Лабораторная работа №3

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

29 сентября 2024

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Прагматика выполнения

Прагматика выполнения

- Освоение шифров гаммированием, в частности шифрование конечной гаммой.

Цели

Цели

Изучить работу алгоритма шифрования гаммированием конечной гаммой, а также реализовать его программно.

Задачи

Задачи

1. Освоить и реализовать алгоритм шифрования гаммированием конечной гаммой на языке Julia.

Выполнение работы

Шифрование конечной гаммой на Julia (1/3)

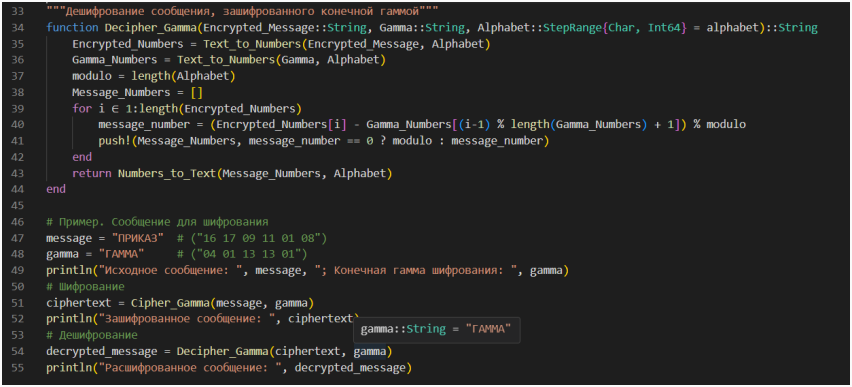
```julia
# Функция для преобразования строки в числовое представление на основе алфавитного порядка
alphabet = 'а':'я'
function Text_to_Numbers(Text::String, Alphabet::StepRange{Char, Int64} = alphabet)::Vector{Any}
    numbers = []
    for char in lowercase(Text)
        push!(numbers, findfirst(c -> c == char, Alphabet))
    end
    return numbers
end

# Функция для преобразования числового представления обратно в строку
function Numbers_to_Text(Numbers::Vector{Any}, Alphabet::StepRange{Char, Int64} = alphabet)::String
    text = ""
    for number in Numbers
        text *= alphabet[number]
    end
    return lowercase(text)
end

"""Шифрование конечной гаммой"""
function Cipher_Gamma(Message::String, Gamma::String, Alphabet::StepRange{Char, Int64} = alphabet)::String
    Message_Numbers = Text_to_Numbers(Message, Alphabet)
    Gamma_Numbers = Text_to_Numbers(Gamma, Alphabet)
    modulo = length(Alphabet)
    Encrypted_Numbers = []
    for i ∈ 1:length(Message_Numbers)
        encrypted_number = (Message_Numbers[i] + Gamma_Numbers[(i-1) % length(Gamma_Numbers) + 1]) % modulo
        push!(Encrypted_Numbers, encrypted_number == 0 ? modulo : encrypted_number)
    end
    return Numbers_to_Text(Encrypted_Numbers, Alphabet)
end

```

Шифрование конечной гаммой на Julia (2/3)



```julia
"""Дешифрование сообщения, зашифрованного конечной гаммой"""
function Decipher_Gamma(Encrypted_Message::String, Gamma::String, Alphabet::StepRange{Char, Int64} = alphabet)::String
    Encrypted_Numbers = Text_to_Numbers(Encrypted_Message, Alphabet)
    Gamma_Numbers = Text_to_Numbers(Gamma, Alphabet)
    modulo = length(Alphabet)
    Message_Numbers = []
    for i ∈ 1:length(Encrypted_Numbers)
        message_number = (Encrypted_Numbers[i] - Gamma_Numbers[(i-1) % length(Gamma_Numbers) + 1]) % modulo
        push!(Message_Numbers, message_number == 0 ? modulo : message_number)
    end
    return Numbers_to_Text(Message_Numbers, Alphabet)
end

# Пример. Сообщение для шифрования
message = "ПРИКАЗ"  # ("16 17 09 11 01 08")
gamma = "ГАММА"     # ("04 01 13 13 01")
println("Исходное сообщение: ", message, "; Конечная гамма шифрования: ", gamma)
# Шифрование
ciphertext = Cipher_Gamma(message, gamma)
println("Зашифрованное сообщение: ", ciphertext)
# Дешифрование
decrypted_message = Decipher_Gamma(ciphertext, gamma)
println("Расшифрованное сообщение: ", decrypted_message)
```

gamma::String = "ГАММА"

Рис. 2: Код реализации алгоритма дешифрования сообщения, зашифрованного конечной гаммой на Julia

```
PS C:\Users\User\Documents\work\study\2024-2025\Математические
thbase-infosec\labs\lab03\report\report> julia .\gamma.jl
Исходное сообщение: ПРИКАЗ; Конечная гамма шифрования: ГАММА
Зашифрованное сообщение: усхчбл
Расшифрованное сообщение: приказ
```

Рис. 3: Результат кода реализации алгоритма шифрования конечной гаммой на Julia

Результаты

Результаты

По результатам работы, я изучил работу алгоритма шифрования гаммированием конечной гаммой, а также реализовал его программно.