Лабораторная работа №2

Математические основы защиты информации и информационной безопасности

Леонтьева Ксения Андреевна | НПМмд-02-23

Содержание

Список иллюстраций

1 Цель работы

Реализовать на языке программирования маршрутное шифрование, шифрование с помощью решеток и таблицу Виженера.

2 Теоретическое введение

Шифры перестановки преобразуют открытый текст в криптограмму путем перестановки его символов.

Маршрутное шифрование разработал французский математик Франсуа Виет. Открытый текст записывают в некоторую геометрическую фигуру, например, прямоугольник, разбив предварительно текст на блоки, длина которых равна количеству букв в пароле (при необходимости дописывают произвольные буквы для достижения нужного количества). Блоки располагаются друг под другом. Затем выписывают столбцы в порядке следования букв в пароле по алфавиту (пароль записывается под прямоугольником).

Более подробно см. в [1].

Шифрование с помощью решеток предложил австрийский криптограф Эдуард Флейснер в 1881 году. Формируется решетка, заполненная цифрами от 1 до 4, из которой удаляются некоторые ячейки так, чтобы при последовательных поворотах этой решетки на 90 градусов и записи букв в удаленные ячейки буквы не накладывались друг на друга и можно было записать весь текст (в идеальном случае $k^2 = N$, где k^2 - длина стороны квадрата решетки, а N - количество букв в исходном тексте). Затем полученная решетка накладывается на аналогичную, но пустую, и, когда заполняются все прорези буквами исходного текста по порядку их следования, решетка поворачивается на 90 градусов и вписывание букв продолжается. Далее подбирается подходящий пароль (число букв пароля должно равняться k^2 и они не должны повторяться), выписываются буквы по столбцами. Очередность столбцов определяется алфавитным порядком букв пароля как в

маршрутном шифровании.

Более подробно см. в [2].

Шифр Виженера опубликовал в 1585 году французский криптограф Блез Виженер в “Трактате о шифрах”. Он считался нераскрываемым до 1863 года, когда австриец Фридрих Казиски взломал его. Опишем одну из схем построения данного шифра. Формируется таблица, где в строчках записаны буквы русского алфавита (рис. 2.1). При переходе от одной строке к другой происходит циклический сдвиг на одну позицию. Пароль же записывается с повторениями над буквами сообщения. Далее в горизонтальном алфавите находим букву из исходного текста, в вертикальном - из пароля. На пересечении столбца и строки в таблице располагается нужная буква. Продолжаем так с остальными буквами.

Более подробно см. в [3].

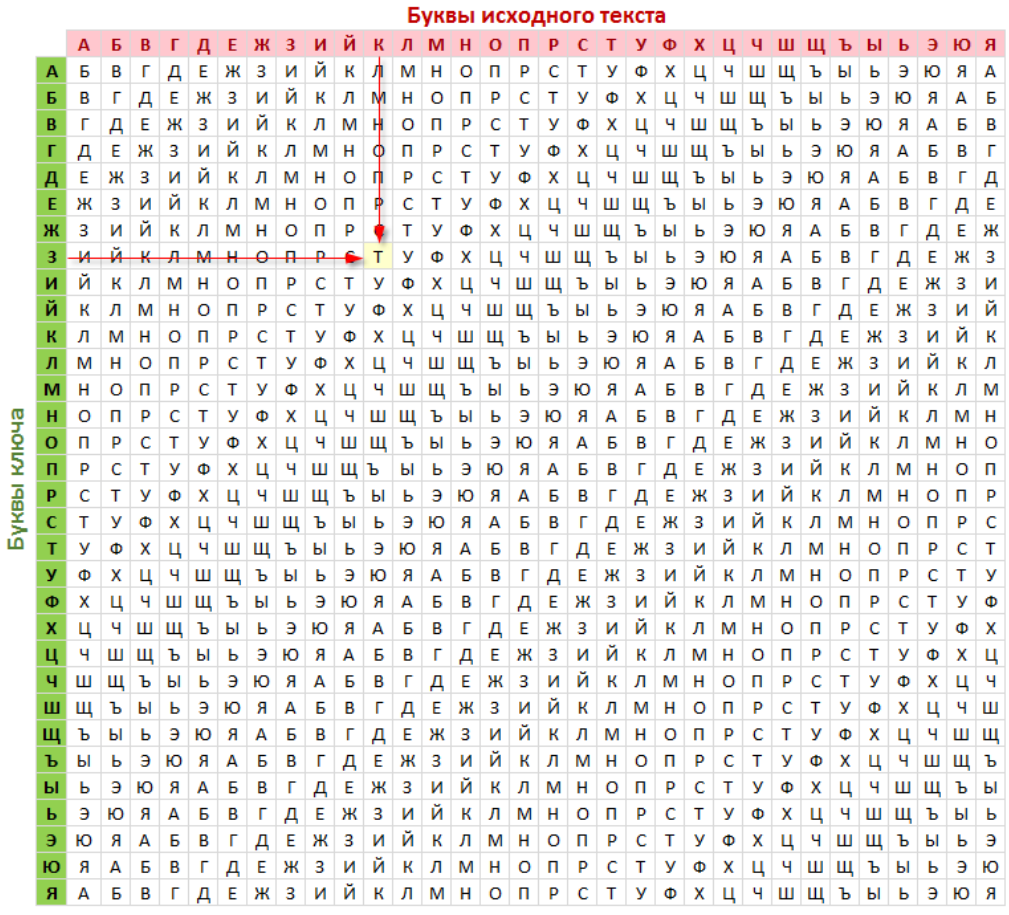

Рис. 2.1: Таблица Виженера для русского алфавита

3 Выполнение лабораторной работы

Начнем с реализации маршрутного шифрования. Переменные phrase и key содержат открытый текст и пароль соответственно. Проверим необходимость добавления в открытый текст дополнительных букв, чтобы текст можно было разбить на равные блоки. Если остаток от деления количества символов в открытом тексте на количество символов в пароле меньше количества символов в пароле, то добавляем в конец текста нужное количество букв. Список block представляет собой список, состоящий из блоков, на которые мы разбили наш открытый текст. Размер каждого блока соответствует количеству символов в пароле.

В список alphabet внесем коды ASCII для букв в пароле. Затем в список new_alphabet внесем индексы букв из пароля в алфавитном порядке. Список word представляет собой зашифрованную последовательность: сначала выбираем индекс буквы в подсписке blocks в соответствии с алфавитным порядком букв в пароле, а затем выписываем буквы с таким индексом последовательно из каждого подсписка. Наконец, выводим полученную последовательность.

Код программы (рис. 3.1).

```
phrase = 'нельзя недооценивать противника'
key = 'пароль'
phrase = list(phrase.replace(" ", ""))
key = list(key)
```

```
m = len(phrase)
n = len(key)
l = m % n

if l < n:
    for i in range(n-l):
        phrase.append('а')

blocks = [phrase[i:i+n] for i in range(0, len(phrase), n)]
blocks
```

```
[['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']]
```

```
alphabet = []

for j in range(n):
    alphabet.append(ord(key[j]))

new_alphabet = sorted(range(len(alphabet)), key=alphabet.__getitem__)
```

```
word = []

for g in range(n):
    for h in range(int(len(phrase)/len(key))):
        word.append(blocks[h][new_alphabet[g]])
```

```
print(''.join(word))
```

```
еенпнзоатаьовокннеьвлдирияцтиа
```

Рис. 3.1: Маршрутное шифрование

В шифровании с помощью решеток начнем с создания решетки, которая впоследствии будет накладываться на пустую для заполнения. Создаем начальный массив a_1 и поворачиваем его три раза подряд на 90 градусов. Затем объединяем соответствующие полученные массивы, чтобы получить решетку 4x4. Далее запоминаем в переменные i_1, i_2, i_3, i_4 индексы цифр 1, 2, 3, 4 и, выбрав, для

каждой цифры произвольный индекс, заменяем ее на -1 (“удаляем”). Оставшиеся цифры для удобства заменяем на 0. Делим полученный массив на блоки по 4 цифры в каждом для дальнейшей работы.

Переменная m содержит полученный разделенный на блоки массив, переменная k - пустой список, куда будут записываться буквы. В phrase и key записываем исходный текст и пароль. В indices сохраняем индексы из m, на месте которых стоят -1. Пока список k не будет заполнен полностью повторяем алгоритм: сохраняем текущие индексы с -1 в indices, далее на место этих -1 записываем последовательно буквы из phrase, удаляя после записи каждой буквы ее из phrase, поворачиваем решетку m на 90 градусов, список indices делаем пустым для дальнейшего заполнения. Выводим итоговую табличку k.

Далее код представляет собой аналогичный маршрутному шифрованию и вывод зашифрованного текста.

Код программы (рис. 3.2 и рис. 3.3).

```
import numpy as np
```

```
a_1 = np.array([[1,2],[3,4]])
a_2 = np.rot90(a_1, 3)
a_3 = np.rot90(a_2, 6)
a_4 = np.rot90(a_3, 1)
a_12 = np.concatenate ((a_1, a_2), axis = 1 )
a_34 = np.concatenate ((a_3, a_4), axis = 1 )
a = np.concatenate ((a_12, a_34), axis = 0 )
aa = np.concatenate((a[0], a[1], a[2], a[3]), axis = 0 )

for i in range(4):
    exec(f"i{i+1} = [j for j in range(0, len(aa)) if aa[j]==i+1]")
    exec(f"ind{i+1} = np.random.randint(0,4)")
    exec(f"aa[i{i+1}[ind{i+1}]] = -1")

for i in range(len(aa)):
    if aa[i] != -1:
        aa[i] = 0
aaa = aa.tolist()
```

```
m = [aaa[i:i+4] for i in range(0, len(aa), 4)]
k = [[0,0,0,0],[0,0,0,0], [0,0,0,0], [0,0,0,0]]
phrase = list('договорподписали')
key = list('шифр')

indices = []

m
```

```
[[-1, 0, 0, 0], [-1, 0, 0, -1], [0, -1, 0, 0], [0, 0, 0, 0]]
```

Рис. 3.2: Шифрование с помощью решеток

```
while any(0 in s for s in k) == True:
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] == -1:
                indices.append([i, j])

    for i in range(4):
        k[indices[i][0]][indices[i][1]] = phrase[0]
        phrase = phrase[1:]

    m = np.rot90(m, 3)

    indices = []
```

```
k
```

```
[['д', 'с', 'в', 'о'],
 ['о', 'р', 'о', 'г'],
 ['д', 'о', 'а', 'п'],
 ['л', 'и', 'п', 'и']]
```

```
alph = []

for j in range(len(key)):
    alph.append(ord(key[j]))

new_alph = sorted(range(len(alph)), key=alph.__getitem__)
```

```
word = []

for g in range(len(key)):
    for h in range(len(m)):
        word.append(k[h][new_alph[g]])
```

```
print(''.join(word))
```

```
сроиогпивоапдодл
```

Рис. 3.3: Шифрование с помощью решеток

В шифре Виженера аналогично в переменные phrase и key записываем исходную последовательность и пароль. В alphabet - русский алфавит без буквы “ё”. Создаем таблицу (table) из повторения алфавита со сдвигом на одну позицию влево, так называемую таблицу Виженера. В k вычисляем количество повторений

пароля целиком и записываем его k раз в key_list. На оставшиеся свободные места (длина key_list равна длине исходной фразы) дописываем только часть пароля. Далее печатаем зашифрованную последовательность следующим образом: из таблицы выбираем строку с буквой из исходной фразы phrase и столбец с буквой из key_list. С помощью цикла последовательно проходимся по соответствующим спискам. Выводим полученную последовательность.

Код программы (рис. 3.4).

```
import math
```

```
phrase = 'криптография серьезная наука'
key = 'математика'
phrase = list(phrase.replace(" ", ""))
key = list(key)
```

```
alphabet = []
for i in range(1072,1104):
    alphabet.append(chr(i))

table = list()
for i in range(len(alphabet)):
    table.append(alphabet[i:] + alphabet[:i])
```

```
k = math.floor(len(phrase)/len(key))
key_list = []
for j in range(k):
    key_list.append(key)
m = len(phrase) % len(key)
part_key = key[:m]
key_list.append(part_key)
key_list = sum(key_list, [])
```

```
cipher = []
for g in range(len(phrase)):
    cipher.append( table[alphabet.index(phrase[g])][alphabet.index(key_list[g])] )
```

```
print(''.join(cipher))
```

```
цръфюохшкффягкььчпчалнтшца
```

Рис. 3.4: Таблица Виженера

4 Выводы

В ходе выполнения данной лабораторной работы были реализованы маршрутное шифрование, шифрование с помощью решеток и таблица Виженера.

Список литературы

1. Маршрутное шифрование [Электронный ресурс]. 2021. URL: https://teletype.in/@hackersacademy/-YilCBTU7Mk.
2. Шифрование с помощью решеток [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D1%88%D1%91%D1%82%D0%BA%D0%B0.
3. Таблица Виженера [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%92%D0%B8%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B0.