Отчет по лабораторной работе №3

дисциплина: Математические основы защиты информации и информационной безопасности

Морозова Ульяна Константиновна

Содержание

1 Цель работы

Целью работы является изучение алгоритмов шифрования гаммированием и реализация его на языке Julia.

2 Выполнение лабораторной работы

2.1 Шифрование гаммированием

Шифрование гаммированием (в англоязычной версии — stream cipher) — метод симметричного шифрования, при котором последовательность случайных символов (гамма) накладывается на открытый текст. Гамма вырабатывается по определённому алгоритму и используется для шифровки открытых данных и дешифровки шифротекста.

Принцип работы - Генерация гаммы. Можно использовать генератор псевдослучайных чисел или аппаратный источник случайных чисел. Длина гаммы должна быть не меньше длины защищаемого сообщения (открытого текста). - Наложение гаммы на открытый текст. Процедура может быть различной: например, символы исходного текста и гаммы заменяются цифровыми эквивалентами, которые затем складываются или вычитаются, или символы представляются в виде двоичного кода, затем соответствующие разряды складываются по модулю 2 (XOR). - Дешифрование — повторная генерация гаммы и наложение гаммы на зашифрованные данные.

Далее приведена реализация шифра на языке Julia.

```
using Random

function gamma_cipher(text::AbstractString, key::AbstractString; encrypt::Bool=true)
    # Преобразуем текст и ключ в массивы байтов
    text_bytes = Vector{UInt8}(text)
```

```
    key_bytes = Vector{UInt8}(key)

    # Если ключ короче текста, повторяем его
    if length(key_bytes) < length(text_bytes)
        key_bytes = repeat(key_bytes, ceil(Int, length(text_bytes) / length(key_bytes)
        key_bytes = key_bytes[1:length(text_bytes)]
    end

    result = Vector{UInt8}(undef, length(text_bytes))

    # Применяем операцию XOR между текстом и гаммой
    for i in 1:length(text_bytes)
        result[i] = text_bytes[i] xor key_bytes[i]
    end

    return String(result)
end

function generate_key(length::Int)
    return String(rand(UInt8, length))
end

function demo()
    # println("=== Демонстрация гаммирования с конечной гаммой ===\n")

    # Исходный текст
    original_text = "Shake it to the max"
    println("Исходный текст: $original_text")
```

```
    # Генерация ключа
    key = generate_key(16)
    println("Ключ (hex): $(bytes2hex(Vector{UInt8}(key)))")

    # Шифрование
    encrypted = gamma_cipher(original_text, key, encrypt=true)
    println("Зашифрованный текст (hex): $(bytes2hex(Vector{UInt8}(encrypted)))")

    # Дешифрование
    decrypted = gamma_cipher(encrypted, key, encrypt=false)
    println("Дешифрованный текст: $decrypted")

    # Проверка
    println("Проверка: $(original_text == decrypted)")
end
```

Результат работы шифра:

```
julia> demo()
Исходный текст: Shake it to the max
Ключ (hex): f556d3c95ed9ed5a6b232ad172eb8699
Зашифрованный текст (hex): a63eb2a23bf9842e4b5745f10683e3b99837ab
Дешифрованный текст: Shake it to the max
Проверка: true

julia> demo()
Исходный текст: All I want for Christmas is you!
Ключ (hex): b82a3af92aa5d3026365c938858f79a7
Зашифрованный текст (hex): f94656d96385a4630d11e95eeafd59e4d058538a5ec8b271430cba18fce
Дешифрованный текст: All I want for Christmas is you!
Проверка: true
```

3 Выводы

Мы изучили работу алгоритмов шифрования гаммированием, а также реализовали его на языке Julia.