Отчёт по лабораторной работе №3  
Математические основы защиты информации и информационной безопасности

Шифрование гаммированием

Выполнила: Коняева Марина Александровна,  
НФИмд-01-25, 1032259383

Содержание

# Теоретическое введение

**Шифр гаммирования** — это симметричный шифр, в котором каждый символ открытого текста объединяется с соответствующим символом гаммы (ключевой последовательности) с помощью операции сложения по модулю.

**Особенности шифра гаммирования:** - Относится к классу потоковых шифров - Криптостойкость зависит от длины и случайности гаммы - При использовании одноразового ключа (гаммы длиной не менее длины текста) обеспечивает абсолютную криптостойкость (шифр Вернама)

# Цель работы

Целью данной работы является изучение алгоритма шифрования гаммированием, принципа его работы и реализация на языке программирования Julia.

## Задание

1. Реализовать алгоритм шифрования гаммированием с заданной гаммой
2. Реализовать алгоритм дешифрования гаммированием
3. Протестировать работу алгоритма на примере

# Выполнение лабораторной работы

## Реализация шифра гаммирования

function gamma\_encrypt(text::Vector{Int}, gamma::Vector{Int}, mod\_value::Int=33)  
 textLen = length(text)  
 gammaLen = length(gamma)  
   
 # Формируем ключевое слово (растягиваем гамму на длину текста)  
 keyText = Int[]  
 for i in 1:(textLen ÷ gammaLen)  
 append!(keyText, gamma)  
 end  
 for i in 1:(textLen % gammaLen)  
 push!(keyText, gamma[i])  
 end  
   
 # Шифрование  
 encrypted = Int[]  
 for i in 1:textLen  
 result = (text[i] + keyText[i]) % mod\_value  
 # Если результат 0, заменяем на mod\_value (для корректной работы с 1-based индексами)  
 if result == 0  
 result = mod\_value  
 end  
 push!(encrypted, result)  
 end  
   
 return encrypted  
end  
  
function gamma\_decrypt(encrypted::Vector{Int}, gamma::Vector{Int}, mod\_value::Int=33)  
 encryptedLen = length(encrypted)  
 gammaLen = length(gamma)  
   
 # Формируем ключевое слово (растягиваем гамму на длину текста)  
 keyText = Int[]  
 for i in 1:(encryptedLen ÷ gammaLen)  
 append!(keyText, gamma)  
 end  
 for i in 1:(encryptedLen % gammaLen)  
 push!(keyText, gamma[i])  
 end  
   
 # Расшифрование  
 decrypted = Int[]  
 for i in 1:encryptedLen  
 result = (encrypted[i] - keyText[i]) % mod\_value  
 # Если результат отрицательный, добавляем mod\_value  
 if result <= 0  
 result += mod\_value  
 end  
 push!(decrypted, result)  
 end  
   
 return decrypted  
end

## Тестирование реализации

# Исходные данные  
text = [16, 17, 9, 11, 1, 8] # ПРИКАЗ  
gamma = [4, 1, 13, 13, 1] # ГАММА  
mod\_value = 33  
  
println("Исходный текст: $text")  
println("Гамма: $gamma")  
println("Модуль: $mod\_value")  
  
# Шифрование  
encrypted = gamma\_encrypt(text, gamma, mod\_value)  
println("Зашифрованный текст: $encrypted")  
  
# Ожидаемый результат  
expected = [20, 18, 22, 24, 2, 12]  
println("Ожидаемый результат: $expected")  
println("Результат совпадает: $(encrypted == expected)")  
  
# Расшифрование  
decrypted = gamma\_decrypt(encrypted, gamma, mod\_value)  
println("Расшифрованный текст: $decrypted")  
println("Исходный текст восстановлен: $(decrypted == text)")

## Результаты выполнения

Исходный текст: [16, 17, 9, 11, 1, 8]  
Гамма: [4, 1, 13, 13, 1]  
Модуль: 33  
Зашифрованный текст: [20, 18, 22, 24, 2, 12]  
Ожидаемый результат: [20, 18, 22, 24, 2, 12]  
Результат совпадает: true  
Расшифрованный текст: [16, 17, 9, 11, 1, 8]  
Исходный текст восстановлен: true

# Вывод

В данной лабораторной работе был успешно реализован алгоритм шифрования гаммированием на языке Julia. Алгоритм корректно выполняет как шифрование, так и дешифрование текста, что подтверждается тестовыми примерами. Особенностью реализации является корректная обработка операций по модулю и циклическое повторение гаммы для текстов произвольной длины.

# Список литературы

[1] Методические материалы курса.

[2] Wikipedia: Stream cipher (URL: https://en.wikipedia.org/wiki/Stream\_cipher)

[3] Официальная документация по языку Julia (URL: https://docs.julialang.org/)