

Презентация по Лабораторной работе №7

Режим Однократного Гаммирования

Вакутайпа М.

16 мая 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Вакутайпа Милдред
- НКАбд 02-23
- Факультет Физико-математических и Естественных Наук
- Российский университет дружбы народов
- 1032239009@rudn.ru
- <https://wakutaipa.github.io>

Цель работы

Научиться применять режим однократного гарммирования.

Задание

Подобрать ключ, чтобы получить сообщение “С Новым Годом”. Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования.

Выполнение лабораторной работы

Функция для генерации ключа

На языке программирования python, создала функцию для генерации случайного ключа.

```
import string
import random

def generate_key (text):
    return ''.join(random.choice(string.ascii_letters + string.digits) for _ in text)
```

Рис. 1: Функция для генерации ключа

Функция для шифрования и дешифрования

Делала одну функцию для шифрования и дешифрования текста.

```
def crypt (text, key):  
    return ''.join(chr(ord(c)^ord(k)) for c, k in zip(text, key))
```

Рис. 2: Функция для шифрования и дешифрования

функция для нахождения возможных ключей

Нужно определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста. Для этого создала функцию для нахождения возможных ключей для фрагмента текста.

```
def find_keys(cipher, fragment):  
    return [''.join(chr(ord(cipher[i+j])^ord(f)))  
            for i in range(len(cipher)-len(fragment)+1)  
            for j, f in enumerate(fragment)]
```

Рис. 3: функция для нахождения возможных ключей

Далее проверила работы программы. Шифрование и дешифрование происходит верно, как и нахождение ключей, с помощью которых можно расшифровать верно кусок текста.

```
bland = 'С Новым Годом'
key = generate_key(bland)
ciphered = crypt(bland, key)
decrypted = crypt(ciphered, key)
fragment = 'С Новым'

possible_keys = find_keys(ciphered, fragment)

print(f"orig: {bland}, \nkey: {key}, \nciphered: {ciphered}, decrypt: {decrypted}")
print(f"possible keys: {possible_keys}, decrypt: {crypt(ciphered, possible_keys[0])}")
```

orig: С Новым Годом, key: Y65x1PofUajDr, ciphered: 0000000000000000, decrypt: С Новым Годом
possible keys: ['V', '6', '5', 'x', 'l', 'P', 'o', 'a', 'J', '[', ' ', 't', '\x18', '0', '\t', 'A', 'C', 'X', 'a', 'й', 'z', 'g', 'ü', '\x06', 'm', 'V', '\n', 'c', '\x7f', 'n', 'H', 'h', 't', '\x14', 'b', ':', 'e', 'h', 'x', 'm', '\x15', 'F', 'r', 'f', '[', 'a', 'l', 'l', 'r'], decrypt: C

Рис. 4: Проверка

Спасибо За Внимание

Спасибо За Внимание

Спасибо За Внимание