Отчёт по лабораторной работе №8

Дисциплина: Основы информационной безопасности

Балакирева Дарья Сергеевна, НПМбд-01-19б

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 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.

3 Выполнение лабораторной работы

Создаём функцию, которая осуществляет однократное гаммирование посредством побитового XOR (рис. 3.1)

```
def cript(text,key):
    if len(text) != len(key):
        return " Ошибка: ключь должен быть той же длины, что и текст"
    result = ''
    for i in range(len(key)):
        n = ord(text[i] ^ ord(key[i]))
        result += chr(n)
        return result
```

Рис. 3.1: Функция шифрования

Задаём две равные по длине текстовые строки и создаём случайный символьный ключ такой же длины (рис. 3.2)

```
text1 = "С Новым годом,друзья!!"
text2 = "Хорошего дня всем вам!"
```

```
from random import randint, seed
seed(21)
key = ''
for i in range(len(text1)):
    key += chr(randint(0,5000))
print(key)
```

Рис. 3.2: Исходные данные

Осуществляем шифрование двух текстов по ключу с помощью написанной функции (рис. 3.3)

```
shifr1 = cript(text1, key)
shifr2 = cript(text2, key)
print(shifr1, shifr2, sep="\n")
```

```
Ŧl?Ↄ੦ϚĠϘ·ᐊϲγЦ_ ι_Ž
Ţ फ ठ ᘓ ᘎ˜ 8U· ⸗ɔϛ:ç###ɋŭŽ
```

Рис. 3.3: Шифрование данных

Создаём переменную, которая, прогнав два шифрованных текста через побитовый XOR, поможет злоумышленнику получить один текст, зная другой, без ключа (рис. 3.4)

```
shifr12 = cript(shifr1, shifr2)
```

```
print(cript(shifr12, text1))
```

```
Хорошего дня всем вам!
```

```
print(cript(shifr12, text2))
```

```
С Новым годом,друзья!!
```

Рис. 3.4: Получение данных без ключа

Таким же способом можно получить часть данных (рис. 3.5)

```
text2[9:17]
```

```
'дня всем'
```

```
shifr12_part = cript(shifr1[8:14],shifr2[8:14])
print(cript(shifr12_part, text2[8:14]))
```

```
годом,
```

Рис. 3.5: Получение части данных

4 Выводы

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

Список литературы