МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Информационных технологий и компьютерные системы»

ОТЧЕТ

о выполнении лабораторной работы №3

по дисциплине: «Методы и алгоритмы защиты информации»

Выполнил:
ст.гр. ПИН/б-21-1-о
Зражевский А.С.
Проверил:
Лебедева М.А.

Севастополь, 2025г.

ЦЕЛЬ РАБОТЫ

Изучить принципы работы сети Фейштеля, научиться шифровать информацию посредством использования блочного криптоалгоритма.

ПОСТАНОВКА ЗАДАЧИ

1. Выбрать из таблицы 3.1 параметры сети Фейштеля в соответствии с вариантом.

2. Разработать программу шифрования и дешифрования текста блоками, В программе предусмотреть ввод криптографического ключа, вычисление образующей функции, зависящей от материала ключа и части блока.

3. Произвести шифрование исходного текста, получить шифрограмму, осуществить ее дешифрование и сравнение с исходным текстом.

4. Результаты работы оформить в виде отчета.

Таблица 1 – Параметры сети Фейштеля

№ варианта	Количество раундов	Образующая функция
10	20	Сложение

ХОД РАБОТЫ

Описание используемого метода:

Метод основан на использовании сети Фейштеля — криптографической структуры, которая применяется в симметричных блочных шифрах. Сеть Фейштеля разбивает входной блок данных на две половины (левую и правую) и многократно применяет к ним преобразования в течение заданного числа раундов (в данном случае 20). В каждом раунде используется образующая функция (f-функция), которая сочетает данные с ключом раунда. Шифрование и дешифрование различаются только порядком применения ключей, что является ключевой особенностью метода. В данной реализации для простоты f-функция использует операцию сложения.

Описание исходных данных:

Текст для шифрования: Произвольная строка, вводимая пользователем, которая преобразуется в последовательность байтов в кодировке UTF-8.

Криптографический ключ: Произвольная строка, также вводимая пользователем, которая используется для генерации ключей для каждого из 20 раундов.

Размер блока: Каждый блок данных составляет 32 бита (4 байта), из которых 16 бит — левая часть, 16 бит — правая часть.

Число раундов: 20.

Алгоритм работы программы:

Генерация ключей для раундов (generate_round_keys):

Входной ключ преобразуется в байтовую строку.

Для каждого из 20 раундов извлекается подмножество байтов длиной 4 из ключа с циклическим сдвигом.

Каждый подключ преобразуется в целое число и ограничивается модулем

2^{16}, чтобы соответствовать размеру половины блока.

Образующая функция (f_function):

Принимает правую часть блока и ключ раунда.

Выполняет простое сложение правой части с ключом.

Результат используется в раунде сети Фейштеля.

Раунд Фейштеля (feistel_round):

Левая часть становится новой правой частью.

Новая левая часть вычисляется как результат сложения исходной левой части с результатом f-функции (по модулю 2^{16}).

Шифрование блока (encrypt_block):

Входной 32-битный блок разбивается на левую (16 бит) и правую (16 бит) части.

Выполняется 20 раундов преобразований с использованием ключей раундов.

После всех раундов части объединяются в 32-битный зашифрованный блок.

Дешифрование блока (decrypt_block):

Аналогично шифрованию, блок разбивается на две части.

Выполняется 20 раундов в обратном порядке (от 19 до 0) с обратной операцией: вычитанием результата f-функции из правой части (по модулю 2^{16}).

Части объединяются в исходный блок.

Преобразование текста в блоки (text_to_blocks):

Входной текст кодируется в байты (UTF-8).

Добавляются пробелы, если длина не кратна 4 байтам.

Разбивается на 32-битные блоки.

Преобразование блоков в текст (blocks_to_text):

Каждый 32-битный блок преобразуется в 4 байта.

Байты объединяются и декодируются обратно в строку с удалением лишних пробелов.

Основной процесс (main):

Запрашивается текст и ключ у пользователя.

Генерируются ключи для 20 раундов.

Текст преобразуется в блоки.

Каждый блок шифруется, затем дешифруется.

Зашифрованный текст выводится в шестнадцатеричном виде, расшифрованный — в текстовом.

Проверяется совпадение исходного и расшифрованного текста.

Текст программы:

```
def generate_round_keys(key, rounds):
    """Генерация ключей для каждого раунда"""
    round_keys = []
    key_length = len(key)
    for i in range(rounds):
```

```
            start = (i * 4) % key_length
            k = key[start:start + 4] if start + 4 <= key_length else key[start:] + key[:(4 - (key_length - start))]
            round_keys.append(int.from_bytes(k.encode(), 'big') % (2**16))
        return round_keys


    def f_function(right, key):
        """Образующая функция: Сложение"""
        N = 16
        return (right + key)


    def feistel_round(left, right, key):
        """Один раунд сети Фейштеля с использованием сложения"""
        N = 16
        new_left = right
        new_right = (left + f_function(right, key)) % (2**N)
        return new_left, new_right


    def encrypt_block(block, round_keys):
        """Шифрование одного блока"""
        left = block >> 16
```

```
    right = block & 0xFFFF

    # Выполняем 20 раундов
    for i in range(20):
        left, right = feistel_round(left, right, round_keys[i])

    return (left << 16) | right

def decrypt_block(block, round_keys):
    """Дешифрование одного блока"""
    left = block >> 16
    right = block & 0xFFFF

    # Выполняем 20 раундов в обратном порядке
    for i in range(19, -1, -1):
        N = 16
        new_right = left
        new_left = (right - f_function(left, round_keys[i])) % (2**N)
        left, right = new_left, new_right

    return (left << 16) | right
```

```
def text_to_blocks(text):
    """Преобразование текста в блоки"""
    byte_data = text.encode('utf-8')
    blocks = []
    while len(byte_data) % 4 != 0:
        byte_data += b' '
    for i in range(0, len(byte_data), 4):
        block = int.from_bytes(byte_data[i:i+4], 'big')
        blocks.append(block)
    return blocks


def blocks_to_text(blocks):
    """Преобразование блоков обратно в текст"""
    byte_data = b''
    for block in blocks:
        byte_data += block.to_bytes(4, 'big')
    return byte_data.decode('utf-8').rstrip()


def main():
    # Ввод данных
```

```
text = input("Введите текст для шифрования: ")

key = input("Введите криптографический ключ: ")


# Параметры сети Фейштеля

rounds = 20  # Изменено с 14 на 20

round_keys = generate_round_keys(key, rounds)


# Преобразуем текст в блоки

blocks = text_to_blocks(text)


# Шифрование

encrypted_blocks = []

for block in blocks:

    encrypted = encrypt_block(block, round_keys)

    encrypted_blocks.append(encrypted)


# Дешифрование

decrypted_blocks = []

for block in encrypted_blocks:

    decrypted = decrypt_block(block, round_keys)

    decrypted_blocks.append(decrypted)
```

```
    # Преобразуем блоки обратно в текст
    encrypted_text = ''.join([hex(block)[2:].zfill(8) for block in encrypted_blocks])
    decrypted_text = blocks_to_text(decrypted_blocks)

    # Вывод результатов
    print("\nРезультаты:")
    print(f"Исходный текст: {text}")
    print(f"Зашифрованный текст (в hex): {encrypted_text}")
    print(f"Расшифрованный текст: {decrypted_text}")

    # Проверка
    if text == decrypted_text.rstrip():
        print("Дешифрование успешно: исходный текст совпадает")
    else:
        print("Ошибка: тексты не совпадают")

if __name__ == "__main__":
    main()
```

Результат работы программы:

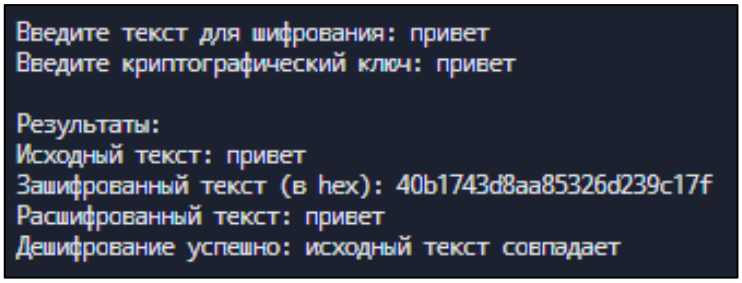

Рисунок 1 – Результат работы программы

ВЫВОД

В ходе лабораторной работы были успешно изучены принципы работы сети Фейштеля, зашифрована информация посредством использования блочного криптоалгоритма.