

Презентация по лабораторной работе № 7

Информационная безопасность

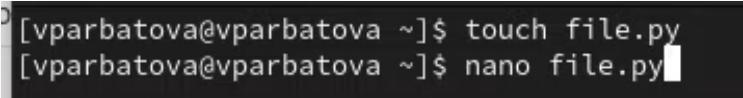
Арбатова В. П.

12 мая 2025

Российский университет дружбы народов, Москва, Россия

Выполнение лабораторной работы

Создаю файл питон, так как буду работать на этом языке и открываю его в редакторе

A terminal window with a dark background and light gray text. The prompt is [vparbatova@vparbatova ~]\$. The first command is touch file.py. The second command is nano file.py, followed by a white cursor block.

```
[vparbatova@vparbatova ~]$ touch file.py  
[vparbatova@vparbatova ~]$ nano file.py
```

Рис. 1: Создание файла

Требуется разработать программу, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Начнем с создания функции для генерации случайного ключа

```
import random
import string

def generate_key_hex(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.ascii_letters + string.digits) #генерация цифры для каждого символа в тексте
    return key
```

Рис. 2: Генерация случайного ключа

Необходимо определить вид шифротекста при известном ключе и известном открытом тексте. Так как операция исключающего или отменяет сама себя, делаю одну функцию и для шифрования и для дешифрования текста

```
def en_de_crypt(text, key):  
    new_text = ''  
    for i in range(len(text)): #проход по каждому символу в тексте  
        new_text += chr(ord(text[i]) ^ ord(key[i % len(key)]))  
    return new_text
```

Рис. 3: Шифрование и дешифрование

Нужно определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. Для этого создаю функцию для нахождения возможных ключей для фрагмента текста

```
def find_possible_key(text, fragment):  
    possible_keys = []  
    for i in range(len(text) - len(fragment) + 1):  
        possible_key = ""  
        for j in range(len(fragment)):  
            possible_key += chr(ord(text[i + j]) ^ ord(fragment[j]))  
        possible_keys.append(possible_key)  
    return possible_keys
```

Рис. 4: Нахождение возможных ключей для фрагмента текста

Вывод результатов

```
t = 'С Новым Годом, друзья!'
key = generate_key_hex(t)
en_t = en_de_crypt(t, key)
de_t = en_de_crypt(en_t, key)
keys_t_f = find_possible_key(en_t, 'С Новым')
fragment = "С Новым"
print('Открытый текст: ', t, "\nКлюч: ", key, '\nШифротекст: ', en_t, '\nИсходный текст: ', de_t,)

print('Возможные ключи: ', keys_t_f)
print('Расшифрованный фрагмент: ', en_de_crypt(en_t, keys_t_f[0]))
```

Рис. 5: Вывод

Проверяю версию питона, запускаю выполнение файла. Шифрование и дешифрование происходит верно, как и нахождение ключей, с помощью которых можно расшифровать верно только кусок текста

```
[vparbatova@vparbatova ~]$ python --version
Python 3.9.19
[vparbatova@vparbatova ~]$ python file.py
Открытый текст: С Новым Годом, друзья!
Ключ: GyhSw3twjjKRiYonMesAEK
Шифротекст: A/vмжЕшWдеGMRU0zЙейЙaj
Исходный текст: С Новым Годом, друзья!
Возможные ключи: ['GyhSw3t', '0єр(3\х0зж', 'ТэX7zME', 'Lк\х1сvж2h', 'dCUwK\х1fC', ' hGf4P', "iwdjM'I", 'V%IA^>w', 'XVbRG0e', '
ццqKжЕ_', '^ьhжC(1', 'MьшQF0', 'Tuђ}}8x', 'Vo~3A\х0f1', '3y\х10MvF6', 'B3nz7A1']
Расшифрованный фрагмент: С НовымЕмьЛпКкРгчQ
[vparbatova@vparbatova ~]$
```

Рис. 6: Результаты