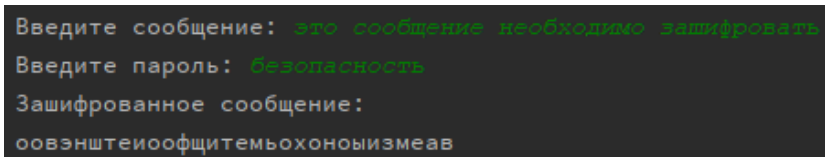
password = make_password(message, password)
table = create_table(alphabet)
print("\nТаблица: ")
print(*table[0:3], sep='\n')
print("...")
print("...")
print(*table[-3:], sep='\n')
result = visioner(message, password, table)
print("\nЗашифрованное сообщение: ")
print(result)

```

4.3 Полученные результаты

4.3.1 Маршрутное шифрование

При запуске программы пользователю предлагается ввести сообщение, которое необходимо зашифровать, и ключ. В результате выполнения пользователь получает зашифрованное сообщение (рис. 4.1).



```

Введите сообщение: это сообщение необходимо зашифровать
Введите пароль: безопасность
Зашифрованное сообщение:
оовэнштейноофщителъохоноыизмеав

```

Figure 4.1: Результаты маршрутного шифрования

4.3.2 Шифрование с помощью решеток

Пользователь вводит сообщение, которое необходимо зашифровать. В конец сообщения добавляются произвольные символы (если это необходимо), затем выводится матрица размерности k . Полученная матрица поворачивается на 90°

и присоединяется к исходной справа. Данная операция повторяется дважды, и полученные матрицы приписываются снизу. Пользователю показывается полученная большая матрица размерности $2k$. Затем выводится зашифрованное сообщение в списковом представлении. После этого пользователю предлагается ввести ключ. Выводится зашифрованное сообщение в виде словаря до сортировки, затем происходит сортировка и выводятся результаты сортировки. В итоге пользователь получает зашифрованное сообщение (рис. 4.2).

```

Введите сообщение: информационная безопасность
Сообщение с учетом добавления произвольных символов:
информационнаябезопасностькххбх
Исходная матрица:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Образованная большая таблица k*2:
[1, 2, 3, 7, 4, 1]
[4, 5, 6, 8, 5, 2]
[7, 8, 9, 9, 6, 3]
[3, 6, 9, 9, 8, 7]
[2, 5, 8, 6, 5, 4]
[1, 4, 7, 3, 2, 1]
Зашифрованное сообщение в списковом представлении:
['o', 'k', 't', 'a', 'n', 'i']
['a', 'p', 'c', 'b', 'j', 'n']
['x', 'c', 'b', 'o', 'e', 'f']
['c', 'm', 't', 'i', 'z', 't']
['a', 'o', 'b', 'b', 'l', 'k']
['n', 'o', 'e', 'n', 'n', 'y']

Введите ключ (длина ключа = 6): кххбх
Зашифрованное сообщение в виде словаря до сортировки:
{'t': ['o', 'k', 't', 'a', 'n', 'i'], 'e': ['a', 'p', 'c', 'b', 'j', 'n'], 'p': ['x', 'c', 'b', 'o', 'e', 'f']}
Зашифрованное сообщение в виде словаря после сортировки:
OrderedDict([('e', ['a', 'p', 'c', 'b', 'j', 'n']), ('m', ['c', 'm', 't', 'i', 'z', 't'])])

Зашифрованное сообщение:
арсбянсмитизтаоьблххцьефпоеннхктани

```

Figure 4.2: Результаты шифрования с помощью решеток

4.3.3 Таблица Виженера

При запуске программы пользователю предлагается ввести сообщение, которое необходимо зашифровать. Из сообщения удаляются пробелы и выводится форматированное сообщение. Затем пользователь вводит ключ, который не должен превышать длину самого сообщения. Ключ дополняется до длины сообщения и демонстрируется пользователю. После этого строится таблица Виженера. В ито-

те пользователь получается зашифрованное сообщение (на рис. 4.3).

```
Введите сообщение: шифрование это безопасно
Форматированное сообщение:
шифрованиеэтобезопасно

Введите пароль (не превышающий длину сообщения): корм
Дополненный ключ до длины сообщения:
кормкормкормкормкормко

Таблица:
абгдезийклмнопрстуфхцщъыьэюя
бвгдезийклмнопрстуфхцщъыьэюяа
вгдезийклмнопрстуфхцщъыьэюяаб
...
...
эюяабгдезийклмнопрстуфхцщъыь
юяабгдезийклмнопрстуфхцщъыьэ
яабгдезийклмнопрстуфхцщъыьэю

Зашифрованное сообщение:
вцдъшррцтунюшпхушэрэчь
```

Figure 4.3: Результаты шифрования с помощью таблица Виженера

5 Выводы

В ходе выполнения данной лабораторной работы было выполнено ознакомление с шифрами перестановки на примере маршрутного шифрования, шифрования с помощью решетов и таблицы Виженера.

В результате проделанной работы были программно реализованы эти методы шифрования.

Как итог, поставленные цели и задачи были успешно достигнуты.

Список литературы

1. Шифр табличной маршрутной перестановки [Электронный ресурс]. Кенвуд, 2020. URL: <https://kenwood-bt.ru/info/shifr-tablichnoj-marshrutnoj-perestanolvki/>.
2. Классические шифры перестановки [Электронный ресурс]. Studme, 2021. URL: https://studme.org/239548/informatika/klassicheskie_shifry_perestanolvki.
3. Шифр Виженера [Электронный ресурс]. Википедия, 2021. URL: https://ru.wikipedia.org/wiki/Шифр_Виженера.