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)}