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 работе №2

по курсу «Защита информации»

на тему: «Симметричный алгоритм DES/3DES»

Вариант № 2 (CBC)

Студент ИУ7-72Б (Группа) _____ (Подпись, дата) Е. О. Карпова (И. О. Фамилия)

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

2023 г.

1 Теоретический раздел

DES (in English, Data Encryption Standard) — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт. 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 DES рекомендовано несколько режимов:

- ECB (electronic code book) — режим «электронной кодовой книги» (простая замена);
- CBC (cipher block chaining) — режим сцепления блоков;
- PCBC (propagating cipher block chaining) — режим распространяющегося сцепления блоков;
- CFB (cipher feed back) — режим обратной связи по шифротексту;
- OFB (output feed back) — режим обратной связи по выходу;
- Counter Mode (CM) — режим счётчика.

Прямым развитием DES в настоящее время является алгоритм Triple DES (3DES). В 3DES шифрование/расшифровка выполняются путём троекратного выполнения алгоритма DES.

1.1 Описание алгоритма DES

На рисунке 1.1 представлена базовая схема работы алгоритма DES.

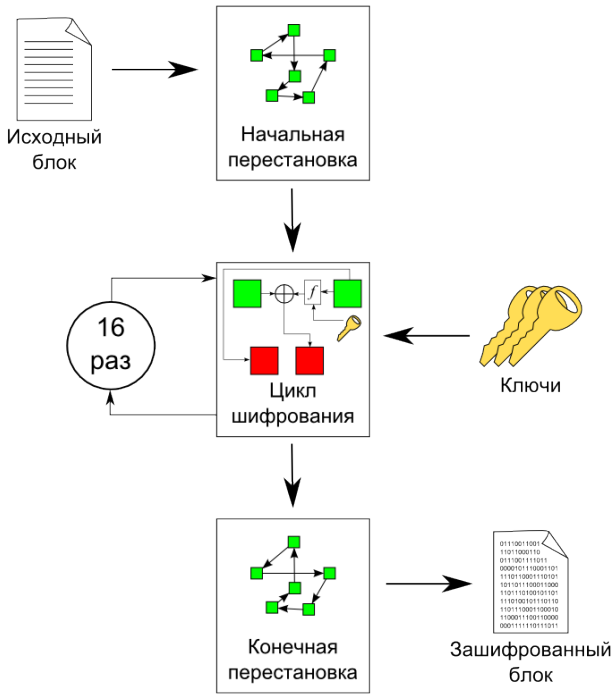

Рисунок 1.1 – Базовая схема работы алгоритма DES

Выполнение алгоритма можно разделить на несколько этапов. Расшифровка представляет собой повторение действий шифрования в обратном порядке. Более подробная схема алгоритма DES представлена на рисунке 1.2.

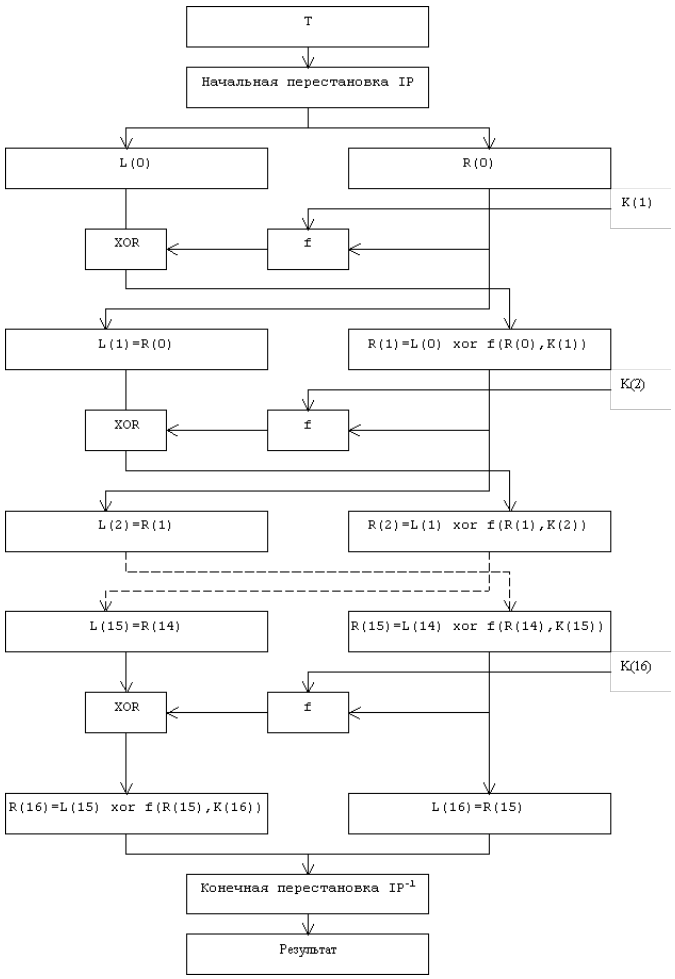

Рисунок 1.2 – Подробная схема работы алгоритма DES

1.1.1 Начальная перестановка

Исходный текст T преобразуется с помощью таблицы IP. Преобразование выполняется по следующему принципу: первый бит результата будет равен значению бита T с номером, равным значению в таблице.

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

1.1.2 Циклы шифрования

После начальной перестановки блок $IP(T)$ участвует в 16 циклах преобразования Фейстеля. Сначала блок разбивается на два 32-битных блока L_0 и R_0 (R_0 — младшая половина).

Тогда в течение 16 итерации можно использовать следующие соотношения:

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

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

Функция Фейстеля

$f(R_{i-1}, k_i)$ — функция Фейстеля. Для вычисления нужно выполнить следующие шаги.

Во-первых, использовать на аргументе (R_{i-1}) функцию расширения E, заданную таблицей. Функция позволяет расширить 32-битный аргумент до 48 бит.

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

Во-вторых, результат расширения сложить по модулю 2 с ключом k_i и разбить на 8 блоков по 6 бит.

$$E(R_{i-1}) \oplus k_i = B_1B_2B_3B_4B_5B_6B_7B_8$$

В-третьих, каждый i-ый блок нужно преобразовать с помощью соответственной таблицы S_i. Пример таблицы S_1 приведен ниже. На основании крайних

битов определется строка таблицы, на основании серединных четырех — столбец. На пересечении стоит четырехбитное число, которое и будет результатом преобразования.

	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
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

После преобразовании четырехбитные блоки склеиваются в один 32-битный, к этому блоку применяется 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

Схема алгоритма функции Фейстеля представлена на рисунке 1.3.

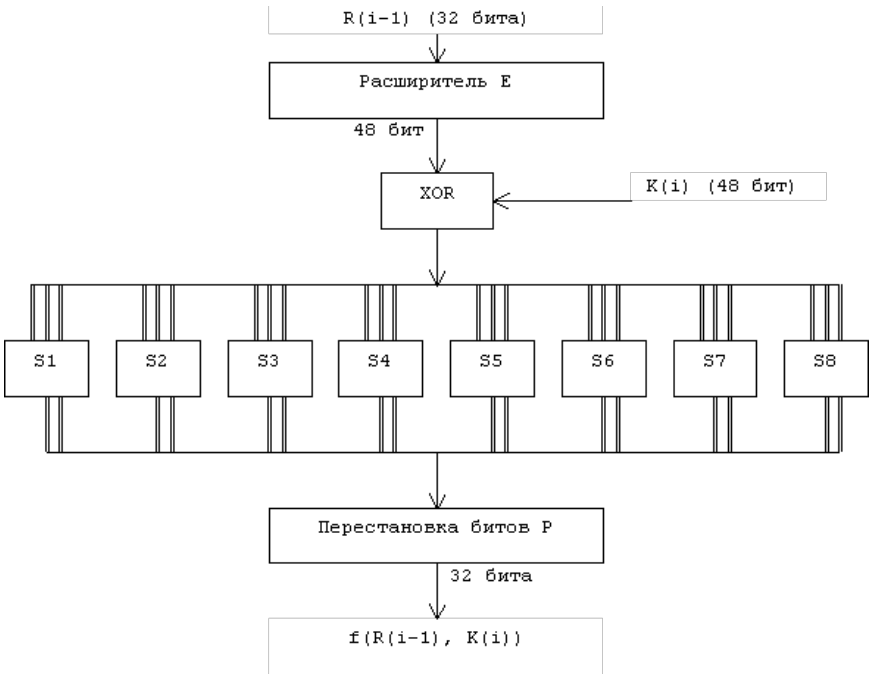

Рисунок 1.3 – Схема алгоритма функции Фейстеля

Генерация ключей

Начальный ключ состоит из 64 бит. С помощью функции G из него убираются контрольные биты и производится перестановка.

57	49	41	33	25	17	9	1	58	50	42	34	26	18
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
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Получившаяся в результате перестановки 56-битная последовательность разбивается на две половины — C_0 и D_0 (D_0 — младшая).

Для i от 1 до 16 с этими половинами производятся циклические сдвиги влево в соответствии с таблицей.

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

Получившиеся блоки склеиваются и производится перестановка в соответствии с таблицей. При этом, из 56-битного ключа, на каждой итерации получается 48-битный ключ.

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

1.1.3 Конечная перестановка

Блоки L_{16} и R_{16} объединяются в блок T_{16} (R_{16} — старшая половина!). Блок T_{16} преобразуется с помощью таблицы IP^{-1}. Преобразование выполняется по следующему принципу: первый бит результата будет равен значению бита T с номером, равным значению в таблице.

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

1.2 Описание режима шифрования CBC

В режиме сцепления блоков (CBC) каждый блок открытого текста складывается по модулю два с предыдущим блоком шифротекста, а затем шифруется. Таким образом, каждый блок шифротекста зависит от всех обработанных блоков открытого текста. Чтобы каждое сообщение было уникальным, при обработке первого блока открытого текста должна использоваться синхропосылка. Режим сцепления блоков является наиболее часто используемым.

На рисунках 1.4 – 1.5 представлена базовая схема работы режима шифрования и расшифрования CBC.

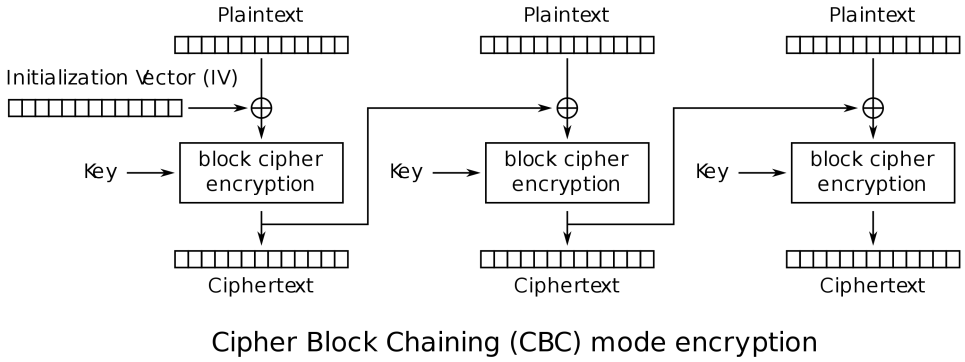

Рисунок 1.4 – Шифрование с использованием CBC

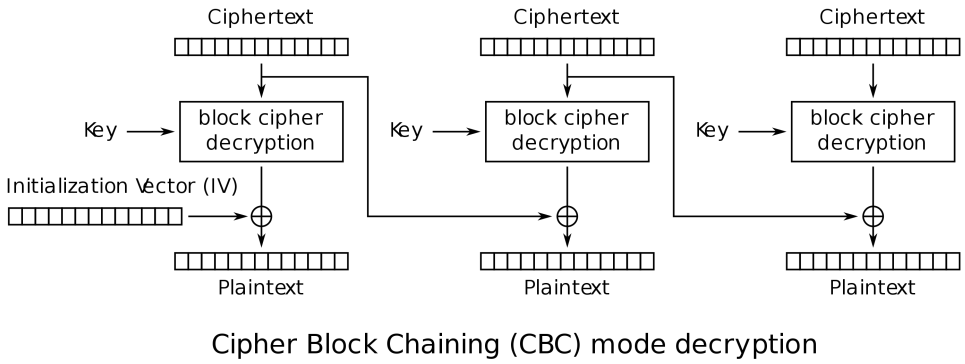

Рисунок 1.5 – Расшифровка с использованием CBC

2 Практический раздел

2.1 Алгоритм шифрования DES

Листинг 2.1 – Реализация алгоритма шифрования DES

```
1  block_t f(des_t des, block_t r, block_t k) {
2      r = e(des, r);
3
4      r = r ^ k;
5
6      block_t b[8];
7      for (int i = 8; i > 0; i--) {
8          b[i - 1] = r & mask(6);
9          r >>= 6;
10     }
11
12     for (int i = 8; i > 0; i--) {
13         block_t row = get_bit_count_from_right(b[i - 1], 5) << 1 |
               get_bit_count_from_right(b[i - 1], 0);
14         block_t col = (b[i - 1] >> 1) & mask(4);
15         b[i - 1] = des.s[i - 1][row][col];
16     }
17
18     r = 0;
19     for (int i = 1; i <= 8; i++) {
20         r <<= 4;
21         r |= b[i - 1];
22     }
23
24     return p(des, r);
25 }
26
27 void fill_keys(des_t des, block_t k[17], block_t raw_key) {
28     k[0] = g(des, raw_key);
29
30     block_t c[17], d[17];
31     c[0] = (k[0] >> HALF_KEY_SIZE) & mask(HALF_KEY_SIZE);
32     d[0] = k[0] & mask(HALF_KEY_SIZE);
```

```
33
34      for (int i = 1; i <= 16; i++) {
35          c[i] = cycle_shift_left(c[i-1], HALF_KEY_SIZE,
                des.shifts[i-1]);
36          d[i] = cycle_shift_left(d[i-1], HALF_KEY_SIZE,
                des.shifts[i-1]);
37
38          k[i] = h(des, d[i] | (c[i] << HALF_KEY_SIZE));
39      }
40  }
41
42  block_t des_encrypt(des_t des, block_t t, block_t key) {
43      block_t t0 = ip(des, t);
44
45      block_t r[17], l[17];
46      r[0] = t0 & mask(HALF_SIZE);
47      l[0] = (t0 >> HALF_SIZE) & mask(HALF_SIZE);
48
49      block_t k[17];
50      fill_keys(des, k, key);
51
52      for (int i = 1; i <= 16; i++) {
53          l[i] = r[i-1];
54          r[i] = l[i-1] ^ f(des, r[i-1], k[i]);
55      }
56
57      block_t t16 = l[16] | (r[16] << HALF_SIZE);
58
59      return ip_reversed(des, t16);
60  }
61
62  block_t des_decrypt(des_t des, block_t t, block_t key) {
63      block_t t16 = ip(des, t);
64
65      block_t r[17], l[17];
66      l[16] = t16 & mask(HALF_SIZE);
67      r[16] = (t16 >> HALF_SIZE) & mask(HALF_SIZE);
68
69      block_t k[17];
```

```
70      fill_keys(des, k, key);
71
72      for (int i = 16; i > 0; i--) {
73          r[i-1] = l[i];
74          l[i-1] = r[i] ^ f(des, l[i], k[i]);
75      }
76
77      block_t t0 = r[0] | (l[0] << HALF_SIZE);
78
79      return ip_reversed(des, t0);
80  }
```

2.2 Режим шифрования CBC

Листинг 2.2 – Реализация режима шифрования CBC

```
1   block_t* cbc_encrypt_blocks(block_t* p, int len, block_t key,
        block_t iv) {
2       block_t* c = calloc(len, sizeof(block_t));
3       if (c == NULL) {
4           return NULL;
5       }
6
7       c[0] = des_encrypt(des_default, p[0] ^ iv, key);
8       for (int i = 1; i < len; i++) {
9           c[i] = des_encrypt(des_default, c[i - 1] ^ p[i], key);
10      }
11
12      return c;
13  }
14
15  block_t* cbc_decrypt_blocks(block_t* c, int len, block_t key,
        block_t iv) {
16      block_t* p = calloc(len, sizeof(block_t));
17      if (p == NULL) {
18          return NULL;
19      }
20
21      p[0] = des_decrypt(des_default, c[0], key) ^ iv;
```

```
22      for (int i = 1; i < len; i++) {
23          p[i] = des_decrypt(des_default, c[i], key) ^ c[i-1];
24      }
25
26      return p;
27  }
```

2.3 Тестирование

Корректность алгоритма проверялось путем применения дешифрации на шифрованное сообщение.

Тестирование было проведено на файлах с типами: текстовый (txt), графический (jpeg, png), архив (zip), несуществующий (ubc). Также, был проведен тест с повреждением зашифрованного файла.

В таблице 2.1 представлены тестовые данные.

Таблица 2.1 – Тестовые данные

Номер теста	Тип файла	Содержимое файла
1	txt	Наглая Пугачева
2	ubc	∅
3	zip	Файлы с тестов 1, 2, 4
4	png	
5	jpeg	
6	jpeg (corrupted) (in english)