

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ИМ. Н. Э. БАУМАНА

УДК № 004.822

№ госрегистрации \_\_\_\_\_

Инв. № \_\_\_\_\_

УТВЕРЖДАЮ

Преподаватель

\_\_\_\_\_  
« \_\_\_\_\_ » \_\_\_\_\_ 2019 г.

ОТЧЁТ  
ПО РУБЕЖНОМУ КОНТРОЛЮ №1

Алгоритм шифрования des  
(промежуточный)

Студент

\_\_\_\_\_ А. А. Куприй

Преподаватели

Л.Л. Волкова, Ю.В. Строганов

Москва 2019

## СПИСОК ИЛЛЮСТРАЦИЙ

1.1	Прямое преобразование сетью Фейстеля .....	7
1.2	Обратное преобразование сетью Фейстеля .....	7
1.3	Схема шифрования алгоритма DES .....	8
1.4	Таблица 3. Преобразования $S_i, i = 1 \dots 8$ .....	11
2.1	Функциональная модель шифрования .....	15
2.2	Алгоритм DES .....	16
4.1	Работа программы .....	20

## **РЕФЕРАТ**

Отчет содержит 22 стр., 7 рис., 7 табл..

## СОДЕРЖАНИЕ

Реферат .....	3
Введение .....	5
1 Аналитический раздел .....	6
1.1 Блочный шифр .....	6
1.2 Преобразования Сетью Фейстеля .....	6
1.3 Схема шифрования алгоритма DES .....	7
1.3.1 Начальная перестановка .....	8
1.3.2 Циклы шифрования .....	9
1.3.3 Основная функция шифрования (функция Фейстеля) .....	9
1.3.4 Генерирование ключей $k_i$ .....	12
1.3.5 Конечная перестановка .....	13
2 Конструкторский раздел .....	15
2.1 IDEF0 Модель .....	15
2.2 Разработка алгоритмов .....	16
2.3 Вывод .....	17
3 Технологический раздел .....	18
3.1 Требования к программному обеспечению .....	18
3.2 Средства реализации .....	18
3.3 Листинги кода .....	18
3.4 Вывод .....	19
4 Исследовательский раздел .....	20
4.1 Примеры работы .....	20
4.2 Вывод .....	20
Заключение .....	21
Список использованных источников .....	22

## ВВЕДЕНИЕ

**DES** (англ. Data Encryption Standard) — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт (FIPS 46-3). Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

Для выполнения данной лабораторной работы необходимо решить следующие задачи:

- изучить алгоритм шифрования **DES**;
- реализовать указанный алгоритм с параллельными вычислениями;

## **1 Аналитический раздел**

DES является блочным шифром. Чтобы понять, как работает DES, необходимо рассмотреть принцип работы блочного шифра, сеть Фейстеля.

### **1.1 Блочный шифр**

Входными данными для блочного шифра служат:

- блок размером  $n$  бит;
- ключ размером  $k$  бит.

На выходе (после применения шифрующих преобразований) получается зашифрованный блок размером  $n$  бит, причём незначительные различия входных данных, как правило, приводят к существенному изменению результата.

Блочные шифры реализуются путём многократного применения к блокам исходного текста некоторых базовых преобразований.

Базовые преобразования:

- сложное преобразование на одной локальной части блока;
- простое преобразование между частями блока.

Так как преобразования производятся поблочно, требуется разделение исходных данных на блоки необходимого размера. При этом формат исходных данных не имеет значения (будь то текстовые документы, изображения или другие файлы). Данные должны интерпретироваться в двоичном виде (как последовательность нулей и единиц) и только после этого должны разбиваться на блоки. Все вышеперечисленное может осуществляться как программными, так и аппаратными средствами.

### **1.2 Преобразования Сетью Фейстеля**

Это преобразование над векторами (блоками), представляющими собой левую и правую половины регистра сдвига. В алгоритме DES используются

прямое преобразование сетью Фейстеля в шифровании (см. Рисунок 1.1) и обратное преобразование сетью Фейстеля в расшифровании (см. Рисунок 1.2).

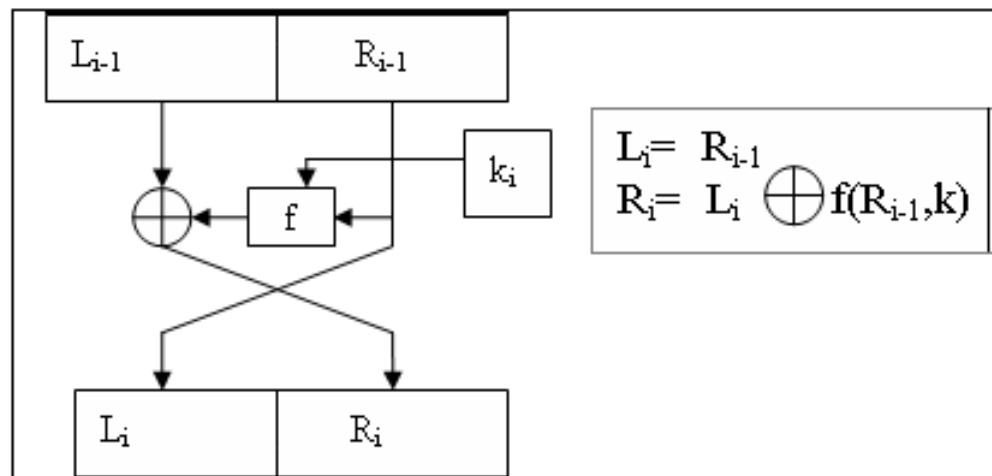


Рисунок 1.1 — Прямое преобразование сетью Фейстеля

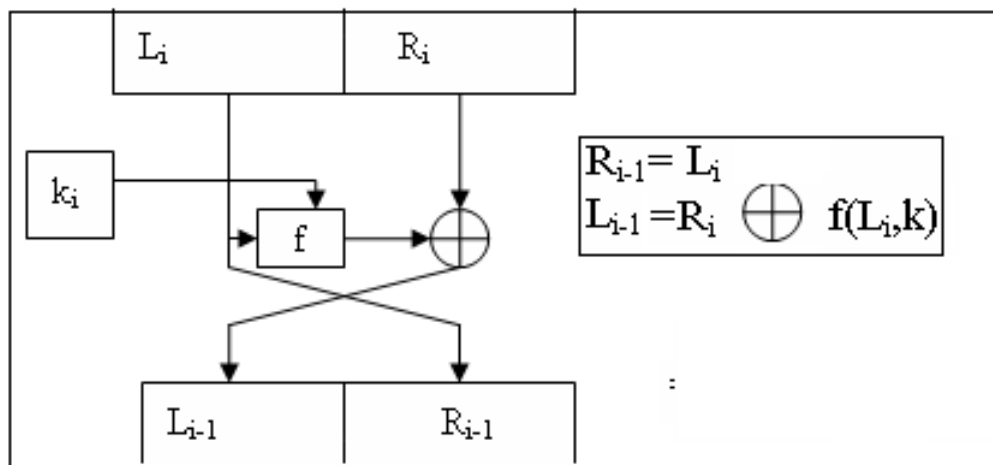


Рисунок 1.2 — Обратное преобразование сетью Фейстеля

### 1.3 Схема шифрования алгоритма DES

Схема шифрования алгоритма DES указана на Рисунке 1.3.

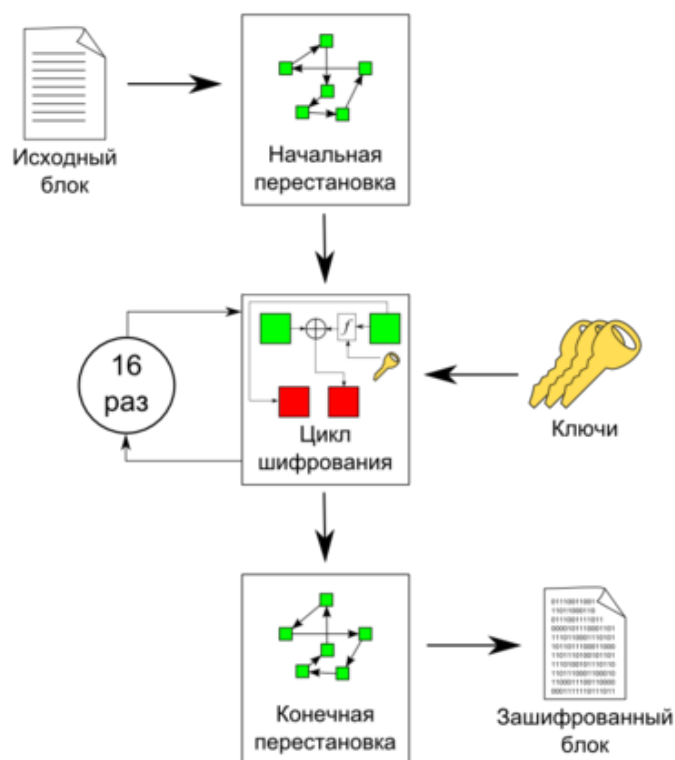


Рисунок 1.3 — Схема шифрования алгоритма DES

Исходный текст — блок 64 бит.

Процесс шифрования состоит из начальной перестановки, 16 циклов шифрования и конечной перестановки.

### 1.3.1 Начальная перестановка

Исходный текст **T** (блок 64 бит) преобразуется с помощью начальной перестановки **IP** которая определяется таблицей 1:

Таблица 1.1 — Начальная перестановка IP

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



По таблице первые 3 бита результирующего блока  $\mathbf{IP}(\mathbf{T})$  после начальной перестановки  $\mathbf{IP}$  являются битами 58, 50, 42 входного блока  $\mathbf{T}$ , а его 3 последние бита являются битами 23, 15, 7 входного блока.

### 1.3.2 Циклы шифрования

Полученный после начальной перестановки 64-битовый блок  $\mathbf{IP}(\mathbf{T})$  участвует в 16 циклах преобразования Фейстеля.

— 16 циклов преобразования Фейстеля:

Разбить  $\mathbf{IP}(\mathbf{T})$  на две части  $L_0, R_0$ , где  $L_0, R_0$  — соответственно 32 старших битов и 32 младших битов блока  $T_0$   $\mathbf{IP}(\mathbf{T}) = L_0 R_0$

Пусть  $T_{i-1} = L_{i-1} R_{i-1}$  результат  $(i-1)$  итерации, тогда результат  $i$ -ой итерации  $T_i = L_i R_i$  определяется:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

Левая половина  $L_i$  равна правой половине предыдущего вектора  $L_{i-1} R_{i-1}$ . А правая половина  $R_i$  — это битовое сложение  $L_{i-1}$  и  $f(R_{i-1}, k_i)$  по модулю 2.

В 16-циклах преобразования Фейстеля функция  $f$  играет роль шифрования. Рассмотрим подробно функцию  $f$ .

### 1.3.3 Основная функция шифрования (функция Фейстеля)

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

- а) функция расширения  $E$ ,
- б) сложение по модулю 2 с ключом  $k_i$
- в) преобразование  $S$ , состоящее из 8 преобразований  $S$ -блоков  $S_1, S_2, S_3 \dots S_8$ ,

г) перестановка  $P$ .

Функция  $E$  расширяет 32-битовый вектор  $R_{i-1}$  до 48-битового вектора  $E(R_{i-1})$  путём дублирования некоторых битов из  $R_{i-1}$ ; порядок битов вектора  $E(R_{i-1})$  указан в таблице 2.

Таблица 1.2 — Функция расширения  $E$

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

Первые три бита вектора  $E(R_{i-1})$  являются битами 32, 1, 2 вектора  $R_{i-1}$ . По таблице 2 видно, что биты 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 32 дублируются. Последние 3 бита вектора  $E(R_{i-1})$  — это биты 31, 32, 1 вектора  $R_{i-1}$ . Полученный после перестановки блок  $E(R_{i-1})$  складывается по модулю 2 с ключами  $k_i$  и затем представляется в виде восьми последовательных блоков  $B_1, B_2, \dots, B_8$ .

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

Каждый  $B_j$  является 6-битовым блоком. Далее каждый из блоков  $B_j$  трансформируется в 4-битовый блок  $B'_j$  с помощью преобразований  $S_j$ . Преобразования  $S_j$  определяются на рисунке 1.4.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	$S_1$
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	$S_2$
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	$S_3$
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	$S_4$
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	$S_5$
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	$S_6$
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	$S_7$
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	$S_8$
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Рисунок 1.4 — Таблица 3. Преобразования  $S_i, i = 1 \dots 8$

Предположим, что  $B_3 = 101111$ , и мы хотим найти  $B'_3$ . Первый и последний разряды  $B_3$  являются двоичной записью числа  $a$ ,  $0 \leq a \leq 3$ , средние 4 разряда представляют число  $b$ ,  $0 \leq b \leq 15$ . Строки таблицы S3 нумеруются от 0 до 3, столбцы таблицы S3 нумеруются от 0 до 15. Пара чисел  $(a, b)$  определяет число, находящееся в пересечении строки  $a$  и столбца  $b$ . Двоичное представление этого числа дает  $B'_3$ . В нашем случае  $a = 11_2 = 3$ ,  $b = 0111_2 = 7$ , а число, определяемое парой  $(3, 7)$ , равно 7. Его двоичное представление  $B'_3$ . Значение функции  $f(R_{i-1}, k_i)$  (32 бит) получается перестановкой  $P$ , применяемой к 32-битовому блоку  $B'_1 B'_2 \dots B'_8$ . Перестановка  $P$  задана таблицей 3.

Таблица 1.3 — Перестановка  $P$

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

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

Согласно таблице 4, первые четыре бита результирующего вектора после действия функции  $f$  — это биты 16, 7, 20, 21 вектора  $B'_1 B'_2 \dots B'_8$

#### 1.3.4 Генерирование ключей $k_i$

Ключи  $k_i$  получаются из начального ключа  $k$  (56 бит = 7 байтов или 7 символов в ASCII) следующим образом. Добавляются биты в позиции 8, 16, 24, 32, 40, 48, 56, 64 ключа  $k$  таким образом, чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64). Такая перестановка определена в таблице 4.

Таблица 1.4 — Перестановка P

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	

Эта перестановка определяется двумя блоками  $C_0$  и  $D_0$  по 28 бит каждый. Первые 3 бита  $C_0$  есть биты 57, 49, 41 расширенного ключа. А первые три бита  $D_0$  есть биты 63, 55, 47 расширенного ключа.  $C_i, D_i, i = 1, 2, 3 \dots$  получаются из  $C_{i-1}, D_{i-1}$  одним или двумя левыми циклическими сдвигами согласно таблице 5.

Таблица 1.5

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

Ключ  $k_i, i=1, \dots, 16$  состоит из 48 бит, выбранных из битов вектора  $C_i D_i$  (56 бит) согласно таблице 7. Первый и второй биты  $k_i$  есть биты 14, 17 вектора  $C_i D_i$

Таблица 1.6

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

### 1.3.5 Конечная перестановка

Конечная перестановка  $IP^{-1}$  действует на  $T_{16}$  и является обратной к первоначальной перестановке. Конечная перестановка определяется таблицей 1.7.

Таблица 1.7

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

## 2 Конструкторский раздел

В данном разделе будет произведена конкретизация задач и проанализированы алгоритмы.

### 2.1 IDEF0 Модель

На рисунке 2.1 приведена функциональная модель шифрования в нотации IDEF0.



Рисунок 2.1 — Функциональная модель шифрования

## 2.2 Разработка алгоритмов

Подробная схема шифрования алгоритма DES представлена на рисунке 2.2.

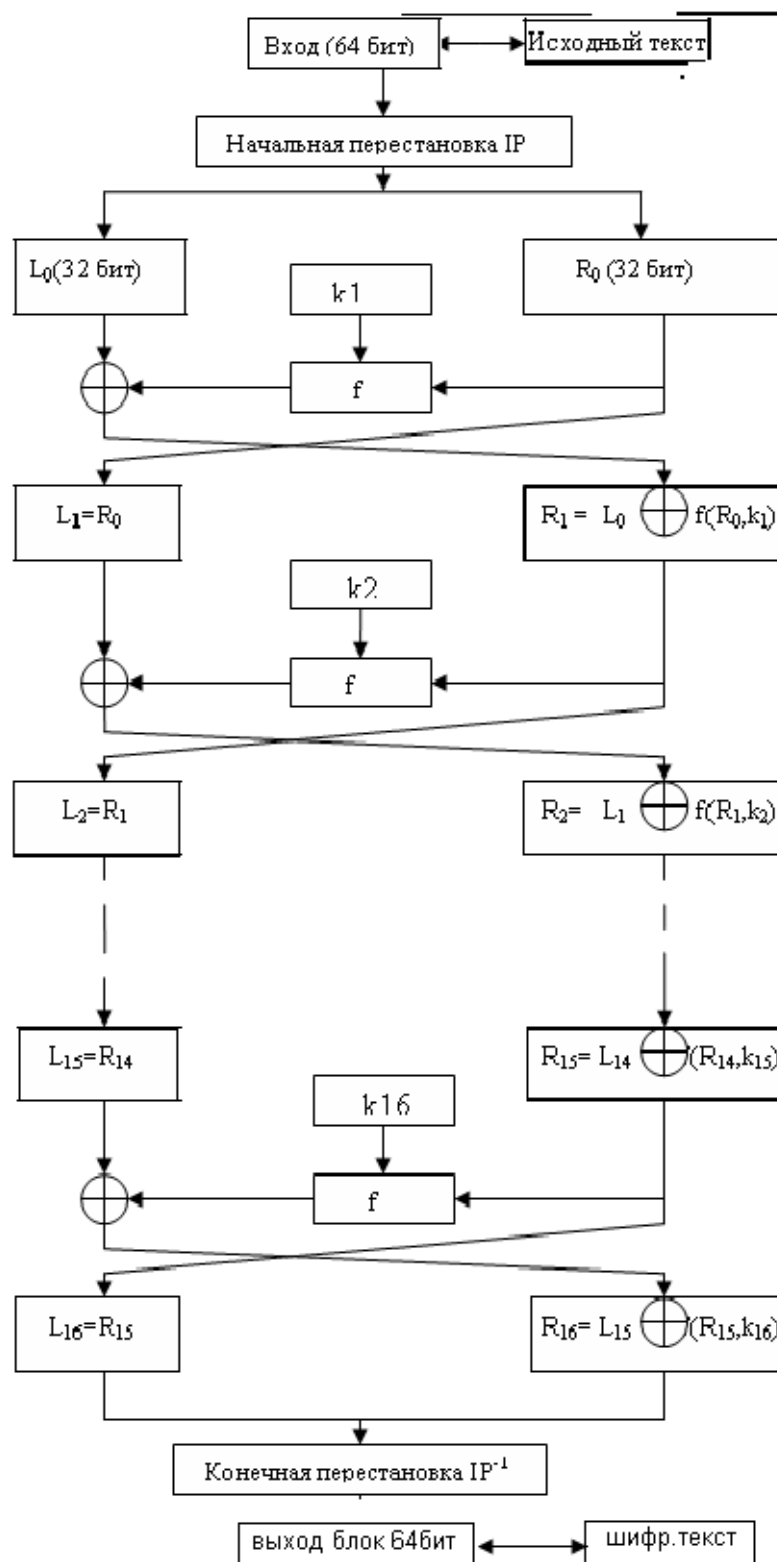


Рисунок 2.2 — Алгоритм DES



## **2.3 Вывод**

В данном разделе была рассмотрена схема алгоритма.

### 3 Технологический раздел

В данном разделе приводятся описания требований к программному обеспечению, средства реализации, листинги кода и описания тестирования.

#### 3.1 Требования к программному обеспечению

Требования к вводу:

- блок размером  $n$  бит;
- ключ размером  $k$  бит;

Требования к выводу:

- зашифрованный блок размером  $n$  бит;

#### 3.2 Средства реализации

В качестве языка программирования был выбран C++. Данный язык имеет высокую скорость и богатую стандартную библиотеку, содержащую необходимые контейнеры для данной работы. Программа, написанная на C++, будет доступна на всех платформах. Для распараллеливания вычислений была использована библиотека с классом нативных потоков `std::thread`.

#### 3.3 Листинги кода

В листинге 3.1 приведена реализация описанного алгоритма.

Листинг 3.1 — Алгоритм шифрования DES

```
1 void Cipher(long long int msg,long long int key){
2     int f=0;
3     int keyr=(int)(key);
4     int keyl=(int)(key>>32);
5     cout<<"Key Left Int:= "<<keyl<<" Key Right Int:= "<<keyr<<endl;
6     int c0=0,d0=0;
7     pKEY(keyl, keyr);
8     c0=keyl;
9     d0=keyr;
10    int kl=0,kr=0;
11    int r0=(int)(msg);
```

```

12     int l0=(int)(msg>>32);
13     cout<<"Msg Left Int:= "<<l0<<"   Msg Right Int:= "<<r0<<endl;
14     pIP(l0,r0);
15     for (int i=1;i<17;i++){
16         int t=2;
17         if (i==1||i==2||i==9||i==16){
18             t=1;
19         }
20         std::thread thr_c(thread_func, std::ref(c0), t);
21         std::thread thr_d(thread_func, std::ref(d0), t);
22
23         thr_c.join();
24         thr_d.join();
25         kl=c0;
26         kr=d0;
27         pSUBKEY(kl,kr);
28         int pl0=l0;
29         l0=r0;
30         r0=pl0^funcF(r0,kl,kr);
31     }
32     int rt=0;
33     rt=r0;
34     r0=l0;
35     l0=rt;
36     finalP(l0,r0);
37     long long int final=l0;
38     final=final<<32;
39     final+=r0;
40     cout<<"[  "<<"Cipher int:#### "<<hex<<final<<" ####\nChipher Left INT:=
        "<<l0<<"   Right INT:= "<<r0 <<endl;
41 }

```

Листинг 3.2 — Функция шифрования в потоке

```

1 void thread_func(int *arg, int t)
2 {
3     *arg = shiftL(*arg, 28, t);
4 }

```

### 3.4 Вывод

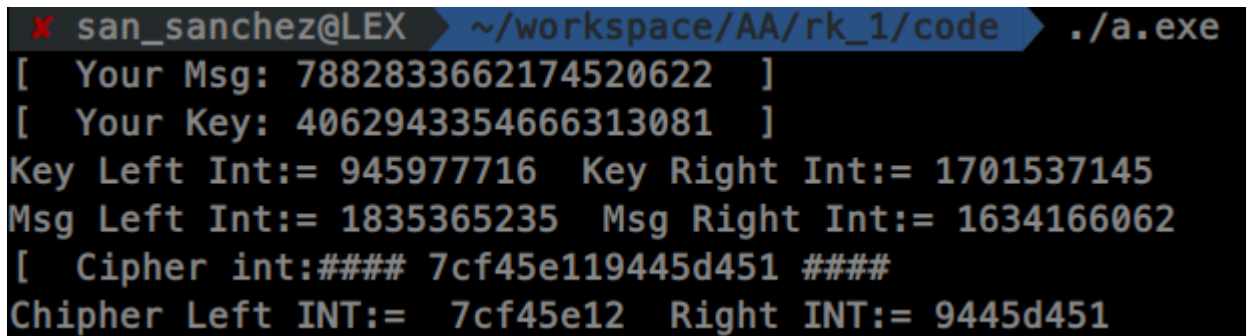
Были сформулированы требования к ПО, выбраны средства реализации.

## 4 Исследовательский раздел

В данном разделе приведены примеры работы программы.

### 4.1 Примеры работы

На рисунках 4.1 показана работа программы.



```
san_sanchez@LEX ~/workspace/AA/rk_1/code ./a.exe
[ Your Msg: 7882833662174520622 ]
[ Your Key: 4062943354666313081 ]
Key Left Int:= 945977716 Key Right Int:= 1701537145
Msg Left Int:= 1835365235 Msg Right Int:= 1634166062
[ Cipher int:#### 7cf45e119445d451 ####
Chipher Left INT:= 7cf45e12 Right INT:= 9445d451
```

Рисунок 4.1 — Работа программы

### 4.2 Вывод

В данном разделе были рассмотрены примеры работы программы.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы мной был изучен метод программирования с несколькими потоками на материале алгоритма шифрования *DES*. Кроме того, была успешно реализована и протестирована программа.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- а) А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин .  
Основы криптографии.
- б) A. Menezes, Pvan Oorschot, S. Vanstone. Handbook of Applied  
Cryptography.
- в) Семенов Ю. А. Алгоритм DES.