Лабораторная работа №7

Элементы криптографии. Однократное гаммирование

Ильин Андрей Владимирович

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования1

# 2 Задачи

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.

# 3 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. [1]

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).

# 4 Выполнение лабораторной работы

1. Для выполнения лабораторной работы восопользуемся открытым ресурсов Google Colab. Создадим новый ноутбук - в нем будем выполнять лабораторную работу. Реализуем класс Gumming, в нем будут следующие методы: xor и \_\_xor - выполняет операцию XOR к двум строкам шестнадцатиричного кода (разделленного пробелами), to\_hex - метод конвертации обычной строки к строке шестнадцатиричного кода, from\_hex - метод для конвертации строки шестнадцатиричного кода к обычной строке. (рис. [1](#fig:001))

class Gumming:  
 def xor(self, hex\_seq1, hex\_seq2):  
 hex1 = hex\_seq1.split()  
 hex2 = hex\_seq2.split()  
 return ' '.join([self.\_\_xor(hex1, hex2) for hex1, hex2 in zip(hex1, hex2)])  
  
 def to\_hex(self, msg):  
 msg\_hex = []  
 for char in msg:  
 char\_cp1251 = char.encode('cp1251')  
 char\_code = int.from\_bytes(char\_cp1251, 'little')  
 char\_hex = hex(char\_code)[-2:].upper()  
 msg\_hex.append(char\_hex)  
 return ' '.join(msg\_hex)  
  
 def from\_hex(self, msg\_hex):  
 msg = ''  
 for char\_hex in msg\_hex.split():  
 char\_code = int(char\_hex, 16)  
 char\_cp1251 = char\_code.to\_bytes(1, 'little')  
 char = char\_cp1251.decode('cp1251')  
 msg += char  
 return msg  
  
 def \_\_xor(self, sym1, sym2):  
 xor = lambda x, y: bytes(a^b for a, b in zip(x, y))  
 b\_sym1 = bytes.fromhex(sym1)  
 b\_sym2 = bytes.fromhex(sym2)  
 r\_result = xor(b\_sym1, b\_sym2)  
 result = r\_result.hex().upper()  
 return result  
  
gumming = Gumming()

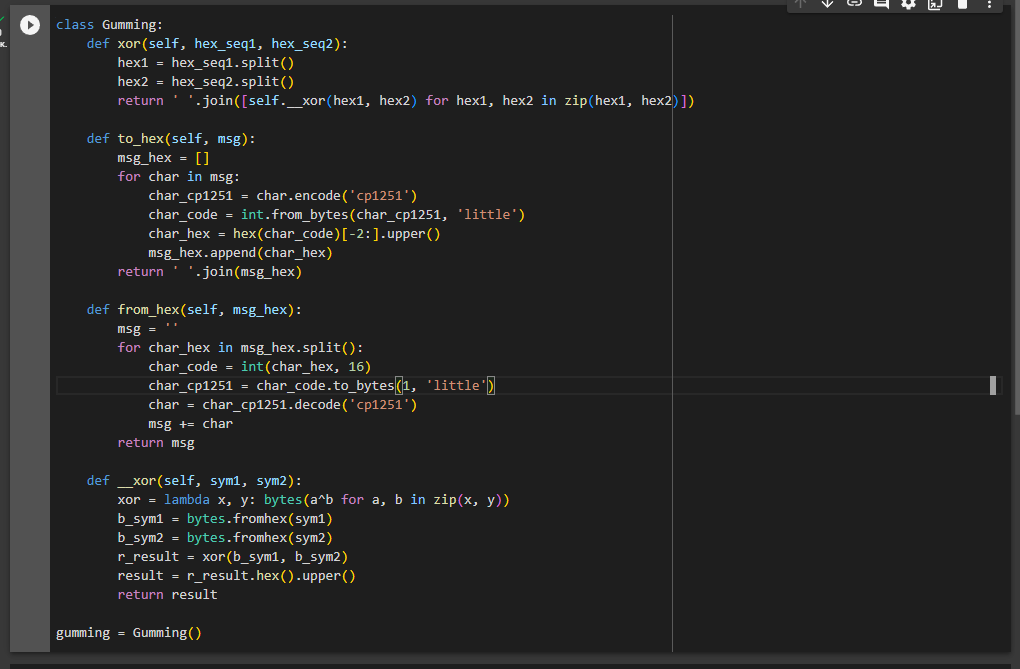


Figure 1: Класс Gumming

1. Проведем эксперимент “Центр и Мюллер”, приведенный в укзаниях к лабораторной работе, используя написанный класс. (рис. [2](#fig:002))

src\_msg = 'Штирлиц – Вы Герой!!'  
hex\_msg = gumming.to\_hex(src\_msg)  
src\_key = '05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54'  
enc\_msg = gumming.xor(hex\_msg, src\_key)  
  
mul\_key = '05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 55 F4 D3 07 BB BC 54'  
mul\_res = gumming.xor(enc\_msg, mul\_key)  
mul\_msg = gumming.from\_hex(mul\_res)  
  
print(src\_msg, '<-- Сообщение Центра')  
print(hex\_msg, '<-- Сообщение Центра (16)')  
print(src\_key, '<-- Ключ Центра')  
print(enc\_msg, '<-- Закодированное cообщение Центра')  
print(mul\_key, '<-- Ключ Мюллера')  
print(mul\_res, '<-- Сообщение Мюллера (16)')  
print(mul\_msg, '<-- Сообщение Мюллера')

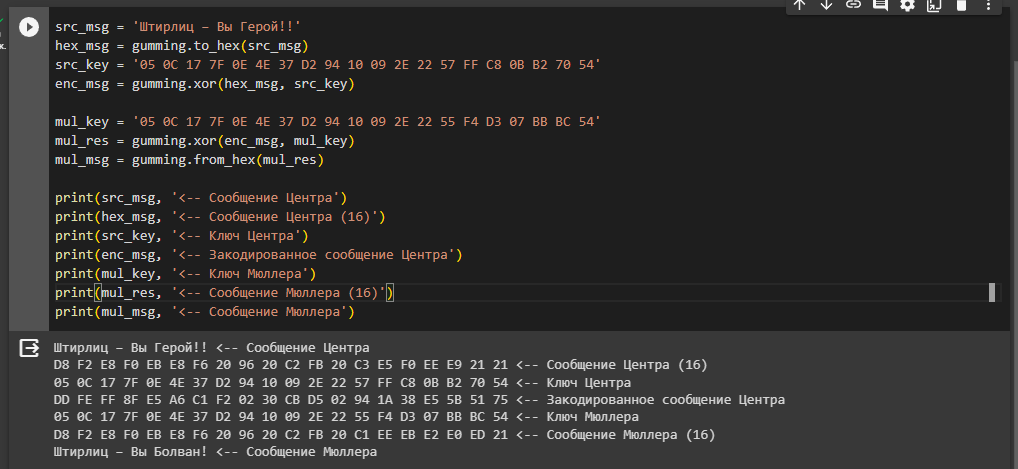


Figure 2: Центр и Мюллер

1. Определим ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста (С Новым Годом,друзья!), (рис. [3](#fig:003))

ng\_msg = 'С Новым Годом,друзья!'  
ng\_hex = gumming.to\_hex(ng\_msg)  
ng\_key = gumming.xor(enc\_msg, ng\_hex)  
ng\_res = gumming.from\_hex(gumming.xor(enc\_msg, ng\_key))  
  
print(ng\_msg, '<-- Необходимое сообщение')  
print(ng\_hex, '<-- Необходимое сообщение (16)')  
print(enc\_msg, '<-- Закодированное cообщение Центра')  
print(ng\_key, '<-- Искомый ключ')  
print(ng\_res, '<-- Необходимое сообщение из cообщения Центра')

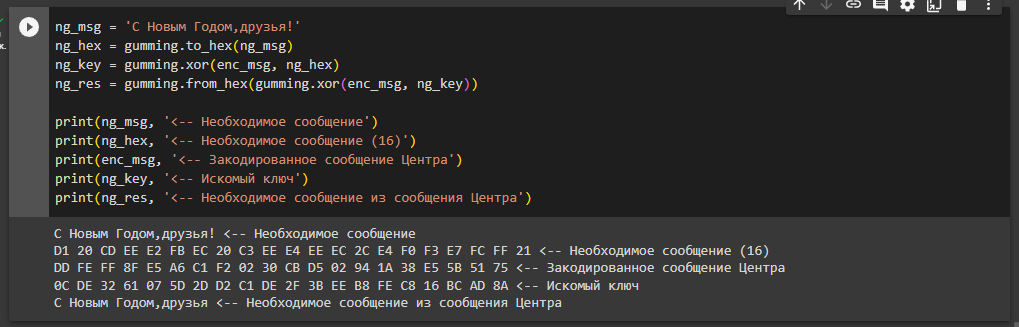


Figure 3: Ключ для С Новым Годом,друзья!

# 5 Анализ результатов

Работа выполненна без непредвиденных проблем в соответствии с руководством. Ошибок и сбоев не произошло.

# 6 Выводы

Нам удалось освоить на практике применение режима однократного гаммирования, в дополнение закрпеили навки владения языками программирования, в частности языком программирования - python.

# Список литературы

1. Материалы по лабораторной работе [Электронный ресурс]. RUDN. URL: <https://esystem.rudn.ru/pluginfile.php/2090284/mod_resource/content/2/007-lab_crypto-gamma.pdf>.