Лабораторная работа №7.

Элементы криптографии. Однократное гаммирование

Силкина Мария Александровна

Содержание

List of Figures

List of Tables

1 Цель работы

Освоить на практике применение режима однократного гаммирования.

2 Задачи

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.
3. Ответить на контрольные вопросы

3 Выполнение лабораторной работы

##Теоретическая справка

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

3.1 Выполнение задач

Первым шагом написала функцию шифрования, которая определяет вид шифротекста при известном ключе и известном открытом тексте “С Новым Годом, друзья!”, который был задан по условию. В выводе я получила наш изначальный текст, его вид в шестнадцатеричной системе, рандомный ключ и зашифрованный текст. (рис - @fig:001)

```
In [2]: # импорт библиотек
        import numpy as np

In [3]: text = 'С Новым Годом, друзья!'

In [25]: def shifr(text):
             print ('Сообщение - ', text)

             text_array = []
             for i in text:
                 text_array.append(i.encode('cp1251').hex())

             print('\nНаше сообщение в 16ричной системе - ', *text_array)

             key_int = np.random.randint(0, 255, len(text))
             key_hex = [hex(i)[2:] for i in key_int]

             print('\nШифр - ', *key_hex)

             text_crypt = []
             for i in range(len(text_array)):
                 text_crypt.append('{:02x}'.format(int(text_array[i], 16)^ int(key_hex[i], 16)))

             print('\nНаше зашифрованное сообщение в 16ричной системе- ', *text_crypt)

             final = bytearray.fromhex(''.join(text_crypt)).decode('cp1251')

             print ('\nЗашифрованное сообщение - ', final)

             return key_hex, final

In [29]: key, final = shifr(text)
         Сообщение -  С Новым Годом, друзья!

         Наше 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

         Шифр -  ba c9 11 19 26 d1 a0 8c 2a 66 57 5c b d6 46 2d f5 1c 2d 70 96 e4

         Наше зашифрованное сообщение в 16ричной системе-  6b e9 dc f7 c4 2a 4c ac e9 88 b3 b2 e7 fa 66 c9 05 ef ca 8c 69 c5

         Зашифрованное сообщение -  kйЬчД*L~й€iIзъfЙ‰nK‰iE
```

Figure 3.1: Функция, шифрующая данные и ее выполнение

Далее я создала функцию для дешифрования, которая определяет ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. (рис - @fig:002)

```
In [38]: def deshifr(text, final):
             print ('Сообщение - ', text)
             print ('\nЗашифрованное сообщение - ', final)

             text_hex = []
             for i in text:
                 text_hex.append(i.encode('cp1251').hex())

             print('\nНаше сообщение в 16ричной системе - ', *text_hex)

             final_hex = []
             for i in final:
                 final_hex.append(i.encode('cp1251').hex())

             print('\nЗашифрованное сообщение в 16ричной системе - ', *final_hex)

             key = [hex(int(i, 16) ^ int(j, 16))[2:] for (i, j) in zip(text_hex, final_hex)]
             print("\nКлюч: ", *key)
             return key

In [39]: keydes = deshifr(text, final)
         Сообщение -  С Новым Годом, друзья!

         Зашифрованное сообщение -  kйЬчД*L~й€iIзъfЙ‰nK‰iE

         Наше 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

         Зашифрованное сообщение в 16ричной системе -  6b e9 dc f7 c4 2a 4c ac e9 88 b3 b2 e7 fa 66 c9 05 ef ca 8c 69 c5

         Ключ:  ba c9 11 19 26 d1 a0 8c 2a 66 57 5c b d6 46 2d f5 1c 2d 70 96 e4

In [36]: print("Ключ верен!") if key == keydes else print("Ключ неверен!")
         Ключ верен!
```

Figure 3.2: Функция, дешифрующая данные и ее выполнение

##Контрольные вопросы

1. Однократное гаммирование - выполнение операции XOR между элементами гаммы и элементами подлежащего сокрытию текста. Если в методе шифрования используется 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.

Недостаток однократного гаммирования:

- Абсолютная стойкость шифра доказана только для случая, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения.

3. Преимущества однократного гаммирования:

- Такой способ симметричен, то есть двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.
- Шифрование и расшифрование может быть выполнено одной и той же программой.
- Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P.

4. Длина открытого текста должна совпадать с длиной ключа, т.к. если ключ короче текста, то операция XOR будет применена не ко всем элементам и конец сообщения будет не закодирован, а если ключ будет длиннее, то появится неоднозначность декодирования.

5. Операция XOR используется в режиме однократного гаммирования. Наложение гаммы по сути представляет собой выполнение побитовой операции сложения по модулю 2, т.е. мы должны сложить каждый элемент гаммы с соответствующим элементом ключа. Данная операция является симметричной, так как прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.

6. Получение шифротекста по открытому тексту и ключу: $C_i = P_i \oplus K_i$

7. Получение ключа по окрытому тексту и шифротексту: $K_i = P_i \oplus C_i$

8. Необходимы и достаточные условия абсолютной стойкости шифра: полная случайность ключа; равенство длин ключа и открытого текста; однократное использование ключа.

4 Выводы

Освоила использования однократного гаммирования для шифрования и дешифрования данных. # Библиография

1. Кулябов Д. С., Королькова А. В., Геворкян М. Н. Информационная безопасность компьютерных сетей. Лабораторная работа № 5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов.