Лабораторная №7

Дисциплина: Основы информационной безопасности

Жибицкая Евгения Дмитриевна

Содержание

# 1 Цель работы

Освоение на практике применения режима однократного гаммирования, разработка приложения, позволяющего шифровать и дешифровать данные в режиме 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 текста.

# 3 Выполнение лабораторной работы

Код будет реализован на языке python, испоьзуем collab для его написания. Для выполнения первого пункта напишем основные функции - генерация ключа с использованием random, шифровка/дешифровка на основе сложения по модулю 2(рис. 1).



Рис. 1: Создание функций

Запустим ее, убедимся, что все работает(рис. 2).

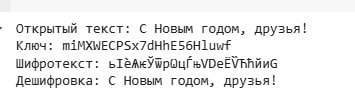


Рис. 2: Результат работы программы

Далее реализуем функцию преобразования фрагмента текста и посмотри на ее вывод(рис. 3).

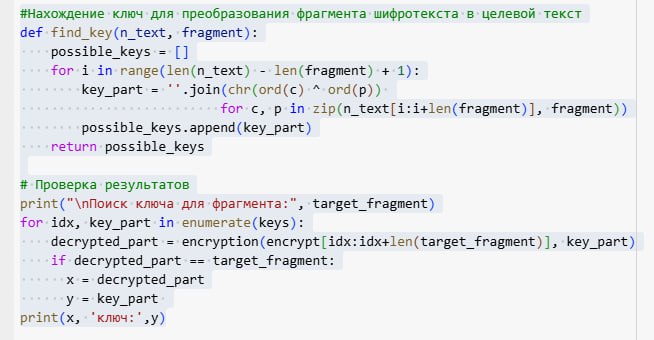


Рис. 3: Функция для 2 части задания

Результат работы программы(рис. 4).

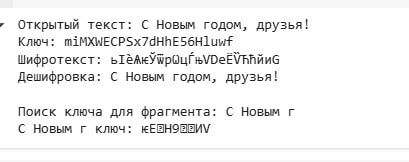


Рис. 4: Результат шифровки и дешифровки

* Листинг программы

import random  
import string  
  
# Генерация ключа  
def key\_generation(text):  
 key = ''  
 for i in range(len(text)):  
 key += random.choice(string.ascii\_letters + string.digits)  
 return key  
  
#Шифровка дешифровка текста  
def encryption(text, key):  
 n\_text = ''  
 for i in range(len(text)):  
 n\_text += chr(ord(text[i]) ^ ord(key[i]))  
 return n\_text  
  
#Нахождение ключ для преобразования фрагмента шифротекста в целевой текст  
def find\_key(n\_text, fragment):  
 possible\_keys = []  
 for i in range(len(n\_text) - len(fragment) + 1):  
 key\_part = ''.join(chr(ord(c) ^ ord(p))   
 for c, p in zip(n\_text[i:i+len(fragment)], fragment))  
 possible\_keys.append(key\_part)  
 return possible\_keys  
  
text = 'С Новым годом, друзья'  
key = key\_generation(text)  
encrypt = encryption(text, key)  
decrypt = encryption(encrypt, key)  
  
  
target\_fragment = 'С Новым г' # Фрагмент текста  
keys = find\_key(encrypt, target\_fragment)  
  
print('Открытый текст:', text)  
print('Ключ:', key)  
print('Шифротекст:', encrypt)  
print('Дешифровка:', decrypt)  
  
# Проверка результатов  
print("\nПоиск ключа для фрагмента:", target\_fragment)  
for idx, key\_part in enumerate(keys):  
 decrypted\_part = encryption(encrypt[idx:idx+len(target\_fragment)], key\_part)  
 if decrypted\_part == target\_fragment:  
 x = decrypted\_part  
 y = key\_part   
print(x, 'ключ:',y)

# 4 Выводы

В ходе работы при получены навыки однократного гаммирования, реализована программа осуществляющая шифровку и дешифровку данных.