Отчет по лабораторной работе №2

Дисциплина: Математические основы защиты информации и информационной безопасности

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

1 Цель работы **4**

2 Задание **5**

3 Теоретическое введение **6**

4 Выполнение лабораторной работы **8**
4.1 Шаг 1 8
4.2 Шаг 2 10
4.3 Шаг 3 12

5 Выводы **14**

Список иллюстраций

4.1 Реализация маршрутного шифрования 9
4.2 Работа функции, выполняющей маршрутное шифрование 9
4.3 Функция поворота 10
4.4 Реализация шифрования с помощью решеток 11
4.5 Работа функции, выполняющей шифрование с помощью решеток 11
4.6 Функция создания таблицы с алфавитом 12
4.7 Реализация шифра Виженера и результат 13

1 Цель работы

Ознакомится и реализовать шифры перестановки.

2 Задание

1. Реализовать маршрутное шифрование.
2. Реализовать шифрование с помощью решеток.
3. Реализовать шифр 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.

Маршрутное шифрование разработал французский математик Франсуа Виет. Открытый текст записывают в некоторую геометрическую фигуру (обычно прямоугольник) по некоторому пути, а затем, выписывая символы по другому пути, получают шифртекст. Пусть *m* и *n* – целые положительные числа, большие 1. Открытый текст разбивается на блоки равной длины, состоящие из числа символов, равному произведению *mn*. Если последний блок получится меньше остальных, то в него следует дописать требуемое количество произвольных символов. Составляется таблица размерности *mn*. Блоки вписывается построчно в таблицу. Криптограмма получается выписыванием букв из таблицы в соответствии с некоторым маршрутом. Ключом такой криптограммы является маршрут и числа *m* и *n*. Обычно буквы выписывают по столбцам, которые упорядочивают согласно паролю: внизу таблицы приписывается слово из *n* неповторяющихся букв и столбцы нумеруются по алфавитному порядку букв пароля.

Шифрование с помощью решеток предложил австрийский криптограф Эдуард Флейснер в 1881 году. Суть этого способа заключается в следующем. Выбирается натуральное число k > 1, строится квадрат размерности k и построчно заполняется числами 1, 2, …, k^2. Этот квадрат поворачивается по часовой стрелке на 90° и присоединяется к исходному квадрату справа. Данная процедура проделывается еще дважды, получившиеся квадраты приписываются снизу. Получается квадрат размерности 2k. Далее из большого квадрата

вырезаются клетки, содержащие числа от 1 до k^2. В каждой клетке должно быть только одно число. Получается своего рода решето. Шифрование осуществляется следующим образом. Решето накладывается на чистый квадрат 2k × 2k и в прорези вписываются буквы исходного текста по порядку их следования. Когда заполнятся все прорези, решето поворачивается на 90° и вписывание букв продолжается. После третьего поворота все клетки большого квадрата окажутся заполненными. Подобрав подходящий пароль (число букв пароля должно равняться k^2 и они не должны повторяться), выписываются буквы по столбцам. Очередность столбцов определяется алфавитным порядком букв пароля.

В 1585 году французский криптограф Блез Виженер опубликовал свой метод шифрования в «Трактате о шифрах». Шифр считался нераскрываемым до 1863 года, когда австриец Фридрих Казиски взломал его. Открытый текст разбивается на блоки длины n. Ключ представляет собой последовательность из n натуральных чисел: $a_1, a_2, a_3, ..., a_n$. Далее в каждом блоке первая буква циклически сдвигается вправо по алфавиту на a_i позиций, вторая буква – на a_2 позиций, последняя – на a_n позиций. Для лучшего запоминания в качестве ключа можно взять осмысленное слово, а алфавитные номера входящих в него букв использовать для осуществления сдвигов.

4 Выполнение лабораторной работы

4.1 Шаг 1

Ознакомилась с предоставленными теоретическими данными. Для выполнения задания решила использовать язык Python. Подключила библиотеку numpy. В качестве сообщения выбрала фразу “осенняя депрессия”. Написала функцию, выполняющую маршрутное шифрование. Код функции и результат ее использования представлен на Рисунке 1 (рис. - fig. 4.1). Функция принимает на вход фразу, которую нужно зашифровать, размерности *n* и *m* и пароль. Вначале если длина сообщения не кратна размерности *n*, в конец сообщения дописывается необходимое количество символов *a*. Затем сообщение переформировывается в матрицу размером $m \times n$. Если длина пароля соответствует размерности *n*, то буквы пароля сортируются по алфавиту и этот порядок запоминается в переменной *b*. Затем для каждого символа из *b* в переменную *n_col* запоминается его порядковый номер в пароле. В соответствии с этими номерами в результирующую криптограмму выписываются поочередно столбцы матрицы.

```
In [2]: import numpy as np

In [3]: message="осенняя депрессия"

In [4]: def sh1 (message, n, m, password):
            shifr=''
            a=len(message)%n
            if (a != 0):
                for i in range(n-a):
                    message=message+'a'
            message1=list(message)
            message1 = np.reshape(message1, (-1, n))
            if (len(password)==n):
                b=sorted(password)
                n_col=np.empty(n)
                j=0
                for i in b:
                    n_col[j]=password.index(i)
                    j+=1
                for i in n_col:
                    g=message1[:,int(i)]
                    d=''.join(g)
                    shifr=shifr+d
                print(shifr)
                print(message1)
                print(n_col)
            else:
                print ("Error")
```

Рис. 4.1: Реализация маршрутного шифрования

На Рисунке 2 (рис. - fig. 4.2) показан результат работы функции.

```
In [5]: sh1(message, 4, 4, 'поле')

н риаеяпсасяесаондея
[['о' 'с' 'е' 'н']
 ['н' 'я' 'я' ' ']
 ['д' 'е' 'п' 'р']
 ['е' 'с' 'с' 'и']
 ['я' 'а' 'а' 'а']]
[3. 2. 1. 0.]
```

Рис. 4.2: Работа функции, выполняющей маршрутное шифрование

4.2 Шаг 2

На Рисунке 3 (рис. - fig. 4.3) представлена реализация функции поворота матрицы на 90° для осуществления шифрования с помощью решеток.

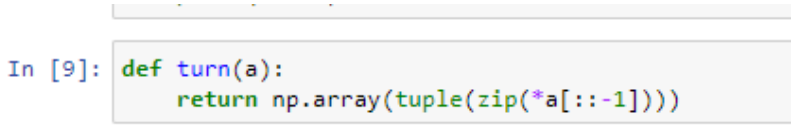

Рис. 4.3: Функция поворота

Функция для шифрования с помощью решеток показана на Рисунке 4 (рис. - fig. 4.4). Функция принимает на вход сообщение и пароль. Размерность k считается равной 2. Создается массив из чисел от 1 до k^2, затем он переформируется в квадрат со стороной k. Последовательно применяя функцию поворота и соединение массивов, получается квадрат размером 2k × 2k. Определенные числа в получившейся матрицы заменила нулями. В цикле, который пройдет 4 раза, значения матрицы *res* проверяются на равенство 0 и в случае, если это так, в матрицу *krypt* в этой позиции записывается буква сообщения; после решетка *res* поворачивается и начинается новая итерация. Затем, аналогично реализации маршрутного шифрования, получившаяся матрица сопоставляется с порядком букв в пароле и выписывается получившаяся криптограмма. Результат показан на Рисунке 5 (рис. - fig. 4.5).

```
In [31]: def sh2 (message, password):
             k=2
             shifr=''
             message1=list(message)
             a=[i for i in range(1,k*k+1)]
             a = np.reshape(a, (-1, k))
             b=turn(a)
             res1=np.concatenate((a, b), axis=1)
             c=turn(turn(b))
             d=turn(b)
             res2=np.concatenate((c, d), axis=1)
             res=np.concatenate((res1, res2), axis=0)
             res[0][3]=0
             res[2][1]=0
             res[2][3]=0
             res[3][2]=0
             krypt=np.full((2*k, 2*k), 'a')
             n=0
             l=0
             while l<4:
                 for i in range(0,2*k):
                     for j in range(0, 2*k):
                         if (res[i][j]==0):
                             krypt[i][j]=message1[n]
                             n=n+1
                         j=j+1
                     i=i+1
                 res=turn(res)
                 l=l+1
             b=sorted(password)
             n_col=np.empty(len(password))
             j=0
             for i in b:
                 n_col[j]=password.index(i)
                 j+=1
             for i in n_col:
                 g=krypt[:,int(i)]
                 d=''.join(g)
                 shifr=shifr+d
             print(res)
             print(krypt)
             print(n_col)
             print(shifr)
```

Рис. 4.4: Реализация шифрования с помощью решеток

```
In [32]: sh2("договорподписали", "шифр")
[[1 2 3 0]
 [3 4 4 2]
 [2 0 4 0]
 [1 3 0 1]]
[['с' 'о' 'а' 'д']
 ['д' 'в' 'п' 'л']
 ['о' 'о' 'и' 'г']
 ['и' 'р' 'о' 'п']]
[1. 3. 2. 0.]
оворддлгпапиосдои
```

Рис. 4.5: Работа функции, выполняющей шифрование с помощью решеток

4.3 Шаг 3

Для реализации шифра Виженера создала переменную, содержащую русский алфавит, и написала вспомогающую функцию, создающую матрицу, где строки - русский алфавит, где все буквы сдвинуты на i, где i - номер строки (рис. - fig. 4.6).

```
In [33]: alphabet="абвгдежзийклмнопрстуфхцчшщьыэюя"

In [55]: def table(alphabet):
             table1=list(alphabet)
             temp=table1
             for i in range(len(alphabet)):
                 temp=temp[1:]+temp[:1]
                 table1=np.concatenate((table1, temp), axis=0)
             table1=np.reshape(table1, (-1, (len(alphabet))))
             return table1
```

Рис. 4.6: Функция создания таблицы с алфавитом

Сама функция реализации принимает на вход сообщение, алфавит и пароль (рис. - fig. 4.7). Если длина сообщения больше длины пароля, то пароль увеличивается с помощью последовательного повторения его букв до длины сообщения. Затем производится поиск номера буквы сообщения в алфавите и номера буквы в пароле в алфавите. В результат криптограммы добавляется буква, находящаяся в созданной таблице на месте, соответствующем полученным номерам. Результат работы функции представлен на Рисунке 7 (рис. - fig. 4.7).

```
In [56]: def sh3(message, alphabet, password):
             shifr=''
             a=len(message)
             pw=password
             l_p=len(password)
             i=0
             if (a>l_p):
                 while a!=l_p:
                     pw=pw+pw[i]
                     i=i+1
                     l_p=l_p+1
             table2=table(alphabet)
             for k in range(a):
                 i=alphabet.find(message[k])
                 j=alphabet.find(pw[k])
                 shifr=shifr+table2[i][j]
             return shifr

In [57]: sh3("криптографиясерьезнаянаука", alphabet, "математика")

Out[57]: 'црьфяохшкффядкэьчпчалнтшца'
```

Рис. 4.7: Реализация шифра Виженера и результат

5 Выводы

Я ознакомилась с тремя типами шифров перестановки и реализовала их. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.