

Защита лабораторной работы №7. Элементы криптографии. Однократное гаммирование

Смородова Дарья Владимировна

2022 Oct 21th

RUDN University, Moscow, Russian Federation

Цель выполнения лабораторной работы

Освоить на практике применение режима однократного гаммирования.

Задание лабораторной работы

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования.

Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

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.

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком \oplus) между элементами гаммы и элементами подлежащего сокрытию текста.

Напомним, как работает операция XOR над битами:

$$0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0.$$

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i$$

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также, а именно, обе части равенства необходимо сложить по модулю 2 с P_i :

$$C_i \oplus P_i = P_i \oplus K_i \oplus P_i = K_i$$

$$K_i = C_i \oplus P_i$$

Результаты выполнения лабораторной работы

Функция шифрования данных

```
Ввод [3]: import numpy as np
def crypt(s):
    array = []
    for i in s:
        array.append(i.encode('cp1251').hex())
    print("Список в 16 cc: ", *array)
    key = np.random.randint(0,255,len(s))
    key_16 = [hex(i)[2:] for i in key]
    print ("Ключ в 16 cc: ", *key_16)
    array_2 = []
    for i in range(len(array)):
        array_2.append("{:02x}".format(int(array[i], 16) ^ int(key_16[i], 16)))
    print("Зашифрованный текст в 16 cc:", *array_2)

    text = bytearray.fromhex(''.join(array_2)).decode('cp1251')
    print("Зашифрованный текст: ", text)
    return key_16, text
```

Figure 1: Функция шифрования данных

Результат работы функции, шифрующей данные

```
Ввод [6]: s = "С Новым Годом, друзья!"  
key, text = crypt(s)  
  
Строка в 16 cc: d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21  
Ключ в 16 cc: c5 f3 a0 22 5c 86 8a 3f 45 28 5 be e3 22 51 9c fc 68 94 72 a3 42  
Зашифрованный текст в 16 cc: 14 d3 6d cc be 7d 66 1f 86 c6 e1 50 0f 0e 71 78 0c 9b 73 8e 5c 63  
Зашифрованный текст: УmM5)£†Ж0Р qх)5E\с
```

Figure 2: Результат работы функции, шифрующей данные

Функция, дешифрующая данные

```
Ввод [5]: def foundkey(text, new_text):
            print("Текст: ", text)
            print("Зашифрованный текст: ", new_text)
            text_16 = []
            for i in text:
                text_16.append(i.encode('cp1251').hex())
            print ("Текст в 16 cc: ", "text_16")
            array = []
            for i in new_text:
                array.append(i.encode('cp1251').hex())
            print("Текст в 16 cc: ", "array")
            key = [hex(int(i, 16))int(j, 16))[2:] for (i, j) in zip(text_16, array)]
            print ("Ключ: ", "key")
            return key
```

Figure 3: Функция, дешифрующая данные

Результат работы функции, дешифрующей данные

```
Ввод [7]: found_key = foundkey(s, text)
```

Текст: С Новым Годом, друзья!

Зашифрованный текст: YmMs}f†Ж6Р qх›sГ\с

Текст в 16 cc: d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0
f3 e7 fc ff 21

Текст в 16 cc: 14 d3 6d cc be 7d 66 1f 86 c6 e1 50 0f 0e 71 78 0c
9b 73 8e 5c 63

Ключ: c5 f3 a0 22 5c 86 8a 3f 45 28 5 be e3 22 51 9c fc 68 94 7
2 a3 42

Figure 4: Результат работы функции, дешифрующей данные

```
Ввод [8]: if key == found_key:
            print("Ключ верный")
            else:
            print("Ключ неверный")

Ключ верный
```

Figure 5: Сравнение ключей

Выводы

Освоили на практике применение режима однократного гаммирования.