Лабораторная работа №8

Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

Ильин Андрей Владимирович

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 2 Задачи

Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты и в режиме однократного гаммирования.Приложение должно:

1. Определить вид шифротекстов и обоих текстов и при известном ключе.
2. Требуется не зная ключа и не стремясь его определить, прочитать текст , при условии, что текста подчиняются шаблону.

# 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. Возьмем класс Gumming, написанный в предыдущей лабораторной работе. (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()

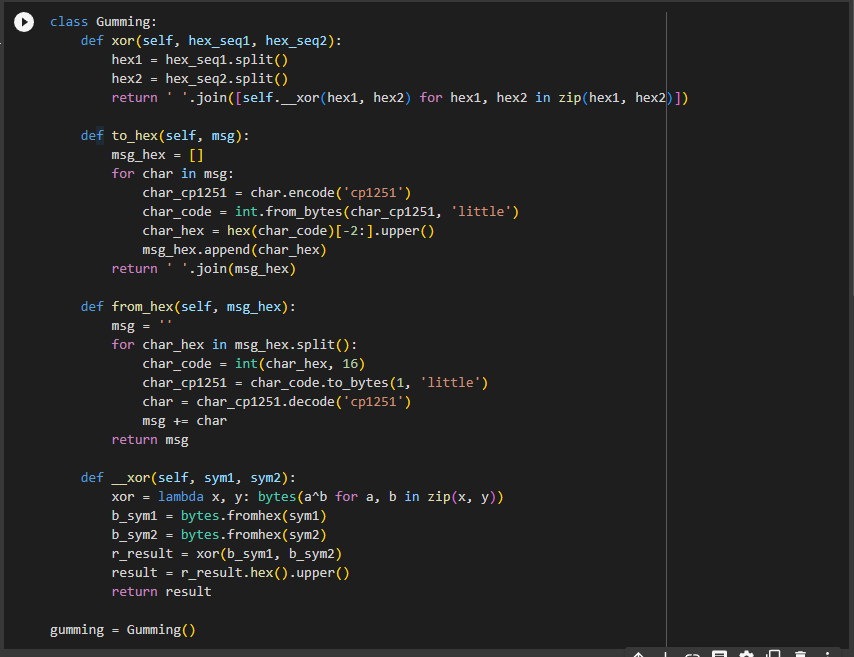


Figure 1: Класс Gumming

1. Напишем скрипт, которой будет шифровать и дешифровать тексты и в режиме однократного гаммирования. (рис. [2](#fig:002))

P1\_raw = 'НаВашисходящийот1204'  
P2\_raw = 'ВСеверныйфилиалБанка'  
P1 = gumming.to\_hex(P1\_raw)  
P2 = gumming.to\_hex(P2\_raw)  
K = '05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54'  
C1 = gumming.xor(P1, K)  
C2 = gumming.xor(P2, K)  
P1\_res = gumming.xor(C1, K)  
P2\_res = gumming.xor(C2, K)  
P1\_fin = gumming.from\_hex(P1\_res)  
P2\_fin = gumming.from\_hex(P2\_res)

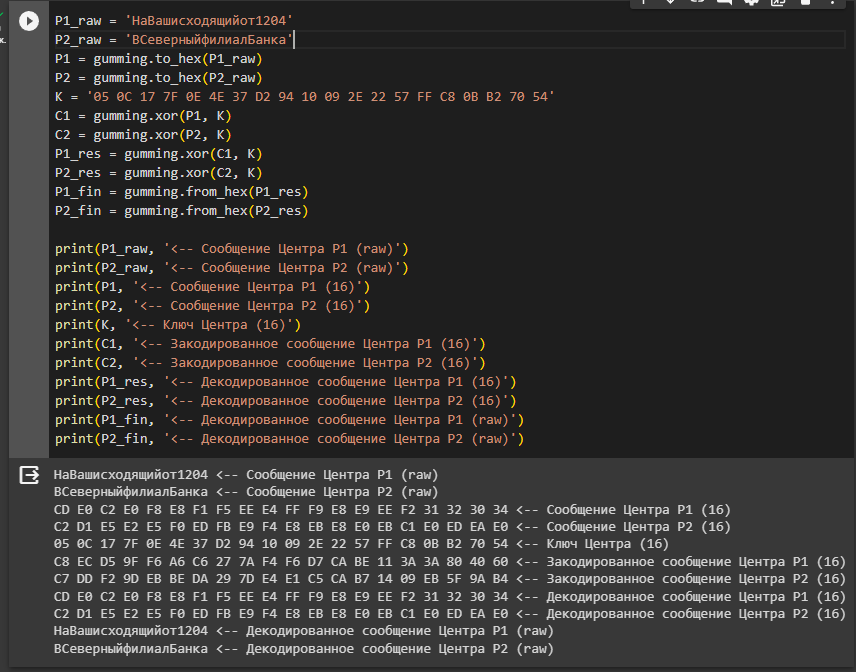


Figure 2: Режим однократного гаммирования

1. Предположим, что каждое сообщение имеет шаблон. Сгенерируем по данному шаблону пару случайных сообщений. Зашифруем их и взломаем второе сообщение, используя шифротексты и известное первое сообщение. (рис. [3](#fig:003))

import random  
  
ALPHABET = list('АаБбВвГгДдЕеЁёЖжЗзИиЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщЪъЫыЬьЭэЮюЯя0123456789')  
IGNORE\_SYM = '\*'  
P\_TEMPLATE = 'На\*\*\*\*сходящий\*\*12\*\*'  
  
def getTextByTemplate(template):  
 msg = ''  
 for char in template:  
 if char == IGNORE\_SYM:  
 msg += random.choice(ALPHABET)  
 else:  
 msg += char  
 return msg  
  
P1\_raw = getTextByTemplate(P\_TEMPLATE)  
P2\_raw = getTextByTemplate(P\_TEMPLATE)  
  
P1 = gumming.to\_hex(P1\_raw)  
P2 = gumming.to\_hex(P2\_raw)  
C1 = gumming.xor(P1, K)  
C2 = gumming.xor(P2, K)  
  
P2\_res = gumming.xor(gumming.xor(C1, C2), P1)  
P2\_fin = gumming.from\_hex(P2\_res)

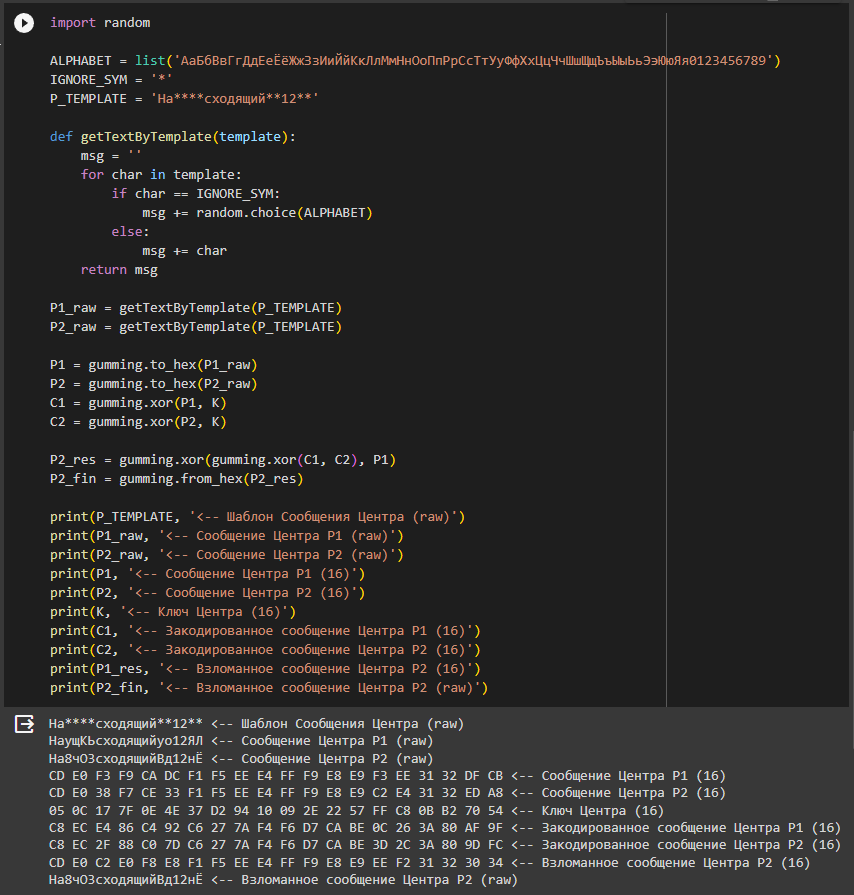


Figure 3: Взлом второго текста

# 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/2090286/mod_resource/content/2/008-lab_crypto-key.pdf>.