

# Лабораторная работа №7

## Элементы криптографии. Однократное гаммирование

---

Щербак Маргарита Романовна

НПИбд-02-21

Студ. билет: 1032216537

2024

RUDN

Освоить на практике применение режима однократного гаммирования.

**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.

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.

Ввела известный открытый текст: `text = "С Новым Годом, друзья!"`, после чего создала ключ:

```
key = ''  
seed(22)  
for i in range(len(text)):  
    key += random.choice(string.ascii_letters  
        + string.digits)
```

Прописала получение шифротекста с использованием функции `xor_text_f`:  
`xor_text = xor_text_f(text, key)`. В этом моменте я передаю известный текст и сгенерированный ключ в функцию `xor_text_f`, которая выполняет операцию XOR. Результат этой операции (шифротекст) сохраняется в переменной `xor_text`. Вывод шифротекста: `print(f'Шифротекст: {xor_text}')`. Так я создала шифротекст на основе известного открытого текста и ключа.

Далее перешла к определению ключа. Дешифрование шифротекста (здесь я беру шифротекст и применяю к нему ту же функцию `xor_text_f`, используя тот же ключ. Это позволяет получить обратно оригинальный открытый текст):

```
decrypted_text = xor_text_f(xor_text, key)
print(f'Расшифрованный текст: {decrypted_text}')
```

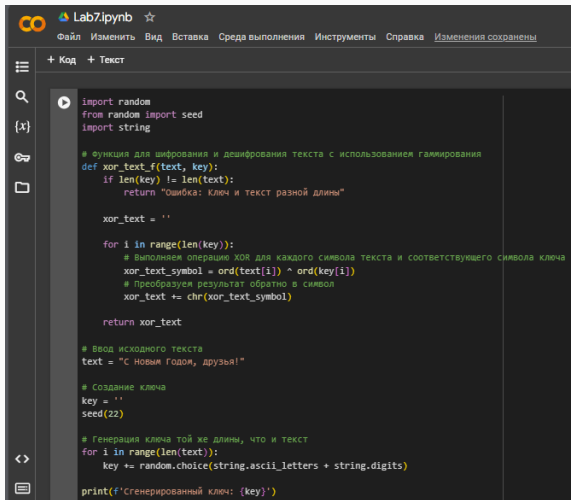
Получение ключа (В этом моменте я выполняю XOR между открытым текстом и шифротекстом, что в результате даст ключ):

```
recovered_key = xor_text_f(text, xor_text)
print(f'Восстановленный ключ: {recovered_key}')
```



# Выполнение лабораторной работы

Код целиком показан на рис.1:



```
Lab7.ipynb
Файл  Изменить  Вид  Вставка  Среда выполнения  Инструменты  Справка  Изменения сохранены

+ Код  + Текст

import random
from random import seed
import string

# функция для шифрования и дешифрования текста с использованием гаммирования
def xor_text_f(text, key):
    if len(key) != len(text):
        return "Ошибка: ключ и текст разной длины"

    xor_text = ''

    for i in range(len(key)):
        # Выполняем операцию XOR для каждого символа текста и соответствующего символа ключа
        xor_text_symbol = ord(text[i]) ^ ord(key[i])
        # Преобразуем результат обратно в символ
        xor_text += chr(xor_text_symbol)

    return xor_text

# Ввод исходного текста
text = "С Новым Годом, друзья!"

# Создание ключа
key = ''
seed(22)

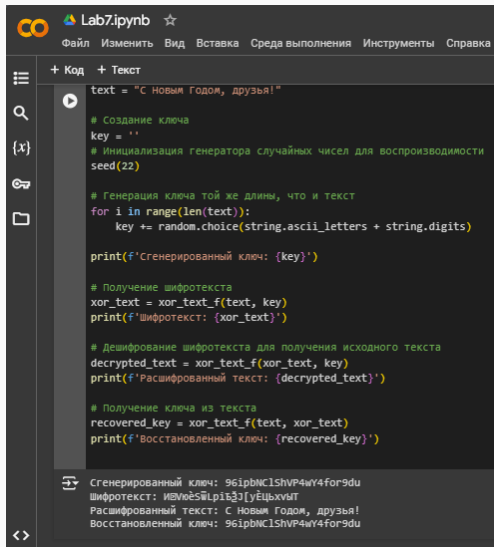
# Генерация ключа той же длины, что и текст
for i in range(len(text)):
    key += random.choice(string.ascii_letters + string.digits)

print(f'Генерированный ключ: {key}')
```

Рис. 1: Код

# Выполнение лабораторной работы

Результат кода (рис.2):



```
Lab7.ipynb ☆
Файл  Изменить  Вид  Вставка  Среда выполнения  Инструменты  Справка

+ Код  + Текст

text = "С Новым Годом, друзья!"

# Создание ключа
key = ''

# Инициализация генератора случайных чисел для воспроизводимости
seed(22)

# Генерация ключа той же длины, что и текст
for i in range(len(text)):
    key += random.choice(string.ascii_letters + string.digits)

print(f'Сгенерированный ключ: {key}')

# Получение шифротекста
xor_text = xor_text_f(text, key)
print(f'Шифротекст: {xor_text}')

# Дешифрование шифротекста для получения исходного текста
decrypted_text = xor_text_f(xor_text, key)
print(f'Расшифрованный текст: {decrypted_text}')

# Получение ключа из текста
recovered_key = xor_text_f(text, xor_text)
print(f'Восстановленный ключ: {recovered_key}')

Сгенерированный ключ: 96ipbNC1ShVP4wY4for9du
Шифротекст: ИЕVpёSЇLp1ё3}[уЁцьхvыГ
Расшифрованный текст: С Новым Годом, друзья!
Восстановленный ключ: 96ipbNC1ShVP4wY4for9du
```

Рис. 2: Продолжение кода и результат

Таким образом, в ходе ЛР№7 я освоила на практике применение режима однократного гаммирования.

- Методические материалы курса.