

# **Отчёт по лабораторной работе**

**Лабораторная №8**

Дерябина Мария Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическая часть</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Вывод</b>	<b>10</b>

# List of Tables

## List of Figures

3.1	Функция для дешифрования сообщения . . . . .	7
3.2	Функция для шифрования сообщения . . . . .	8
3.3	Функция для дешифрования сообщения без ключа . . . . .	8
3.4	Вызов функций для тестирования . . . . .	8
3.5	Тестирование программы . . . . .	9

# 1 Цель работы

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

## 2 Теоретическая часть

Если даны две телеграммы Центра, то шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:  $C1 = P1 \wedge K$ ,  $C2 = P2 \wedge K$ . Чтобы найти открытый текст, зная шифротекст двух телеграмм, зашифрованных одним ключом, надо сложить по модулю 2 эти два равенства. Тогда с учётом свойства операции XOR

$$1 \wedge 1 = 0, 1 \wedge 0 = 1,$$

получаем:

$$C1 \wedge C2 = P1 \wedge K \wedge P2 \wedge K = P1 \wedge P2.$$

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар  $C1 \wedge C2$  (известен вид обеих шифровок). Тогда зная  $P1$ , имеем:

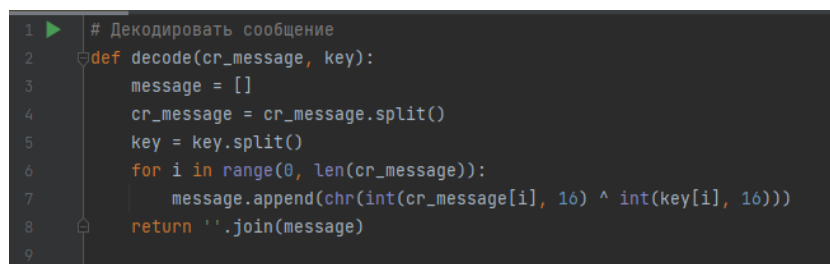
$$C1 \wedge C2 \wedge P1 = P1 \wedge P2 \wedge P1 = P2.$$

Таким образом, злоумышленник получает возможность определить те символы сообщения  $P2$ , которые находятся на позициях известного шаблона сообщения  $P1$ . В соответствии с логикой сообщения  $P2$ , злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения  $P2$ . Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

### 3 Выполнение лабораторной работы

Написала программу на языке Python, позволяющую шифровать и дешифровать данные в режиме одноразового гаммирования. Программа имеет 3 функции:

1. `decode(cr_message, key)`. Данная функция принимает зашифрованное сообщение и ключ (в виде строк с шестнадцатиричными значениями). Для каждого значения зашифрованного сообщения выполняется сложение по модулю 2 с соответствующим значением ключа. Функция возвращает строку с расшифрованным сообщением (рис. 3.1).

A screenshot of a code editor showing the implementation of the `decode` function. The code is as follows:

```
1 # Декодировать сообщение
2 def decode(cr_message, key):
3     message = []
4     cr_message = cr_message.split()
5     key = key.split()
6     for i in range(0, len(cr_message)):
7         message.append(chr(int(cr_message[i], 16) ^ int(key[i], 16)))
8     return ''.join(message)
9
```

Figure 3.1: Функция для дешифрования сообщения

2. `def encode(message, key)`. Данная функция принимает исходное сообщение и ключ. Каждый символ сообщения преобразовывается в число, соответствующее его коду в системе Unicode. Далее выполняется сложение по модулю 2 между полученными кодами и соответствующими значениями ключа. Функция возвращает зашифрованное сообщение в виде строки с шестнадцатиричными значениями (рис. 3.2).

```

11 # Закодировать сообщение
12 def encode(message, key):
13     cr_message = []
14     key = key.split()
15     for i in range(0, len(message)):
16         cr_message.append(hex(ord(message[i]) ^ int(key[i], 16)).lstrip('0x')).upper()
17         if len(cr_message[i]) == 1:
18             cr_message[i] = '0' + cr_message[i]
19     return ' '.join(cr_message)

```

Figure 3.2: Функция для шифрования сообщения

3. `get_message(cr_message1, cr_message2, message2)`. Данная функция принимает зашифрованное сообщение, шаблон исходного сообщения и зашифрованное шаблонное сообщение. Выполняется сложение по модулю 2 между значениями закодированного сообщения, кодами символов шаблонного сообщения и значениями закодированного шаблонного сообщения. Функция возвращает строку с расшифрованным сообщением(рис. 3.3).

```

22 def get_message(cr_message1, cr_message2, message2):
23     message1 = []
24     cr_message1 = cr_message1.split()
25     cr_message2 = cr_message2.split()
26     for i in range(0, len(cr_message1)):
27         message1.append(chr(int(cr_message1[i], 16) ^ int(cr_message2[i], 16) ^ ord(message2[i])))
28     return ' '.join(message1)

```

Figure 3.3: Функция для дешифрования сообщения без ключа

Написала код с вызовом функций для тестирования (рис. 3.4).

```

36 print('Определим вид шифротекста_1 при известном ключе и известном открытом тексте')
37 message = input('Введите текст сообщения_1: ')
38 key = input('Введите ключ: ')
39 cr_message_test = encode(message, key)
40 print('Закодированное сообщение_1:', cr_message_test)
41
42 print()
43 print('Определим вид шифротекста_2 при известном ключе и известном открытом тексте')
44 message = input('Введите текст сообщения_2: ')
45 key = input('Введите ключ: ')
46 cr_message_test = encode(message, key)
47 print('Закодированное сообщение_2:', cr_message_test)
48
49 print()
50 print('Декодируем сообщение при его известном шифротексте, имея известный шаблон и его шифротекст')
51 cr_message1 = input('Введите текст закодированного сообщения: ')
52 message2 = input('Введите текст шаблонного сообщения: ')
53 cr_message2 = input('Введите текст закодированного шаблонного сообщения: ')
54 message1 = get_message(cr_message1, cr_message2, message2)
55 print('Декодированное сообщение:', message1)

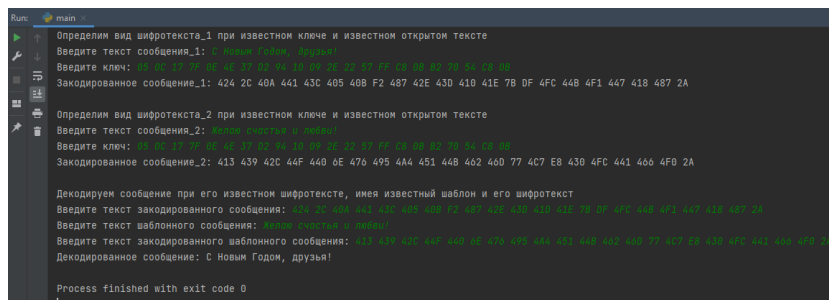
```

Figure 3.4: Вызов функций для тестирования

Протестировала программу на сообщениях 'С Новым Годом, друзья!' и 'Желаю счастья и любви!'. Вначале программа определила вид шифротекста сообщений (при



одинаковом ключе). Далее была вызвана функция `get_message()`, в которую были переданы шифротекст первого сообщения, а также исходный текст и шифротекст второго сообщения. После обработки этих данных, функция корректно определила исходный текст первого сообщения (рис. 3.5).



```
Run: main
↑
Определим вид шифротекста_1 при известном ключе и известном открытом тексте
Введите текст сообщения_1: С Новым Годом, друзья!
Введите ключ: 12 34 56 78 9A BC DE F0 12 34 56 78 9A BC DE
Закодированное сообщение_1: 424 2C 40A 441 43C 405 40B F2 487 42E 43D 410 41E 7B DF 4FC 44B 4F1 447 418 487 2A

Определим вид шифротекста_2 при известном ключе и известном открытом тексте
Введите текст сообщения_2: С Новым Годом, друзья!
Введите ключ: 12 34 56 78 9A BC DE F0 12 34 56 78 9A BC DE
Закодированное сообщение_2: 413 439 42C 44F 440 0E 470 495 4A4 451 44B 462 46D 77 4C7 E8 430 4FC 441 460 4F0 2A

Декодируем сообщение при его известном шифротексте, имея известный шаблон и его шифротекст
Введите текст закодированного сообщения: 424 2C 40A 441 43C 405 40B F2 487 42E 43D 410 41E 7B DF 4FC 44B 4F1 447 418 487 2A
Введите текст шаблонного сообщения: С Новым Годом, друзья!
Введите текст закодированного шаблонного сообщения: 413 439 42C 44F 440 0E 470 495 4A4 451 44B 462 46D 77 4C7 E8 430 4FC 441 460 4F0 2A
Декодированное сообщение: С Новым Годом, друзья!

Process finished with exit code 0
```

Figure 3.5: Тестирование программы

## 4 Вывод

Я освоила на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом. Определила способ, при котором злоумышленник может прочесть оба текста, не зная ключа и не стремясь его определить.