Лабораторная работа №2

Шифры перестановки

Хохлачева Яна Дмитриевна, НПМмд-02-22

Содержание

# 1 Цель работы

Знакомство с шифрами перестановки: маршрутным шифрование, шифрованием с помощью решеток, шифрованием при помощи таблицы Виженера.

# 2 Задание

1. Релизовать маршрутное шифрование.
2. Реализовать шифрование с помощью решеток.
3. Реализовать шифрование при помощи таблицы Виженера.

# 3 Теоретическое введение

Шифры перестановки преобразуют открытый текст в криптограмму путем перестановки его символов. Способ, каким при шифровании переставляются буквы открытого текста, и является ключом шифра. Важным требованием является равенство длин ключа и исходного текста [1]. В классической криптографии шифры перестановки можно разделить на два класса:

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

## 3.1 Маршрутное шифрование

Широкое распространение получили так называемые маршрутные перестановки, использующие некоторую геометрическую фигуру (плоскую или объемную). Преобразования состоят в том, что отрезок открытого текста записывается в такую фигуру по некоторой траектории, а выписывается по другой траектории.

Например, для шифрования текста *нельзя недооценивать противника*, разобьем его на блоки . Блоков получится . К последнему блоку припишем букву . В качестве пароля выберем слово *пароль*. Теперь выпишем буквы по столбцам в соответствии с алфавитным порядком букв пароля и получим следующую криптограмму: *ЕЕНПНЗОАТАЬОВОКННЕЬВЯЦТИА*.

## 3.2 Шифрование с помощью решеток

Данный способ шифрования предложил австрийский крипатограф Эдуард Флейснер в 1881 году. Суть этого способа заключается в следующем. Выбирается натуральное число , строится квадрат размерности и построчно заполняется числами . В качестве примера рассмотрим квадрат размерности .

Повернем его по часовой стрелке на 90 градусов и присоединим к исходному квадрату справа.

Проделаем еще дважды такую процедуру и получим квадрат размерности .

Далее из большого квадрата вырезаются клетки с числами от до , для каждого числа одна клетка. Процесс шифрования происходит следующим образом. Сделанная решетка (квадрат с прорезями) накладывается на чистый квадрат и в прорези по строчкам (т.е. слева направо и сверху вниз) вписываются первые буквы открытого текста. Затем решетка поворачивается на девяносто градусов по часовой стрелке и накладывается на частично заполненный квадрат, вписывание продолжается. После третьего поворота, наложения и вписывания все клетки квадрата будут заполнены. Правило выбора прорезей гарантирует, что при заполнении квадрата буква на букву никогда не попадет. Из заполненного квадрата буквы можно выписать по столбцам, выбрав подходящий пароль [2].

## 3.3 Таблица Виженера

Шифр Виженера — метод полиалфавитного шифрования буквенного текста с использованием ключевого слова.

Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джовани Баттиста Белласо в книге *La cifra del. Sig. Giovan Battista Bellasо* в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, но является недоступным для простых методов криптоанализа [3].

# 4 Выполнение лабораторной работы

## 4.1 Структура программной реализации

### 4.1.1 Маршрутное шифрование

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 Шифрование 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 Шифрование 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 shifr(message, password, table) для шифрования сообщения.

## 4.2 Листинг

### 4.2.1 Реализация маршрутного шифрования.

from \_collections import OrderedDict  
  
def get\_nm(message, password):  
 n = len(password)  
 result = len(message) / len(password)  
 while True:  
 if int(result) != result:  
 message += 'a'  
 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  
  
  
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 Реализация шифрования при помощи решеток.

from \_collections import OrderedDict  
import math  
import random  
  
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('йцукенгшщзхъфывапролджэячсмитьбю')  
 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  
  
  
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 Реализация шифра Виженера.

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  
  
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  
   
  
message = input("Введите сообщение: ").replace(' ', '')  
print("Форматированное сообщение: ")  
print(message)  
print()  
password = input("Введите пароль (не превышающий длину сообщения): ")  
print("Дополненный пароль 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 = shifr(message, password, table)  
print("\nЗашифрованное сообщение: ")  
print(result)

## 4.3 Полученные результаты

Результаты работы программы, реализующей маршрутное шифрование показаны на рис. 1.

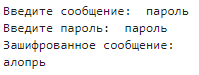


Рис. 1: Маршрутное шифрование

Результаты работы программы, реализующей шифрование при помощи решеток показаны на рис. 2.

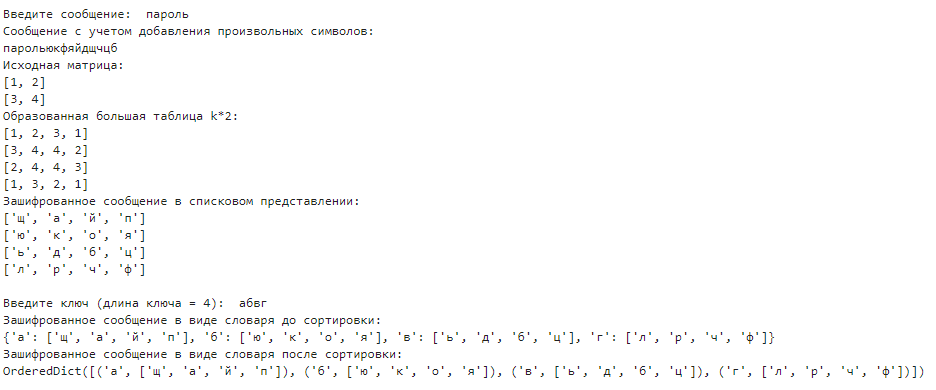


Рис. 2: Шифрование решетками

Результаты работы программы, реализующей шифрование методом Виженера показаны на рис. 3.

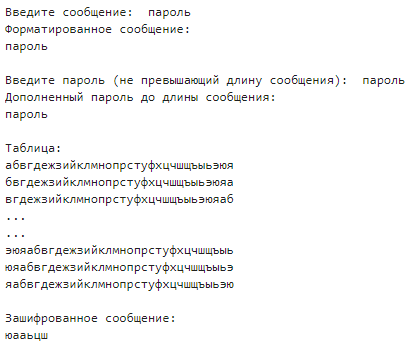


Рис. 3: Шифр Виженера

Для выхода из программы нужно ввести “0”.

# 5 Выводы

Таким образом в процессе лабораторной работы я изучила и реализовала следующие методы шифрования:

* Маршрутное шифрование.
* Шифрование с помощью решеток.
* Шифрование при помощи таблицы Виженера.

# Список литературы

1. Перестановочный шифр [Электронный ресурс]. Википедия, 2022. URL: <https://ru.wikipedia.org/wiki/Перестановочный_шифр>.

2. Шифровальная решетка [Электронный ресурс]. Википедия, 2022. URL: <https://ru.wikipedia.org/wiki/Шифровальная_решетка>.

3. Шифр Виженера [Электронный ресурс]. Википедия, 2022. URL: <https://ru.wikipedia.org/wiki/Шифр_Виженера>.