



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 и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

## ОТЧЕТ

по лабораторной работа №2  
по курсу «Защита информации»  
на тему: «Алгоритм DES»  
Вариант № 2

Студент ИУ7-73Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

Лысцев Н. Д.  
(И. О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

Чиж И. С.  
(И. О. Фамилия)

2024 г.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>3</b>
<b>1 Аналитический раздел</b>	<b>4</b>
1.1 История создания алгоритма DES . . . . .	4
1.2 Общие положения алгоритма DES . . . . .	4
1.3 Алгоритм DES . . . . .	4
1.3.1 Раунды шифрования . . . . .	6
1.3.2 Функция Фейстеля . . . . .	7
1.3.3 Генерация ключей $k_i$ . . . . .	9
1.4 Расшифрование . . . . .	10
1.5 Режим шифрования PCBC . . . . .	11
<b>2 Конструкторский раздел</b>	<b>13</b>
<b>3 Технологический раздел</b>	<b>16</b>
3.1 Требования к программному обеспечению . . . . .	16
3.2 Средства реализации . . . . .	16
3.3 Сведения о модулях программы . . . . .	16
3.4 Тестирование . . . . .	16
3.5 Реализации алгоритмов . . . . .	17
<b>ЗАКЛЮЧЕНИЕ</b>	<b>21</b>

## ВВЕДЕНИЕ

Целью данной лабораторной работы является реализация программы шифрования симметричным алгоритмом DES с применением режима PCBC.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) описать алгоритм DES с режимом PCBC;
- 2) выбрать средства программной реализации;
- 3) реализовать данный алгоритм.

# 1 Аналитический раздел

В данном разделе будет кратко описана история создания алгоритма DES, будет приведено его формальное описание, а также будет формально описан режим шифрования PCBC.

## 1.1 История создания алгоритма DES

Стандарт шифрования данных (DES) - блочный шифр с симметричными ключами, разработан Национальным Институтом Стандартов и Технологии (NIST - National Institute of Standards and Technology).

В 1973 году NIST издал запрос для разработки предложения национальной криптографической системы с симметричными ключами. Предложенная IBM модификация проекта, названная Lucifer, была принята как DES. DES был издан как FIPS 46 в Федеральном Регистре в январе 1977 года. FIPS объявил DES как стандарт для использования в неофициальных приложениях. Позже NIST предложил новый стандарт (FIPS 46-3), который рекомендует использование тройного DES (трехкратно повторенный шифр DES) для будущих приложений.

## 1.2 Общие положения алгоритма DES

Общие положения алгоритма DES:

- 1) на стороне шифрования DES принимает 64-битовый исходный текст и порождает 64-битовый зашифрованный текст;
- 2) на стороне дешифрования DES принимает 64-битовый зашифрованный текст и порождает 64-битовый исходный текст;
- 3) на обеих сторонах для шифрования и дешифрования применяется один и тот же 56-битовый ключ.

## 1.3 Алгоритм DES

Общее описание алгоритма:

- исходный текст - блок 64 бит;

- процесс шифрования состоит из начальной перестановки, 16 циклов шифрования (раундов Фейстеля) и конечной перестановки;
- каждый раунд использует различные сгенерированные 48-битовые ключи.

На рисунке 1.1 представлена схема шифрования алгоритма DES:

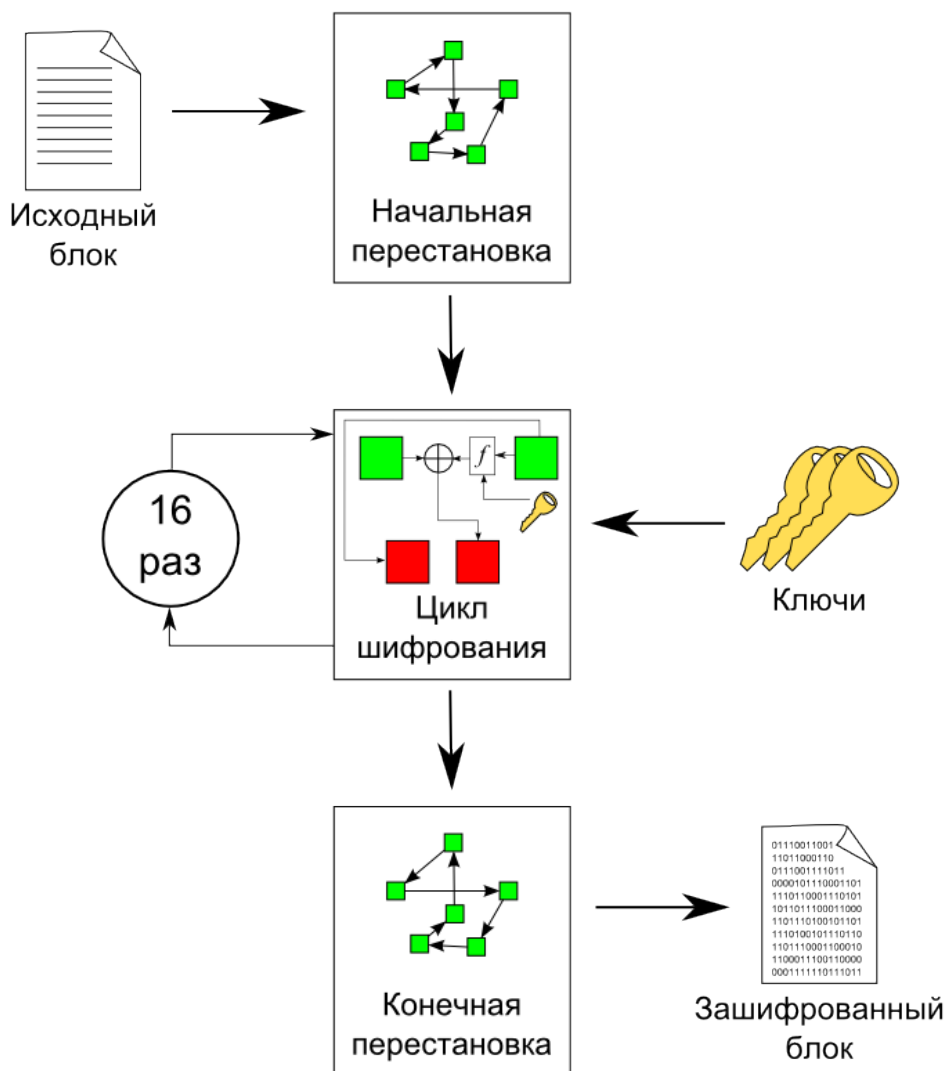


Рисунок 1.1 – Схема шифрования алгоритма DES

Начальная и конечная перестановки в основном служат для облегчения побайтовой загрузки данных открытого текста и шифротекста в микросхему DES (DES появился раньше 16- и 32-битовых микропроцессорных шин).

Каждая из перестановок принимает 64-битовый вход и переставляет его элементы по заданному правилу. Эти перестановки - прямые перестановки без

ключей, которые инверсны друг другу. Значение каждого элемента определяет номер входного порта, а порядковый номер (индекс) элемента определяет номер выходного порта.

На рисунках 1.2 и 1.3 представлены таблицы начальной и конечной перестановок:

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Рисунок 1.2 – Начальная перестановка  $IP$

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Рисунок 1.3 – Конечная перестановка  $IP^{-1}$

### 1.3.1 Раунды шифрования

Полученный после начальной перестановки 64-битовый блок  $IP(T)$  участвует в 16 раундах преобразования Фейстеля.

Раунд шифрования:

- 1) разбить  $IP(T)$  на две части  $L_0$  и  $R_0$ , где  $L_0$  и  $R_0$  соответственно 32 старших битов и 32 младших битов блока  $T_0$   $IP(T) = L_0R_0$ ;
- 2) пусть  $T_{i-1} = L_{i-1}R_{i-1}$  – результат  $i - 1$  итерации, тогда результат  $i$ -ой итерации  $T_i = L_iR_i$  определяется по формуле 1.4;

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, k_i) \end{aligned} \tag{1.1}$$

- 3) в 16-ти раундах преобразования Фейстеля функция  $f$  играет роль шифрования.

На рисунке 1.4 представлена схема работы одного раунда шифрования:

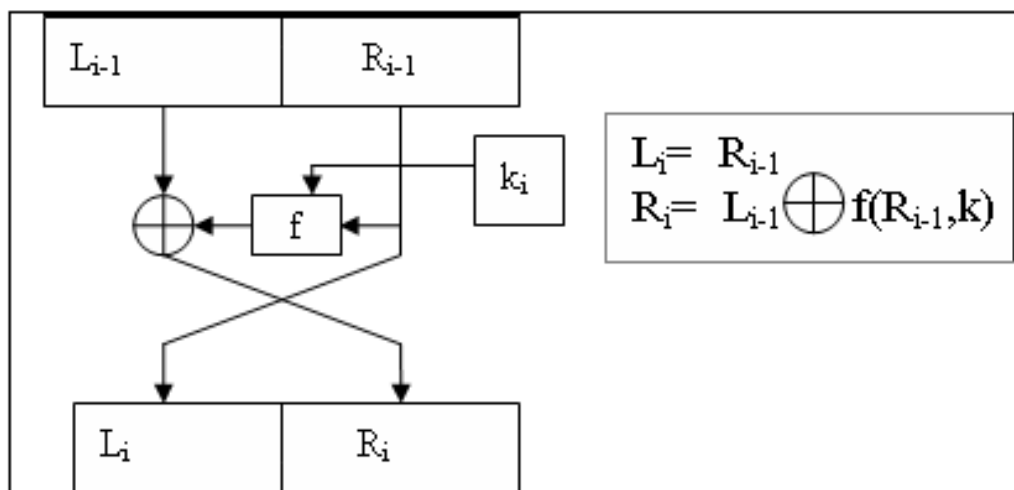


Рисунок 1.4 – Прямое преобразование сетью Фейстеля

### 1.3.2 Функция Фейстеля

Аргументами функции  $f$  являются 32-битный вектор  $R_{i-1}$  и 48-битовый ключ  $k_i$ , который является результатом преобразования 56-битового исходного ключа шифра  $k$ . Для вычисления значения функции  $f$  последовательно используются:

- 1) функция расширения  $E$ ;
- 2) XOR с ключом  $k_i$ ;
- 3) преобразование  $S$ , состоящее из 8-ми преобразований  $S$ -блоков  $S_1, S_2, S_3, \dots, S_8$ ;
- 4) перестановка  $P$ .

Функция  $E$  расширяет 32-битовый вектор  $R_{i-1}$  до 48-битового вектора  $E(R_{i-1})$  путём выполнения следующих действий:

- 1) 32-битовый вектор  $R_{i-1}$  делится на 8 блоков по 4 бита каждый;
- 2) в каждый блок слева добавляется крайний правый бит предыдущего блока, а справа – крайний левый бит следующего блока;

- 3) в качестве бита слева для первого блока выступает крайний правый бит последнего блока, а для бита справа для последнего блока – крайний левый бит первого блока.

Порядок битов указан на рисунке 1.5.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Рисунок 1.5 – Функция расширения  $E$

Полученный после перестановки блок  $E(R_{i-1})$  складывается по модулю 2 с ключом  $k_i$  и представляется в виде 8-ми последовательных блоков  $B_1, B_2, B_3, \dots B_8$  каждый по 6 бит (формула 1.2).

$$E(R_{i-1}) \oplus k_i = B_1, B_2, B_3, \dots B_8 \quad (1.2)$$

Каждый  $B_j$  является 6-битовым блоком. Далее, каждый из блоков  $B_j$  трансформируется в 4-битовый блок  $B'_j$  с помощью преобразования  $S_i$  [info\_DES].

Значение функции  $f(R_{i-1}, k_i)$  получается перестановкой  $P$ , применяемой к 32-битному блоку  $B'_1, B'_2, B'_3, \dots B'_8$  (формула 1.3).

$$f(R_{i-1}, k_i) = P(B'_1, B'_2, B'_3, \dots B'_8) \quad (1.3)$$

Таблица перестановки  $P$  указана на рисунке 1.6.



16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Рисунок 1.6 – Перестановка  $P$

### 1.3.3 Генерация ключей $k_i$

Ключ  $k$  представляет собой 64-битовый блок с восемью битами контроля по четности, расположенными в позициях 8,16,24,32,40,48,56,64. Для удаления контрольных битов и перестановки остальных используется функция  $G$  первоначальной подготовки ключа (рисунок 1.7).

57	49	41	33	25	17	9	1	58	50	42	34	26	18	$C_0$
10	2	59	51	43	35	27	19	11	3	60	52	44	36	
63	55	47	39	31	23	15	7	62	54	46	38	30	22	$D_0$
14	6	61	53	45	37	29	21	13	5	28	20	12	4	

Рисунок 1.7 – Перестановка первоначальной подготовки ключа

Результат преобразования  $G(k)$  разбивается на два 28-битовых блока  $C_0$  и  $D_0$ , причем  $C_0$  будет состоять из битов 57, 49, ..., 44, 36 ключа  $k$ , а  $D_0$  будет состоять из битов 63, 55, ..., 12, 4 ключа  $k$ .

После определения  $C_0$  и  $D_0$  рекурсивно определяются  $C_i$  и  $D_i$ ,  $i = 1 \dots 16$ . Для этого применяют циклический сдвиг влево на один или два бита в зависимости от номера итерации (рисунок 1.8).

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число сдвига	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Рисунок 1.8 – Сдвиги для вычисления ключа

Ключ  $k_i$ ,  $i = 1 \dots 16$  состоит из 48 бит, выбранных из вектора  $C_i D_i$  согласно рисунку 1.9.

14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

Рисунок 1.9 – Завершающая обработка ключа

Первый и второй биты ключа  $k_i$  есть биты 14, 17 вектора  $C_i D_i$ .

## 1.4 Расшифрование

При расшифровании данных все действия выполняются в обратном порядке. В 16 циклах расшифрования, в отличие от шифрования с помощью прямого преобразования сетью Фейстеля, здесь используется обратное преобразование сетью Фейстеля:

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, k_i) \end{aligned} \quad (1.4)$$

На рисунке 1.10 представлена схема работы одного раунда расшифрования:

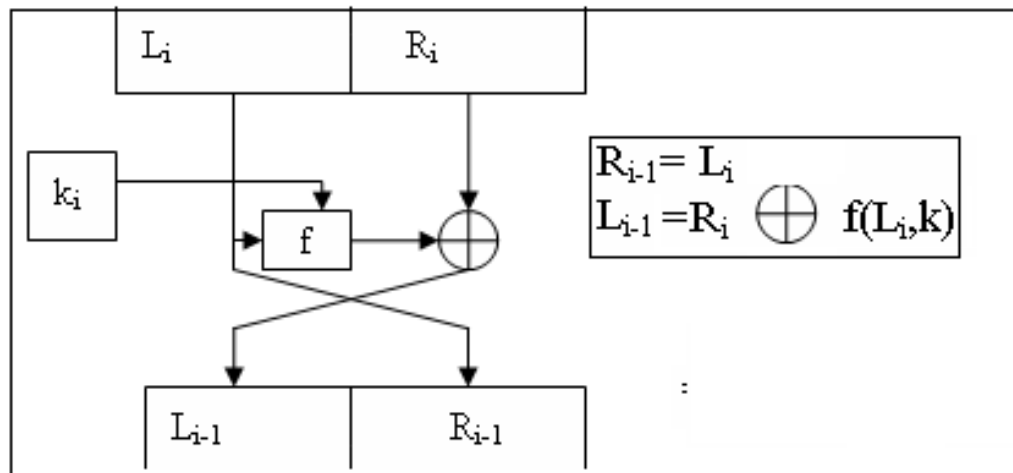


Рисунок 1.10 – Обратное преобразование сетью Фейстеля

Ключ  $k_i$ ,  $i = 16 \dots 1$  такие же, как и в процессе шифрования. Функция  $f$ , перестановка  $IP$  и  $IP^{-1}$  такие же, как и в процессе шифрования. Алгоритм генерации ключей зависит только от ключа пользователя, поэтому при расшифровании они идентичны.

## 1.5 Режим шифрования PCBC

Для шифрования некоторого сообщения  $P$  выполняются следующие действия:

- 1) сообщение разбивается на блоки одинакового размера. Размер (длина) блока равен  $n$  и измеряется в битах. При необходимости последний блок дополняется до длины  $n$ ;
- 2) шифрование очередного ( $i$ -го) блока сообщения  $P_i$  выполняется с использованием предыдущего блока открытого текста и предыдущего блок шифротекста после применения операции XOR над ними. Для первого блока ( $P_1$ ) зашифрованного блока не существует, поэтому первый блок шифруют с использованием «вектора инициализации»  $IV$  (формула 1.5);

$$\begin{aligned} C_0 &= IV \\ C_i &= E_k(P_{i-1} \oplus C_{i-1} \oplus P_i, k) \end{aligned} \quad (1.5)$$

- 3) расшифрование очередного ( $i$ -го) блока сообщения происходит по формуле 1.6.

$$\begin{aligned} C_0 &= IV \\ P_i &= D_k(C_i, k) \oplus C_{i-1} \oplus P_{i-1} \end{aligned} \quad (1.6)$$

На рисунке 2.1 представлена схема режима шифрования PCBC.

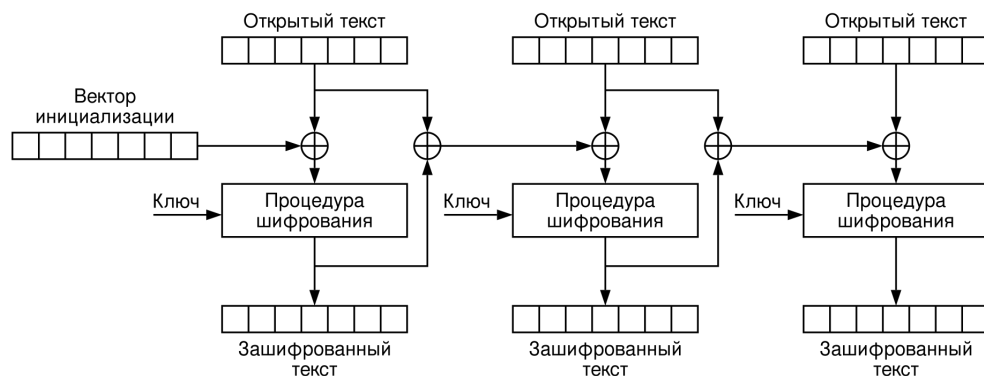


Рисунок 1.11 – Схема режима шифрования PCBC

## Вывод

В данном разделе была кратко описана история создания алгоритма DES, было приведено его формальное описание, а также был формально описан режим шифрования PCBC.

## 2 Конструкторский раздел

На рисунках 2.1–2.4 представлены схемы алгоритмов DES, раунда DES, функции Фейстеля, а также схема работы шифрования в режиме PCBC.

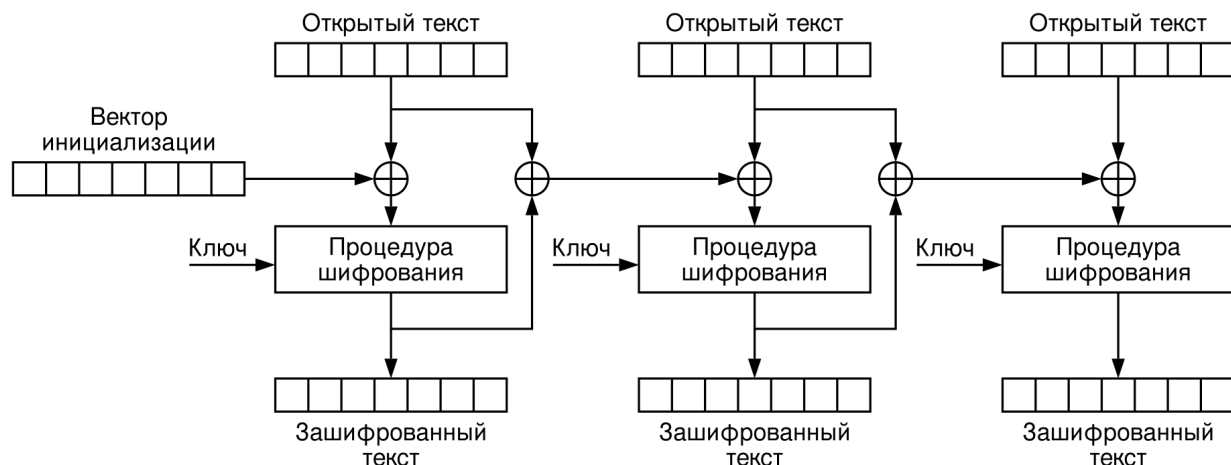


Рисунок 2.1 – Схема алгоритма работы шифрования в режиме PCBC



Рисунок 2.2 – Схема алгоритма DES

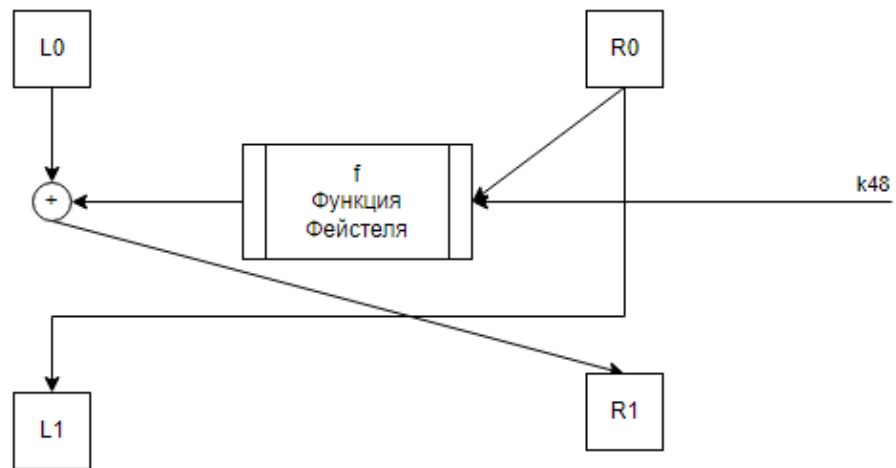


Рисунок 2.3 – Схема алгоритма раунда шифрования в алгоритме DES

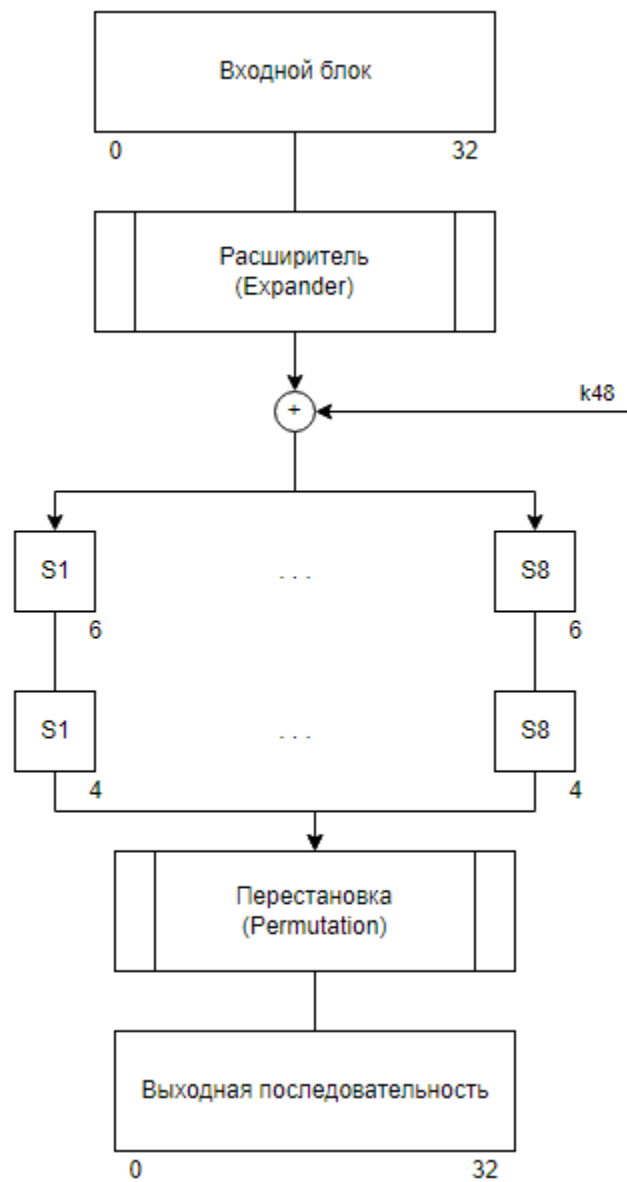


Рисунок 2.4 – Схема алгоритма функции Фейстеля

## Вывод

В данном разделе были представлены схемы алгоритмов DES, раунда DES, функции Фейстеля, а также схема работы шифрования в режиме PCBC.

## 3 Технологический раздел

В данном разделе будут перечислены требования к программному обеспечению, средства реализации и листинги кода.

### 3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

- программа должна предоставлять возможность шифрования и расшифровки произвольного файла, а также текстового сообщения;
- программа должна корректно работать с пустым однобайтовым файлом;
- программа должна уметь обрабатывать файл архива.

### 3.2 Средства реализации

В качестве языка программирования для этой лабораторной работы был выбран `C++`.

### 3.3 Сведения о модулях программы

Программа состоит из трех модулей:

- `main.cpp` — файл, содержащий точку входа в программу;
- `DES.cpp` — модуль, реализующий алгоритм DES;
- `PCBC.cpp` — модуль, реализующий шифрование в режиме PCBC.

### 3.4 Тестирование

Таблица 3.1 – Функциональные тесты

Входная строка	Шифрованная строка
Hello	gùÄv
gùÄv	Hello
B	hí'(!
hí'(!	B



Все тесты были успешно пройдены. Тесты с файлами были также успешно пройдены.

## 3.5 Реализации алгоритмов

На листинге 3.1 представлена функция, реализующая алгоритм DES.

На листингах 3.2 и 3.3 представлена функция, реализующая генерацию ключей для каждого раунда шифрования.

На листинге 3.4 представлена функция, реализующая раунд шифрования.

На листингах 3.5 и 3.6 представлена реализация функции Фейстеля.

На листинге 3.7 представлена реализация шифрования в режиме PCBC.

Листинг 3.1 – Функция, реализующая алгоритм DES

```
bitset<64> DES::encryptBlock(bitset<64> block, bitset<64> key) {
    bitset<64> IPBlock = this->_applyIP(block);

    vector<bitset<48>> roundKeys = this->_genRoundKeys(key);

    auto [Li, Ri] = DES::_iPBlockToLOR0(IPBlock);

    for (auto roundKey: roundKeys) {
        pair<bitset<32>, bitset<32>> value =
            this->_roundEncrypt(Li, Ri, roundKey);
        Li = value.first;
        Ri = value.second;
    }
    bitset<64> encryptBlock = this->_uniteRiAndLi(Ri, Li);

    return this->_applyIpInv(encryptBlock);
}
```

Листинг 3.2 – Функция, реализующая генерацию ключей для каждого раунда шифрования (начало)

```
vector<bitset<48>> DES::_genRoundKeys(bitset<64> key) {
    vector<bitset<48>> keys;

    bitset<56> P1Key = this->_applyP1Ki(key);

    for (auto cntBitsForShift: this->_kiLeftShiftCnt) {
        bitset<56> tmp = P1Key;
```

Листинг 3.3 – Функция, реализующая генерацию ключей для каждого раунда шифрования (конец)

```
    auto [C0, D0] = DES::_p1KeyToC0D0(tmp);

    C0 = DES::_applyCycleLeftShift(C0, cntBitsForShift);
    D0 = DES::_applyCycleLeftShift(D0, cntBitsForShift);

    for (int i = 0; i < 28; i++) {
        P1Key[i] = C0[i];
        P1Key[i + 27] = D0[i];
    }

    bitset<48> P2Key = this->_applyP2Ki(P1Key);

    keys.push_back(P2Key);
}

return keys;
}
```

Листинг 3.4 – Функция, реализующая раунд шифрования

```
pair<bitset<32>, bitset<32>> DES::_roundEncrypt(bitset<32>
    Li_minus_1, bitset<32> Ri_minus_1, bitset<48> ki) {
    bitset<32> Li = Ri_minus_1;
    bitset<32> Ri = Li_minus_1 ^ this->_f(Ri_minus_1, ki);

    return pair<bitset<32>, bitset<32>>{Li, Ri};
}
```

Листинг 3.5 – Реализация функции Фейстеля (начало)

```
bitset<32> DES::_f(bitset<32> Ri_minus_1, bitset<48> ki) {
    bitset<48> extendRi_minus_1 = this->_applyE(Ri_minus_1);
    bitset<48> xorRi_minus_1 = extendRi_minus_1 ^ ki;

    bitset<6> tmp = {};
    bitset<32> result = {};

    int k = 0;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 6; j++) {
            tmp[j] = xorRi_minus_1[i * 6 + j];
        }
    }
}
```

Листинг 3.6 – Реализация функции Фейстеля (конец)

```

    }

    bitset<4> s_res = this->_applyS(tmp, i);

    for (int p = 0; p < 4; p++) {
        result[p + k] = s_res[p];
    }

    k += 4;
}

return this->_applyP(result);
}

```

Листинг 3.7 – Реализация шифрования в режиме PCBC

```

vector<char> PCBC::_encrypt(vector<bitset<64>> &strBlocks,
    bitset<64> &bitsetKey) {
    bitset<64> Ci_minus_1 = this->_iv;
    vector<char> encryptedStr;

    for (auto strBlock: strBlocks) {
        bitset<64> encryptBlock =
            this->_des.encryptBlock(Ci_minus_1 ^ strBlock,
                bitsetKey);

        vector<char> tmpRes =
            PCBC::_bitset64ToVChar(encryptBlock);

        for (auto symbol: tmpRes) {
            encryptedStr.push_back(symbol);
        }

        Ci_minus_1 = encryptBlock ^ strBlock;
    }

    return encryptedStr;
}

```

## Вывод

В данном разделе были перечислены требования к программному обеспечению, средства реализации и листинги кода.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были решены следующие задачи:

- 1) описан алгоритм DES с режимом PCBC;
- 2) выбраны средства программной реализации;
- 3) реализован данный алгоритм.

Цель работы, а именно реализация программы шифрования симметричным алгоритмом DES с применением режима PCBC, также была достигнута.