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 1](#_Toc88324945)

[2 Задание 2](#_Toc88324946)

[3 Теоретическое введение 2](#_Toc88324947)

[3.1 Маршрутное шифрование 2](#_Toc88324948)

[3.2 Шифрование с помомщью решеток 2](#_Toc88324949)

[3.3 Таблица Виженера 3](#_Toc88324950)

[4 Выполнение лабораторной работы 4](#_Toc88324951)

[4.1 Описание реализации методов 4](#_Toc88324952)

[4.1.1 Описание реализации маршрутного шифрования 4](#_Toc88324953)

[4.1.2 Описание реализации шифрования с помощью решеток 4](#_Toc88324954)

[4.1.3 Описание реализации шифрования с помощью таблицы Виженера 5](#_Toc88324955)

[4.2 Листинг 5](#_Toc88324956)

[4.2.1 Маршрутное шифрование 5](#_Toc88324957)

[4.2.2 Шифрование с помощью решеток 6](#_Toc88324958)

[4.2.3 Таблица виженера 8](#_Toc88324959)

[4.3 Полученные результаты 9](#_Toc88324960)

[4.3.1 Маршрутное шифрование 9](#_Toc88324961)

[4.3.2 Шифрование с помощью решеток 10](#_Toc88324962)

[4.3.3 Таблица Виженера 11](#_Toc88324963)

[5 Выводы 12](#_Toc88324964)

[Список литературы 12](#_Toc88324965)

# 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 определенному 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.



Figure 1: Маршрутное шифрование

В результате отработки алгоритма возвращаются отсортированные столбцы таблицы по алфавитному порядку букв ключа. На рис. 1 ключом является пароль, соответственно в результирующее сообщение сначала записывается столбец под буквой a ключа, и заканчивается столбцом под ь.

## 3.2 Шифрование с помомщью решеток

Поворотная решетка — это прямоугольная или квадратная карточка с четным числом строк и столбцов 2k X 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 2изображена последовательность поворота решетки для заполнения её буквами сообщения:

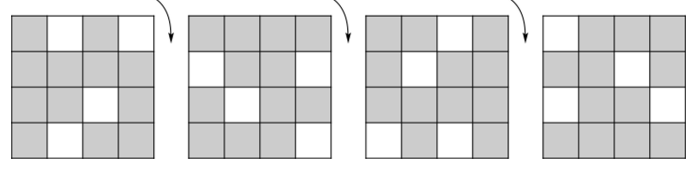


Figure 2: Шифрование с помощью решеток

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:

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

Если — количество букв в алфавите, — номер буквы открытого текста, — номер буквы ключа в алфавите, то шифрование Виженера можно записать следующим образом:

Пример таблицы виженера для латинского алфавита изображен на рис. 3:

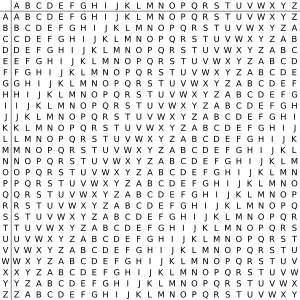


Figure 3: Таблица Виженера

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 собщение 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 = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'  
  
  
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("Дополненый 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).

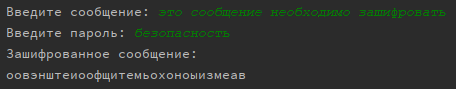


Figure 4: Результаты маршрутного шифрования

### 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 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. 5).

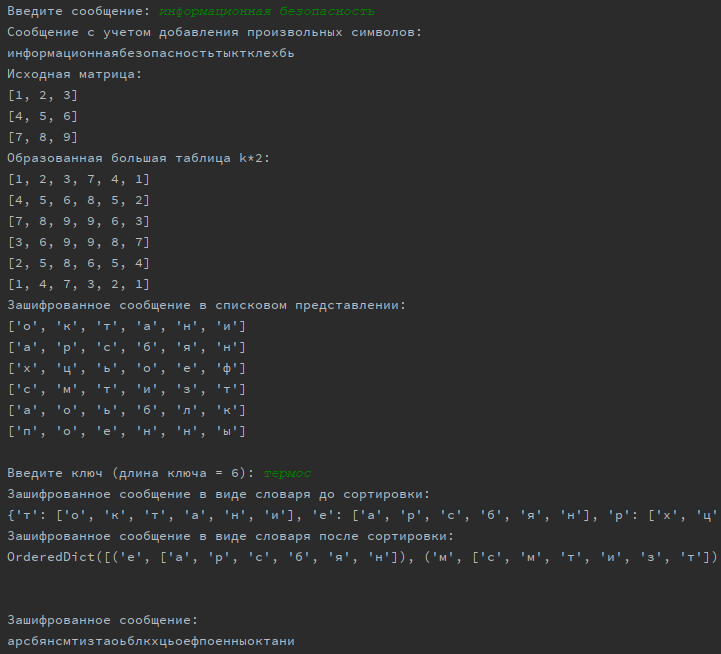


Figure 5: Результаты шифрования с помощью решеток

### 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 дополянется до длины сообщения и демонстрируется пользователю. После этого строится таблица Виженера. В итоге пользователь получается зашифрованное сообщение (на рис. 6).

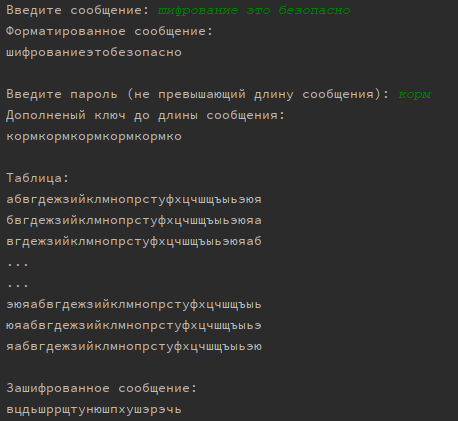


Figure 6: Результаты шифрования с помощью таблица Виженера

# 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-perestanovki/>.

2. Классические шифры перестановки [Электронный ресурс]. Studme, 2021. URL: <https://studme.org/239548/informatika/klassicheskie_shifry_perestanovki>.

3. Шифр Виженера [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Шифр_Виженера>.