

Отчёт по лабораторной работе №7

дисциплина: Информационная безопасность

Морозова Ульяна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	7

Список иллюстраций

2.1	Импорт	5
2.2	Генерация ключа	5
2.3	(Де)шифрование	5
2.4	Функция нахождения всех возможных ключей	6
2.5	Проверка работы программы	6

1 Цель работы

Освоить на практике применение режима однократного гаммирования

2 Выполнение лабораторной работы

Для выполнения лабораторной работы я написала программу на языке программирования Python.

1. Для начала импортируем необходимые для работы библиотеки (рис. 2.1).

```
[1]: import random  
import string
```

Рис. 2.1: Импорт

2. Создадим функцию для генерации ключа (рис. 2.2).

```
[2]: 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. 2.2: Генерация ключа

3. Затем напишем функцию для (де)шифрования (рис. 2.3).

```
[3]: #для шифрования и дешифрования  
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. 2.3: (Де)шифрование

4. 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 (рис. 2.4).

```
[4]: 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. 2.4: Функция нахождения всех возможных ключей

5. Проверка работы всех функций. Шифрование и дешифрование происходит верно, как и нахождение ключей, с помощью которых можно расшифровать верно только кусок текста (рис. 2.5).

```
[5]: 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, "\n")

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

Открытый текст: С Новым Годом, друзья!
Ключ: ВУККСснFliиизЕРиRgA
Шифротекст: ky10V7uиs0ищd0bAзйиШ
Исходный текст: С Новым Годом, друзья!

Возможные ключи: ['ВУККСс', 'jV'H:\x14f', 'uikk6m1i', '\\i\x15aü\x1eи', 'иB0ag9a', 'j0ëëk@x16p', '-ииLо\x07u', 'иVос-\x02j', 'tй@r{RR', 's0Qui3и', 'iRтй
С:', 'тмэжК{\x0b', 'и090'[С', 'x31.\x054 ', 'иë\r\тйи\и14', 'Pa*А.сk']
Расшифрованный фрагмент: С НовымГодоми0иуии
```

Рис. 2.5: Проверка работы программы

3 Выводы

Мы освоили на практике применение режима однократного гаммирования