## Front matter

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

## Generic otions

lang: ru-RU toc-title: “Содержание”

# Цель работы

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

# Задание

1. Сгенерировать случайную двоичную последовательность, которая будет использоваться в качестве шифрования
2. Выполнить сложение индексов букв шифруемой строки и ключа для получения зашифрованного 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

Ключем является последовательность длинной, равной длинне сообщения.

[ k = k\_1,k\_2…k\_m ]

(k\_i ) - биты ключа, **m** - длинна ключа. Ключ генерируется случайно.

## Непосредственно шифрование

Исходное сообщение также представляется в виде двоичной последовательности:

[ p = p\_1,p\_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.

# Выполнение лабораторной работы

## Написаны программы на языке Julia.

function encrypt(plaintext::String, key::String)   
 plaintext = filter(c -> c in russian\_letters, uppercase(plaintext))  
 key = filter(c -> c in russian\_letters, uppercase(key))  
   
   
 plaintext\_nums = [letter\_to\_num[c] for c in plaintext]   
 key\_nums = [letter\_to\_num[c] for c in key]

1. Убираем не буквы, делаем буквы большими
2. Сопоставляем буквам сообщения число по алфавиту
3. Сопоставляем буквам ключа число по алфавиту

## Непосредственно шифрование

m = length(plaintext\_nums)  
 key\_nums\_extended = [key\_nums[(i-1) % length(key\_nums) + 1] for i in 1:m]   
   
 cipher\_nums = [(plaintext\_nums[i] + key\_nums\_extended[i]) % 32 for i in 1:m]   
 cipher\_nums = [num == 0 ? 32 : num for num in cipher\_nums]   
   
 cipher\_text = [num\_to\_letter[num] for num in cipher\_nums]   
 return join(cipher\_text)   
end

1. Делаем длинну ключа и строки одинаковой
2. Суммируем номера букв сообщения и ключа, затем делим с остатком
3. В зависимости от нового индекса получаем новые буквы сообщения

## Пример

plaintext = "ПРИКАЗ"  
key = "ГАММА"  
  
ciphertext = encrypt(plaintext, key)  
println("Сообщение: $plaintext")  
println("Ключ: $key")  
println("Зашифрованное Сообщение: $ciphertext")

Сообщение: ПРИКАЗ  
Ключ: ГАММА  
Зашифрованное Сообщение: УСХЧБЛ

# Выводы

В ходе выполнения лабораторной работы №3 был реализован алгоритм шифрования граммированием конечной гаммой на языке Julia. Программа успешно выполняет шифрование сообщения, как видно из примера.

# Список литературы

::: {#[Лабораторная\_работа\_3](https://esystem.rudn.ru/mod/folder/view.php?id=1150970)}