

# **Отчет по лабораторной работе №7**

**Дисциплина: Основы информационной безопасности**

Иванов Сергей Владимирович

# Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Ответы на контрольные вопросы	9
5	Вывод	11

## Список иллюстраций

3.1	Функция генерации ключа . . . . .	6
3.2	Функция шифрования и дешифрования . . . . .	6
3.3	Поиск возможных ключей . . . . .	7
3.4	Проверка работы программы . . . . .	7

# **1 Цель работы**

Освоить на практике применение режима однократного гаммирования.

## 2 Задание

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.

### 3 Выполнение лабораторной работы

Я выбрал для написания программы язык Python. Для начала напишем функцию для генерации случайного ключа (рис. 1)

```
1 import random
2 import string
3
4 # генерация ключа
5 usage
6 def generate_key(text):
7     key = ''
8     for i in range(len(text)):
9         key += random.choice(string.ascii_letters + string.digits)
10    return key
```

Рис. 3.1: Функция генерации ключа

Далее необходимо определить вид шифротекста при известном ключе и известном открытом тексте. Т.к операция XOR исключает сама себя, пишу одну функцию для шифрования и дешифрования текста(рис. 2).

```
11 # шифрование и дешифрование текста
12 3 usages
13 def en_de_crypt(text, key):
14     new_text = ''
15     for i in range(len(text)):
16         new_text += chr(ord(text[i]) ^ ord(key[i % len(key)]))
17    return new_text
```

Рис. 3.2: Функция шифрования и дешифрования

Далее 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 пишу функцию для поиска возможных ключей для фрагмента текста. (рис. 3).

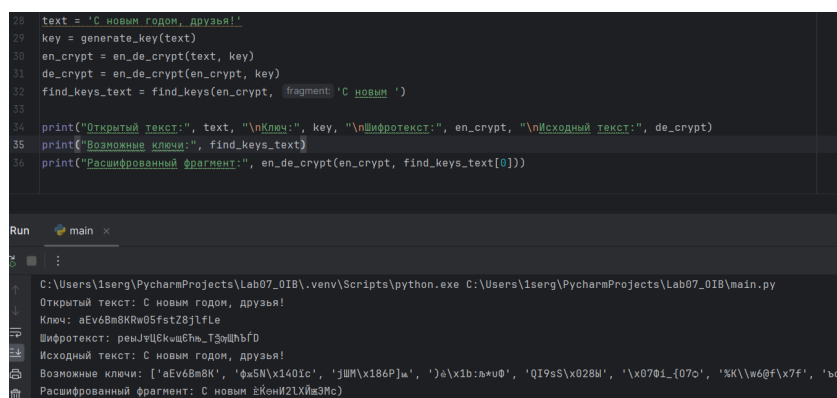
```

18 # функция для нахождения возможных ключей
19 def find_keys(text, fragment):
20     possible_keys = []
21     for i in range(len(text) - len(fragment) + 1):
22         possible_key = ''
23         for j in range(len(fragment)):
24             possible_key += chr(ord(text[i + j]) ^ ord(fragment[j]))
25         possible_keys.append(possible_key)
26     return possible_keys

```

Рис. 3.3: Поиск возможных ключей

Проверяем всех функций. Убеждаемся, все работает корректно. (рис. 4).



```

28 text = 'С новым годом, друзья!'
29 key = generate_key(text)
30 en_crypt = en_de_crypt(text, key)
31 de_crypt = en_de_crypt(en_crypt, key)
32 find_keys_text = find_keys(en_crypt, fragment='С новым ')
33
34 print("Открытый текст:", text, "\nКлюч:", key, "\nШифротекст:", en_crypt, "\nИсходный текст:", de_crypt)
35 print("Возможные ключи:", find_keys_text)
36 print("Расшифрованный фрагмент:", en_de_crypt(en_crypt, find_keys_text[0]))

```

Run main x

```

C:\Users\Iserg\PycharmProjects\Lab07_01B\venv\Scripts\python.exe C:\Users\Iserg\PycharmProjects\Lab07_01B\main.py
Открытый текст: С новым годом, друзья!
Ключ: aEv6Bm8Krw05fstZ8jlfle
Шифротекст: remJvQ6kwaEhь_T3щhьbD
Исходный текст: С новым годом, друзья!
Возможные ключи: ['aEv6Bm8K', 'фа5N\x140Ic', 'jШN\x186P]м', '')a\x1b:ъmu0', 'QI9sS\x028u', '\x0701_070', 'ЖK\w6f\x7f', 'ьс']
Расшифрованный фрагмент: С новым ьКенW2LXИa3Mc)

```

Рис. 3.4: Проферка работы программы

Листинг программы

```

import random
import string

# генерация ключа
def generate_key(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.ascii_letters + string.digits)
    return key

```

```

# шифрование и дешифрование текста
def en_de_crypt(text, key):
    new_text = ''
    for i in range(len(text)):
        new_text += chr(ord(text[i]) ^ ord(key[i % len(key)]))
    return new_text

# функция для нахождения возможных ключей
def find_keys(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

text = 'С новым годом, друзья!'
key = generate_key(text)
en_crypt = en_de_crypt(text, key)
de_crypt = en_de_crypt(en_crypt, key)
find_keys_text = find_keys(en_crypt, 'С новым ')

print("Открытый текст:", text, "\nКлюч:", key, "\nШифротекст:", en_crypt, "\nИсходный текст:", de_crypt)
print("Возможные ключи:", find_keys_text)
print("Расшифрованный фрагмент:", en_de_crypt(en_crypt, find_keys_text[0]))

```



## 4 Ответы на контрольные вопросы

1. Поясните смысл одноразового гаммирования. - Одноразовое гаммирование - это метод шифрования, при котором каждый символ открытого текста гаммируется с соответствующим символом ключа только один раз.
2. Перечислите недостатки одноразового гаммирования. - Недостатки одноразового гаммирования:
  - Уязвимость к частотному анализу из-за сохранения частоты символов открытого текста в шифротексте.
  - Необходимость использования одноразового ключа, который должен быть длиннее самого открытого текста.
  - Нет возможности использовать один ключ для шифрования разных сообщений.
3. Перечислите преимущества одноразового гаммирования. - Преимущества одноразового гаммирования:
  - Высокая стойкость при правильном использовании случайного ключа.
  - Простота реализации алгоритма.
  - Возможность использования случайного ключа.
4. Почему длина открытого текста должна совпадать с длиной ключа? - Длина открытого текста должна совпадать с длиной ключа, чтобы каждый символ открытого текста гаммировался с соответствующим символом ключа.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности? - В режиме однократного гаммирования используется операция XOR (исключающее ИЛИ), которая объединяет двоичные значения символов открытого текста и ключа для получения шифротекста. Особенность XOR - если один из битов равен 1, то результат будет 1, иначе 0.
6. Как по открытому тексту и ключу получить шифротекст? - Для получения шифротекста по открытому тексту и ключу каждый символ открытого текста гаммируется с соответствующим символом ключа с помощью операции XOR.
7. Как по открытому тексту и шифротексту получить ключ? - По открытому тексту и шифротексту невозможно восстановить действительный ключ, так как для этого нужна информация о каждом символе ключа.
8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра - Необходимые и достаточные условия абсолютной стойкости шифра:
- Ключи должны быть случайными и использоваться только один раз.
  - Длина ключа должна быть не менее длины самого открытого текста.
  - Ключи должны быть храниться и передаваться безопасным способом.

## **5 Вывод**

В ходе выполнения лабораторной работы мной было освоено на практике применение режима однократного гаммирования.