федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе № 2

«Блочное симметричное шифрование»

по дисциплине «**Информационная безопасность**»

Вариант 12

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: P34312

Преподаватель: Маркина Т. А.

Санкт-Петербург 2023

1. Цель работы

Изучить структуры и основные принципы работы современных алгоритмов блочного симметричного шифрования, приобрести навыки программной реализации блочных симметричных шифров.

2. Описание

Реализовать систему симметричного блочного шифрования, позволяющую шифровать и дешифровать файл на диске с использованием заданного блочного шифра в заданном режиме шифрования.

3. Выполнение

Вариант 6. Алгоритм DES. Режим шифрования CBC.

3.1. Описание программы

Сначала получается 64 битный ключ из 56 битного таким образом, чтобы каждый бит содержал нечетное число единиц. Затем осуществляется перестановка сжатия (key56) и генерируются 16 ключей, необходимых для шифрования реализацией сети Фейстеля.

Над самим блоком данных осуществляется начальная перестановка, и после данные деляется на левую и правую часть. Затем производится 16 циклов шифрования и конечная перестановка, которая объединяет получаемые половины.

Цикл шифрования следующий:

1. Осуществляется функция расширения над правой частью.
2. Над ключом и результатом 1) производится операция xor.

3. Над результатом 2) производится преобразование S, состоящее из 8 преобразований S-блоков.
4. Производится перестановка над результатом 3).
5. Осуществляется операция xor над результатом 4) и левой часть. Результат — левая часть для следующей итерации.
6. Левой частью становится правая часть.

Цикл расшифрования аналогичен циклу шифрования, но ключи применяются в обратном порядке.

3.2. Листинг

Ниже представлен частичный листинг алгоритма, описываемый в Пункте 3.1.

```
def des_encrypt(key: bitarray, data: bitarray, verbose=False) -> bitarray:
    key64 = des_key64(key)
    key56 = des_key_permutation(key64) # get 56-bit key
    keys = des_keys(key56)

    ip = initial_permutation(data)
    lp, rp = split_half(ip, BLOCK_BITS)

    for k in keys:
        ext_r = expansion_permutation(rp)
        xor_r = ext_r ^ k
        sb_r = sbox_transformation(xor_r)
        f_r_k = straight_permutation(sb_r)
        r = lp ^ f_r_k
        l = rp
        lp, rp = l, r

    return final_permutation(rp, lp)


def des_decrypt(key, data: bitarray, verbose=False):
    key64 = des_key64(key)
    key56 = des_key_permutation(key64)
    keys = des_keys(key56)
```

```
    keys.reverse()

    # same as above
    ip = initial_permutation(data)
    lp, rp = split_half(ip, BLOCK_BITS)

    for k in keys:
        ext_r = expansion_permutation(rp)
        xor_r = ext_r ^ k
        sb_r = sbox_transformation(xor_r)
        f_r_k = straight_permutation(sb_r)
        r = lp ^ f_r_k
        l = rp
        lp, rp = l, r

    return final_permutation(rp, lp)
```

3.3. Результат работы

```
> python main.py -c -k 0x123123123213 -v data/in.txt data/out.txt
key:  00123123123213
read from data/in.txt: b'hello world!\n'
written to data/out.txt: b'\xf3 vzha\xcf\xee\x14\x91\x1ba^&\x90!'
> python main.py -z -k 0x123123123213 -v data/out.txt data/in.txt
key:  00123123123213
read from data/out.txt: b'\xf3 vzha\xcf\xee\x14\x91\x1ba^&\x90!'
written to data/in.txt: b'hello world!\n\x00\x00\x00'
```

Как можно заметить, в конце расшифрованной последовательности добавляются нулевые байты. Это связано с тем, что алгоритм добивает входные данные до размера блока нулевыми байтами.

4. Выводы

В ходе выполнения лабораторной работы был реализован алгоритм DES в режиме шифрования CBC.

Алгоритм DES базируется на сети Фейстеля и является симметричным алгоритмом шифрования. В настоящее время именно DES не используется, в качестве замены ему применяется 3DES или AES.

Режим шифрования CBC добавляет элемент зависимости между блоками данных, так что каждый блок зависит от предыдущего зашифрованного блока. Он обеспечивает больший уровень секретности, чем режим ECB, однако не может быть распараллелен, так как каждый следующий блок зависит от предыдущего.