Отчет по лабораторной работе № 7. Элементы криптографии. Однократное гаммирование

дисциплина: Информационная безопасность

Смирнова Мария Александровна

Цель работы

Освоить на практике применение режима однократного гаммирования.

Теоретические сведения

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 $\oplus$) между элементами гаммы и элементами подлежащего сокрытию текста. Напомним, как работает операция XOR над битами: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой. Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов. К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. 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 . Необходимые и достаточные условия абсолютной стойкости шифра: – полная случайность ключа; – равенство длин ключа и открытого текста; – однократное использование ключа.

Задание

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.

Выполнение лабораторной работы

1. Подключим необходимые библиотеки (рис. -@fig:001).

Ввод [1]:
```
import numpy as np
import sys
import operator as op
```

Библиотеки

2. Напишем функцию coder, которая будет генерировать случайный ключ и с помощью операции XOR будет шифровать открытый текст. Проверим ее работу (рис. -@fig:002).

Ввод [2]:
```
def coder (otext):
    a1 = []
    for i in otext:
        a1.append(i.encode("cp1251").hex())
    print("Открытый текст: ", a1)
    b = np.random.randint(0, 255, len(otext))
    key = [hex(i)[2:] for i in b]
    print("Ключ: ", key)
    key_b = bytearray.fromhex("".join(key)).decode("cp1251")
    print("Текст ключа: ", key_b)
    xor = []
    for i in range(len(a1)):
        xor.append("{:02x}".format(int(key[i],16) ^ int(a1[i],16)))
    print("Шифротекст: ", xor)
    shtext = bytearray.fromhex("".join(xor)).decode("cp1251")
    print("Текст шифра: ", shtext)
    return shtext
```

Ввод [4]:
```
mytext = "С Новым Годом, друзья!"
```

Ввод [5]:
```
shifr = coder(mytext)
```

```
Открытый текст:  ['d1', '20', 'cd', 'ee', 'e2', 'fb', 'ec', '20', 'c3', 'ee', 'e4', 'ee', 'e
c', '2c', '20', 'e4', 'f0', 'f3', 'e7', 'fc', 'ff', '21']
Ключ:  ['c6', 'b1', 'c3', '74', 'aa', 'f', 'a4', '25', 'bc', '4e', '24', '87', '87', 'd4', '
b0', '59', '71', 'a3', '16', '6e', 'e', '56']
Текст ключа:  Ж±Гt€ъВ[ДвНх}К–1foV
Шифротекст:  ['17', '91', '0e', '9a', '48', 'f4', '48', '05', '7f', 'a0', 'c0', '69', '6b',
'f8', '90', 'bd', '81', '50', 'f1', '92', 'f1', '77']
Текст шифра:  ‘љНфН AikшђSЃPc’cw
```

Шифрование

3. Напишем функцию decoder, которая будет с помощью операции XOR при известном открытом тексте и шифротексте определит подходящий ключ. Проверим ее работу (рис. -@fig:003).

Ввод [8]:
```
def decoder (otext, shtext):
    a1 = []
    for i in otext:
        a1.append(i.encode("cp1251").hex())
    a2 = []
    for i in shtext:
        a2.append(i.encode("cp1251").hex())
    xor = [hex(int(k,16)^int(j,16))[2:] for (k,j) in zip(a1, a2)]
    print("Ключ: ", xor)
    xor1 = bytearray.fromhex("".join(xor)).decode("cp1251")
    print("Текст ключа: ", xor1)
```

Ввод [9]:
```
decoder(mytext, shifr)
```

```
Ключ:  ['c6', 'b1', 'c3', '74', 'aa', 'f', 'a4', '25', 'bc', '4e', '24', '87', '87', 'd4', '
b0', '59', '71', 'a3', '16', '6e', 'e', '56']
Текст ключа:  Ж±ГtЄъВ[ДвНх}К–1foV
```

Определение ключа

Контрольные вопросы

1. Поясните смысл однократного гаммирования. Это симметричный метод шифрования, смысл которого заключается в наложении последовательности элементов других данных на данные для их шифрования - наложение гаммы. По сути, это сложение элементов гаммы с открытым текстом по модулю (в нашем случае по модулю 2). Поскольку шифрование симметричное, то мы можем не только накладывать информацию для шифрования, но и снимать ее для расшифровки. Данными, необходимыми для накладывания(снятия) является открытый ключ.
2. Перечислите недостатки однократного гаммирования. Короткую информацию, зашифрованную с помощью гаммирования, будет достаточно просто узнать. Ключ необходимо конфиденциально передавать обеим сторонам, что тоже достаточно сложно, особенно в зависимости от его длины. Также ключ необходимо менять каждый раз. И злоумышленник все равно может подобрать часть открытого текста, что может в определенных ситуациях плохо сказаться.
3. Перечислите преимущества однократного гаммирования. Это простой и интуитивно понятный способ шифрования. Также при случайном ключе, равным по длине открытому тексту и однократно используемом, шифр является абсолютно стойким. Алгоритм не дает никакой информации об открытом тексте.
4. Почему длина открытого текста должна совпадать с длиной ключа? Поскольку идет посимвольное шифрование. Это является условием аболютной стойкости алгоритма.
5. Какая операция используется в режиме однократного гаммирования, назовите её особенности? Операция XOR (исключающее или) или сложение по модулю 2. Главная особенность для симметричного шифрования - двойное прибавление одной и той же величины по модулю два восстанавливает исходное значение.
6. Как по открытому тексту и ключу получить шифротекст? Сложением по модулю 2 элементы открытого текста и элементы ключа.

7. Как по открытому тексту и шифротексту получить ключ? Повторить предыдущую операцию для открытого текста и шифротекста.
8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

- полная случайность ключа
- равенство длин ключа и открытого текста
- однократное использование ключа

Выводы

В процессе выполнения лабораторной работы мы освоили на практике применение режима однократного гаммирования на языке python.