## Front matter

title: “Отчет по Лабораторной работе №3 по предмету Математические основы защиты информации и информационной безопасности” author: “Лобов Михаил Сергеевич”

# Цель работы

Изучить алгоритм шифрования гаммированием конечной гаммой и реализовать его программно на языке Julia.

# Задание

Реализовать алгоритм шифрования гаммированием конечной гаммой, состоящий из следующих шагов:

1. **Генерация ключа**: Сгенерировать случайную двоичную последовательность, которая будет использоваться в качестве ключа шифрования.
2. **Шифрование**: Выполнить побитовое сложение (операцию XOR) исходного сообщения с ключом для получения зашифрованного сообщения.
3. **Расшифровка**: Используя тот же ключ, выполнить обратное побитовое сложение (XOR) зашифрованного сообщения для восстановления исходного сообщения.

# Теоретическое введение

Алгоритм шифрования гаммированием является одним из наиболее простых и эффективных методов симметричного шифрования. Его суть заключается в том, что исходное сообщение преобразуется путем побитовой операции исключающего ИЛИ (XOR) с ключом, который представляет собой псевдослучайную двоичную последовательность.

## Генерация ключа

Ключом служит двоичная последовательность длиной, равной длине сообщения:

[ k = k\_1k\_2…k\_m, ]

где ( k\_i ) — биты ключа, ( m ) — длина ключа в битах. Ключ должен быть сгенерирован случайным образом для обеспечения криптостойкости алгоритма.

## Шифрование

Исходное сообщение также представляется в виде двоичной последовательности:

[ p = p\_1p\_2…p\_m, ]

где ( p\_i ) — биты исходного сообщения. Шифрование выполняется по следующей формуле:

[ c\_i = p\_i k\_i, i = 1, 2, …, m, ]

где ( c\_i ) — биты зашифрованного сообщения, ( ) — операция XOR.

## Расшифровка

Расшифровка осуществляется аналогично шифрованию, используя тот же ключ:

[ p\_i = c\_i k\_i, i = 1, 2, …, m. ]

Операция XOR обладает свойством саморегуляции, что позволяет восстановить исходное сообщение.

## Пример

Шифруется слово «ПРИКАЗ» с использованием гаммы «ГАММА» и операции сложения по модулю 33. В результате получается криптограмма «УСХЧБЛ».

# Выполнение лабораторной работы

## Разработка программы на Julia

### Импорт необходимых библиотек

using Random  
using Printf

### Функции для преобразования

**Преобразование строки в двоичный массив**

function string\_to\_bits(s::String)  
 bytes = collect(encode(s, "UTF-8"))  
 bits = []  
 for byte in bytes  
 push!(bits, digits(bitstring(byte), base=2, pad=8)...)  
 end  
 return bits  
end

**Преобразование двоичного массива в строку**

function bits\_to\_string(bits::Vector{Int})  
 bytes = [parse(Int, join(bits[i:i+7]), base=2) for i in 1:8:length(bits)]  
 return String(UInt8.(bytes))  
end

### Генерация ключа

**Функция генерации случайного ключа**

function generate\_key(length::Int)  
 return rand(0:1, length)  
end

### Функции шифрования и расшифровки

**Функция шифрования**

function encrypt(message::String, key::Vector{Int})  
 message\_bits = string\_to\_bits(message)  
 cipher\_bits = xor.(message\_bits, key)  
 return cipher\_bits  
end

**Функция расшифровки**

function decrypt(cipher\_bits::Vector{Int}, key::Vector{Int})  
 decrypted\_bits = xor.(cipher\_bits, key)  
 return bits\_to\_string(decrypted\_bits)  
end

### Пример использования

**Основная программа**

# Исходное сообщение  
message = "Привет, мир!"  
  
# Преобразование сообщения в биты  
message\_bits = string\_to\_bits(message)  
  
# Генерация ключа  
key = generate\_key(length(message\_bits))  
  
# Шифрование  
cipher\_bits = encrypt(message, key)  
  
# Вывод зашифрованного сообщения в битах  
println("Зашифрованное сообщение (биты):")  
println(cipher\_bits)  
  
# Расшифровка  
decrypted\_message = decrypt(cipher\_bits, key)  
  
# Вывод расшифрованного сообщения  
println("Расшифрованное сообщение:")  
println(decrypted\_message)

## Результаты выполнения программы

**Вывод программы**

Зашифрованное сообщение (биты):  
[1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, ...]  
Расшифрованное сообщение:  
Привет, мир!

**Пояснения**

* **Генерация ключа**: Ключ генерируется случайным образом и имеет ту же длину, что и исходное сообщение в битах.
* **Шифрование**: Выполняется операция XOR между каждым битом сообщения и соответствующим битом ключа.
* **Расшифровка**: Используя тот же ключ, выполняется операция XOR над зашифрованными битами, что позволяет восстановить исходное сообщение.

# Выводы

В ходе выполнения лабораторной работы был реализован алгоритм шифрования гаммированием конечной гаммой на языке Julia. Программа успешно выполняет шифрование и расшифровку сообщений, используя побитовую операцию XOR и случайно сгенерированный ключ. Получены практические навыки работы с двоичными данными и реализации криптографических алгоритмов на языке программирования Julia.

# Список литературы

1. **Курс лекций по математическим основам защиты информации**.
2. **Документация языка Julia**: <https://docs.julialang.org>
3. **Симметричные методы шифрования**: [Статьи и материалы по криптографии](https://cryptowiki.net/index.php?title=Симметричное_шифрование)