ЛАБОРАТОРНАЯ РАБОТА №2

Шифры перестановки

Кузнецов Юрий Владимирович

Введение

В данном отчёте будет представлена реализация шифров перестановки

Основное содержание

Шифры простой замены:

- Маршрутное шифрование
- Шифрование с помощью решеток
- Таблица Виженера

Кодовая реализация

Маршрутное шифрование

```
alph = [
    "а", "б", "в", "г", "д", "е", "ж", "з", "и", "к",
    "л", "м", "н", "о", "п", "р", "с", "т", "у", "ф",
    "х", "ц", "ч", "ш", "щ", "ъ", "ы", "ь", "э", "ю", "я"
]

str = "нельзя недооценивать противника"
password = "пароль"
dimension = (m = 5, n = 6)

function matrixify(str, dimension)
    matr = replace(str, " " => "")
    matr_chars = collect(matr)
    res = Vector{Vector{Char}}(undef, dimension.m)
    counter = 1

    for i in 1:dimension.m
        res[i] = matr_chars[counter:min(counter + dimension.n - 1, end)]
        counter += dimension.n
```

```
    end

    return res
end
```

Маршрутное шифрование

```
function matrixFill(matrix, dimension)

    while length(matrix[end]) < dimension.n
        push!(matrix[end], 'a')
    end

    return matrix
end

function pushindex(str, alph)
  indexes = Int[]
  for ch in collect(str)
      index = findfirst(==(string(ch)), alph)
      if index !== nothing
          push!(indexes, index)
      end
  end
  return indexes
end

function result(arr, indexes, dimension)
```

```
    result_str = ""
    map = sort([Dict(:item => indexes[i], :key => i) for i in 1:length(indexes)],
               by = x -> x[:item])

    print(map)

    for i in 1:dimension.n
        counter = map[i][:key]
        for j in 1:dimension.m
            char = arr[j][counter]
            result_str *= string(char)
        end
    end

    return uppercase(result_str)
end
```

Маршрутное шифрование

```
matrix = matrixFill(matrixify(str, dimension), dimension)
indexes = pushindex(password, alph)
result_str = result(matrix, indexes, dimension)

println(result_str)
end
```

Шифрование с помощью решеток

```
function rot90(matrix::Array{T,2}) where {T}
  return [matrix[j, end-i+1] for i in 1:size(matrix, 1), j in 1:size(matrix, 2)]
end

function encrypt_with_rails(input_text::String, cipher_key::String, matrix_dim::I
  base_matrix = fill("", matrix_dim, matrix_dim)
  encryption_matrix = fill("", 2 * matrix_dim, 2 * matrix_dim)
  char_sequence = 1
  result_text = ""
  sanitized_text = replace(input_text, " " => "")

  for i in 1:matrix_dim, j in 1:matrix_dim
    base_matrix[i, j] = string(char_sequence)
    encryption_matrix[i, j] = string(char_sequence)
    char_sequence += 1
  end

  for row in 1:(2*matrix_dim)
    for col in (2*matrix_dim):-1:1
      if encryption_matrix[row, col] == ""
        base_matrix = rot90(base_matrix)
        for i in 0:(matrix_dim-1), j in 0:(matrix_dim-1)
          encryption_matrix[row+i, col-j] = base_matrix[matrix_dim-i, matrix_dim-
        end
      end
    end
  end
```

Шифрование с помощью решеток

```
  char_sequence = 1
  used_positions = String[]
  for char in sanitized_text
    placed = false
    for i in 1:(2*matrix_dim), j in 1:(2*matrix_dim)
      if encryption_matrix[i, j] == string(char_sequence) && !placed
        position_key = string(i, ",", j)
        if !(position_key in used_positions)
          encryption_matrix[i, j] = string(char)
          push!(used_positions, position_key)
          placed = true
        end
      end
    end
    char_sequence += 1
    if char_sequence > matrix_dim^2
      char_sequence = 1
      empty!(used_positions)
    end
  end
```

Шифрование с помощью решеток

```
sorted_key = sort(collect(cipher_key))
  for key_char in sorted_key
    column_index = findfirst(==(key_char), cipher_key)
```

```
    for row in 1:(2*matrix_dim)
      cell = encryption_matrix[row, column_index]
      result_text *= cell != "" ? cell : " "
    end
  end

  return result_text
end

text="word"
key="key"
matrix_size = 2

println(encrypt_with_rails(text, key, matrix_size))
```

Таблица Виженера

```
function cipher_vigenere(msg::String, secret::String)
  alpha_range = 'a':'z'
  result_text = ""
  pos_in_key = 1

  for char in msg
    if isletter(char)
      shift = findfirst(==(secret[pos_in_key]), alpha_range) - 1
      char_pos = findfirst(==(char), alpha_range) + shift
      char_pos > 26 && (char_pos -= 26)
      result_text *= alpha_range[char_pos]
```

```
      pos_in_key += 1
      pos_in_key > length(secret) && (pos_in_key = 1)
    else
      result_text *= char
    end
  end

  return result_text
end

msg="hello"
secret="key"

println(cipher_vigenere(msg, secret))
```

Заключение

В данной лабораторной работе были реализованы шифры перестановки (Маршрутное шифрование, Шифрование с помощью решеток, Таблица Виженера)