Лабораторная работа №8

Основы информационной безопасности

Кондрашова Анастасия Андреевна

Содержание

Список иллюстраций

Список таблиц

1 Теоретическое введение

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте. Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком ⊠) между элементами гаммы и элементами подлежащего сокрытию текста.

2 Цель работы

Освоить на практике применение однократного гаммирования при работе с различными текстами на одном ключе.

3 Выполнение лабораторной работы

Лабораторная работа выполнена на языке Python 3 в среде Jupiter Notebook.

1. Создаём функцию, которая осуществляет однократное гаммирование посредством побитового XOR

```
def cript(text, key):
    if len(text)!= len(key):
        return "Error: key must be the same len as text"
    result = ''
    for i in range(len(key)):
        p = ord(text[i]) ^ ord(key[i])
        result += chr(p)
    return result
```

Рис. 3.1: Функция шифрования

2. Задаём две равные по длине текстовые строки и создаём случайный символьный ключ такой же длины

```
text1 = "С новым годом, друзья!"
text2 = "С днем рождения тебя!!"
```

```
from random import randint, seed
seed(31)
key = ''
for i in range(len(text1)):
    key += chr(randint(0,5000))
print(key)
```

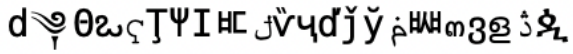

Рис. 3.2: Исходные данные

3. Осуществляем шифрование двух текстов по ключу с помощью написанной функции

```
cipher1 = cript(text1, key)
cipher2 = cript(text2, key)
print(cipher1, cipher2, sep="\n")
```

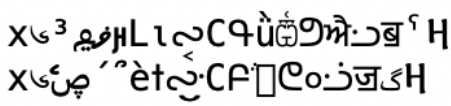

Рис. 3.3: Шифрование данных

4. Создаём переменную, которая, прогнав два шифрованных текста через побитовый XOR, поможет злоумышленнику получить один текст, зная другой, без ключа

```
In [32]: zlo = cript(cipher1, cipher2)

In [33]: print(cript(zlo, text1))
         С днем рождения тебя!!

In [34]: print(cript(zlo, text2))
         С новым годом, друзья!
```

Рис. 3.4: Получение данных без ключа

5. Таким же способом можно получить часть данных

```
In [35]: text2[7:15]
Out[35]: 'рождения'

In [29]: zlo_part = cript(cipher1[7:15], cipher2[7:15])
         print(cript(zlo_part, text2[7:15]))
         годом,
```

Рис. 3.5: Получение части данных

4 Выводы

Я освоила на практике применение режима однократного гаммирования при работе с несколькими текстами.