

# **Лабораторная работа №2**

**Математические основы защиты информации и информационной безопасности**

Николаев Дмитрий Иванович, НПМмд-02-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
2.1	Маршрутное шифрование . . . . .	6
2.2	Шифрование с помощью решеток . . . . .	7
2.3	Таблица Виженера . . . . .	9
<b>3</b>	<b>Ход работы</b>	<b>10</b>
3.1	Маршрутное шифрование . . . . .	10
3.2	Шифрование с помощью решеток . . . . .	13
3.3	Таблица Виженера . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

2.1	Шифрование с помощью решеток . . . . .	8
2.2	Соответствия в шифровании с помощью таблицы Вижинёра . . .	9
3.1	Код реализации алгоритма маршрутного шифрования на Julia (1/3)	11
3.2	Код реализации алгоритма маршрутного шифрования на Julia (2/3)	12
3.3	Код реализации алгоритма маршрутного шифрования на Julia (3/3)	12
3.4	Результат кода реализации алгоритма маршрутного шифрования на Julia . . . . .	13
3.5	Код реализации алгоритма шифрования с помощью решёток на Julia (1/4) . . . . .	13
3.6	Код реализации алгоритма шифрования с помощью решёток на Julia (2/4) . . . . .	14
3.7	Код реализации алгоритма шифрования с помощью решёток на Julia (3/4) . . . . .	14
3.8	Код реализации алгоритма шифрования с помощью решёток на Julia (4/4) . . . . .	15
3.9	Результат кода реализации алгоритма шифрования с помощью решёток на Julia . . . . .	16
3.10	Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (1/4) . . . . .	17
3.11	Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (2/4) . . . . .	17
3.12	Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (3/4) . . . . .	18
3.13	Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (4/4) . . . . .	18
3.14	Результат кода реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia . . . . .	19

## Список таблиц

# 1 Цель работы

Изучить работу шифров перестановки — маршрутного шифрования, шифрования с помощью решёток и таблицы Вижинёра, а также реализовать их программно.

## 2 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.

Существует два широко распространенных метода перестановок: маршрутное шифрование и шифрование с помощью решеток.

### 2.1 Маршрутное шифрование

Данный способ шифрования 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 шифротекст. Пусть  $m$  и  $n$  — целые положительные числа, большие 1. Открытый текст разбивается на блоки равной длины, состоящие из числа символов, равного произведению  $m \times n$ . Если последний блок получится меньше остальных, то в него следует дописать требуемое количество произвольных символов. Составляется таблица размерности  $mn$ . Блоки вписываются 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 паро-

ля.

Например, для шифрования текста “нельзя недооценивать противника”, разобьем его на блоки длины  $n = 6$ . Блоков получится  $m = 5$ . К последнему блоку припишем букву “а”. В качестве пароля выберем слово “пароль”. Теперь будем выписывать буквы по столбцам в соответствии с алфавитным порядком букв пароля и получим следующую криптограмму: ЕЕНПНЗОАТАЬОВОКН-НЕВВЯЦТИА.

Рассмотренный способ шифрования (столбцовая перестановка) в годы первой мировой войны использовала легендарная немецкая шпионка Мата Хари.

## 2.2 Шифрование с помощью решеток

Данный способ шифрования предложил австрийский криптограф Эдуард Флейснер в 1881 году. Суть этого способа заключается в следующем. Выбирается натуральное число  $k > 1$ , строится квадрат размерности  $k$ , который построчно заполняется числами  $1, 2, \dots, k^2$ . В качестве примера рассмотрим квадрат размерности  $k = 2$ :

$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}$$

Повернем его по часовой стрелке на  $90^\circ$  и присоединим к исходному квадрату справа:

$$\begin{array}{cccc} 1 & 2 & 3 & 1 \\ 3 & 4 & 4 & 2 \end{array}$$

Проделаем такую процедуру еще дважды и припишем получившиеся квадраты снизу. Получился большой квадрат размерности  $2k \times 2k$ .

1 2 3 1  
 3 4 4 2  
 2 4 4 3  
 1 3 2 1

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  $2k \times 2k$ , и в прорези вписываются буквы исходного текста по порядку их следования. Когда заполнятся все прорези, решето поворачивается на  $90^\circ$  и вписывание букв продолжается. После третьего поворота все клетки большого квадрата окажутся заполненными. Подобрал подходящий пароль (число букв пароля должно равняться  $k^2$ , и они не должны повторяться), выпишем буквы по столбцам. Очередность столбцов определяется алфавитным порядком букв пароля.

Пример. Исходный текст — “договор подписали”, пароль — “шифр”.

			д
	о		г
		о	

			д
	в		
о	о		г
	р	о	п

	о		д
а	в	п	
о	о		г
и	р	о	п

с	о	а	д
д	в	п	л
о	о	и	г
и	р	о	п
ш	и	ф	р

Рис. 2.1: Шифрование с помощью решеток

С применением вышеуказанной ([2.1]) решетки за пять шагов получаем следующую криптограмму: ОВОРДЛГПАПИОСДОИ.

Важно отметить, что число  $k$  подбирается в соответствии с количеством букв  $N$  исходного текста. В идеальном случае  $k^2 = N$ . Если такого равенства достичь невозможно, то можно либо дописать произвольную букву к последнему слову открытого текста, либо убрать её.



## 2.3 Таблица Виженера

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:  $a_1, a_2, \dots, a_n$ . Далее в каждом блоке первая буква циклически сдвигается вправо по алфавиту на  $a_1$  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. Рассмотрим еще одну схему построения шифра Виженера. В таблице в строчках записаны буквы русского алфавита смещённые на соответствующий номер строки (первая — исходный алфавит). При переходе от одной строке к другой происходит циклический сдвиг на одну позицию.

Пример. Исходный текст – “криптография серьезная наука”, пароль – “математика”. Пароль записывается с повторениями над буквами сообщения ([2.2]):

м	а	т	е	м	а	т	и	к	а	м	а	т	е	м	а	т	и	к	а	м	а	т	е	м	а
к	р	и	п	т	о	г	р	а	ф	и	я	с	е	р	ь	е	з	н	а	я	н	а	у	к	а

Рис. 2.2: Соответствия в шифровании с помощью таблицы Вижинёра

В итоге получается следующая криптограмма: ЦРЬФЯОХШКФФЯДКЭБЧП-ЧАЛНТШЩА.

## 3 Ход работы

Следуем указаниям из [1].

### 3.1 Маршрутное шифрование

Реализуем алгоритм маршрутного шифрования и его расшифрование на Julia ([3.1-3.3]), в результате получим следующий вывод ([3.4]).

```

1  alphabet = 'а':'я'
2  function Find_Alphabetical_Indices(word::String)
3      Temp_Char_Indices = Int[]
4      # Находим порядковые (в алфавите) значения символов в слове
5      for char in lowercase(word)
6          if char in alphabet
7              position = findfirst(x -> x == char, alphabet)
8              push!(Temp_Char_Indices, position)
9          end
10     end
11     # Находим индексы символом в алфавитном порядке в слове
12     return sortperm(Temp_Char_Indices)
13 end
14
15 function Fill_Table(Input_Text::String, rows::Int, cols::Int)
16     # cols - число столбцов, rows - число строк (длина ключа)
17     table = [" " for _ in 1:rows, _ in 1:cols]
18     Text_Indices = collect(enumerate(Input_Text))
19     index = 1
20     for i in 1:rows # Строки
21         for j in 1:cols # Столбцы
22             if index <= length(Input_Text)
23                 # Заполнение таблицы символами
24                 # сообщения по строкам
25                 table[i, j] = string(Text_Indices[index][2])
26                 index += 1
27             else # Заполнение оставшегося пр-ва таблицы
28                 table[i, j] = string('а')
29             end
30         end
31     end
32     return table
33 end
34

```

Рис. 3.1: Код реализации алгоритма маршрутного шифрования на Julia (1/3)

```

35 function Route_Cipher(Message::String, key::String)::String
36     # n (cols) - число столбцов, m (rows) - число строк (длина ключа)
37     n = length(key) # Число столбцов
38     m = div(length(Message), n) + 1 # Число строк
39     table = Fill_Table(Message, m, n)
40     Encrypted_Message = ""
41     Cols_Indices = Find_Alphabetical_Indices(key)
42     for j in Cols_Indices # Столбцы
43         for i in 1:m # Строки
44             # Читаем по столбцам в
45             # алфавитном порядке индексов ключа
46             Encrypted_Message *= table[i, j]
47         end
48     end
49     return Encrypted_Message
50 end
51
52 function Route_Decipher(Encrypted_Message::String, key::String)::String
53     # n (cols) - число столбцов, m (rows) - число строк (длина ключа)
54     m = length(key) # Ключ - число строк
55     n = div(length(Encrypted_Message), m)
56     # Записываем шифр в таблицу по строкам
57     # (исходная была n x m, данная - m x n)
58     table = Fill_Table(Encrypted_Message, m, n)
59     Initial_Message = ""
60     Rows_Indices = Find_Alphabetical_Indices(key)
61     for j in 1:n # Строки
62         for i in sortperm(Rows_Indices) # Столбцы
63             # Читаем по столбцам, выбирая
64             # элементы в соответствии с изначальным
65             # расположением столбцов
66             Initial_Message *= table[i, j]
67         end
68     end
69     return Initial_Message
70 end

```

Рис. 3.2: Код реализации алгоритма маршрутного шифрования на Julia (2/3)

```

71 # Пример использования
72 Message = "нельзя недооценивать противника"
73 key = "пароль"
74 println("Исходное сообщение: $Message")
75 # Шифрование
76 Encrypted_Message = Route_Cipher(Message, key)
77 println("Зашифрованный текст (Маршрутное шифрование): $Encrypted_Message")
78 # Расшифрование
79 Initial_Message = Route_Decipher(Encrypted_Message, key)
80 println("Расшифрованный текст (Маршрутное шифрование): $Initial_Message")

```

Рис. 3.3: Код реализации алгоритма маршрутного шифрования на Julia (3/3)

```

thbase-infosec\labs\lab02\report\report> julia .\lab_2.jl
Исходное сообщение: нельзя недооценивать противника
Зашифрованный текст (Маршрутное шифрование): еенпнзоатаьовокннеьвдиряцта
Расшифрованный текст (Маршрутное шифрование): нельзя недооценивать противникаа
PS C:\Users\User\Documents\work\study\2024-2025\Математические основы защиты и
thbase-infosec\labs\lab02\report\report>

```

Рис. 3.4: Результат кода реализации алгоритма маршрутного шифрования на Julia

## 3.2 Шифрование с помощью решеток

Реализуем алгоритм шифрования с помощью решёток и его расшифрование на Julia ([3.5-3.8]), в результате получим следующий вывод ([3.9]).

```

1  include("lab_2_Route_Cipher.jl")
2
3  # Функция для заполнения сетки с использованием решетки (шаблона) и текста
4  function Fill_Grid_With_Grille(Message::String, Grille::Matrix{Bool})::Matrix{String}
5      Mat_size = size(Grille)[1]
6      Grid = [" " for _ in 1:Mat_size, _ in 1:Mat_size]
7      Text_Indices = collect(enumerate(Message))
8      index = 1
9      # Вращения 4 раза по 90 градусов
10     for rotation in 1:4
11         for i in 1:Mat_size
12             for j in 1:Mat_size
13                 # Если в решетке есть прорезь
14                 if Grille[i, j]
15                     if index <= length(Message)
16                         Grid[i, j] = string(Text_Indices[index][2])
17                         index += 1
18                     else # Заполнение оставшегося пр-ва таблицы
19                         Grid[i, j] = string(' ')
20                     end
21                 end
22             end
23         end
24         Grille = rotr90(Grille)
25     end
26     return Grid
27 end
28

```

Рис. 3.5: Код реализации алгоритма шифрования с помощью решёток на Julia (1/4)

```

29 function Grid_Cipher(Message::String, Grille::Matrix{Bool}, key::String)::String
30     Mat_size = length(key) # Ключ - размерность матрицы
31     Grid = Fill_Grid_With_Grille(Message, Grille)
32     Encrypted_Message = ""
33     Cols_Indices = Find_Alphabetical_Indices(key)
34     for j in Cols_Indices # Столбцы
35         for i in 1:Mat_size # Строки
36             # Читаем по столбцам в
37             # алфавитном порядке индексов ключа
38             Encrypted_Message *= Grid[i, j]
39         end
40     end
41     return Encrypted_Message
42 end

```

Рис. 3.6: Код реализации алгоритма шифрования с помощью решёток на Julia  
(2/4)

```

44 function Grid_Decipher(Encrypted_Message::String, Grille::Matrix{Bool}, key::String)::String
45     Mat_size = length(key) # Ключ - размерность матрицы
46     # Записываем шифр в таблицу по строкам
47     Grid = Fill_Table(Encrypted_Message, Mat_size, Mat_size)
48     Initial_Message = ""
49     Rows_Indices = Find_Alphabetical_Indices(key)
50     Temp_Grid = [" " for _ in 1:Mat_size, _ in 1:Mat_size]
51     # Преобразуем таблицу с зашифрованным текстом к
52     # виду аналогичному записанному с помощью
53     # решета исходного сообщения
54     for j in 1:Mat_size # Строки
55         temp_col = []
56         for i in sortperm(Rows_Indices) # Столбцы
57             push!(temp_col, Grid[i, j])
58         end
59         # Из выбранных из столбца элементов
60         # формируем строку
61         for k in 1:Mat_size
62             Temp_Grid[j, k] = temp_col[k]
63         end
64     end
65     for rotation in 1:4
66         for i in 1:Mat_size
67             for j in 1:Mat_size
68                 # Если в решете есть прорезь
69                 if Grille[i, j]
70                     Initial_Message *= Temp_Grid[i, j]
71                 end
72             end
73         end
74         Grille = rotr90(Grille)
75     end
76     return Initial_Message
77 end

```

Рис. 3.7: Код реализации алгоритма шифрования с помощью решёток на Julia  
(3/4)

```

79 # Пример использования
80 text = "договорподписали"
81 key = "шифр"
82 Mat_size = Int64(floor(sqrt(length(text))))
83 #size = 4 # Размер сетки (должен быть квадратом)
84 grille = [false for i in 1:Mat_size, j in 1:Mat_size]
85 indexes = [(1, 4) (3, 2) (3, 4) (4, 3)]
86 for (i, j) in indexes
87     grille[i, j] = true
88 end
89 println("\nИсходное сообщение: $text")
90 println("\nКлюч шифрования: $key")
91 println("\nРешето для записи сообщения в таблицу: ")
92 for i in 1:Mat_size
93     for j in 1:Mat_size
94         print(grille[i, j], " ")
95     end
96     println("\n")
97 end
98
99 println("\nЗаписанное в таблицу сообщение: ")
100 grid = Fill_Grid_With_Grille(text, grille)
101 for i in 1:Mat_size
102     for j in 1:Mat_size
103         print(grid[i, j], " ")
104     end
105     println("\n")
106 end
107
108 # Шифрование
109 Encrypted_Message = Grid_Cipher(text, grille, key)
110 println("Зашифрованный текст (Шифрование с помощью решёток): $Encrypted_Message")
111
112 # Расшифрование
113 Decrypted_Message = Grid_Decipher(Encrypted_Message, grille, key)
114 println("Расшифрованный текст (Шифрование с помощью решёток): $Decrypted_Message")

```

Рис. 3.8: Код реализации алгоритма шифрования с помощью решёток на Julia (4/4)

```

Исходное сообщение: договорподписали

Ключ шифрования: шифр

Решето для записи сообщения в таблицу:
false false false true
false false false false
false true false true
false false true false

Записанное в таблицу сообщение:
с о а д
д в п л
о о и г
и р о п

Зашифрованный текст (шифрование с помощью решёток): овордлгпапиосдои
Расшифрованный текст (шифрование с помощью решёток): договорподписали

```

Рис. 3.9: Результат кода реализации алгоритма шифрования с помощью решёток на Julia

### 3.3 Таблица Виженера

Реализуем алгоритм шифрования с помощью таблицы Вижинёра и его расшифрование на Julia ([3.10-3.13]), в результате получим следующий вывод ([3.14]).



```

1  alphabet = 'а':'я'
2  # Функция нахождения массива порядковых
3  # номеров букв в слове из алфавита
4  function Word_Alphabet_Serial_Numbers(Word::String)
5      Temp_Char_Indices = []
6      for char in lowercase(Word)
7          if char in alphabet
8              position = findfirst(x -> x == char, alphabet)
9              push!(Temp_Char_Indices, position)
10         end
11     end
12     return Temp_Char_Indices
13 end
14 # Функция для создания таблицы Виженера
15 function Create_Vigener_Table()::Matrix{String}
16     Tab_size = length(alphabet)
17     Vigener_Table = [" " for _ in 1:Tab_size, _ in 1:Tab_size]
18     for i in 1:Tab_size
19         for j in 1:Tab_size
20             Vigener_Table[i, j] = string(alphabet[(mod(i + j - 2, Tab_size) + 1)])
21         end
22     end
23     return Vigener_Table
24 end
25

```

Рис. 3.10: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (1/4)

```

26 # Функция шифрования с использованием таблицы Виженера
27 function Vigener_Cipher(Message::String, key::String)::String
28     Message = lowercase(Message)
29     key = lowercase(key)
30     Vigener_Table = Create_Vigener_Table()
31     Tab_size = length(alphabet)
32     # Алфавитные индексы исходного сообщения
33     Message_Indices = Word_Alphabet_Serial_Numbers(Message)
34     # Алфавитные индексы ключа
35     Key_Indices = Word_Alphabet_Serial_Numbers(key)
36     Encrypted_Message = ""
37     key_length = length(key)
38     for (i, j) in enumerate(Message_Indices) # В 1-ой строке
39         Encrypted_Message *= Vigener_Table[Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)], j]
40     end
41     return Encrypted_Message
42 end

```

Рис. 3.11: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (2/4)

```

44 # функция расшифровки с использованием таблицы Виженера
45 function Vigenere_Decipher(Encrypted_Message::String, key::String)::String
46     Encrypted_Message = lowercase(Encrypted_Message)
47     key = lowercase(key)
48     Vigenere_Table = Create_Vigenere_Table()
49     Tab_size = length(alphabet)
50     # Алфавитные индексы зашифрованного сообщения
51     Message_Indices = Word_Alphabet_Serial_Numbers(Encrypted_Message)
52     # Алфавитные индексы ключа
53     Key_Indices = Word_Alphabet_Serial_Numbers(key)
54     Initial_Message = ""
55     key_length = length(key)
56     # Первый способ
57     #for (i, j) in enumerate(Message_Indices) # В первом столбце
58     #    # i - номер "номера", j - номер строки, так как зашифрованную строку
59     #    # расположили как левый крайний столбец, тогда
60     #    # надо сместиться влево на (размер алфавита - (порядковый номер ключа - 1) + 1)
61     #    # так как число скачков на 1 меньше порядкового номера ключа,
62     #    # где первый уводит на последний столбец, поэтому + 1
63     #    Key_Real_Index = Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)]
64     #    Left_col_Shift = Tab_size - Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)] + 2
65     #    Initial_Message *= Vigenere_Table[j, Key_Real_Index == 1 ? Key_Real_Index : mod(Left_col_Shift, Tab_size)]
66 end =#
67 # Второй способ
68 for (i, j) in enumerate(Message_Indices) # В последнем столбце
69     # Алфавит в правом крайнем столбце смещён на единицу вниз,
70     # Поэтому если номер зашифрованной буквы 32, то она
71     # находится в правом верхнем углу, а относительно столбца
72     # происходит (размер алфавита - (порядковый номер ключа - 1))
73     # скачков влево
74     Key_Real_Index = Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)]
75     Left_col_Shift = Tab_size - Key_Real_Index + 1
76     Initial_Message *= Vigenere_Table[j % Tab_size == 0 ? 1 : j + 1, Left_col_Shift]
77 end
78 return Initial_Message
79 end

```

Рис. 3.12: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (3/4)

```

81 # Пример использования
82 text = "криптографиясерьезнаянаука"
83 key = "математика"
84 println("\n\nИсходное сообщение: $text")
85 println("Ключ шифрования: $key")
86
87 # Шифрование
88 Encrypted_Message = Vigenere_Cipher(text, key)
89 println("Зашифрованный текст (Шифрование с помощью таблицы Вижинёра): $Encrypted_Message")
90
91 # Расшифрование
92 Decrypted_Message = Vigenere_Decipher(Encrypted_Message, key)
93 println("Расшифрованный текст (Шифрование с помощью таблицы Вижинёра): $Decrypted_Message")

```

Рис. 3.13: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (4/4)

```
thbase-infosec\labs\lab02\report\report> julia .\lab_2_Viginer_Cipher.jl
Исходное сообщение: криптографиясерьезнаянаука
Ключ шифрования: математика
Зашифрованный текст (Шифрование с помощью таблицы Вижинёра): црѣфюохшкффягкѣьчпчалнтщца
Расшифрованный текст (Шифрование с помощью таблицы Вижинёра): криптографиясерьезнаянаука
PS C:\Users\User\Documents\work\study\2024-2025\Математические основы защиты информации и ин
```

Рис. 3.14: Результат кода реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia

## 4 Выводы

В ходе выполнения лабораторной работы я изучил работу перестановочных шифров — маршрутного шифрования, шифрования с помощью решёток и таблицы Вижинёра, а также реализовать их программно на языке Julia.

## Список литературы

1. Лабораторная работа № 2. Шифры перестановки [Электронный ресурс]. Саратовский государственный университет имени Н.Г. Чернышевского, 2024.