# Отчет по лабораторной работе 7

Даниил Генералов, 1032212280

# Содержание

# Список иллюстраций

# Список таблиц

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования.

# 2 Выполнение лабораторной работы

Эта лабораторная работа подразумевает разработку программы, которая использует шифрование с режимом однократного гаммирования: это подразумевает, что байты сообщения смешиваются с помощью XOR с байтами ключа.

Если ключ используется только один раз, то этот способ шифрования называется *one-time pad*.

Это единственный теоретически-безопасный способ шифрования: если не знать ключа, то, пытаясь подобрать ключ, можно получить любое сообщение длины, совпадающей с исходной длиной. Нет никакой возможности определить, какое из сообщений является настоящим.

Этот алгоритм является симметричным алгоритмом шифрования: тот же самый ключ используется как для того, чтобы получить шифртекст, так и для того, чтобы из шифртекста получить исходный текст. Он также является симметричным в более широком смысле: нет никакой разницы между ключом и шифртекстом.

Благодаря этому свойству мы можем решить задачу в лабораторной работе: нам дан шифртекст `DD FE FF 8F E5 A6 C1 F2 B9 30 CB D5 02 94 1A 38 E5 5B 51 75`, который можно сочетать с двумя разными ключами, чтобы получить разные сообщения: `Штирлиц – Вы Герой!!` и `Штирлиц - Вы Болван!`. Нам требуется найти ключ, используя который, можно получить `С Новым Годом, друзья!`.

Поскольку все сообщения написаны в кодировке CP1251, нужно, чтобы и это сообщение было написано в ней: это можно сделать командой на Python, как на

рис. fig. 2.1.

```
>>> text = 'С Новым Годом, друзья!'
>>> encoded = text.encode('cp1251')
>>> encoded
b'\xd1 \xcd\xee\xe2\xfb\xec \xc3\xee\xe4\xee\xec, \xe4\xf0\xf3\xe7\xfc\xff!'
>>> encoded.hex(' ')
'd1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21'
>>>
```

Рис. 2.1: python

Значит, это сообщение имеет байты `d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21`. Из-за того, что это сообщение на два байта длинее, чем исходные, нужно его сократить: если удалить пробел после буквы `С` и после запятой, то получится `d1 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c e4 f0 f3 e7 fc ff 21`

После этого мы пишем программу, которая читает два значения: простой текст и ключ – и выдает шифртекст. Из-за симметрии алгоритма, эта же программа может принять шифртекст и ключ и выдать исходный текст. Эту программу можно увидеть на рис. fig. 2.2.

labs > lab7 > report > crypt.py > ...

```python
try:
    plaintext_text = plaintext.decode('cp1251')
    print('Plaintext как CP1251:')
    print(plaintext_text)
except:
    print('Plaintext не CP1251')

print()
print('Key:')
key = bytes.fromhex(input('> '))

if len(plaintext) != len(key):
    print('Plaintext и key должны быть одной и той же длины')

print()
print('Ciphertext:')
ciphertext = bytes([a ^ b for a, b in zip(plaintext, key)])
print(ciphertext.hex(' '))
try:
    ciphertext_text = ciphertext.decode('cp1251')
    print('Ciphertext как CP1251:')
    print(ciphertext_text)
except:
    print('Ciphertext не CP1251')
```

Рис. 2.2: python

Затем можно использовать эту программу, чтобы проверить, что расшифровка

происходит таким образом как мы ожидаем: на рис. fig. 2.3 я передаю туда параметры из примера и вижу, что результат действительно тот, который должен быть.

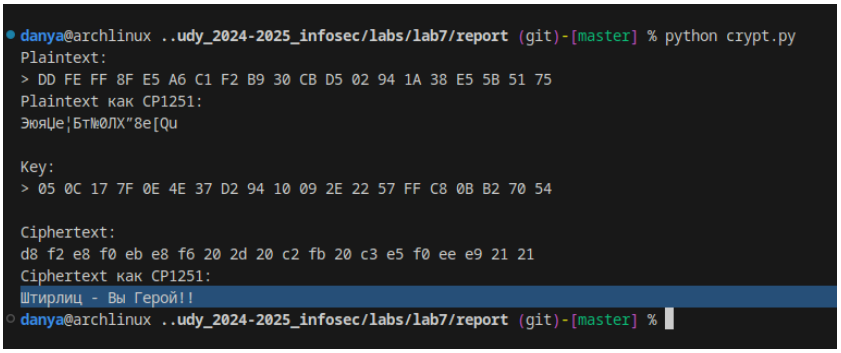


Рис. 2.3: python

После этого можно попробовать передать как один из параметров шифртекст, а как другой – тот текст, который нужно, чтобы получился.

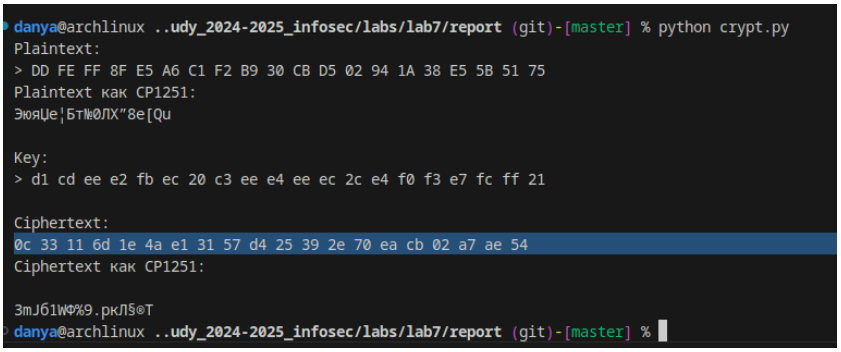


Рис. 2.4: python

В результате комбинации этих двух строк мы получаем `0c 33 11 6d 1e 4a e1 31 57 d4 25 39 2e 70 ea cb 02 a7 ae 54` – строку, которую нужно использовать как ключ. Наконец, мы комбинируем этот ключ с исходным зашифрованным сообщением, и на рис. fig. 2.5 можно увидеть, что сообщение `CНовым Годом,друзья!` получилось из исходного шифртекста. Это доказывает, что один и тот же шифртекст можно интерпретировать как любое сообщение данной длины.

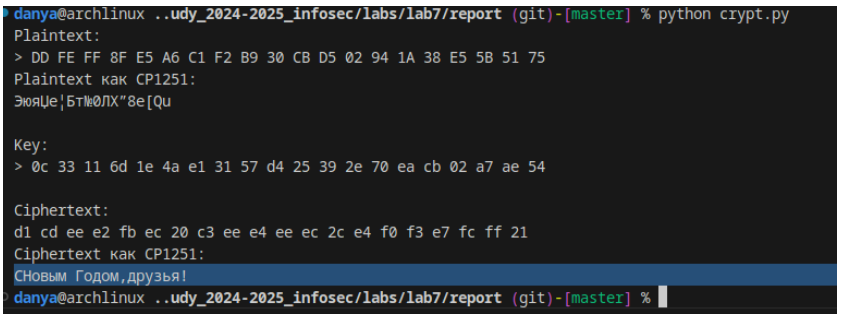


Рис. 2.5: python

# 3 Контрольные вопросы

1. Поясните смысл однократного гаммирования.

Однократное гаммирование заключается в комбинации простого текста и ключа с помощью операции XOR для получения шифртекста, а затем комбинации этого же ключа и шифртекста для получения исходного текста.

2. Перечислите недостатки однократного гаммирования.

Ключ сообщения должен иметь такую же длину, как и само сообщение, и он не может быть использован больше одного раза (иначе тривиально определить, какой это ключ, зная два разных шифртекста). Из-за этого практические использования такого шифра очень ограниченны.

3. Перечислите преимущества однократного гаммирования.

Этот шифр единственный предоставляет абсолютную теоретическую безопасность: изменяя значение ключа, можно получить из одного и того же шифртекста любой исходный текст, и нет возможности определить, какой из них правильный.

4. Почему длина открытого текста должна совпадать с длиной ключа?

Шифртекст – это открытый текст, у которого каждый бит был XOR-ен с каждым битом ключа. Если битов ключа не хватает, то нужно найти их где-то еще, но любые способы расширения ключа не имеют теоретической

безопасности. Практические алгоритмы симметричного шифрования фактически используют однократное гаммирование, но используя ключ, который создан из псевдослучайного генератора: этот компонент не имеет теоретической безопасности, и поэтому такая схема не является настоящим однократным гаммированием.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности?

Операция XOR комбинирует два бита, возвращая 1, если эти биты были разными, и 0, если они были одинаковыми. Эта операция обратимая и коммутативная, из-за чего в алгоритме нет строго различия между шифртекстом, простым текстом и ключом.

6. Как по открытому тексту и ключу получить шифротекст?

Нужно сделать XOR между этими двумя строками.

7. Как по открытому тексту и шифротексту получить ключ?

Также, нужно сделать XOR между этими двумя строками. Именно из-за этого нет принципиальной разницы между ключом и шифртекстом.

8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

Необходимо и достаточно использование уникальных ключей для каждого сообщения, которые никогда больше не повторяются. Если сделать не так, то можно определить биты ключа по совпадениям между разными шифртекстами.

# 4 Выводы

В этой лабораторной работе мы рассмотрели алгоритм однократного гаммирования и показали, каким образом можно шифровать и дешифровать сообщения с помощью него.