Лабораторная работа №3

Шифрование гаммированием

Хохлачева Яна, НПМмд-02-22

Содержание

# 1 Цель работы

Знакомство с шифрованием гаммированием на примере гаммирования конечной гаммой.

# 2 Задание

Реализовать алгоритм шифрования гаммированием конечной гаммой.

# 3 Теоретическое введение

Гаммирование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле.

При шифровании гаммированием формируется m - разрядная случайная последовательность. Пусть k - передаваемое сообщение

а p - последовательность, которая является ключом:

тогда i-ый символ криптограммы будет равен:

где - операция побитового сложения XOR [1]. В результате криптограмму можно записать следующим образом:

Более подробно о шифровании гаммированием: [2]

# 4 Выполнение лабораторной работы

## 4.1 Описание реализации метода шифрования

В данной работе применяется схема однократного использования (рис. 1). К элементам ключа и исходного сообщения применяется побитовое сложение XOR, в результате чего формируется зашифрованное сообщение:

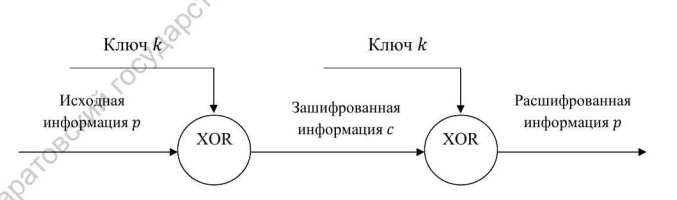


Рис. 1: Схема однократного использования

Для того, чтобы применить операцию побитового сложения, необходимо, чтобы ключ и исходное сообщение были одной длины. Для достижения данной цели, ключ растягивается до тех пор, пока не сравняется длиной с исходным сообщением следующим образом: пусть сообщение будет

длина ‘m’ которого равна 12, тогда ключ растягивается следующим образом:

Таким образом, к сообщению и ключу одинаковой длины можно применить операцию побитового сложения XOR.

## 4.2 Листинг

alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"  
ignore\_symbols = "!.,@#$%^&\*()-+={}[]<>/ "  
  
def gamma\_shifr(gamma, message\_length):  
 new = ""  
 for i in range(message\_length):  
 new += gamma[i % len(gamma)]  
 return new  
  
def get\_indexes(text):  
 indexes = []  
 for letter in text:  
 try:  
 indexes.append(alphabet.index(letter) + 1)  
 except ValueError:  
 indexes.append(ignore\_symbols.index(letter) + 10000)  
 return indexes  
  
def encrypt(message\_indexes, gamma\_indexes):  
 indexes = []  
 for ix, item in enumerate(message\_indexes):  
 if message\_indexes[ix] > 10000 or gamma\_indexes[ix] > 10000:  
 result = item  
 indexes.append(result)  
 continue  
 result = (message\_indexes[ix] + gamma\_indexes[ix]) % len(alphabet)  
 indexes.append(result)  
 return indexes  
  
def to\_text(indexes):  
 text = ""  
 for index in indexes:  
 if index > 10000:  
 text += ignore\_symbols[index - 10000]  
 continue  
 text += alphabet[index - 1]  
 return text  
  
message = input("Введите сообщение: ")  
gamma = input("Введите ключ (гамма): ")  
new = gamma\_shifr(gamma, len(message))  
print("\nПреобразование {} -> {}".format(gamma.upper(), new.upper()))  
message\_indexes = get\_indexes(message)  
print("\nВаше сообщение:\n{} ({})".format(message.upper(), message\_indexes))  
gamma\_indexes = get\_indexes(new)  
print("\nВаша гамма:\n{} ({})".format(gamma.upper(), gamma\_indexes))  
encrypted\_message\_indexes = encrypt(message\_indexes, gamma\_indexes)  
encrypted\_message = to\_text(encrypted\_message\_indexes)  
print("\nЗашифрованное сообщение:\n{} ({})"  
 .format(encrypted\_message.upper(), encrypted\_message\_indexes))

## 4.3 Полученные результаты

При запуске программы пользователю предлагается ввести сообщение, которое необходимо зашифровать, и ключ. После этого, если ключ и сообщения разной длины, из ключа исключаются лишние символы или добавляются в конец новые. Знаки препинания и пробелы, в свою очередь, игнорируются и остаются в неизменном виде. Для удобства, они нумеруются начиная с индекса 10000. В результате выполнения пользователь получает зашифрованное сообщение (рис. 2).

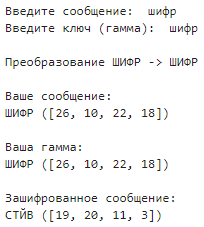


Рис. 2: Результаты шифрования гаммированием

# 5 Выводы

Таким образом в процессе лабораторной работы изучено и реализовано шифрование гаммирования конечной гаммой.

# Список литературы

1. Исключающее «или» [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Исключающее_«или»>.

2. Шифр табличной маршрутной перестановки [Электронный ресурс]. Википедия, 2020. URL: <https://ru.wikipedia.org/wiki/Гаммирование>.