Лабораторная работа №7

Элементы криптографии. Однократное гаммирование

Дугаева Светлана Анатольевна

Содержание

Список иллюстраций

Цель работы

Освоить на практике применение режима однократного гаммирования.

Выполнение лабораторной работы

1. Сначала подключим нужные библеотеки. Затем в переменную hi запишем наше сообщение, которое требуется зашифровать. Наиболее удобным вариантом будет написать функцию (crypto), которая принимет на вход открытое сообщение(словами). После этого переведем сообщение в 16ричную систему исчисления(для этого используется библеотека sys). Создадим случайный ключ(выбор случайного значения от 0 до 255, и затем переведем ключ в 16ричную систему счисления. В дополнение я перевела ключ в символьный вид. Наложим гамму, то есть выполним операцию сложения по модулю 2(XOR) между элементами полученного случайного ключа и элементами подлежащего сокрытию текста Для получания шифротекста переведем значение из 16ричной системы в символную. (рис. @fig:001):

```python
import numpy as np
import sys
import operator as op
```

```python
hi = "С Новым Годом, друзья!"
```

```python
def crypto (message):
    tmp1 = []
    for i in message:
        tmp1.append(i.encode("cp1251").hex())
    print("Открытое сообщение в 16ричной системе:", tmp1)
    d = np.random.randint(0, 255, len(message))
    key = [hex(i)[2:] for i in d]
    print("Ключ в 16ричной системе:", key)
    key_d = bytearray.fromhex("".join(key)).decode("cp1251")
    print("Текст ключа:", key_d)
    xor = []
    for i in range(len(tmp1)):
        xor.append("{:02x}".format(int(key[i],16) ^ int(tmp1[i], 16)))
    print("Шифротекст в 16ричной системе:", xor)
    code = bytearray.fromhex("".join(xor)).decode("cp1251")
    print("Текст шифра:", code)
    return code
```

Рис. 0.1: Функция кодирования

2. Вызовем функцию crypto, она возвращает закодированное сообщение в символьном виде, оно нам понадобится для следущего задания. Также именно на скриншоте можем увидеть все, что выводит функция (рис. @fig:002):

```python
code = crypto(hi)
```

```
Открытое сообщение в 16ричной системе: ['d1', '20', 'cd', 'ee', 'e2', 'fb', 'ec', '20', 'c3', 'ee', 'e4', 'ee', 'ec', '2c', '20', 'e4', 'f0', 'f3', 'e7', 'fc', 'ff', '21']
Ключ в 16ричной системе: ['c2', 'c', '1', '7c', '7', '8d', '9', 'e1', '47', '3d', 'c4', 'a', '1', '1d', 'f7', '3e', '7e', '7f', '2d', 'b5', '84', 'd0']
Текст ключа: ВБ|хЩбG=ДЎ?ч>~?-µ„Р
Шифротекст в 16ричной системе: ['13', '2c', 'cc', '92', 'e5', '76', 'e5', 'c1', '84', 'd3', '20', 'e4', 'ed', '31', 'd7', 'da', '8e', '8c', 'ca', '49', '7b', 'f1']
Текст шифра: ?,М’eveБ„У дн1ЧЬЋЊKI{с
```

Рис. 0.2: Вызов функции crypto

3. Напишем функция для нахождения ключа для декодирования сообщения. На вход принимается открытое сообщение и зашифрованное. Для начала переведем оба сообщения в 16ричную систему счисления. А затем снова выполним

операцию гаммирования, т.е. сложим по модулю 2. Таким образом получим ключ. Затем для наглядности переведем ключ в символьный вид (рис. @fig:003):

```
def decoder (message, code):
    tmp1 = []
    for i in message:
        tmp1.append(i.encode("cp1251").hex())
    tmp2 =[]
    for i in code:
        tmp2.append(i.encode("cp1251").hex())
    xor = [hex(int(k,16)^int(j,16))[2:] for (k,j) in zip(tmp1, tmp2)]
    print("Нашли ключ в 16ричной системе:", xor)
    xor_dec = bytearray.fromhex("".join(xor)).decode("cp1251")
    print("Текст ключа:", xor_dec)
```

Рис. 0.3: Функция для получения ключа

4. Вызовем функию decoder, передадим ей открытый ключ и шифротекст в символьном виде. На эран выводится ключ в 16ричной системе счисления и в символьном виде (рис. @fig:004):

```
decoder(hi, code)
```

```
Нашли ключ в 16ричной системе: ['c2', 'c', '1', '7c', '7', '8d', '9', 'e1', '47', '3d', 'c4', 'a', '1', '1d', 'f7', '3e', '7e',
'7f', '2d', 'b5', '84', 'd0']
Текст ключа: ВБ|хЩбG=ДЎ□ч>~□-µ„Р
```

Рис. 0.4: Вызов функции decoder

Выводы

Освоила на практике применение режима однократного гаммирования. Закодировала сообщение с помощью создания случайного ключа и нашла ключ исходя из открытого текста и шифротекста.