Лабораторная работа №7.

Элементы криптографии. Однократное гаммирование.

Ишанова А.И. группа НФИ-02-19

Содержание

# Цель работы

Освоить на практике применение режима однократного гаммирования.

# Теоретическое введение

Гамми́рование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле. Например, в поле Галуа GF(2) суммирование принимает вид операции «исключающее ИЛИ (XOR)». [2]

# Выполнение лабораторной работы

1. Была реализована программа на Python:

import numpy as np  
from random import randrange  
  
# 1/ задаем строку для шифрования  
  
t = "С Новым Годом, друзья!"  
  
# 2/ переведем строку в hex   
  
hex\_message = []  
for i in t:  
 hex\_message.append(i.encode("cp866").hex())  
print (hex\_message)  
  
# 3/ задаем ключ, такой же длины, что и строка для шифрования   
  
def gen\_key(length: int):  
 key = []  
 for i in range(0,length):  
 key.append(hex(randrange(255))[2:])  
 return key  
  
key\_1 = gen\_key(len(hex\_message))  
print(key\_1)  
  
# 4/ кодируем строку с помощью ключа  
  
def encode\_message(hex\_message, key):  
 return ["%x" % (int(x,16) ^ int(y,16)) for (x, y) in zip(hex\_message, key)]  
   
encoded\_message = encode\_message(hex\_message, key\_1)  
print(encoded\_message)  
  
# сообщение текстом  
  
def code\_to\_lang(encoded\_message):  
 return bytearray.fromhex("".join(encoded\_message)).decode("cp866")  
  
encoded\_text = code\_to\_lang(encoded\_message)  
print(encoded\_text)  
  
# 5/ задаем ключ для расшифровки  
key\_2 = gen\_key(len(hex\_message))  
print(key\_2)  
  
# 6/ декодируем с помощью нового ключа  
  
def decode\_message(key, encoded\_message):  
 return ["%x" % (int(x,16) ^ int(y,16))   
 for (x, y) in zip(key, encoded\_message)]  
  
decoded\_message = decode\_message(key\_2, encoded\_message)  
print(decoded\_message)  
  
decoded\_text = code\_to\_lang(decoded\_message)  
print(decoded\_text)  
  
# 7/ декодируем с помощью правильного ключа  
  
decoded\_message\_r = decode\_message(key\_1, encoded\_message)  
decoded\_text\_r = code\_to\_lang(decoded\_message\_r)  
print(decoded\_text\_r)

1. Запустили программу. Получили:

* сообщение в hex

[‘91’, ‘20’, ‘8d’, ‘ae’, ‘a2’, ‘eb’, ‘ac’, ‘20’, ‘83’, ‘ae’, ‘a4’, ‘ae’, ‘ac’, ‘2c’, ‘20’, ‘a4’, ‘e0’, ‘e3’, ‘a7’, ‘ec’, ‘ef’, ‘21’]

* ключ для кодировки

[‘a6’, ‘d6’, ‘e8’, ‘35’, ‘f3’, ‘1d’, ‘41’, ‘e1’, ‘88’, ‘d1’, ‘bd’, ‘2a’, ‘16’, ‘80’, ‘a2’, ‘20’, ‘ed’, ‘6a’, ‘fc’, ‘67’, ‘ce’, ‘9d’]

* закодированное сообщение

[‘37’, ‘f6’, ‘65’, ‘9b’, ‘51’, ‘f6’, ‘ed’, ‘c1’, ‘b’, ‘7f’, ‘19’, ‘84’, ‘ba’, ‘ac’, ‘82’, ‘84’, ‘d’, ‘89’, ‘5b’, ‘8b’, ‘21’, ‘bc’]

* закодированное сообщение в виде текста (fig. 1)

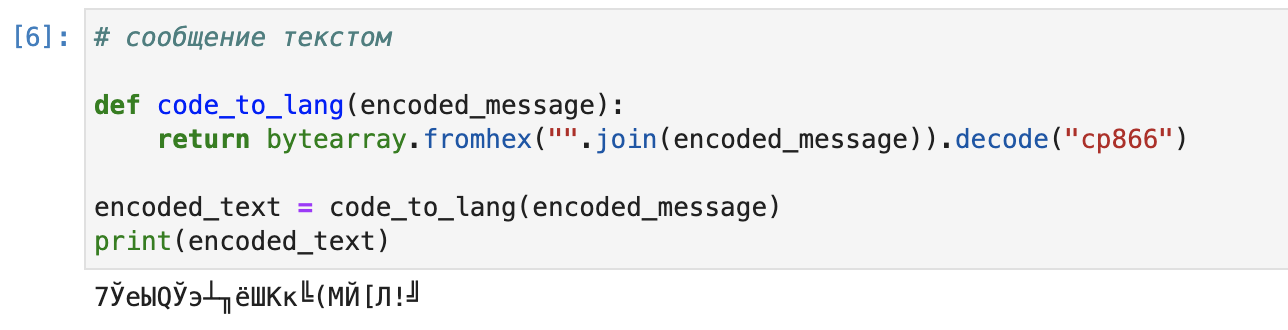


Figure 1: Вывод программы: закодированное сообщение в виде текста

* ключ для расшифровки

[‘7a’, ‘f1’, ‘5b’, ‘3e’, ‘ea’, ‘d’, ‘9e’, ‘23’, ‘d6’, ‘3e’, ‘40’, ‘d9’, ‘de’, ‘6b’, ‘d8’, ‘9b’, ‘b’, ‘4f’, ‘3a’, ‘6e’, ‘14’, ‘eb’]

* сообщение, раскодированное ключом для расшифровки

[‘4d’, ‘7’, ‘3e’, ‘a5’, ‘bb’, ‘fb’, ‘73’, ‘e2’, ‘dd’, ‘41’, ‘59’, ‘5d’, ‘64’, ‘c7’, ‘5a’, ‘1f’, ‘6’, ‘c6’, ‘61’, ‘e5’, ‘35’, ‘57’]

* раскодированное сообщение текстом (fig. 2)

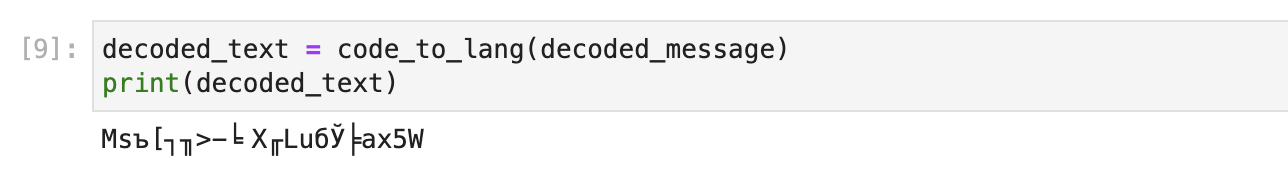


Figure 2: Вывод программы: раскодированное сообщение в виде текста (неверный ключ)

* текст сообщения, раскодированного ключом для кодировки (fig. 3)

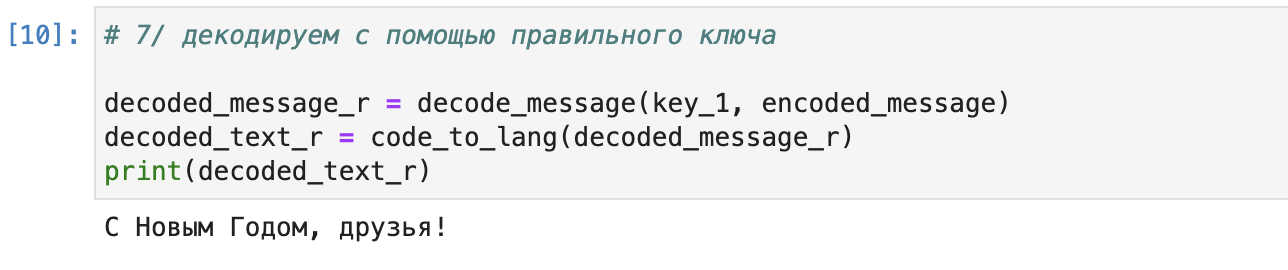


Figure 3: Вывод программы: раскодированное сообщение в виде текста (верный ключ)

# Вывод

В ходе выполнения лабораторной работы было изучено шифрование методом однократного гаммирования и реализована программа на python, шифрующая и расшифровавующая заданную строку этим методом.

# Библиография

1. Методические материалы курса.
2. Wikipedia: Гаммирование (URL: https://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)