Отчёт по лабораторной работе №7

Дисциплина: Основы информационной безопасности

Мишина Анастасия Алексеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить на практике применение режима однократного гаммирования [1].

2 Выполнение лабораторной работы

Создаем функцию encrypt() (рис. 2.1), которая будет шифровать заданный текст с помощью гаммирования. Также есть возможность подавать на вход определенный ключ шифрования. Если ключа нет, то он генерируется рандомно. Сначала исходный текст и ключ шифрования преобразуются в 16-ую систему счисления, затем, применяется операция XOR для каждого элемента ключа и текста. Полученный шифротекст декодируется из 16-ой СС и получается набор из символов.

```
def encrypt(text: str, key: list = None):
    '''
    Выводит шифротекст для заданного текста.
    Если ключа нет, то генерируется случайный ключ
    '''

    text_16 = [char.encode(encoding='cp1251').hex().upper() for char in text]
    if not key:
        key = generate_key(length=len(text))

    print(f"Ключ шифрования:", ' '.join(str(s) for s in key))
    print(f"Исходный текст:", ' '.join(text_16))

    encrypted_text = []
    for i in range(len(text)):
        xor_char = int(text_16[i], 16) ^ int(key[i], 16)
        encrypted_text.append(int2hex(xor_char))

    encrypted_text = validate(encrypted_text)
    ciphertext = bytes.fromhex(''.join(encrypted_text)).decode('cp1251')
    print(f"Шифротекст: {ciphertext}\n\n")

    return {
        'key': key,
        'ciphertext': ciphertext
    }
```

Рис. 2.1: Функция encrypt()

Вызов функции функции encrypt() (рис. 2.2) и результаты ее работы (рис. 2.3):

```
encryption = encrypt('С Новым Годом, друзья!')
```

Рис. 2.2: Вызов функции encrypt()

```
[aamishina@aamishina Documents]$ python main.py
Ключ шифрования: 7A C0 EB A1 20 79 F5 71 68 75 32 8B 4C 52 F3 BE 3A C0 FF A3 CB 66
Исходный текст: D1 20 CD EE E2 FB EC 20 C3 EE E4 EE EC 2C 20 E4 F0 F3 E7 FC FF 21
Шифротекст: «a&OB,Q«›Це ~YZK3_4G
```

Рис. 2.3: Результаты работы функции encrypt()

Далее создаем функцию decrypt() (рис. 2.4). Она по заданному шифротексту выводит исходный текст. Также есть возможность опционально задать ключ дешифровки, или же он будет создан автоматически. Функция преобразует шифротекст в 16-ую СС и применяет XOR для шифротекста и ключа.

```
def decrypt(ciphertext: str, key: list = None):
    ciphertext_16 = [char.encode('cp1251').hex().upper() for char in ciphertext]
    if not key:
        key = generate_key(length=len(ciphertext))

    print(f"Ключ шифрования:", ' '.join(str(s) for s in key))
    print(f"Исходный шифротекст:", ciphertext)

    decrypted_text = []
    for i in range(len(ciphertext)):
        xor_char = int(ciphertext_16[i], 16) ^ int(key[i], 16)
        decrypted_text.append(int2hex(xor_char))

    decrypted_text = validate(decrypted_text)
    decrypted_text = bytes.fromhex(''.join(decrypted_text)).decode('cp1251')

    print('Расшифрованный текст: ', decrypted_text)

    return {
        'key': key,
        'text': decrypted_text
    }
```

Рис. 2.4: Функция decrypt()

Вызов функции функции decrypt() с тем же ключом, что и в шифровании, (рис. 2.5) и результаты ее работы (рис. 2.6):

```
encryption = encrypt('С Новым Годом, друзья!')
decrypt(encryption['ciphertext'], key=encryption['key'])

:wq
```

Рис. 2.5: Вызов функции denrypt()

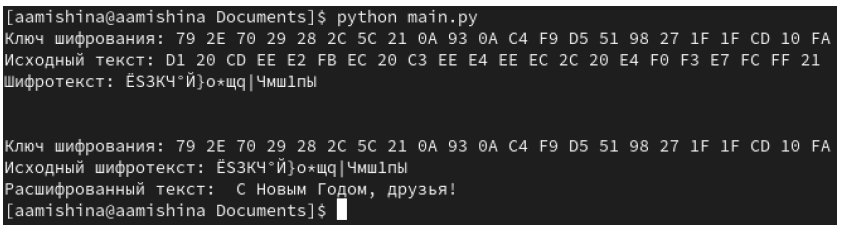

Рис. 2.6: Результаты работы функции denrypt()

Вызов функции функции decrypt() со случайным ключом (рис. 2.7) и результаты ее работы (рис. 2.8):

```
encryption = encrypt('С Новым Годом, друзья!')
decrypt(encryption['ciphertext'])
```

Рис. 2.7: Вызов функции denrypt()

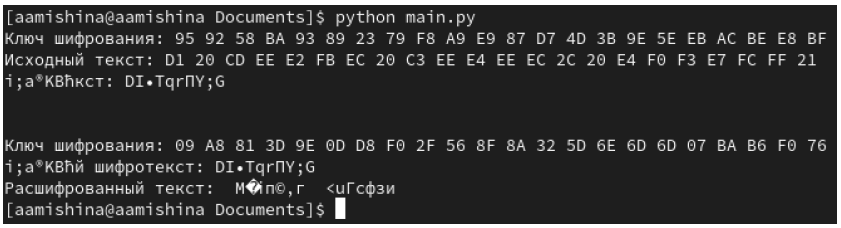

Рис. 2.8: Результаты работы функции denrypt()

Также создаем функцию find_key() (рис. 2.9). Она вызывает функцию decrypt() до тех пор, пока расшифрованный и исходный текст не совпадут, т.е. пытается подобрать ключ дешифровки.

```python
def find_key(text):
    '''
    Подбирает ключ, с помощью которого сообщение было зашифровано
    '''

    decrypted_text = ''
    encryption = encrypt(text)

    while decryption_text != text:
        decryption = decrypt(encryption['ciphertext'])
        decrypted_text = decryption['text']
        print(f'Полученный текст: {decrypted_text}')
    print(f"Ключ успешно подобран! {decryption['key']}")
```

Рис. 2.9: Функция find_key()

3 Контрольные вопросы

1. Поясните смысл однократного гаммирования

Гаммирование – выполнение операции XOR между элементами гаммы и элементами подлежащего сокрытию текста. Если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

2. Перечислите недостатки однократного гаммирования

Шифр абсолютно стойкий только тогда, когда ключ сгенерирован из случайной двоичной последовательности

3. Перечислите преимущества однократного гаммирования

Это симметричный способ шифрования; алгоритм не дает никакой информации об исходном сообщении; шифрование/дешифрование может быть применено одной программой (в обратном порядке)

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

Если ключ длиннее, то часть текста (разница между длиной ключа и открытого текста) не будет зашифрована. Если же ключ короче, то однозначное дешифрование невозможно

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

операция XOR (сложение по модулю 2), ее особенность - симметричность, т.к. если ее применить 2 раза, то вернется исходное значение

6. Как по открытому тексту и ключу получить шифротекст?

Сначала исходный текст и ключ шифрования преобразуются в 16-ную СС, затем, применяется операция XOR для каждого элемента ключа и текста. Полученный шифротекст декодируется из 16-ной СС и получается набор из символов.

7. Как по открытому тексту и шифротексту получить ключ?

Применить операцию XOR для каждого элемента шифротекста и открытого текста: key[i] = crypted[i] XOR text[i]

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

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа
- равенство длин ключа и открытого текста
- однократное использование ключа

4 Выводы

В ходе выполнения данной лабораторной работы я освоила на практике применение режима однократного гаммирования.

Список литературы

1. Кулябов Д.С., Королькова А.В., Геворкян М.Н. Информационная безопасность компьютерных сетей. Лабораторные работы, учебное пособие. Москва: РУДН, 2015. 64 с.