Отчет по лабораторной работе №8

Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

Татьяна Александровна Буллер

Содержание

Список иллюстраций

1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

2 Выполнение лабораторной работы

2.1 Генерация ключа

В отличие от предыдущей работы, закодированные сообщения пользователь может выбрать сам. Поэтому для упрощения реализации возьмем стандартную кодировку ASCII, известную без дополнительных включений всем языкам программирования. Для обоих шифротекстов необходимо сгенерировать ключ. Помним, что для обеспечения шифру адекватной криптостойкости, длина ключа должна быть не менее длины шифротекста. Напишем функцию, которая по переданной ей длине текста сгенерирует для него случайную строку ключа шифрования:

```
1 import random
2 import string
3
4 # сгенерировать ключ ascii
5 def key_hex(n):
6     key = ''
7     for i in range(n):
8         key += random.choice(string.ascii_letters + string.digits)
9     return key
10
```

Рис. 2.1: Функция генерацияи ключа

2.2 Функция операции XOR для двух строк

Далее необходимо было написать код, с помощью которого можно было бы зашифровать сообщение. Для этого напишем простую функцию, которая в цикле по каждому символу исходного текста и соответствующему символу ключа

сгенерирует символ нового текста.

```
# зашифровать по ключу
def cipher(plain, key):
    new_text = ''
    for i in range(len(plain)):
        new_text += chr(ord(plain[i]) ^ ord(key[i % len(key)]))
    return new_text
```

Рис. 2.2: Функция шифрования

2.3 Тело программы

В теле программы зададим две строки открытого текста одинаковой длины (plain1 и plain2), взяв за текст стандартную рыбу Lorem Ipsum. По одному из текстов (в представленном примере - по первому) сгенерируем ключ с помощью написанной ранее функции, затем продемонстрируем шифрование строки этим ключом и дешифрование полученных шифротекстов.

```
plain1 = 'Neque porro quisquam est qui dolorem' # открытый текст 1
key = key_hex(len(plain1)) # сгенерировать ключ для обоих текстов
cipher1 = cipher(plain1, key) # шифротекст 1
deciphered1 = cipher(cipher1, key) # расшифровать шифротекст 1 по ключу

plain2 = 'Fusce eu venenatis erat. Aliquam sed' # открытый текст 2
cipher2 = cipher(plain2, key) # шифротекст 2
deciphered2 = cipher(cipher2, key) # расшифровать шифротекст 2 по ключу

# человекочитаемый вывод

print('оригинальный текст 1: ', plain1, "\nсгенерированный ключ: ", key,
'\nшифротекст: ', cipher1, '\nрасшифровка: ', deciphered1,'\n')
print('оригинальный текст: ', plain2, "\nсгенерированный ключ: ", key,
'\nшифротекст: ', cipher2, '\nрасшифровка: ', deciphered2,'\n')
```

Рис. 2.3: Тело программы: открытый текст, генерация ключа, шифротекст и расшифровка по ключу

2.4 Расшифровать один текст через второй

В случае, когда мы имеем два сообщения одинаковой длины и известно, что они были зашифрованы одним ключом, мы можем расшифровать одно сообщение через второе. Для этого проведем простую операцию: XOR для двух шифротекстов.

Это позволит получить ключ шифрования, который мог бы быть использован, чтобы операцией XOR на одном из открытых текстов получить второй. Владея одним из открытых текстов и этим ключом, мы можем получить второй открытый текст.

```
xored = cipher(cipher2, cipher1) # проксорить второй через первый
print('второй по первому: ', cipher(plain1, xored))
print('первый по второму: ', cipher(plain2, xored))
```

Рис. 2.4: Расшифровка одного текста через второй

2.5 Вывод программы

Человекочитаемый вывод программы состоит из трех частей: операции над первым текстом (сам открытый текст, сгенерированный ключ, шифротекст и открытый текст, полученный от шифротекста и ключа), аналогичные операции для второго текста и получение открытых сообщений по двум шифротекстам и одному открытому.

```
┌──(tabuller㉿jordi)-[~/work/study/2024-2025/Основы информационной безопасности/info
sec-intro/labs/lab8]
└─$ python3 lab8.py
оригинальный текст 1:  Neque porro quisquam est qui dolorem
сгенерированный ключ:  LlTPLJbGi2FWfOinM3Lkjw3OoRKjEz4UmGzh
шифротекст:  %%)j()w:<F-J@;O#>e[95
расшифровка:  Neque porro quisquam est qui dolorem

оригинальный текст:  Fusce eu venenatis erat. Aliquam sed
сгенерированный ключ:  LlTPLJbGi2FWfOinM3Lkjw3OoRKjEz4UmGzh
шифротекст:
'3)j2ID#9 $@l GaO'4U8M4

расшифровка:  Fusce eu venenatis erat. Aliquam sed

второй по первому:  Fusce eu venenatis erat. Aliquam sed
первый по второму:  Neque porro quisquam est qui dolorem
```

Рис. 2.5: Вывод программы

3 Листинг программы

```
import random
import string

# сгенерировать ключ ascii
def key_hex(n):
    key = ''
    for i in range(n):
        key += random.choice(string.ascii_letters + string.digits)
    return key

# зашифровать по ключу
def cipher(plain, key):
    new_text = ''
    for i in range(len(plain)):
        new_text += chr(ord(plain[i]) ^ ord(key[i % len(key)]))
    return new_text

plain1 = 'Neque porro quisquam est qui dolorem' # открытый текст 1
key = key_hex(len(plain1)) # сгенерировать ключ для обоих текстов
cipher1 = cipher(plain1, key) # шифротекст 1
deciphered1 = cipher(cipher1, key) # расшифровать шифротекст 1 по ключу
```

```
plain2 = 'Fusce eu venenatis erat. Aliquam sed' # открытый текст 2
cipher2 = cipher(plain2, key) # шифротекст 2
deciphered2 = cipher(cipher2, key) # расшифровать шифротекст 2 по ключу

# человекочитаемый вывод

print('оригинальный текст 1: ', plain1, "\ncгенерированный ключ: ", key, '\nшифротекст
print('оригинальный текст: ', plain2, "\ncгенерированный ключ: ", key, '\nшифротекст:

xored = cipher(cipher2, cipher1) # проксорить второй через первый
print('второй по первому: ', cipher(plain1, xored))
print('первый по второму: ', cipher(plain2, xored))
```

4 Ответы на контрольные вопросы

1. Как, зная один из текстов (P_1 или P_2), определить другой, не зная при этом ключа?

В случае, когда мы имеем два сообщения одинаковой длины и известно, что они были зашифрованы одним ключом, мы можем расшифровать одно сообщение через второе. Для этого проведем простую операцию: XOR для двух шифротекстов. Это позволит получить ключ шифрования, который мог бы быть использован, чтобы операцией XOR на одном из открытых текстов получить второй. Владея одним из открытых текстов и этим ключом, мы можем получить второй открытый текст.

2. Что будет при повторном использовании ключа при шифровании текста?

При повторном применении того же ключа к уже зашифрованному им тексту будет получен открытый текст.

3. Как реализуется режим шифрования однократного гаммирования одним ключом двух открытых текстов?

Режим шифрования однократного гаммирования одним ключом двух открытых текстов реализуется через использование операции XOR над каждым битом открытого текста сквозь ключ. Для шифрования двух текстов применяется один и тот же ключ.

4. Перечислите недостатки шифрования одним ключом двух открытых текстов

Повторное использование ключа позволяет расшифровать любое сообщение, зашифрованное этим ключом, если известена хотя бы одна пара шифротекст:открытый текст.

5. Перечислите преимущества шифрования одним ключом двух открытых текстов

Снижение затрат ресурсов памяти на генерацию нового ключа для каждого следующего сообщения.

5 Выводы

Было освоено на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом, написана программа, шифрующая две строки одним случайно сгенерированным ключом и дешифрующая шифротексты обратно по одному открытому тексту и двум шифрам.