

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе 2

*Дисциплина: Математические основы защиты
информации и информационной безопасности*

Студент: Румянцева Александра Сергеевна, 1132223493

Группа: НПМмд-02-22

Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2022

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Маршрутное шифрование	9
4.2	Метод решеток	10
4.3	Метод Виженера	13
5	Библиография	14
6	Выводы	15

List of Figures

4.1	Рис. 1. 1 часть программного кода реализации маршрутного шифрования.	9
4.2	Рис. 2. 2 часть программного кода реализации маршрутного шифрования.	10
4.3	Рис. 3. 1 часть программного кода реализации шифрования с помощью метода решеток.	11
4.4	Рис. 4. 2 часть программного кода реализации шифрования с помощью метода решеток.	11
4.5	Рис. 5. 3 часть программного кода реализации шифрования с помощью метода решеток.	12
4.6	Рис. 6. 4 часть программного кода реализации шифрования с помощью метода решеток.	12
4.7	Рис. 7. Программного кода реализации шифра Виженера.	13
4.8	Рис. 8. Программного кода реализации расшифровки шифра Виженера.	13

List of Tables

1 Цель работы

Целью данной лабораторной работы является ознакомление с тремя методами шифрования: маршрутным шифрованием, шифрованием с помощью решеток, таблицей Виженера, а так же их реализация на произвольном языке программирования.

2 Задание

1. Реализовать метод маршрутного шифрования.
2. Реализовать метод шифрования с помощью решеток.
3. Реализовать метод таблицы Вижера.

3 Теоретическое введение

Математическая часть подробно описана в задании к лабораторной работе. Я поставила перед собой задачу найти исторические сведения, факты о методах шифрования.

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

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

Метод **маршрутного шифрования** изобрел французский математик и криптограф Франсуа Виет. Этот способ относится к перестановочным шифрам. Шифр называется перестановочным, если все связанные с ним криптограммы получаются из соответствующих открытых текстов перестановкой букв. Способ, каким при шифровании переставляются буквы открытого текста, и является ключом шифра. Такой метод шифрования (столбцовая перестановка) в годы первой мировой войны использовала легендарная немецкая шпионка Мата Хари.

Шифровальная решётка — трафарет с прорезями-ячейками (из бумаги, картона или аналогичного материала), использовавшийся для шифрования открытого текста. Текст наносился на лист бумаги через такой трафарет по определённым правилам, и расшифровка текста была возможна только при наличии такого же трафарета.

Вращающаяся решётка: Прямоугольные решётки Кардано можно размещать в четырёх позициях. Шифр с сеткой в виде шахматной доски имеет только две позиции, но именно этот вариант вращающейся решётки послужил для разработки более сложной решётки с четырьмя позициями, которую можно вращать в двух направлениях.

Шифр Виженера является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джован Баттиста Беллазо (итал. Giovan Battista Bellaso) в книге *La cifra del. Sig. Giovan Battista Bellaso* в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, он является недоступным для простых методов криптоанализа.

4 Выполнение лабораторной работы

4.1 Маршрутное шифрование

В соответствии с заданием, первой была написана программа для маршрутного шифрования (рис. 1-2). В качестве параметров системы были взяты данные из описательной части лабораторной работы портала ТУИС.

Примечание: комментарии по коду реализации маршрутного шифрования представлены на скриншотах.

Маршрутное шифрование

```
In [15]: 1 import numpy as np

In [17]: 1 def marsh_shift():
2     m=5 #длина блока
3     n=6 #число блоков
4     text="Нельзя недооценивать противника" #текст для шифрования
5     text1=text.upper() #заглавными буквами
6     result=list(text1) #сделали список из строки
7
8     #исключаем пробелы, точки, запятые, тире и тп
9
10    for symbol in result:
11        if (symbol==" ") or (symbol==".") or (symbol==",") or (symbol=="-") or (symbol=="?") or (symbol=="'") or (symbol=="'"):
12            index = result.index(symbol)
13            element = result.pop(index) #вырезаем символ по заданному индексу
14            result1=result.copy()
15            result2=[]
16
17    #создаем необходимую матрицу с шагом n
18    for i in range(0,len(result1),n):
19        result2.append(list(result1[i:n+i]))
20    #недостающие элементы заполняем буквами A
21    while (len(result2[m-1])<n):
22        result2[m-1].append('A')
23
24    #ввели пароль в корректной для работы форме
25    text2="пароль"
26    text2=text2.upper()
27    password=list(text2)
28
29    result3=list(result2)
30    result3.append(password) #добавили к матрице пароль
31    alphabet="АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЪЭЮЯ" #ввели алфавит
32    indices=[]
```

Figure 4.1: Рис. 1. 1 часть программного кода реализации маршрутного шифрования.

```

34 #Смотрим на индексы пароля в алфавите
35 for pas in password:
36     for letter in alphabet:
37         if pas==letter:
38             ind=alphabet.find(letter)
39             indices.append(ind)
40 result4=list(result3)
41 result4.append(indices) #добавили индексы в матрицу
42 result5=np.array(result4)
43 result6=result5[:,np.argsort(result5[-1,:])] #сортировка
44 result7=list(result6)
45
46 #убрали две последние строки в матрице
47 del (result7[-1] )
48 del (result7[-1])
49 result8=np.array(result7)
50 result9=result8.transpose() #транспонирует для того чтобы вывести шифр
51 result10=[]
52
53 #Начали работу над выводом шифра
54 for i in range (n):
55     result10.extend(result9[i])
56     print("".join(result10))#вывели строку шифра
57
58 marsh_shift( )

```

Figure 4.2: Рис. 2. 2 часть программного кода реализации маршрутного шифрования.

Результат выполнения программы видно на рис. 2. Результат можно назвать успешным, он совпадает с шифрованием из примера в ТУИСе РУДН, при этом само по себе шифрование однозначно, поэтому совпадение результата говорит о правильности выполнения работы.

4.2 Метод решеток

Далее была написана программа реализации шифрования методом решеток (рис. 3-6). В качестве параметров системы были взяты данные из описательной части лабораторной работы портала ТУИС.

Примечание: комментарии по коду реализации маршрутного шифрования представлены на скриншотах.

Метод решеток

```
In [18]: 1 import numpy as np
2 import random

In [23]: 1 def turning_grille():
2         k=2 #вводим k
3
4         #заполняем маленькую матрицу
5         osnova=np.linspace(1,k**2,k**2)
6         result=[]
7         for i in range(0,len(osnova),k):
8             result.append(list(osnova[i:i+k]))
9
10        #вводим функцию для поворота матрицы
11        def rot90(matrix):
12            return [list(reversed(col)) for col in zip(*matrix)]
13
14        matrix=np.full((2*k,2*k),0) #создали и заполнили нулями матрицу 2k x 2k
15
16        #заполняем матрицу matrix по четвертям
17        #1 четверть
18        matrix[k:k,0:k]=result
19        #2 четверть
20        result2=rot90(result)
21        matrix[k,k:2*k]=result2
22        #3 четверть
23        result3=rot90(result2)
24        matrix[k:2*k,k:2*k]=result3
25        #4 четверть
26        result4=rot90(result3)
27        matrix[k:2*k,k:2*k]=result4
28
```

Figure 4.3: Рис. 3. 1 часть программного кода реализации шифрования с помощью метода решеток.

```
29 #работа с отверстиями (определение координат)
30 holes=[]
31 for i in range(1,k**2+1):#прогонка по отдельному числу, например, по единичкам
32     indexes=[ ]
33     for m in range(0,2*k):#прогонка по строкам
34         for j in range(0,2*k):#прогонка по столбцам
35             if matrix[m][j]==1:
36                 coords=tuple([m, j])
37                 indexes.append(coords)
38     find=random.randint(0,3)#выбираем 1 из 4 координат
39     holes.append(indexes[find])
40
41 #работа с отверстиями (продолжение) визуализация поворотов и случаев размещения отверстий
42 template=np.full((2*k,2*k),0)
43 for d in range(k**2):
44     template[holes[d][0],holes[d][1]] =1
45 #1 поворот
46 template1=rot90(template)
47 #2 поворот
48 template2=rot90(template1)
49 #3 поворот
50 template3=rot90(template2)
51 text="ДОГОВОР ПОДПИСАЛИ"
52
```

Figure 4.4: Рис. 4. 2 часть программного кода реализации шифрования с помощью метода решеток.

```

53 #прогоняем templates для нахождения координат для заполнения буквами
54 #1 поворот
55 indexes1=[]
56 for m1 in range (0,2*k):
57     for j1 in range (0,2*k):
58         if template1[m1][j1]==1:
59             coords1=tuple([m1,j1])
60             indexes1.append(coords1)
61 #2 поворот
62 indexes2=[]
63 for m2 in range (0,2*k):
64     for j2 in range (0,2*k):
65         if template2[m2][j2]==1:
66             coords2=tuple([m2,j2])
67             indexes2.append(coords2)
68 #3 поворот
69 indexes3=[]
70 for m3 in range (0,2*k):
71     for j3 in range (0,2*k):
72         if template3[m3][j3]==1:
73             coords3=tuple([m3,j3])
74             indexes3.append(coords3)
75
76 #Переходим к образованию матрицы с буквами
77 letters_matrix=np.full((2*k,2*k),'0')
78 #0
79 for d in range (k**2):
80     letters_matrix[holes[d][0], holes[d][1]]=text[d]
81 #1
82 for d in range (k**2):
83     letters_matrix[indexes1[d][0],indexes1[d][1]]=text[d+k**2]
84 #2
85 for d in range (k**2):
86     letters_matrix[indexes2[d][0], indexes2[d][1]]=text[d+2*(k**2)]
87 #3
88 for d in range (k**2):
89     letters_matrix[indexes3[d][0],indexes3[d][1]]=text[d+3*(k**2)]
90 #####

```

Figure 4.5: Рис. 5. 3 часть программного кода реализации шифрования с помощью метода решеток.

```

91 #####
92 letter_matrix=list(letters_matrix)
93 text2="шифр"
94 text2=text2.upper()
95 password=list(text2)
96 letter_matrix.append(password)
97 alphabet="АВВГДЕЖЗИЙКЛМНОПРСТУХЦЧШЩЪЫЬЭЮЯ"
98 indices=[]
99 #Смотрим на индексы пароля в алфавите
100 for pas in password:
101     for letter in alphabet:
102         if pas==letter:
103             ind=alphabet.find(letter)
104             indices.append(ind)
105 letter_matrix.append(indices)
106 letter_matrix=np.array(letter_matrix)
107 letter_matrix=letter_matrix[:,np.argsort(letter_matrix[-1,:])]#упорядочили
108 letter_matrix=list(letter_matrix)
109 del (letter_matrix[-1])#убрали строку с индексами букв из пароля в алфавите
110 del (letter_matrix[-1])#убрали строку с индексами букв из пароля в алфавите
111 letter_matrix=np.array(letter_matrix)
112 letter_matrix=letter_matrix.transpose()
113 letter_matrix=list(letter_matrix)
114
115 #Выводим ответ в виде строки
116 result1=[]
117 for i in range (2*k):
118     result1.extend(letter_matrix[i])
119 print("".join(result1))
120
121 turning_grille()

```

ОВОРДЛГПАИОСДОИ

Figure 4.6: Рис. 6. 4 часть программного кода реализации шифрования с помощью метода решеток.

Результат выполнения программы видно на рис. 6. Результат можно назвать успешным, он совпадает с шифрованием из примера в ТУИСе РУДН. При этом само по себе шифрование неоднозначно, поскольку используется модуль random для порядка размещения символов, поэтому могут появляться и другие варианты шифрования.

4.3 Метод Виженера

Последним реализованным методом был метод Виженера (рис. 7). В данном методе я зашифровала фразы из английского алфавита.

Шифр Виженера

```
In [11]: 1 import string
2 def encrypt_vigenere(plaintext: str, keyword: str) -> str:
3     letters = string.ascii_letters #abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
4     abc = letters[:len(letters)//2] #abcdefghijklmnopqrstuvwxyz
5     ABC = letters[len(letters)//2:] #ABCDEFGHIJKLMNOPQRSTUVWXYZ
6
7     while len(plaintext) > len(keyword):
8         keyword += keyword
9     keyword = keyword[:len(plaintext)].upper()
10
11     ciphertext = ""
12     for i in range(len(plaintext)):
13         n = ABC.find(keyword[i])
14         cipher_letters = abc[n:] + abc[:n] + ABC[n:] + ABC[:n]
15         if plaintext[i] in letters:
16             ciphertext += cipher_letters[letters.find(plaintext[i])]
17         else:
18             ciphertext += plaintext[i]
19
20     return ciphertext
21
22 In [12]: 1 a = encrypt_vigenere("ATTACKATdawn", "LeNON")
2         a
23
24 Out[12]: 'LXFOPVEFrnh'
```

Figure 4.7: Рис. 7. Программного кода реализации шифра Виженера.

Далее было реализовано дешифрование текста (рис. 8). Важно отметить, что нужно использовать одинаковый ключ (кодировое слово) как для шифровки, так и для дешифрования.

Расшифровка Виженера

```
In [13]: 1 def decrypt_vigenere(ciphertext: str, keyword: str) -> str:
2
3     import string
4     letters = string.ascii_letters #abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
5     abc = letters[:len(letters)//2] #abcdefghijklmnopqrstuvwxyz
6     ABC = letters[len(letters)//2:] #ABCDEFGHIJKLMNOPQRSTUVWXYZ
7
8     while len(ciphertext) > len(keyword):
9         keyword += keyword
10    keyword = keyword[:len(ciphertext)].upper()
11
12    plaintext = ""
13    for i in range(len(ciphertext)):
14        n = ABC.find(keyword[i])
15        cipher_letters = abc[n:] + abc[:n] + ABC[n:] + ABC[:n]
16        if ciphertext[i] in letters:
17            plaintext += letters[cipher_letters.find(ciphertext[i])]
18        else:
19            plaintext += ciphertext[i]
20
21    return plaintext
22
23 In [14]: 1 b = decrypt_vigenere(a, "LeNON")
2         b
24
25 Out[14]: 'ATTACKATdawn'
```

Figure 4.8: Рис. 8. Программного кода реализации расшифровки шифра Виженера.

Как можно видеть, расшифровка совпадает с изначальным текстом на рис. 7, что говорит об успешной реализации метода.

5 Библиография

1. ТУИС РУДН
2. Википедия. Перестановочный шифр [Электронный ресурс]. Википедия, свободная энциклопедия, 2021. URL: https://en.wikipedia.org/wiki/Transposition_cipher.
3. VK. Метод маршрутного шифрования [Электронный ресурс]. VK, 2018. URL: <https://vk.com/@cryptandcod-metod-marshrutnogo-shifrovaniya>.
4. Википедия. Шифровальная решетка [Электронный ресурс]. Википедия, свободная энциклопедия, 2021. URL: [https://en.wikipedia.org/wiki/Grille_\(cryptography\)](https://en.wikipedia.org/wiki/Grille_(cryptography)).
5. Wix. Криптография [Электронный ресурс]. Wixsite, 2021. URL: [https://en.wikipedia.org/wiki/Grille_\(cryptography\)](https://en.wikipedia.org/wiki/Grille_(cryptography)).
6. Википедия. Шифр Виженера [Электронный ресурс]. Википедия, свободная энциклопедия, 2021. URL: https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher.

6 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: я ознакомилась с тремя методами шифрования – маршрутным шифрованием, шифрованием с помощью решеток, таблицей Виженера, а так же мне удалось реализовать их на языке программирования Python.