Отчет по лабораторной работе №7

Основы информационной безопасности

Машков Илья Евгеньевич

Содержание

# 1 Цель

Освоить на практике применение режима однократного гаммирования

# 2 Задание

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

# 3 Выполнение лабораторной работы

Я выполнала лабораторную работа на языке программирования Python, листинг программы и результаты выполнения приведены в отчете.

Требуется разработать программу, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Начнем с создания функции для генерации случайного ключа (рис. 1).

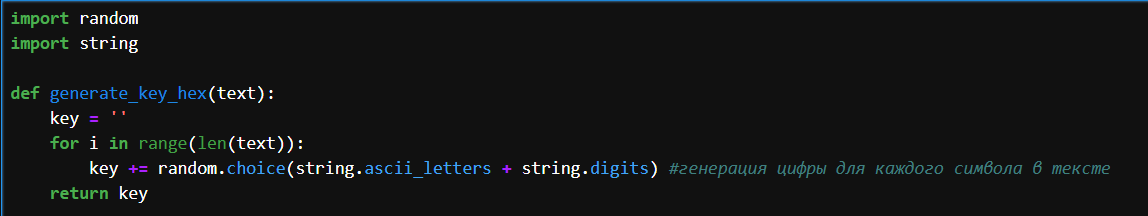


Рис. 1: Функция генерации ключа

Необходимо определить вид шифротекста при известном ключе и известном открытом тексте. Так как операция исключающего или отменяет сама себя, делаю одну функцю и для шифрования и для дешифрования текста (рис. 2).

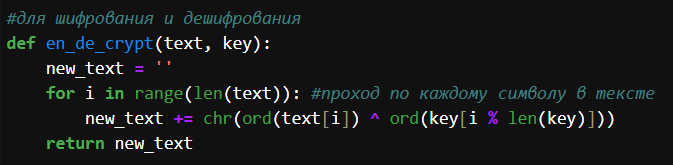


Рис. 2: Функция для шифрования текста

Нужно определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. Для этого создаю функцию для нахождения возможных ключей для фрагмента текста (рис. 3).

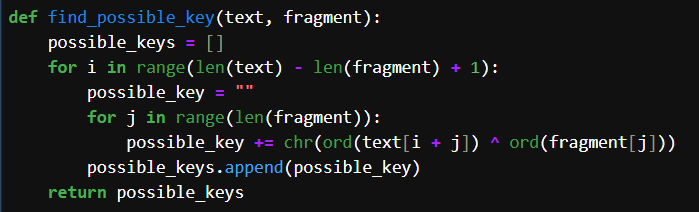


Рис. 3: Подбор возможных ключей для фрагмента

Проверка работы всех функций. Шифрование и дешифрование происходит верно, как и нахождение ключей, с помощью которых можно расшифровать верно только кусок текста (рис. 4).

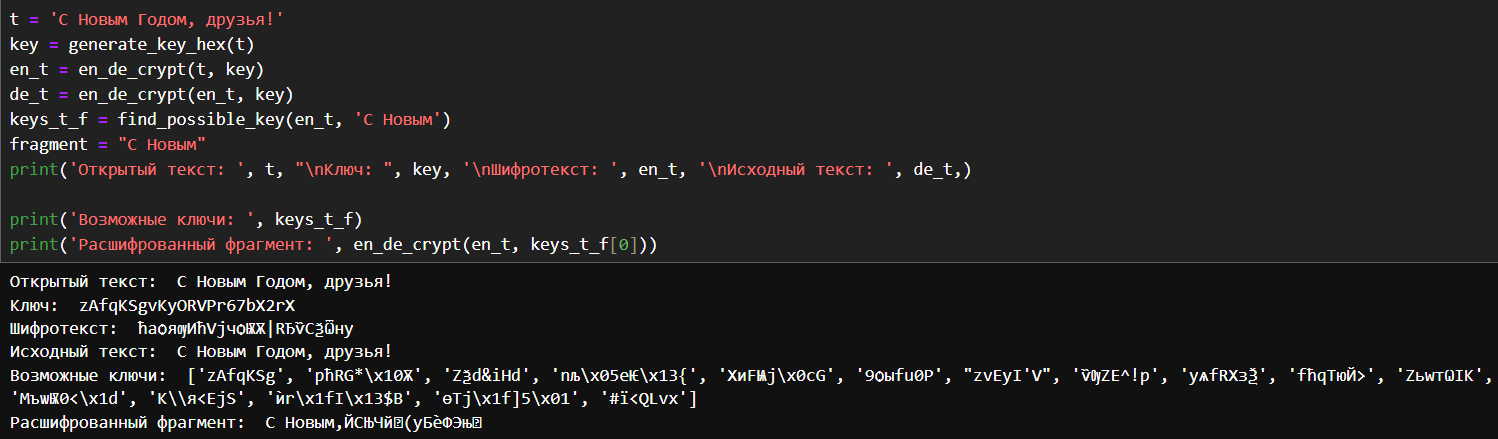


Рис. 4: Результат работы программы

Листинг программы 1:

import random  
import string  
  
def generate\_key\_hex(text):  
 key = ''  
 for i in range(len(text)):  
 key += random.choice(string.ascii\_letters + string.digits) #генерация цифры для каждого символа в тексте  
 return key  
  
#для шифрования и дешифрования  
def en\_de\_crypt(text, key):  
 new\_text = ''  
 for i in range(len(text)): #проход по каждому символу в тексте  
 new\_text += chr(ord(text[i]) ^ ord(key[i % len(key)]))  
 return new\_text  
  
def find\_possible\_key(text, fragment):  
 possible\_keys = []  
 for i in range(len(text) - len(fragment) + 1):  
 possible\_key = ""  
 for j in range(len(fragment)):  
 possible\_key += chr(ord(text[i + j]) ^ ord(fragment[j]))  
 possible\_keys.append(possible\_key)  
 return possible\_keys  
  
t = 'С Новым Годом, друзья!'  
key = generate\_key\_hex(t)  
en\_t = en\_de\_crypt(t, key)  
de\_t = en\_de\_crypt(en\_t, key)  
keys\_t\_f = find\_possible\_key(en\_t, 'С Новым')  
fragment = "С Новым"  
print('Открытый текст: ', t, "\nКлюч: ", key, '\nШифротекст: ', en\_t, '\nИсходный текст: ', de\_t,)  
  
print('Возможные ключи: ', keys\_t\_f)  
print('Расшифрованный фрагмент: ', en\_de\_crypt(en\_t, keys\_t\_f[0]))

# 4 Выводы

В ходе выполнения данной лабораторной работы мной было освоено на практике применение режима однократного гаммирования.

# Список литературы