Лабораторная работа №8

Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

Дугаева Светлана Анатольевна

Содержание

Список иллюстраций

Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Выполнение лабораторной работы

1. Сначала подключим нужные библеотеки. Затем в переменные hi1 и hi2 запишем 2 наших сообщения, которые требуется зашифровать. Наиболее удобным вариантом будет написать функцию (crypto), которая принимет на вход 2 открытых сообщения(словами). После этого переведем оба сообщения 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. Наложим гамму, то есть выполним операцию сложения по модулю 2(XOR) между элементами полученного случайного ключа и элементами подлежащего сокрытию текста (проведем данную операцию 2 раза, поскольку у нас 2 сообщения). 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
hi1 = "С Новым Годом, друзья!"
```

```python
hi2 = "До НГ осталось 15 дней"
```

```python
def crypto (mes1, mes2):
    tmp1 = []
    for i in mes1:
        tmp1.append(i.encode("cp1251").hex())
    print("Открытое сообщение 1 в 16ричной системе:", tmp1)
    tmp2 = []
    for i in mes2:
        tmp2.append(i.encode("cp1251").hex())
    print("Открытое сообщение 2 в 16ричной системе:", tmp2)
    d = np.random.randint(0, 255, len(mes1))
    key = [hex(i)[2:] for i in d]
    print("Ключ в 16ричной системе:", key)
    xor1 = []
    for i in range(len(tmp1)):
        xor1.append("{:02x}".format(int(key[i],16) ^ int(tmp1[i], 16)))
    print("Шифротекст 1 в 16ричной системе:", xor1)
    code1 = bytearray.fromhex("".join(xor1)).decode("cp1251")
    print("Текст шифра 1:", code1)
    xor2 = []
    for i in range(len(tmp2)):
        xor2.append("{:02x}".format(int(key[i],16) ^ int(tmp2[i], 16)))
    print("Шифротекст 2 в 16ричной системе:", xor2)
    code2 = bytearray.fromhex("".join(xor2)).decode("cp1251")
    print("Текст шифра 2:", code2)

    return code1, code2
```

Рис. 0.1: Функция кодирования

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

```python
code1, code2 = crypto(hi1, hi2)
```

```
Открытое сообщение 1 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']
Открытое сообщение 2 в 16ричной системе: ['c4', 'ee', '20', 'cd', 'c3', '20', 'ee', 'f1', 'f2', 'e0', 'eb', 'ee', 'f1', 'fc',
'20', '31', '35', '20', 'e4', 'ed', 'e5', 'e9']
Ключ в 16ричной системе: ['bf', 'c5', '92', 'bb', 'd6', '46', 'fc', '9a', '5c', 'ee', 'ec', 'bf', '47', '54', 'b2', 'aa', 'ba',
'95', '0', '7a', 'c', 'ae']
Шифротекст 1 в 16ричной системе: ['6e', 'e5', '5f', '55', '34', 'bd', '10', 'ba', '9f', '00', '08', '51', 'ab', '78', '92', '4
e', '4a', '66', 'e7', '86', 'f3', '8f']
Текст шифра 1: ne_U4S⊡єцQ«x’NJfзtyЦ
Шифротекст 2 в 16ричной системе: ['7b', '2b', 'b2', '76', '15', '66', '12', '6b', 'ae', '0e', '07', '51', 'b6', 'a8', '92', '9
b', '8f', 'b5', 'e4', '97', 'e9', '47']
Текст шифра 2: {+Iv⊡f⊡k®⊡⊡Q¶Ё’›Ццд—йG
```

Рис. 0.2: Вызов функции crypto

3. Напишем функцию decoder для нахождения ключа для декодирования 2го сообщения. На вход принимается 2 зашифрованных сообщения и одно открытое. Для начала переведем оба зашифрованных сообщения и одно открытое в 16ричную систему счисления. А затем снова выполним операцию гаммирования, т.е. сложим по модулю 2 между 2мя зашифрованными сообщениями, а потом проделаем ту же операцию между полученным резьтатом и открытым сообщением. Таким образом получим второе открытое сообщение в 16 системе сичисления. Затем для наглядности переведем данное сообщение в символьный вид (рис. @fig:003):

```
def decoder (code1, code2, mes1):
    tmp1 = []
    for i in code1:
        tmp1.append(i.encode("cp1251").hex())
    print("Шифротекст 1 в 16ричной системе:", tmp1)
    tmp2 = []
    for i in code2:
        tmp2.append(i.encode("cp1251").hex())
    print("Шифротекст 2 в 16ричной системе:", tmp2)
    tmp3 = []
    for i in mes1:
        tmp3.append(i.encode("cp1251").hex())
    print("Открытое сообщение 1 в 16ричной системе:", tmp3)
    xor = []
    result = []
    for i in range(len(tmp2)):
        xor.append("{:02x}".format(int(tmp1[i],16) ^ int(tmp2[i], 16)))
        result.append("{:02x}".format(int(xor[i],16) ^ int(tmp3[i], 16)))
    print("Открытое сообщение 2 в 16ричной системе:", result)
    mes2 = bytearray.fromhex("".join(result)).decode("cp1251")
    print("Открытое сообщение 2:", mes2)
```

Рис. 0.3: Функция для получения второго открытого сообщения

4. Вызовем функию decoder, передадим ей 2 зашифрованных сообщения и одно открытое в символьном виде. На эран выводятся сообщения в 16ричной системе счисления, а так же полученное открытое сообщение в символьном виде. Декодировка произведена верно, сообщение совпадает с изначальным (рис. @fig:004):

```
1 decoder(code1, code2, hi1)
```

```
Шифротекст 1 в 16ричной системе: ['6e', 'e5', '5f', '55', '34', 'bd', '10', 'ba', '9f', '00', '08', '51', 'ab', '78', '92', '4
e', '4a', '66', 'e7', '86', 'f3', '8f']
Шифротекст 2 в 16ричной системе: ['7b', '2b', 'b2', '76', '15', '66', '12', '6b', 'ae', '0e', '07', '51', 'b6', 'a8', '92', '9
b', '8f', 'b5', 'e4', '97', 'e9', '47']
Открытое сообщение 1 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']
Открытое сообщение 2 в 16ричной системе: ['c4', 'ee', '20', 'cd', 'c3', '20', 'ee', 'f1', 'f2', 'e0', 'eb', 'ee', 'f1', 'fc',
'20', '31', '35', '20', 'e4', 'ed', 'e5', 'e9']
Открытое сообщение 2: До НГ осталось 15 дней
```

Рис. 0.4: Вызов функции decoder

Выводы

Освоила на практике применение режима однократного гаммирования на примере кодирования различных исходных тестов одним ключом. Закодировала два сообщения с помощью создания случайного ключа и нашла второе открытое сообщение исходя из открытого текста одного сообщения и двух зашифрованных.