Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 2

по курсу «Защита Информации»

на тему: «Шифрование симметричным алгоритмом DES»

Вариант № 2

Студент ИУ7-71Б _____ Корниенко К. Ю.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Чиж И. С.
(Подпись, дата) (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель лабораторной работы — разработать программу, осуществляющую шифрование в соответствии с алгоритмом «DES»

Задачи лабораторной работы:

1. провести анализ алгоритма шифрования «DES»;
2. описать алгоритм шифрования;
3. релизовать описанный алгоритм.

1 Аналитический раздел

1.1 Алгоритм шифрования «DES»

DES (Data Encryption Standard) [1] — это симметричный шифровальный алгоритм, разработанный в 1970-х годах, который использует блочное шифрование с фиксированной длиной блока в 64 бита. Основные шаги и логика работы DES:

1. **Начальная перестановка** (Initial Permutation): Исходный текст (64 бита) проходит через начальную перестановку, где биты переставляются в определенном порядке согласно предопределенной таблице перестановок.

2. **Раунды шифрования** (Rounds): DES состоит из 16 раундов шифрования, каждый из которых включает несколько шагов:

- **Расширение** (Expansion): 32-битный входной блок расширяется до 48 бит путем перестановки и дублирования некоторых битов.
- **Ключ раунда** (Round Key): к 48-битному расширенному блоку применяется 48-битный ключ раунда, полученный из основного ключа DES.
- **Скремблирование** (Substitution): 48-битный блок проходит через S-блоки (Substitution-boxes), которые заменяют блоки по 6 бит на блоки по 4 бита с использованием заранее определенных таблиц замен.
- **Перестановка** (Permutation): после замены, полученный блок по 32 бита проходит через таблицу перестановки, которая перемешивает биты в блоке.
- **Обработка ключа** (Key Mixing): к полученному блоку применяется операция XOR с ключом раунда для обеспечения взаимодействия ключа и данных.

3. **Завершающая перестановка** (Final Permutation): После 16 раундов, 64-битный блок проходит через последнюю перестановку, обратную начальной перестановке, чтобы получить зашифрованный текст.

Основным элементом DES является ключ, который состоит из 56 бит, и который используется для генерации ключей раунда. Ключ разбивается на

две половины, и каждая половина сдвигается влево на определенное количество бит в зависимости от номера раунда. Затем, из полученных половинок формируется ключ раунда.

Таким образом, DES использует комбинацию перестановок, замен и операций XOR для шифрования данных. Эти шаги повторяются 16 раз, в каждом раунде используется уникальный ключ. Результат — зашифрованный блок данных, который без знания правильного ключа практически невозможно расшифровать.

Алгоритм шифрования DES может использоваться в следующих режимах.

1. **ECB** (Electronic Code Book) — режим «электронной кодовой книги» (простая замена);
2. **CBC** (Cipher Block Chaining) — режим сцепления блоков;
3. **CFB** (Cipher Feed Back) — режим обратной связи по шифротексту;
4. **OFB** (Output Feed Back) — режим обратной связи по выходу.

2 Конструкторская часть

2.1 Разработка алгоритма

На рисунке 2.1 представлена схема алгоритма шифрования DES.

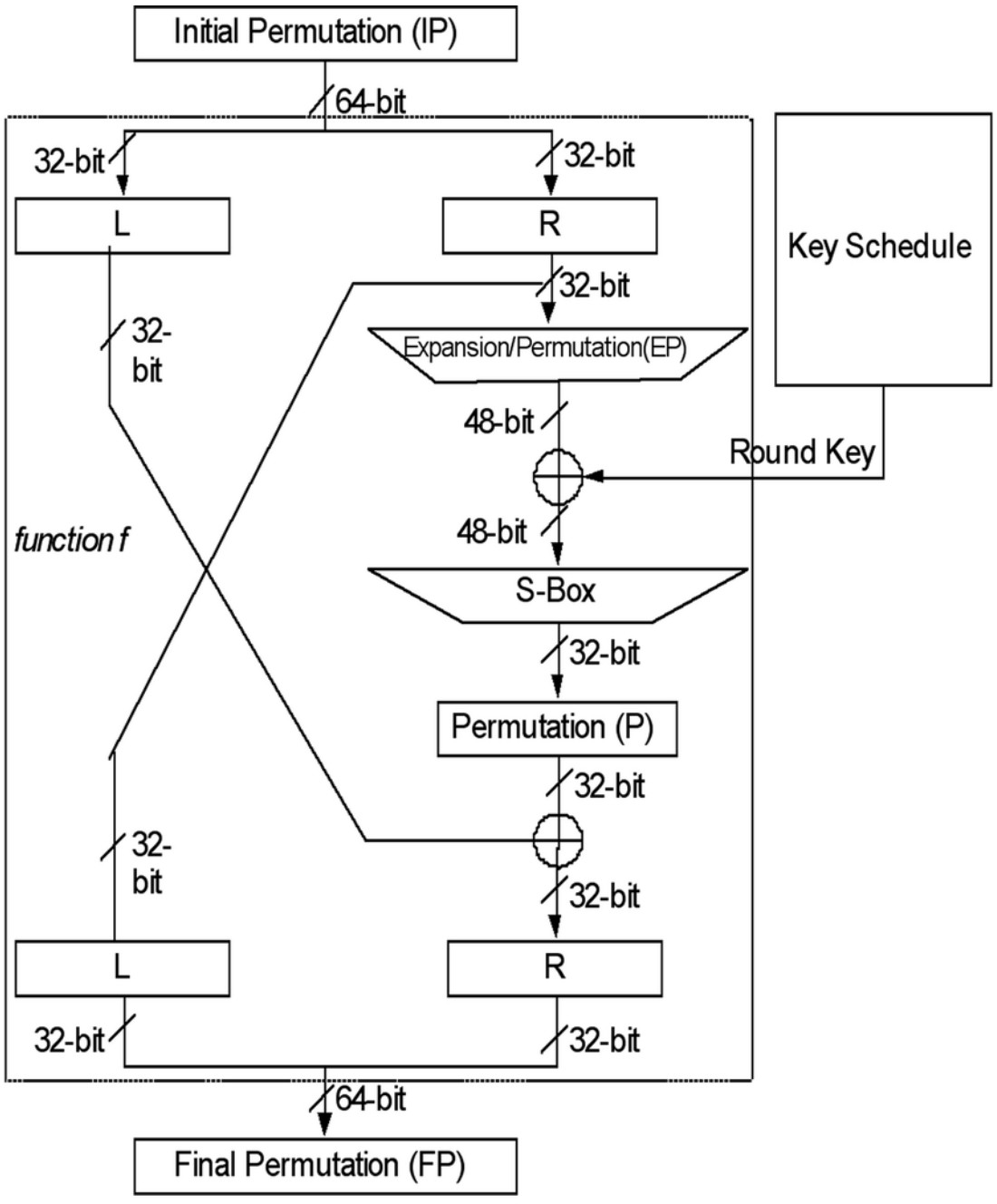

Рисунок 2.1 – Схемы алгоритма DES

3 Технологическая часть

3.1 Средства реализации

Для реализации ПО был выбран язык C++ [2]. В данном языке есть все требующиеся инструменты для данной лабораторной работы. В качестве среды разработки была выбрана среда VS code [3].

3.2 Реализация алгоритма

Листинг 3.1 – Методы шифрования и дешифровки произвольного сообщения

```
void DesCryptor::encrypt(std::istream& input, std::ostream&
   output) const
{
    auto round_keys = generate_round_keys(key);
    bool run = true;
    bool first_pass = true;
    uint64_t prev_cyphered_block;
    do
    {
        uint8_t buffer[8];
        input.read(reinterpret_cast<char*>(buffer), 8);
        auto nread = input.gcount();
        if (nread < 8) {
            extend_block(buffer, nread); // extend to 64bit block
            run = false;
        }
        uint64_t block = buffer_to_block(buffer);
        if (first_pass)
            first_pass = false;
        else
            block = block xor prev_cyphered_block;
        block = encrypt_block(block, round_keys);
        prev_cyphered_block = block;
        block_to_buffer(block, buffer);
        output.write(reinterpret_cast<char*>(buffer), 8);
    } while (run);
}

void DesCryptor::decrypt(std::istream& input, std::ostream&
   output) const
```

```
{
    auto round_keys = generate_round_keys(key);
    uint64_t prev_block;
    uint8_t prev_buffer[8];
    uint8_t buffer[8];
    bool run = true;
    bool first_pass = true;
    do
    {
        input.read(reinterpret_cast<char*>(buffer), 8);
        auto nread = input.gcount();
        if (nread == 0)
            run = false;
        else if (nread < 8)
            throw std::runtime_error("invalid cyphered input
               size");

        uint64_t block = buffer_to_block(buffer);
        auto decyphered_block = decrypt_block(block, round_keys);
        if (!first_pass)
            decyphered_block = decyphered_block xor prev_block;
        prev_block = block;
        block_to_buffer(decyphered_block, buffer);

        // handle last block as extended
        size_t nwrite = run ? 8 : prev_buffer[7];
        if (first_pass)
            first_pass = false;
        else if (nwrite > 0)
            output.write(reinterpret_cast<char*>(prev_buffer),
               nwrite);
        memcpy(prev_buffer, buffer, 8);
    } while (run);
}
```

3.3 Тестирование ПО

В таблице 3.1 представлены тестовые данные, для проверки корректности работы программы. Применена методология черного ящика. Тесты пройдены *успешно*.

Таблица 3.1 – Функциональные тесты для текстовых файлов

Файл	16-ричный дамп файла	
Ключ	TODO	
Входной файл 1	00000000: 7365 6372 6574 206d 00000008: 6573 7361 6765	secret m essage
Зашифрованный файл 1	00000000: 327b 33f9 248f 9d39 00000008: 655d 73df 265b	2{3.$..9 e]s.&.
Дешифрованный файл 1	00000000: 7365 6372 6574 206d 00000008: 6573 7361 6765	secret m essage
Ключ	TODO	
Входной файл 2	00000000: 3131 3131 3131 3131 00000008: 3131 3131 3131 3131 00000010: 3131 3131 3131 3131 00000018: 3131 3131 3131 310a	11111111 11111111 11111111 1111111.
Зашифрованный файл 2	00000000: d072 b537 61bd e940 00000008: 0e2b 9179 3144 1265 00000010: 31bc 879b cfb7 2268 00000018: 98e6 3535 67f1 2d0a	.r.7a..@ .+.y1D.e 1.....\"h ..55g.-.
Дешифрованный файл 2	00000000: 3131 3131 3131 3131 00000008: 3131 3131 3131 3131 00000010: 3131 3131 3131 3131 00000018: 3131 3131 3131 310a	11111111 11111111 11111111 1111111.

Таблица 3.2 – Функциональные тесты для бинарных файлов

Файл	16-ричный дамп начала файла
Ключ	TODO
Входной файл 3	00000000: 8950 4e47 0d0a 1a0a .PNG.... 00000008: 0000 000d 4948 4452 ....IHDR 00000010: 0000 02c9 0000 02c7 .... 00000018: 0806 0000 007c 3fdf .....|?. ...
Зашифрованный файл 3	00000000: 113d 4afc 85b4 3982 .=J...9. 00000008: a449 000d d9d8 fb52 .I.....R 00000010: 9b78 a44c a200 027f .x.L.... 00000018: 27dd 70aa 0026 4cdf '.p..&L. ...
Дешифрованный файл 3	00000000: 8950 4e47 0d0a 1a0a .PNG.... 00000008: 0000 000d 4948 4452 ....IHDR 00000010: 0000 02c9 0000 02c7 .... 00000018: 0806 0000 007c 3fdf .....|?. ...
Ключ	TODO
Входной файл 4	00000000: 7f45 4c46 0201 0100 .ELF.... 00000008: 0000 0000 0000 0000 .... 00000010: 0300 3e00 0100 0000 ..>..... 00000018: 4011 0000 0000 0000 @....... ...
Зашифрованный файл 4	00000000: 48cb 0cad d4d3 0100 H....... 00000008: a449 00f0 0000 40aa .I....@. 00000010: 0378 3a00 0100 2300 .x:...#. 00000018: 40c4 70aa 0000 d2d7 @.p..... ...
Дешифрованный файл 4	00000000: 7f45 4c46 0201 0100 .ELF.... 00000008: 0000 0000 0000 0000 .... 00000010: 0300 3e00 0100 0000 ..>..... 00000018: 4011 0000 0000 0000 @....... ...

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе:

— проведен анализ работы алгоритма «DES»;

— описан алгоритм шифрования;

— реализован описанный алгоритм.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Prasad K.*, *Kumari M.* A review on mathematical strength and analysis of Enigma // arXiv preprint arXiv:2004.09982. — 2020.
2. *Josuttis N. M.* The C++ standard library: a tutorial and reference. — 2012.
3. *Code V. S.* Visual studio code // línea]. Available: https://code. visualstudio. com. — 2019.