Лабораторная работа №2

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

28 сентября 2024

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Прагматика выполнения

Прагматика выполнения

- Освоение алгоритмов шифров перестановки — маршрутного шифрования, шифрования с помощью решёток и таблицы Вижинёра

Цели

Цели

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

Задачи

Задачи

1. Освоить и реализовать алгоритм маршрутного шифрования (и расшифрования) на языке Julia;
2. Освоить и реализовать алгоритм шифрования с помощью решёток (и расшифрования) на языке Julia;
3. Освоить и реализовать алгоритм шифрования с помощью таблицы Вижинёра (и расшифрования) на языке Julia.

Выполнение работы

Маршрутное шифрование (1/4)

```
alphabet = 'а':'я'
function Find_Alphabetical_Indices(word::String)
    Temp_Char_Indices = Int[]
    # Находим порядковые (в алфавите) значения символов в слове
    for char in lowercase(word)
        if char in alphabet
            position = findfirst(x -> x == char, alphabet)
            push!(Temp_Char_Indices, position)
        end
    end
    # Находим индексы символом в алфавитном порядке в слове
    return sortperm(Temp_Char_Indices)
end

function Fill_Table(Input_Text::String, rows::Int, cols::Int)
    # cols - число столбцов, rows - число строк (длина ключа)
    table = ["" for _ in 1:rows, _ in 1:cols]
    Text_Indices = collect(enumerate(Input_Text))
    index = 1
    for i in 1:rows # Строки
        for j in 1:cols # Столбцы
            if index <= length(Input_Text)
                # Заполнение таблицы символами
                # сообщения по строкам
                table[i, j] = string(Text_Indices[index][2])
                index += 1
            else # Заполнение оставшегося пр-ва таблицы
                table[i, j] = string('а')
            end
        end
    end
    return table
end
```

Маршрутное шифрование (2/4)

```
function Route_Cipher(Message::String, key::String)::String
    # n (cols) - число столбцов, m (rows) - число строк (длина ключа)
    n = length(key)  # Число столбцов
    m = div(length(Message), n) + 1  # Число строк
    table = Fill_Table(Message, m, n)
    Encrypted_Message = ""
    Cols_Indices = Find_Alphabetical_Indices(key)
    for j in Cols_Indices # Столбцы
        for i in 1:m  # Строки
            # Читаем по столбцам в
            # алфавитном порядке индексов ключа
            Encrypted_Message *= table[i, j]
        end
    end
    return Encrypted_Message
end

function Route_Decipher(Encrypted_Message::String, key::String)::String
    # n (cols) - число столбцов, m (rows) - число строк (длина ключа)
    m = length(key)  # Ключ - число строк
    n = div(length(Encrypted_Message), m)
    # Записываем шифр в таблицу по строкам
    # (исходная была n x m, данная - m x n)
    table = Fill_Table(Encrypted_Message, m, n)
    Initial_Message = ""
    Rows_Indices = Find_Alphabetical_Indices(key)
    for j in 1:n # Строки
        for i in sortperm(Rows_Indices)  # Столбцы
            # Читаем по столбцам, выбирая
            # элементы в соответствии с изначальным
            # расположением столбцов
            Initial_Message *= table[i, j]
        end
    end
    return Initial_Message
end
```

```julia
# Пример использования
Message = "нельзянедооцениватьпротивника"
key = "пароль"
println("Исходное сообщение: $Message")
# Шифрование
Encrypted_Message = Route_Cipher(Message, key)
println("Зашифрованный текст (Маршрутное шифрование): $Encrypted_Message")
# Расшифрование
Initial_Message = Route_Decipher(Encrypted_Message, key)
println("Расшифрованный текст (Маршрутное шифрование): $Initial_Message")
```

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

Маршрутное шифрование (4/4)

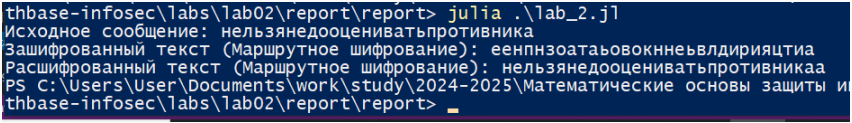

Рис. 4: Результат кода реализации алгоритма маршрутного шифрования на Julia

Шифрование с помощью решеток (1/5)

```julia
include("lab_2_Route_Cipher.jl")

# Функция для заполнения сетки с использованием решетки (шаблона) и текста
function Fill_Grid_With_Grille(Message::String, Grille::Matrix{Bool})::Matrix{String}
    Mat_size = size(Grille)[1]
    Grid = ["" for _ in 1:Mat_size, _ in 1:Mat_size]
    Text_Indices = collect(enumerate(Message))
    index = 1
    # Вращения 4 раза по 90 градусов
    for rotation in 1:4
        for i in 1:Mat_size
            for j in 1:Mat_size
                # Если в решете есть прорезь
                if Grille[i, j]
                    if index <= length(Message)
                        Grid[i, j] = string(Text_Indices[index][2])
                        index += 1
                    else # Заполнение оставшегося пр-ва таблицы
                        Grid[i, j] = string('а')
                    end
                end
            end
        end
        Grille = rotr90(Grille)
    end
    return Grid
end
```

Шифрование с помощью решеток (2/5)

```julia
function Grid_Cipher(Message::String, Grille::Matrix{Bool}, key::String)::String
    Mat_size = length(key)  # Ключ - размерность матрицы
    Grid = Fill_Grid_With_Grille(Message, Grille)
    Encrypted_Message = ""
    Cols_Indices = Find_Alphabetical_Indices(key)
    for j in Cols_Indices # Столбцы
        for i in 1:Mat_size  # Строки
            # Читаем по столбцам в
            # алфавитном порядке индексов ключа
            Encrypted_Message *= Grid[i, j]
        end
    end
    return Encrypted_Message
end
```

Рис. 6: Код реализации алгоритма шифрования с помощью решёток на Julia (2/4)

Шифрование с помощью решеток (3/5)

```julia
function Grid_Decipher(Encrypted_Message::String, Grille::Matrix{Bool}, key::String)::String
    Mat_size = length(key)  # Ключ - размерность матрицы
    # Записываем шифр в таблицу по строкам
    Grid = Fill_Table(Encrypted_Message, Mat_size, Mat_size)
    Initial_Message = ""
    Rows_Indices = Find_Alphabetical_Indices(key)
    Temp_Grid = ["" for _ in 1:Mat_size, _ in 1:Mat_size]
    # Преобразуем таблицу с зашифрованным текстом к
    # виду аналогичному записанному с помощью
    # решета исходного сообщения
    for j in 1:Mat_size # Строки
        temp_col = []
        for i in sortperm(Rows_Indices)  # Столбцы
            push!(temp_col, Grid[i, j])
        end
        # Из выбранных из столбца элементов
        # формируем строку
        for k in 1:Mat_size
            Temp_Grid[j, k] = temp_col[k]
        end
    end
    for rotation in 1:4
        for i in 1:Mat_size
            for j in 1:Mat_size
                # Если в решете есть прорезь
                if Grille[i, j]
                    Initial_Message *= Temp_Grid[i, j]
                end
            end
        end
        Grille = rotr90(Grille)
    end
    return Initial_Message
end
```

Шифрование с помощью решеток (4/5)

```
# Пример использования
text = "договорподписали"
key = "шифр"
Mat_size = Int64(floor(sqrt(length(text))))
#size = 4  # Размер сетки (должен быть квадратом)
grille = [false for i in 1:Mat_size, j in 1:Mat_size]
indexes = [(1, 4) (3, 2) (3, 4) (4, 3)]
for (i, j) in indexes
    grille[i, j] = true
end
println("\n\nИсходное сообщение: $text")
println("\n\nКлюч шифрования: $key")
println("\nРешётка для записи сообщения в таблицу: ")
for i in 1:Mat_size
    for j in 1:Mat_size
        print(grille[i, j], " ")
    end
    println("\n")
end

println("\nЗаписанное в таблицу сообщение: ")
grid = Fill_Grid_With_Grille(text, grille)
for i in 1:Mat_size
    for j in 1:Mat_size
        print(grid[i, j], " ")
    end
    println("\n")
end

# Шифрование
Encrypted_Message = Grid_Cipher(text, grille, key)
println("Зашифрованный текст (Шифрование с помощью решёток): $Encrypted_Message")

# Расшифрование
Decrypted_Message = Grid_Decipher(Encrypted_Message, grille, key)
println("Расшифрованный текст (Шифрование с помощью решёток): $Decrypted_Message")
```

Шифрование с помощью решеток (5/5)

```
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
```

Таблица Виженера (1/5)

```julia
alphabet = 'а':'я'
# Функция нахождения массива порядковых
# номеров букв в слове из алфавита
function Word_Alphabet_Serial_Numbers(Word::String)
    Temp_Char_Indices = []
    for char in lowercase(Word)
        if char in alphabet
            position = findfirst(x -> x == char, alphabet)
            push!(Temp_Char_Indices, position)
        end
    end
    return Temp_Char_Indices
end
# Функция для создания таблицы Виженера
function Create_Vigener_Table()::Matrix{String}
    Tab_size = length(alphabet)
    Vigenere_Table = ["" for _ in 1:Tab_size, _ in 1:Tab_size]
    for i in 1:Tab_size
        for j in 1:Tab_size
            Vigenere_Table[i, j] = string(alphabet[mod(i + j - 2, Tab_size) + 1])
        end
    end
    return Vigenere_Table
end
```

Таблица Виженера (2/5)

```
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      Vigenere_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 *= Vigenere_Table[Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)], j]
40      end
41      return Encrypted_Message
42  end
```

Рис. 11: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (2/4)

Таблица Виженера (3/5)

```julia
# Функция расшифрования с использованием таблицы Виженера
function Vigener_Decipher(Encrypted_Message::String, key::String)::String
    Encrypted_Message = lowercase(Encrypted_Message)
    key = lowercase(key)
    Vigenere_Table = Create_Vigener_Table()
    Tab_size = length(alphabet)
    # Алфавитные индексы зашифрованного сообщения
    Message_Indices = Word_Alphabet_Serial_Numbers(Encrypted_Message)
    # Алфавитные индексы ключа
    Key_Indices = Word_Alphabet_Serial_Numbers(key)
    Initial_Message = ""
    key_length = length(key)
    # Первый способ
    #=for (i, j) in enumerate(Message_Indices) # В первом столбце
        # i - номер "номера", j - номер строки, так как зашифрованную строку
        # расположили как левый крайний столбец, тогда
        # надо сместиться влево на (размер алфавита - (порядковый номер ключа - 1) + 1)
        # так как число скачков на 1 меньше порядкового номера ключа,
        # где первый уводит на последний столбец, поэтому + 1
        Key_Real_Index = Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)]
        Left_Col_Shift = Tab_size - Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)] + 2
        Initial_Message *= Vigenere_Table[j, Key_Real_Index == 1 ?  Key_Real_Index : mod(Left_Col_Shift, Tab_size)]
    end =#
    # Второй способ
    for (i, j) in enumerate(Message_Indices) # В последнем столбце
        # Алфавит в правом крайнем столбце смещён на единицу вниз,
        # Поэтому если номер зашифрованной буквы 32, то она
        # находится в правом верхнем углу, а относительно столбца
        # происходит (размер алфавита - (порядковый номер ключа - 1))
        # скачков влево
        Key_Real_Index = Key_Indices[i % key_length == 0 ? key_length : mod(i, key_length)]
        Left_Col_Shift = Tab_size - Key_Real_Index + 1
        Initial_Message *= Vigenere_Table[j % Tab_size == 0 ? 1 : j + 1, Left_Col_Shift]
    end
    return Initial_Message
end
```

```julia
# Пример использования
text = "криптографиясерьезнаянаука"
key = "математика"
println("\n\nИсходное сообщение: $text")
println("Ключ шифрования: $key")

# Шифрование
Encrypted_Message = Vigener_Cipher(text, key)
println("Зашифрованный текст (Шифрование с помощью таблицы Вижинёра): $Encrypted_Message")

# Расшифрование
Decrypted_Message = Vigener_Decipher(Encrypted_Message, key)
println("Расшифрованный текст (Шифрование с помощью таблицы Вижинёра): $Decrypted_Message")
```

Рис. 13: Код реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia (4/4)

Таблица Виженера (5/5)

```
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): цръфюохшкффягкььчпчалнтшца
Расшифрованный текст (Шифрование с помощью таблицы Вижинёра): криптографиясерьезнаянаука
PS C:\Users\User\Documents\work\study\2024-2025\Математические основы защиты информации и ин
```

Рис. 14: Результат кода реализации алгоритма шифрования с помощью таблицы Вижинёра на Julia

Результаты

Результаты

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