Лабораторная работа №2

Математические основы защиты информации и информационной безопасности

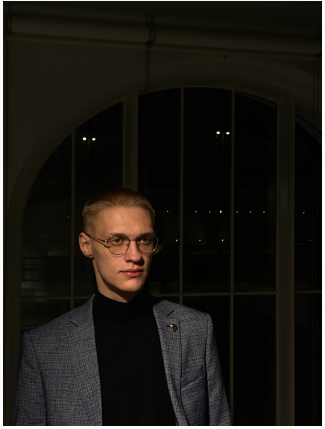
Махорин И. С.

2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Докладчик

- Махорин Иван Сергеевич
- Студент группы НФИмд-02-25
- Студ. билет 1032259380
- Российский университет дружбы народов имени Патриса Лумумбы



Цель лабораторной работы

- Изучить шифры перестановки и научиться их реализовывать.

Выполнение лабораторной работы

Реализация маршрутного шифрования

```julia
# Маршрутное шифрование
function columnar_transposition(text::String, password::String)::String
    # Удаляем пробелы и приводим к нижнему регистру
    text = replace(lowercase(text), " " => "")
    password = lowercase(password)
    text_chars = collect(text)
    password_chars = collect(password)

    # Определяем размеры таблицы
    n = length(password_chars)
    # Дополняем текст если необходимо
    remainder = length(text_chars) % n
    if remainder != 0
        append!(text_chars, collect("а"^(n - remainder)))
    end
    m = length(text_chars) ÷ n

    # Создаем таблицу размером mxn (строки x столбцы) - заполняем по строкам
    table = reshape(text_chars, (n, m))  # Сначала создаем nxm
    table = permutedims(table)  # Транспонируем чтобы получить mxn

    # Сортируем столбцы по алфавитному порядку букв пароля
    sorted_indices = sortperm(password_chars)

    # Формируем шифртекст по отсортированным столбцам
    ciphertext_chars = Char[]
    for col in sorted_indices
        append!(ciphertext_chars, table[:, col])
    end

    return uppercase(String(ciphertext_chars))
end
```

Рис. 1: Реализация маршрутного шифрования

Реализация маршрутного шифрования

```julia
function columnar_decryption(ciphertext::String, password::String)::String
    password_chars = collect(lowercase(password))
    ciphertext_chars = collect(lowercase(ciphertext))
    n = length(password_chars)
    m = length(ciphertext_chars) ÷ n

    # Получаем порядок столбцов при шифровании
    sorted_indices = sortperm(password_chars)

    # Создаем таблицу из шифртекста (заполняем по столбцам в порядке sorted_indices)
    encrypted_table = Matrix{Char}(undef, m, n)
    for (i, col_idx) in enumerate(sorted_indices)
        encrypted_table[:, col_idx] = ciphertext_chars[(i-1)*m+1:i*m]
    end

    # Восстанавливаем текст из таблицы (читаем по строкам)
    plaintext_chars = Char[]
    for i in 1:m
        for j in 1:n
            push!(plaintext_chars, encrypted_table[i, j])
        end
    end

    return String(plaintext_chars)
end
```

Рис. 2: Реализация маршрутного шифрования

Проверка

```
# Тестирование маршрутного шифрования
text = "нельзя недооценивать противника"
password = "пароль"
println("Исходный текст: ", text)
println("Пароль: ", password)

encrypted_col = columnar_transposition(text, password)
println("Зашифрованный: ", encrypted_col)

decrypted_col = columnar_decryption(encrypted_col, password)
println("Расшифрованный: ", decrypted_col)
println()
```

```
Исходный текст: нельзя недооценивать противника
Пароль: пароль
Зашифрованный: ЕЕНПНЗОАТАЬОВОКННЕЬВЛДИРИЯЦТИА
Расшифрованный: нельзянедооцениватьпротивникаа
```

Рис. 3: Проверка

Реализация шифрования с помощью решеток

```julia
# Шифрование с помощью решеток

# Вспомогательная функция для поворота матрицы
function rot90(matrix::Matrix{T}, k::Int=1) where T
    result = copy(matrix)
    for _ in 1:k
        result = reverse(permutedims(result, (2, 1)), dims=2)  # Поворот на 90°
    end
    return result
end

function generate_fleissner_grille(k::Int)::Matrix{Int}
    # Создаем базовый квадрат k×k с числами от 1 до k²
    base = Matrix{Int}(undef, k, k)
    for i in 1:k, j in 1:k
        base[i, j] = (i-1)*k + j  # Заполняем числами от 1 до k²
    end

    # Создаем большой квадрат 2k×2k
    big_square = zeros(Int, 2k, 2k)

    # Заполняем поворотами базового квадрата
    big_square[1:k, 1:k] = base                    # Исходная позиция
    big_square[1:k, k+1:2k] = rot90(base, 1)       # Поворот на 90° вправо
    big_square[k+1:2k, k+1:2k] = rot90(base, 2)    # Поворот на 180°
    big_square[k+1:2k, 1:k] = rot90(base, 3)       # Поворот на 270°

    return big_square
end
```

Рис. 4: Реализация шифрования с помощью решеток

Реализация шифрования с помощью решеток

```
function fleissner_encrypt(text::String, k::Int, password::String)::String
    text = replace(lowercase(text), " " => "")
    password = lowercase(password)

    # Проверяем длину текста
    required_length = 4*k^2
    if length(text) != required_length
        error("Длина текста должна быть равна $required_length, получено $(length(text))")
    end

    # Генерируем решетку
    grille = generate_fleissner_grille(k)

    # Создаем таблицу для заполнения
    table = Matrix{Char}(undef, 2k, 2k)
    text_chars = collect(text)
    text_index = 1

    # Заполняем через повороты решетки (4 поворота по 90°)
    for rotation in 0:3
        rotated_grille = rot90(grille, rotation)  # Поворачиваем решетку

        # Проходим по всем ячейкам решетки
        for i in 1:2k, j in 1:2k
            # Если в ячейке число от 1 до k² - это прорезь
            if 1 <= rotated_grille[i, j] <= k^2
                # Заполняем эту ячейку текстом
                if text_index <= length(text_chars)
                    table[i, j] = text_chars[text_index]
                    text_index += 1
                end
            end
        end
    end
```

Рис. 5: Реализация шифрования с помощью решеток

```
    # Сортируем столбцы по алфавитному порядку букв пароля
    password_chars = collect(password)
    sorted_indices = sortperm(password_chars)

    ciphertext_chars = Char[]
    for col in sorted_indices
        # Безопасно добавляем столбец
        for i in 1:2k
            push!(ciphertext_chars, table[i, col])
        end
    end

    return uppercase(String(ciphertext_chars))
end
```

Рис. 6: Реализация шифрования с помощью решеток

Проверка

```julia
# Тестирование шифра решеток
k = 2
grille_text = "договор подписали"  # 16 символов для k=2 (4*2²=16)
grille_text_clean = replace(lowercase(grille_text), " " => "")

# Проверяем длину
required_length = 4*k^2
if length(grille_text_clean) != required_length
    println("Дополняем текст до $required_length символов")
    if length(grille_text_clean) < required_length
        grille_text_clean *= "а"^(required_length - length(grille_text_clean))
    else
        grille_text_clean = grille_text_clean[1:required_length]
    end
end

grille_password = "шифр"
println("Исходный текст: ", grille_text)
println("Подготовленный текст: ", grille_text_clean)
println("Длина текста: ", length(grille_text_clean))
println("k: ", k)
println("Пароль: ", grille_password)

encrypted_grille = fleissner_encrypt(grille_text_clean, k, grille_password)
println("Зашифрованный: ", encrypted_grille)
```

```
Исходный текст: договор подписали
Подготовленный текст: договорподписали
Длина текста: 16
k: 2
Пароль: шифр
Зашифрованный: ООДАОПИИГРПЛДВОС
```

Рис. 7: Проверка

Реализация таблицы Виженера

```
# Таблица Виженера
function vigenere_cipher(text::String, key::String)::String
    text = replace(lowercase(text), " " => "")  # Очищаем текст
    key = lowercase(key)

    alphabet = collect("абвгдежзийклмнопрстуфхцчшщьыэюя")  # Русский алфавит
    n = length(alphabet)

    # Повторяем ключ до длины текста
    extended_key = Char[]
    key_chars = collect(key)
    for i in 1:length(text)
        # Циклически повторяем ключ
        push!(extended_key, key_chars[(i-1) % length(key) + 1])
    end

    ciphertext_chars = Char[]
    text_chars = collect(text)
    for (i, char) in enumerate(text_chars)
        char_idx = findfirst(isequal(char), alphabet)  # Ищем индекс символа
        if char_idx !== nothing
            key_char = extended_key[i]
            key_idx = findfirst(isequal(key_char), alphabet)  # Ищем индекс ключа

            if key_idx !== nothing
                # Применяем преобразование Виженера: (текст + ключ) mod n
                new_idx = (char_idx - 1 + key_idx - 1) % n + 1
                push!(ciphertext_chars, alphabet[new_idx])
            else
                push!(ciphertext_chars, char)  # Если ключ не найден, оставляем как есть
            end
        else
            push!(ciphertext_chars, char)  # Если символ не в алфавите, оставляем
        end
```

Рис. 8: Реализация таблицы Виженера

Реализация таблицы Виженера

```
function vigenere_decipher(ciphertext::String, key::String)::String
    ciphertext = lowercase(ciphertext)
    key = lowercase(key)

    alphabet = collect("абвгдежзийклмнопрстуфхцчшщъыьэюя")
    n = length(alphabet)

    # Повторяем ключ до длины текста
    extended_key = Char[]
    key_chars = collect(key)
    for i in 1:length(ciphertext)
        push!(extended_key, key_chars[(i-1) % length(key) + 1])
    end

    plaintext_chars = Char[]
    cipher_chars = collect(ciphertext)
    for (i, char) in enumerate(cipher_chars)
        char_idx = findfirst(isequal(char), alphabet)
        if char_idx !== nothing
            key_char = extended_key[i]
            key_idx = findfirst(isequal(key_char), alphabet)

            if key_idx !== nothing
                # Обратное преобразование: (шифр - ключ) mod n
                new_idx = (char_idx - 1 - (key_idx - 1) + n) % n + 1
                push!(plaintext_chars, alphabet[new_idx])
            else
                push!(plaintext_chars, char)
            end
        else
            push!(plaintext_chars, char)
        end
    end
```

Рис. 9: Реализация таблицы Виженера

Проверка

```
# Тестирование шифра Виженера
vigenere_text = "криптография серьезная наука"
vigenere_key = "математика"
println("Исходный текст: ", vigenere_text)
println("Ключ: ", vigenere_key)

encrypted_vig = vigenere_cipher(vigenere_text, vigenere_key)
println("Зашифрованный: ", encrypted_vig)

decrypted_vig = vigenere_decipher(encrypted_vig, vigenere_key)
println("Расшифрованный: ", decrypted_vig)
println()
```

```
Исходный текст: криптография серьезная наука
Ключ: математика
Зашифрованный: ЦРЪФЮОХШКФФЯГКЬЬЧПЧАЛНТШЦА
Расшифрованный: криптографиясерьезнаянаука
```

Рис. 10: Проверка

Вывод

Вывод

- В ходе выполнения лабораторной работы были изучены шифры перестановки, а также написаны их алгоритмы на языке Julia.

Список литературы. Библиография

Список литературы. Библиография

[1] Julia: https://docs.julialang.org/en/v1/