

# **Отчёт по лабораторной работе №8**

**Дисциплина: Основы информационной безопасности**

Полиенко Анастасия Николаевна, НПМбд-01-19

# Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
4	Выводы	9
	Список литературы	10

## Список иллюстраций

3.1	Функция шифрования . . . . .	6
3.2	Исходные данные . . . . .	7
3.3	Шифрование данных . . . . .	7
3.4	Получение данных без ключа . . . . .	8
3.5	Получение части данных . . . . .	8

# 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  $\boxplus$ ) 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 crypt(text, key):  
    if len(text) != len(key):  
        return "Error: key must be the same lenght 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

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

```
text1 = "С Новым годом, друзья!"
text2 = "С днём рождения тебя!!"
```

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

[illegible]

Рис. 3.2: Исходные данные

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

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

ॐ नमो भगवते वासुदेवाय ॥ ॐ नमो भगवते वासुदेवाय ॥  
 ॐ नमो भगवते वासुदेवाय ॥ ॐ नमो भगवते वासुदेवाय ॥

Рис. 3.3: Шифрование данных

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

```
zlo = cript(cipher1, cipher2)
```

```
print(cript(zlo, text1))
```

С днём рождения тебя!!

```
print(cript(zlo, text2))
```

С Новым годом, друзья!

Рис. 3.4: Получение данных без ключа

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

```
text2[7:15]
```

'рождения'

```
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.

## **Список литературы**