Лабораторная работа №3

Шифр гаммирования

Дугаева Светлана Анатольевна

15 октября 2022

Российский университет дружбы народов, Москва, Россия

Цель работы

Цель работы

- Цель данной лабораторной работы изучение реализация алгоритма шифрования гаммированием.

Теоретическое введение

Гамми́рование, 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 [@cypher].

В этом способе шифрование выполняется путем сложения символов исходного текста и ключа по модулю, равному числу букв в алфавите. Если в исходном алфавите, например, 33 символа, то сложение производится по модулю 33. Такой процесс сложения исходного текста и ключа называется в криптографии **наложением гаммы** [@intuit].

Выполнение лабораторной работы

Выполнение лабораторной работы

Для реализации шифров мы будем использовать Python, так как его синтаксис позволяет быстро реализовать необходимые нам алгоритмы.

Реализация маршрутного шифрования

В качестве начальных значений берется гамма “гамма”. Алфавитом может быть любая строка неповторяющихся символов. Я использую кириллицу. Также задаю строку сообщение, которое будет шифроваться.

```
import numpy as np
text = 'приказ'
key = 'гамма'
a = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
```

Рис. 1: Начальные значения для шифрования гаммированием

Реализация маршрутного шифрования

Реализация маршрутного шифрования

Задам функцию *code()*, в качестве параметров передаются заданные начальные данные. Внутри функции ключ-гамма, алфавит и сообщение преобразую в массив. Затем увеличу длину ключа-гаммы, чтобы число символов совпадало с сообщением, делаю это дописывая ключ пока длина не будет равной или больше сообщению, лишние символы отсекаю. Затем нахожу индексы символов сообщения и ключа в алфавите и сохраняю их в массиве. В новый массив сохраняю символы, рассчитав индексы по формуле $z = x + k \ (mod\ N)$. Полученный массив преобразую в строку и возвращаю.

Реализация маршрутного шифрования

Реализация маршрутного шифрования

```
def code(t, k, a):
    t = list(t)
    k = list(k)
    a = list(a)
    n = len(a)
    while len(k)<len(t):
        k += k
    k = k[:len(t)]
    res1 = []
    for i in range(len(t)):
        for j in range(n):
            if t[i] == a[j]:
                res1.append(j)
    res2 = []
    for i in range(len(k)):
        for j in range(n):
            if k[i] == a[j]:
                res2.append(j)
    res = []
    for i in range(len(res1)):
        res.append(a[(res1[i]+res2[i])%n+1])
    return res
```

Полученный результат

Выведу результат работы программы для заданных начальных значений.

```
print(code(text, key, a))
```

```
['у', 'с', 'ц', 'ш', 'б', 'л']
```

Рис. 3: Результат выполнения программы

Выводы

Выводы

В ходе данной лабораторной работы я реализовала алгоритм шифрования гаммированием.