

Отчёт по лабораторной работе №7

Дисциплина: Основы информационной безопасности

Дудырев Глеб Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	9

Список иллюстраций

2.1	Функция <code>generate_key()</code>	6
2.2	Функция <code>encrypt()</code>	7
2.3	Функция <code>encrypt()</code>	8

Список таблиц

1 Цель работы

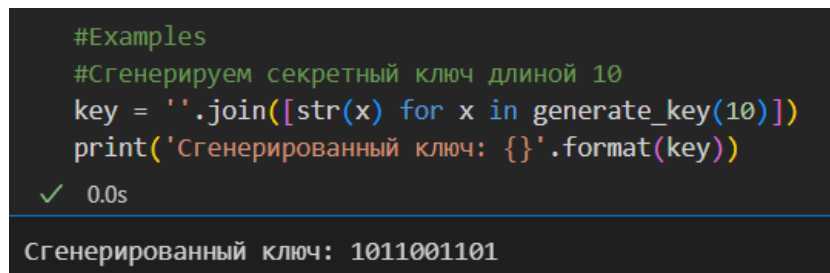
Освоить на практике применение режима однократного гаммирования¹

2 Выполнение лабораторной работы

1. Для начала реализую функцию `generate_key`, которая генерирует случайную последовательность бит, то есть секретный ключ.

```
def generate_key(key_len: int) -> list:
    """
    Функция генерирует псевдослучайную последовательность,
    которая будет использоваться в качестве ключа для шифрования.
    """
    key = [] #Объект, который будет содержать итоговую последовательность
    for i in range(key_len):
        #Генерируем последовательность
        key.append(np.random.randint(0, 2))
    return key
```

Посмотрим на результат работы `generate_key()`: (рис. 2.1)



```
#Examples
#Генерируем секретный ключ длиной 10
key = ''.join([str(x) for x in generate_key(10)])
print('Сгенерированный ключ: {}'.format(key))
```

✓ 0.0s

Сгенерированный ключ: 1011001101

Рис. 2.1: Функция `generate_key()`

2. Далее я реализую функцию `encrypt`, которая производит шифрование открытого текста, с помощью применения однократного кодирования.

```
def encrypt(open_text: str, key: list = None) -> str:
```


Функция шифрует данные в режиме однократного гаммирования.

#Из открытого текста получаем бинарную последовательность

```
open_text_bin = ".join(format(ord(x), '08b') for x in open_text)
```

#Если ключ не передается, то сгенерируем его

if not key:

```
key_len = len(open_text_bin)
```

```
key = generate_key(key_len)
```

#Получаем последовательность бит шифротекста, применяя последовательно XOR к биту из открытого текста и

```
ciphertext_bin = []
```

```
for idx, bit in enumerate(open_text_bin):
```

```
ciphertext_bin.append(int(bit) ^ key[idx])
```

```
ciphertext bin = ".join([str(x) for x in ciphertext bin])
```

#Преобразуем последовательность бит в текст

```
ciphertext = ".join(chr(int(ciphertext_bin[(i * 8):(i * 8 + 8)],2)) for i in range(len(ciphertext_bin) // 8))
```

```
return ciphertext, key
```

Посмотрим на пример работы этой функции: (рис. 2.2)

```
#examples:

open_text = 'Happy New Year, my friends!'
print("Открытый текст: {open_text}")

cipher_text, key = encrypt(open_text)
print("Секретный ключ: {}".format(",".join([str(x) for x in key])))
print("Шифротекст: {cipher_text}")
```

Рис. 2.2: Функция encrypt()

3. Следующим шагом я реализовал функцию, которая призводит дешифрования шифротекста.

```

def decrypt(cipher_text: str, key) -> str:
    """
    Функция, которая производит дешифрование
    """

    if not key: #Если ключ не передали, то завершаем работу программы
        return 'You should enter the secret key.'

    #Из зашифрованного текста получаем бинарную последовательность
    cipher_text_bin = ''.join(format(ord(x), '08b') for x in cipher_text)

    #Получаем последовательность бит открытого текста, применяя последовательно XOR к биту из шифротекста и
    open_text_bin = []

    for idx, bit in enumerate(cipher_text_bin):
        open_text_bin.append(int(bit) ^ key[idx])


    open_text_bin = ''.join([str(x) for x in open_text_bin])

    #Преобразуем последовательность бит в текст
    open_text = ''.join(chr(int(open_text_bin[(i * 8):(i * 8 + 8)], 2)) for i in range(len(open_text_bin) // 8))

    return open_text, key

```

Посмотрим на пример ее работы: (рис. 2.3)



```

#Examples
print(f'Шифротекст: {cipher_text}')
res_text, key = decrypt(cipher_text, key)
print('Серкетный ключ: {}'.format(''.join([str(x) for x in key])))
print(f'Текст после дешифровки: {res_text}')
✓ 0.0s

Шифротекст: 00ä0ä#ÜÀ@ää00ä0oáLü0L"Ü~]
Серкетный ключ: 101100111111001001001010011111101100111011011011011000110010001101010111011100
Текст после дешифровки: Happy New Year, my friends!

```

Рис. 2.3: Функция encrypt()

3 Выводы

Я приобрел навыки написания программ с использованием циклов и обработкой аргументов командной строки. Написал программу для вычисления суммы значений от заданной функции, в которой аргументы вводятся с командной строки