Лабораторная работа №7

Элементы криптографии. Однократное гаммирование

Голощапова Ирина Борисовна

21 октября 2023

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Голощапова Ирина Борисовна
- студентка уч. группы НФИбд-01-20
- Российский университет дружбы народов
- 1032201666@pfur.ru
- https://github.com/ibgoloshchapowa

Вводная часть

Объект и предмет исследования

- Дистрибутив Rocky
- Дискреционное разграничение доступа
- Элементы криптографии
- Однократное гаммирование

Цели и задачи

Освоить на практике применение режима однократного гаммирования

Теоретическое введение

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» является простой, но надёжной схемой шифрования данных. 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.

Выполнение работы

Условие задачи

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.

Программный код

1. Определить вид шифротекста при известном ключе и известном открытом тексте

```
import numpy as np
import pandas as pd
import sys
```

```
x = "С Новым Годом, друзья!"
def cryptography(x):
    print ("text:" , x)
    text = []
    for i in x:
        text.append(i.encode("cp1251").hex())
    print ("text in 16:" , *text)
    k= np.random.randint(0, 255, len(x))
    key=[hex(i)[2:] for i in k]
    newkey=[]
    for i in key:
        newkey.append(i.encode("cp1251").hex().upper())
    print("key in 16: ", *key)
    b=[]
    for i in range(len(text)):
        b.append("{:02x}".format(int(key[i], 16)^int(text[i],16)))
    print("cypher text in 16: " , *b)
    fintext=bytearray.fromhex("".join(b)).decode("cp1251")
    print("cypher text: ", fintext)
    return key, b, fintext
```

Рис. 1: функция №1

Программный код

Вывод функции №1:

```
key, b, fintext = cryptography(x)
```

```
text: С Новым Годом, друзья!
text in 16: d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
key in 16:  b8 6c 81 32 31 85 79 88 e7 6c dc 9a 4 80 55 6e 7c 2c 74 db 41 3f
cypher text in 16:  69 4c 4c dc d3 7e 95 a8 24 82 38 74 e8 ac 75 8a 8c df 93 27 be 1e
cypher text:  iLLbУ~•Ё$,8tи~uЊЊЯ"'s
```

Рис. 2: Работа функции №1

Программный код

2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста

```
def findkey(x, fintext):
    print("text: ", x, "\ncypher text: ", fintext)
    newtext=[]
    for i in x:
        newtext.append(i.encode("cp1251").hex())
    print("text in 16: ", *newtext)
    findtext=[]
    for i in fintext:
        findtext.append(i.encode("cp1251").hex())
    print("cypher text in 16: ", *findtext)
    key=[hex(int(i,16)^int(j,16))[2:] for (i,j) in zip(newtext, findtext)]
    print("found key in 16: ", *key)
    return key
```

Рис. 3: функция №2

Программный код

Вывод функции №2:

```
key1 = findkey(x, fintext)
```

```
text:  С Новым Годом, друзья!
cypher text:  iLlbУ~•Ё$,8ti~uЊЊЯ‘'s
text in 16:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
cypher text in 16:  69 4c 4c dc d3 7e 95 a8 24 82 38 74 e8 ac 75 8a 8c df 93 27 be 1e
found key in 16:  b8 6c 81 32 31 85 79 88 e7 6c dc 9a 4 80 55 6e 7c 2c 74 db 41 3f
```

Рис. 4: Работа функции №2

3. Сравнение ключей

```
if key == key1:
    print("yes, correct")
else:
    print("no, incorrect")
```

```
yes, correct
```

Рис. 5: Сравнение ключей

Вывод

В ходе лабораторной работы мне удалось освоить на практике применение режима однократного гаммирования