Отчёт по лабораторной работе №7

Дисциплина: Основы информационной безопасности

Дудырев Глеб Андреевич

Содержание

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования1

# 2 Выполнение лабораторной работы

1. Для начала реализую функцию generate\_key, которая генерирует случайнную последовательность бит, то есть секретный ключ.

def generate\_key(key\_len: int) -> list:  
 """  
 Функция генерирует пвсевдослучайную последовательность,   
 которая будет использоваться в качестве ключа для шифрования.  
 """  
 key = [] #Объект, который будет содержать итоговую последовательность  
 for i in range(key\_len):  
 #Генерируем последовательность  
 key.append(np.random.randint(0, 2))  
 return key

Посмотрим на результат работы generate\_key():(рис. 1)

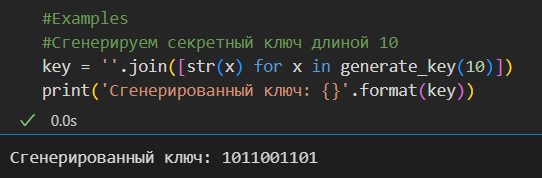


Рис. 1: Функция generate\_key()

1. Далее я реализую функцию encrypt, которая производит шифрование открытого текста, с помощью применения однократного кодирования.

def encrypt(open\_text: str, key: list = None) -> str:  
 """  
 Функция шифрует данные в режиме однократного гаммирования.  
 """  
 #Из открытого текста получаем бинарную последовательность  
 open\_text\_bin = ''.join(format(ord(x), '08b') for x in open\_text)  
 #Если ключ не передается, то сгенерируем его  
 if not key:  
 key\_len = len(open\_text\_bin)  
 key = generate\_key(key\_len)  
 #Получаем последовательность бит шифротекста, применяя последовательно XOR к биту из открытого текста и соответсвующему биту из секретного ключа  
 ciphertext\_bin = []  
 for idx, bit in enumerate(open\_text\_bin):  
 ciphertext\_bin.append(int(bit) ^ key[idx])  
 ciphertext\_bin = ''.join([str(x) for x in ciphertext\_bin])  
 #Преобразуем последовательность бит в текст  
 ciphertext = ''.join(chr(int(ciphertext\_bin[(i \* 8):(i \* 8 + 8)],2)) for i in range(len(ciphertext\_bin) // 8))  
 return ciphertext, key

Посмотрим на пример работы этой функции: (рис. 2)

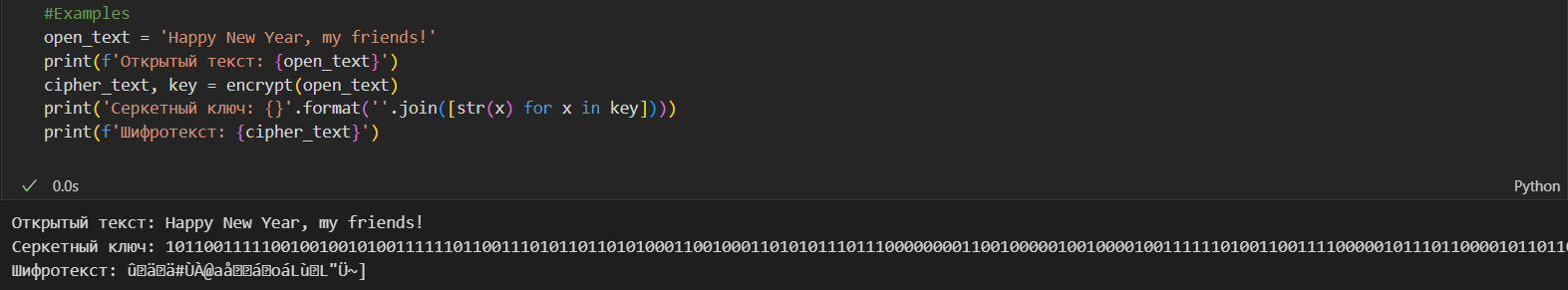


Рис. 2: Функция encrypt()

1. Следующим шагом я реализовал функцию, которая призводит дешифрования шифротекста.

def decrypt(cipher\_text: str, key) -> str:  
 """  
 Функция, которая производит дешифрование  
 """  
 if not key: #Если ключ не передали, то завершаем работу программы  
 return 'You should enter the secret key.'  
 #Из зашифрованного текста получаем бинарную последовательность  
 cipher\_text\_bin = ''.join(format(ord(x), '08b') for x in cipher\_text)  
 #Получаем последовательность бит открытого текста, применяя последовательно XOR к биту из шифротекста и соответсвующему биту из секретного ключа  
 open\_text\_bin = []  
 for idx, bit in enumerate(cipher\_text\_bin):  
 open\_text\_bin.append(int(bit) ^ key[idx])  
 open\_text\_bin = ''.join([str(x) for x in open\_text\_bin])  
 #Преобразуем последовательность бит в текст  
 open\_text = ''.join(chr(int(open\_text\_bin[(i \* 8):(i \* 8 + 8)],2)) for i in range(len(open\_text\_bin) // 8))  
 return open\_text, key

Посмотрим на пример ее работы: (рис. 3)

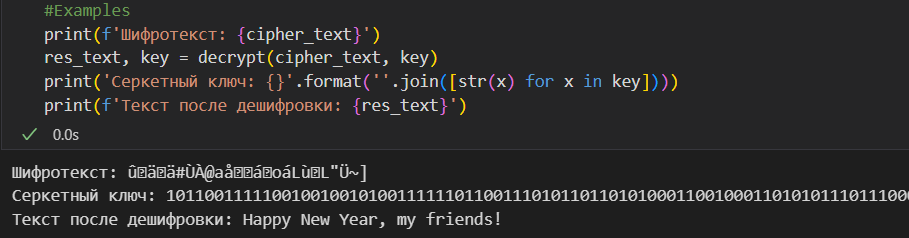


Рис. 3: Функция encrypt()

# 3 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки. Написал программу для вычисления суммы значений от заданной функции, в которой аргументы вводятся с командной строки