**Отчет к лабораторной работе №7**

# Common information

discipline: Основы информационной безопасности group: НПМбд-02-21 author: Ермолаев А.М.

**Цель работы**

Освоить на практике применение режима однократного гаммирования.

# Выполнение работы

Напишем код на языке программирования Python.

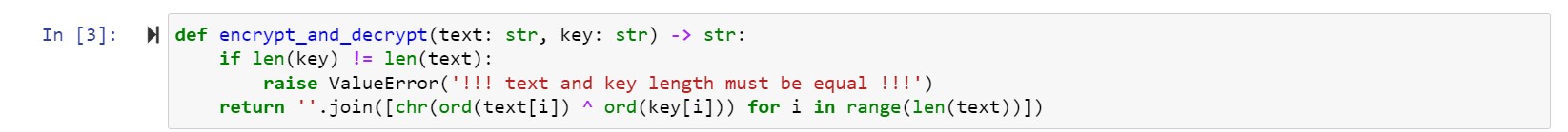
Импортируем все необходимые библиотеки:



Напишем функцию для генерации случайного ключа заданной длины:



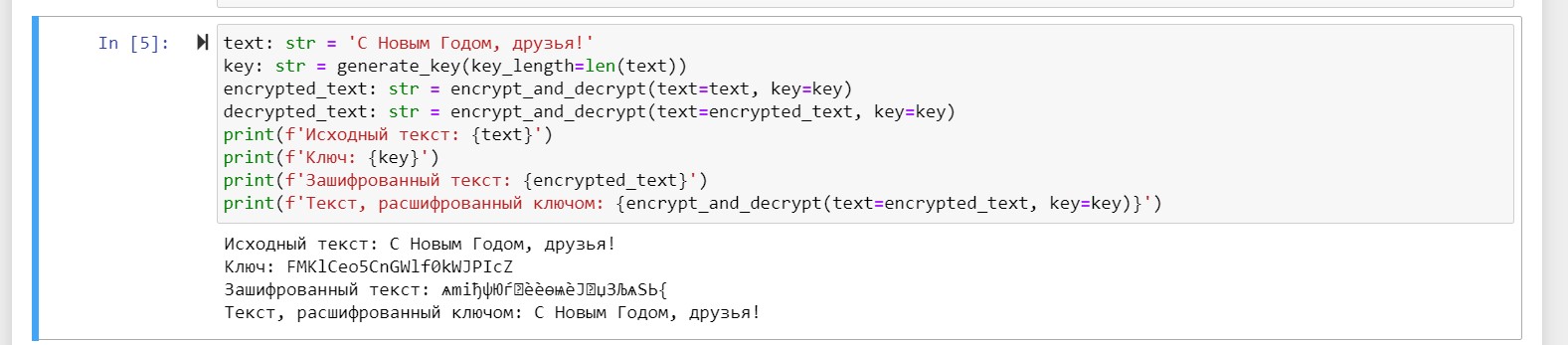
Напишем функцию для шифорования и дешифрования текста по заданному ключу (реализация одной функции для двух процессов возможна ввиду того, что операция исключающего или отменяет сама себя):

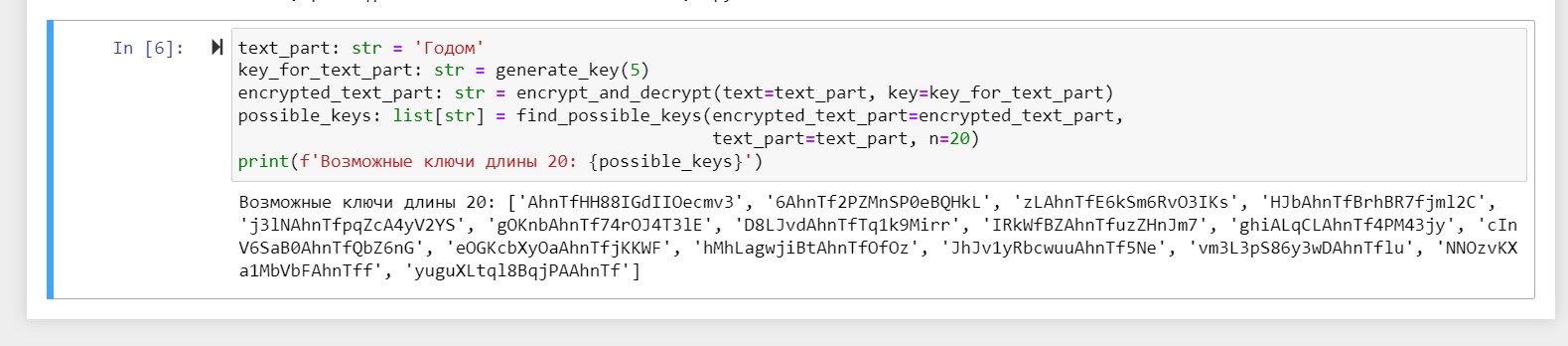


После чего напишем функцию, которая формирует ключ заданной длины, содержащий в себе в качестве фрагмента ключ, расшифровывающий заданные зашифрованный и исходный тексты, который для дешифрования исходного текста:



Проверим корректность работы написанных нами функций:





В итоге имеем данну программу:

import random from string import ascii\_letters, digits

def generate\_key(key\_length: int) -> str:

return ''.join([random.choice(ascii\_letters + digits) for \_ in range(key\_length)])

def encrypt\_and\_decrypt(text: str, key: str) -> str: if len(key) != len(text):

raise ValueError('!!! text and key length must be equal !!!')

return ''.join([chr(ord(text[i]) ^ ord(key[i])) for i in range(len(text))])

def find\_possible\_keys(encrypted\_text\_part: str, text\_part: str, n: int) -> list[str]:

possible\_keys: list[str] = [] for i in range(n - len(text\_part) + 1):

possible\_key\_part1: str = generate\_key(i) possible\_key\_part2: str = encrypt\_and\_decrypt(text=encrypted\_text\_part, key=text\_part) possible\_key\_part3: str = generate\_key(n - i - len(encrypted\_text\_part)) possible\_key: str = possible\_key\_part1 + possible\_key\_part2 + possible\_key\_part3 possible\_keys.append(possible\_key) return possible\_keys

text: str = 'С Новым Годом, друзья!' key: str = generate\_key(key\_length=len(text)) encrypted\_text: str = encrypt\_and\_decrypt(text=text, key=key) decrypted\_text: str = encrypt\_and\_decrypt(text=encrypted\_text, key=key) print(f'Исходный текст: {text}') print(f'Ключ: {key}')

print(f'Зашифрованный текст: {encrypted\_text}') print(f'Текст, расшифрованный ключом: {encrypt\_and\_decrypt(text=encrypted\_text,

key=key)}

'

)

text\_part: str =

'Годом'

key\_for\_text\_part: str = generate\_key(

5

)

encrypted\_text\_part: str = encrypt\_and\_decrypt(text=text\_part,

key=key\_for\_text\_part)

possible\_keys: list[str] =

find\_possible\_keys(encrypted\_text\_part=encrypted\_text\_part,

text\_part=text\_part, n=

20

)

print(

f'Возможные ключи длины 20:

{

possible\_keys

}

'

)

# Ответы на контрольные вопросы

1. Однократное гаммирование - это метод шифрования, при котором каждый символ открытого текста гаммируется с соответствующим символом ключа только один раз.
2. Недостатки однократного гаммирования:

Уязвимость к частотному анализу из-за сохранения частоты символов открытого текста в шифротексте.

Необходимость использования одноразового ключа, который должен быть длиннее самого открытого текста.

Нет возможности использовать один ключ для шифрования разных сообщений.

1. Преимущества однократного гаммирования:

Высокая стойкость при правильном использовании случайного ключа.

Простота реализации алгоритма.

Возможность использования случайного ключа.

1. Длина открытого текста должна совпадать с длиной ключа, чтобы каждый символ открытого текста гаммировался с соответствующим символом ключа.
2. В режиме однократного гаммирования используется операция XOR (исключающее ИЛИ), которая объединяет двоичные значения символов открытого текста и ключа для получения шифротекста.

Особенность XOR - если один из битов равен 1, то результат будет 1, иначе 0.

1. Для получения шифротекста по открытому тексту и ключу каждый символ открытого текста гаммируется с соответствующим символом ключа с помощью операции XOR.
2. По открытому тексту и шифротексту невозможно восстановить действительный ключ, так как для этого нужна информация о каждом символе ключа.
3. Необходимые и достаточные условия абсолютной стойкости шифра:

Ключи должны быть случайными и использоваться только один раз.

Длина ключа должна быть не менее длины самого открытого текста.

Ключи должны быть храниться и передаваться безопасным способом.

**Вывод**

В рамках выполнения работы я освоил на практике применение режима однократного гаммирования.

# Список литературы

https://bugtraq.ru/library/books/crypto/chapter7/ https://xakep.ru/2019/07/18/crypto-xor/ https://www.youtube.com/watch?v=tAjBULW\_OjQ