Лабораторная работа №7

Элементы криптографии. Однократное гаммирование

Маслова Анастасия Сергеевна

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования.

# 2 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.

# 3 Порядок выполнения работы

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.

# 4 Выполнение работы

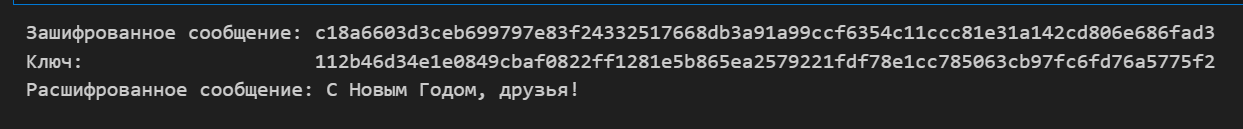
Для выполнения работы я использовала язык программирования python, библиотеки binascii, secrets. Работа кода построена следующим образом:

* Сначала сообщение преобразуется в шестнадцатиричный вид, чтобы было проще оперировать битами.
* Далее сообщение кодируется с помощью случайно сгенерированного ключа.
* С помощью функции decrypt\_message сообщение может быть раскодировано обратно.

Весь код представлен ниже.

import binascii  
import secrets  
  
def str\_to\_hex(s):  
 return binascii.hexlify(s.encode('utf-8')).decode('utf-8')  
  
def hex\_to\_str(h):  
 return binascii.unhexlify(h.encode('utf-8')).decode('utf-8')  
  
def xor\_hex\_str(hex1, hex2):  
 bytes1 = bytes.fromhex(hex1)  
 bytes2 = bytes.fromhex(hex2)  
 xor\_result = bytes(a ^ b for a, b in zip(bytes1, bytes2))  
 return xor\_result.hex()  
  
def gen\_random\_key(l):  
 random\_bytes = secrets.token\_bytes(l)  
 return random\_bytes  
  
def encrypt\_message(message):  
 message\_bytes = message.encode('utf-8')  
 key = gen\_random\_key(len(message\_bytes))  
 cipher\_bytes = bytes(a^b for a,b in zip(message\_bytes, key))  
 return cipher\_bytes, key  
  
def decrypt\_message(cipher\_bytes, key):  
 decrypted\_bytes = bytes(a^b for a,b in zip(cipher\_bytes, key))  
 return decrypted\_bytes.decode('utf-8')  
  
open\_message = "С Новым Годом, друзья!"  
encrypted\_message, key = encrypt\_message(open\_message)  
print("Зашифрованное сообщение:", encrypted\_message.hex())  
print("Ключ: ", key.hex())  
decrypted\_message = decrypt\_message(encrypted\_message, key)  
print("Расшифрованное сообщение:", decrypted\_message)

Так выглядит вывод моей программы (рис. [??]).



Вывод программы

# 5 Выводы

В ходе лабораторной работы я освоила на практике применение режима однократного гаммирования.

# Список литературы