Стандарт шифрования данных Data Encryption Standard

В 1977 году Национальное бюро Стандартов США (NBS) опубликовало стандарт шифрования данных Data Encryption Standard (DES), предназначенный для использования в государственных и правительственных учреждениях США для защиты от несанкционированного доступа важной, но несекретной информации. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 году был одобрен ANSI. С этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. Появляются программное обеспечение и специализированные микроЭВМ, предназначенные для шифрования/расшифрования информации в сетях передачи данных и на магнитных носителях. К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. Более того реализация алгоритма DES в таких системах является просто признаком хорошего тона! За примерами далеко ходить не надо. Программа DISKREET из пакета Norton Utilities, предназначенная для создания зашифрованных разделов на диске, использует именно алгоритм DES. "Собственный алгоритм шифрования" отличается от DES только числом итераций при шифровании. Почему же DES добился такой популярности?

Основные достоинства алгоритма DES:

- используется только один ключ длиной 56 битов;
- зашифровав сообщение с помощью одного пакета, для расшифровки вы можете использовать любой другой;
- относительная простота алгоритма обеспечивает высокую скорость обработки информации;
- достаточно высокая стойкость алгоритма.

DES осуществляет шифрование 64-битовых блоков данных с помощью 56-битового ключа. Расшифрование в DES является операцией обратной шифрованию и выполняется путем повторения операций шифрования в обратной последовательности (несмотря на кажущуюся очевидность, так делается далеко не всегда. Позже мы рассмотрим шифры, в которых шифрование и расшифрование осуществляются по разным алгоритмам).

Процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, обратной перестановки битов (рис.1).

Рис.1. Обобщенная схема шифрования в алгоритме DES

Необходимо сразу же отметить, что ВСЕ таблицы, приведенные в данной статье, являются СТАНДАРТНЫМИ, а следовательно должны включаться в вашу реализацию алгоритма в неизменном виде. Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс расшифровки путем подбора ключа. Структура алгоритма DES приведена на рис.2.

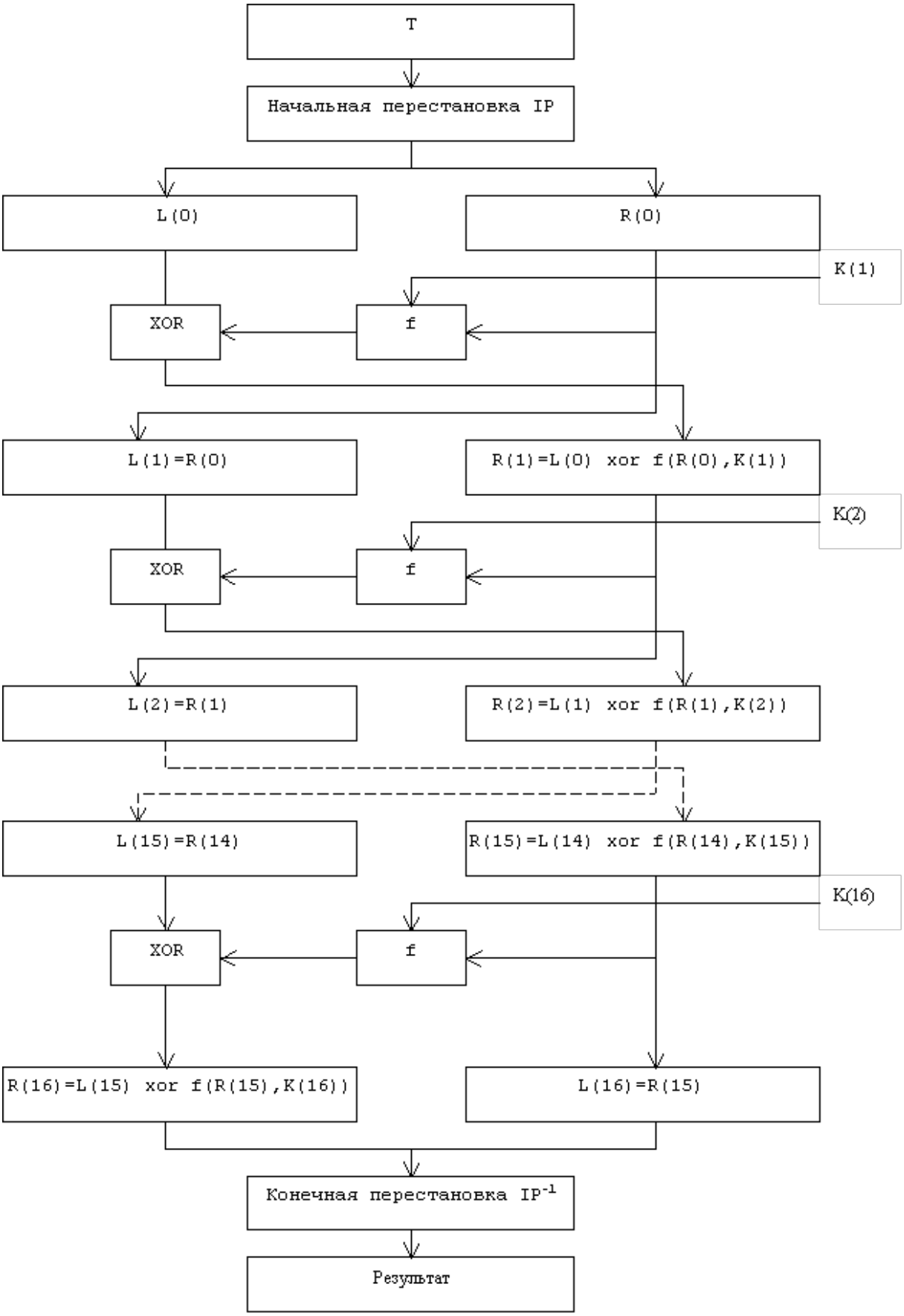

Рис.2. Структура алгоритма шифрования DES

Пусть из файла считан очередной 8-байтовый блок T, который преобразуется с помощью матрицы начальной перестановки IP (табл.1) следующим образом: бит 58 блока T становится битом 1, бит 50 - битом 2 и т.д., что даст в результате: T(0) = IP(T).

Полученная последовательность битов T(0) разделяется на две последовательности по 32 бита каждая: L(0) - левые или старшие биты, R(0) - правые или младшие биты.

Таблица 1
Матрица начальной перестановки IP

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Затем выполняется шифрование, состоящее из 16 итераций. Результат i-й итерации описывается следующими формулами:

$L(i) = R(i-1)$

$R(i) = L(i-1)$ xor $f(R(i-1), K(i))$,

где xor - операция ИСКЛЮЧАЮЩЕЕ ИЛИ.

Функция f называется функцией шифрования. Ее аргументы - это 32-битовая последовательность R(i-1), полученная на (i-1)-ой итерации, и 48-битовый ключ K(i), который является результатом преобразования 64-битового ключа K. Подробно функция шифрования и алгоритм получения ключей K(i) описаны ниже.

На 16-й итерации получают последовательности R(16) и L(16) (без перестановки), которые конкатенируют в 64-битовую последовательность R(16)L(16).

Затем позиции битов этой последовательности переставляют в соответствии с матрицей IP^{-1} (табл.2).

Таблица 2
Матрица обратной перестановки IP^{-1}

40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

Матрицы IP^{-1} и IP соотносятся следующим образом: значение 1-го элемента матрицы IP^{-1} равно 40, а значение 40-го элемента матрицы IP равно 1, значение 2-го элемента матрицы IP^{-1} равно 8, а значение 8-го элемента матрицы IP равно 2 и т.д.

Процесс расшифрования данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP^{-1}, а затем над последовательностью бит R(16)L(16) выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифрования может быть описан следующими формулами:

R(i-1)=L(i), i=1,2, ...,16;

L(i-1)=R(i) xor f(L(i),K(i)), i=1,2, ... ,16.

На 16-й итерации получают последовательности L(0) и R(0), которые конкатенируют в 64-битовую последовательность L(0)R(0).

Затем позиции битов этой последовательности переставляют в соответствии с матрицей IP. Результат такой перестановки - исходная 64-битовая последовательность.

Теперь рассмотрим функцию шифрования f(R(i-1),K(i)). Схематически она показана на рис. 3.

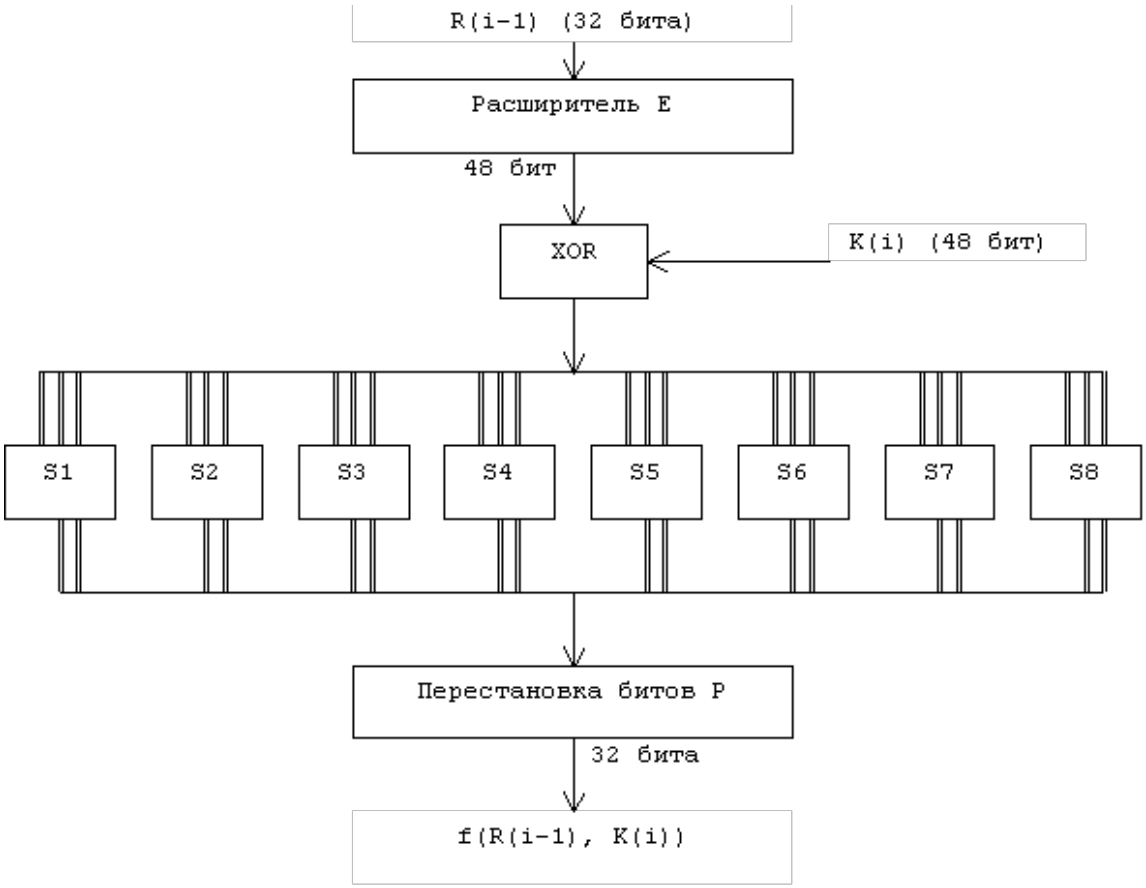

Рис.3. Вычисление функции f(R(i-1),K(i))

Для вычисления значения функции f используются следующие функции-матрицы:

- E - расширение 32-битовой последовательности до 48-битовой,
- S1, S2, ... , S8 - преобразование 6-битового блока в 4-битовый,
- P - перестановка бит в 32-битовой последовательности.

Функция расширения E определяется табл.3. В соответствии с этой таблицей первые 3 бита E(R(i-1)) - это биты 32, 1 и 2, а последние - 31, 32 и 1.

Таблица 3

Функция расширения E

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	01

Результат функции E(R(i-1)) есть 48-битовая последовательность, которая складывается по модулю 2 (операция xor) с 48-битовым ключом K(i). Получается 48-битовая последовательность, которая разбивается на восемь 6-битовых блоков B(1)B(2)B(3)B(4)B(5)B(6)B(7)B(8). То есть:

E(R(i-1)) xor K(i) = B(1)B(2)...B(8).

Функции S1, S2, ... , S8 определяются табл.4.

Таблица 4

Функции преобразования S1, S2, ..., S8

Номер строки	Номер столбца 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	S1
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	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
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	S3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
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	S4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
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	S5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
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	S6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
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	S7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
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	S8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
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	

К табл.4. требуются дополнительные пояснения. Пусть на вход функции-матрицы Sj поступает 6-битовый блок B(j) = b1b2b3b4b5b6, тогда двухбитовое число b1b6 указывает номер строки матрицы, а b2b3b4b5 - номер столбца. Результатом Sj(B(j)) будет 4-битовый элемент, расположенный на пересечении указанных строки и столбца.

Например, B(1)=011011. Тогда S1(B(1)) расположен на пересечении строки 1 и столбца 13. В столбце 13 строки 1 задано значение 5. Значит, S1(011011)=0101.

Применив операцию выбора к каждому из 6-битовых блоков B(1), B(2), ..., B(8), получаем 32-битовую последовательность S1(B(1))S2(B(2))S3(B(3))...S8(B(8)).

Наконец, для получения результата функции шифрования надо переставить биты этой последовательности. Для этого применяется функция перестановки P (табл.5). Во входной последовательности биты перестанавливаются так, чтобы бит 16 стал битом 1, а бит 7 - битом 2 и т.д.

Таблица 5
Функция перестановки P

16	07	20	21
29	12	28	17
01	15	23	26
05	18	31	10
02	08	24	14
32	27	03	09
19	13	30	06
22	11	04	25

Таким образом,

$$f(R(i-1), K(i)) = P(S1(B(1)),...S8(B(8)))$$

Чтобы завершить описание алгоритма шифрования данных, осталось привести алгоритм получения 48-битовых ключей K(i), i=1...16. На каждой итерации используется новое значение ключа K(i), которое вычисляется из начального ключа K. 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 (табл.6).

Таблица 6
Матрица G первоначальной подготовки ключа

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

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...16. Для этого применяют циклический сдвиг влево на один или два бита в зависимости от номера итерации, как показано в табл.7.

Таблица 7
Таблица сдвигов для вычисления ключа

Номер итерации	Сдвиг (бит)

01	1
02	1
03	2
04	2
05	2
06	2
07	2
08	2
09	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Полученное значение вновь "перемешивается" в соответствии с матрицей H (табл.8).

Таблица 8
Матрица H завершающей обработки ключа

14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Ключ K(i) будет состоять из битов 14, 17, ..., 29, 32 последовательности C(i)D(i). Таким образом:

K(i) = H(C(i)D(i))

Блок-схема алгоритма вычисления ключа приведена на рис.4.

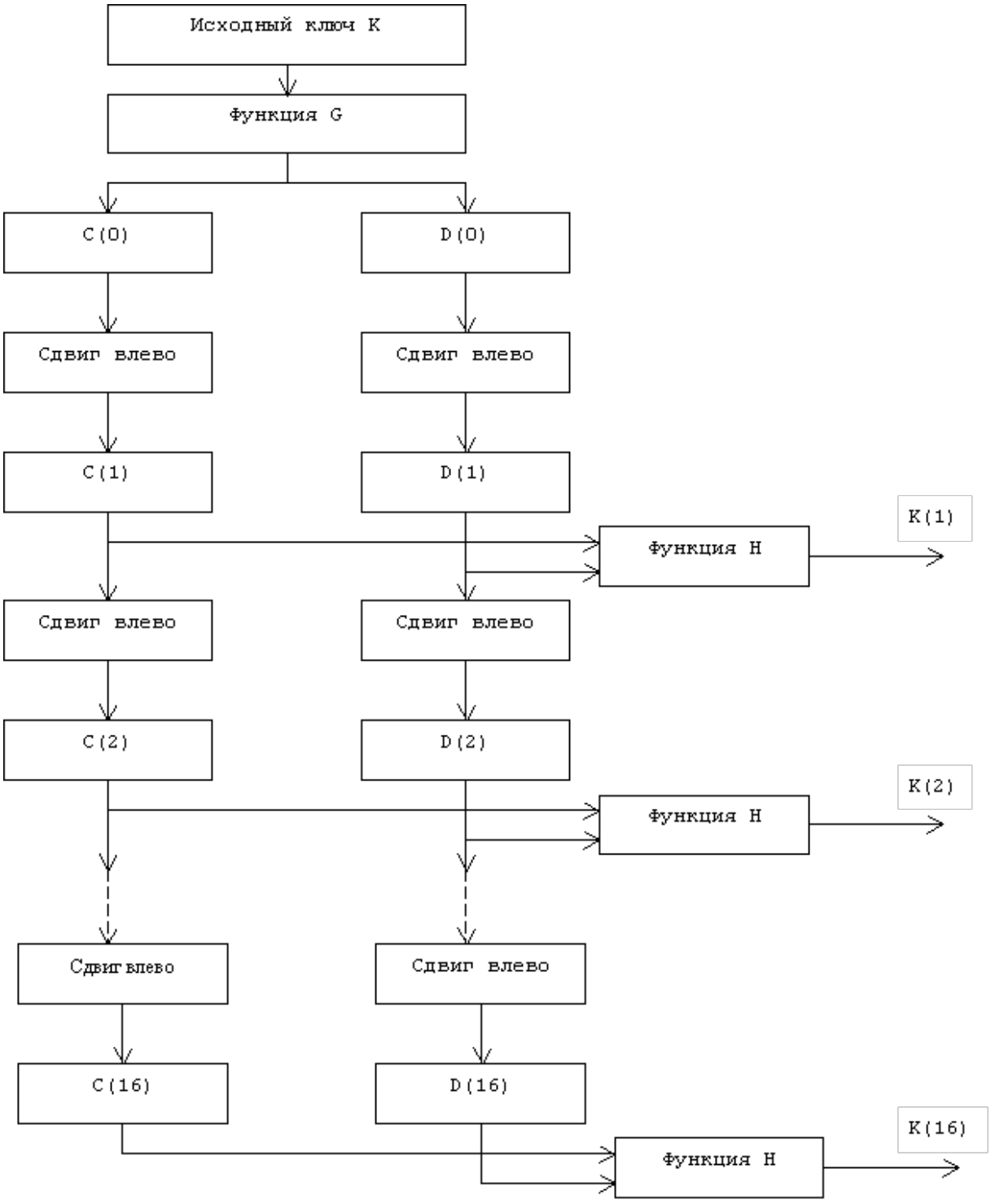

Рис.4. Блок-схема алгоритма вычисления ключа K(i)

Восстановление исходного текста осуществляется по этому алгоритму, но вначале вы используете ключ
K(15), затем - K(14) и так далее. Теперь вам должно быть понятно, почему автор настойчиво рекомендует использовать приведенные матрицы. Если вы начнете самовольничать, вы, должно быть, получите очень секретный шифр, но вы сами не сможете его потом раскрыть!

Режимы работы алгоритма DES

Для наиболее полного удовлетворения всем требованиям, предъявляемым к коммерческим системам шифрования, реализованы несколько режимов работы алгоритма DES. Наиболее широкое распространение получили режимы:

- электронный шифроблокнот (Electronic Codebook) - ECB;
- цепочка цифровых блоков (Cipher Block Chaining) - CBC;
- цифровая обратная связь (Cipher Feedback) - CFB;
- внешняя обратная связь (Output Feedback) - OFB.

Кстати, если вы написали программу защиты данных и хотите дать полноценную рекламу, то автор рекомендует прямо указывать, какие из режимов поддерживает ваше детище. Это, вообще говоря, признак хорошего тона на рынке программного обеспечения средств защиты. Нет смысла раскрывать весь алгоритм, вы просто указываете : DES-CBC или DES-CFB и, как говорится, "умный догадается, а дураку и не надо".

Давайте рассмотрим перечисленные выше режимы

DES-ECB

В этом режиме исходный файл M разбивается на 64-битовые блоки (по 8 байтов): M = M(1)M(2)...M(n). Каждый из этих блоков кодируется независимо с использованием одного и того же ключа шифрования (рис.5). Основное достоинство этого алгоритма - простота реализации. Недостаток - относительно слабая устойчивость против квалифицированных криптоаналитиков.

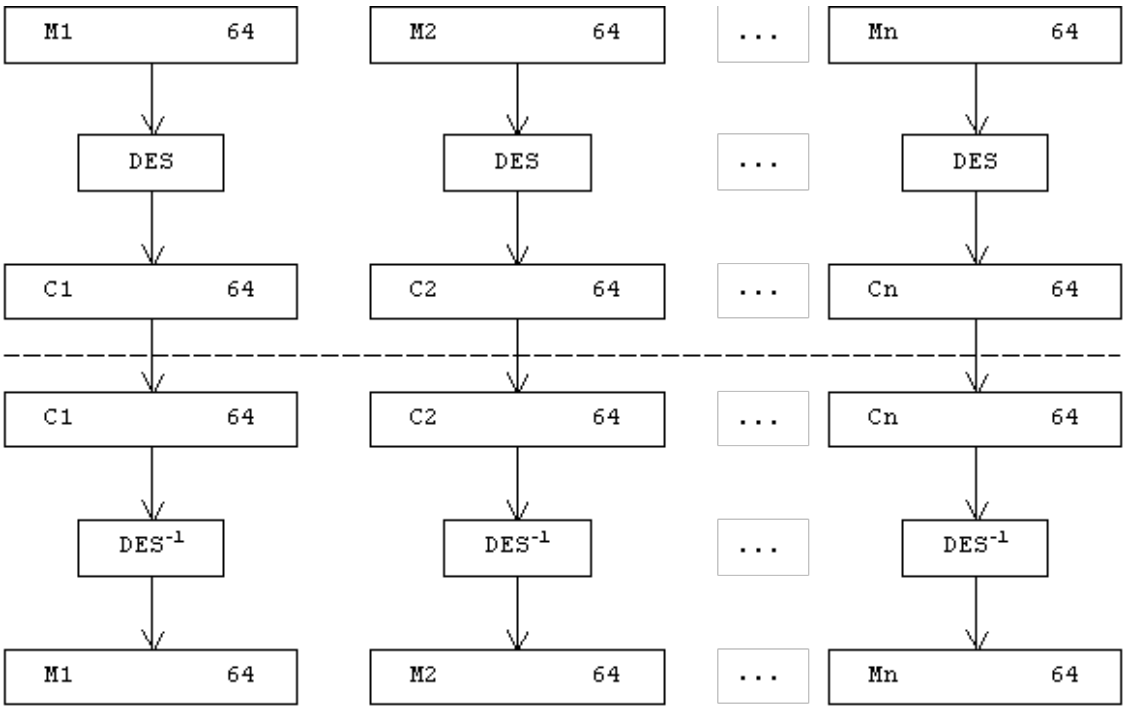

Рис.5. Работа алгоритма DES в режиме ECB

В частности, не рекомендуется использовать данный режим работы для шифрования EXE файлов, потому что первый же блок - заголовок файла, является вполне удачным началом для взлома всего шифра.

В то же время следует признать, что этот режим в силу своей простой реализации наиболее популярен среди любительских разработок.

DES-CBC

В этом режиме исходный файл M также, как и в режиме ECB, разбивается на 64-битовые блоки: M = M(1)M(2)...M(n). Первый блок M(1) складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете. Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый блок шифртекста C(1) складывается по модулю 2 со вторым блоком исходного текста, результат шифруется и получается второй 64-битовый блок шифртекста C(2) и т.д. Процедура повторяется до тех пор, пока не будут обработаны все блоки исходного текста (рис.6).

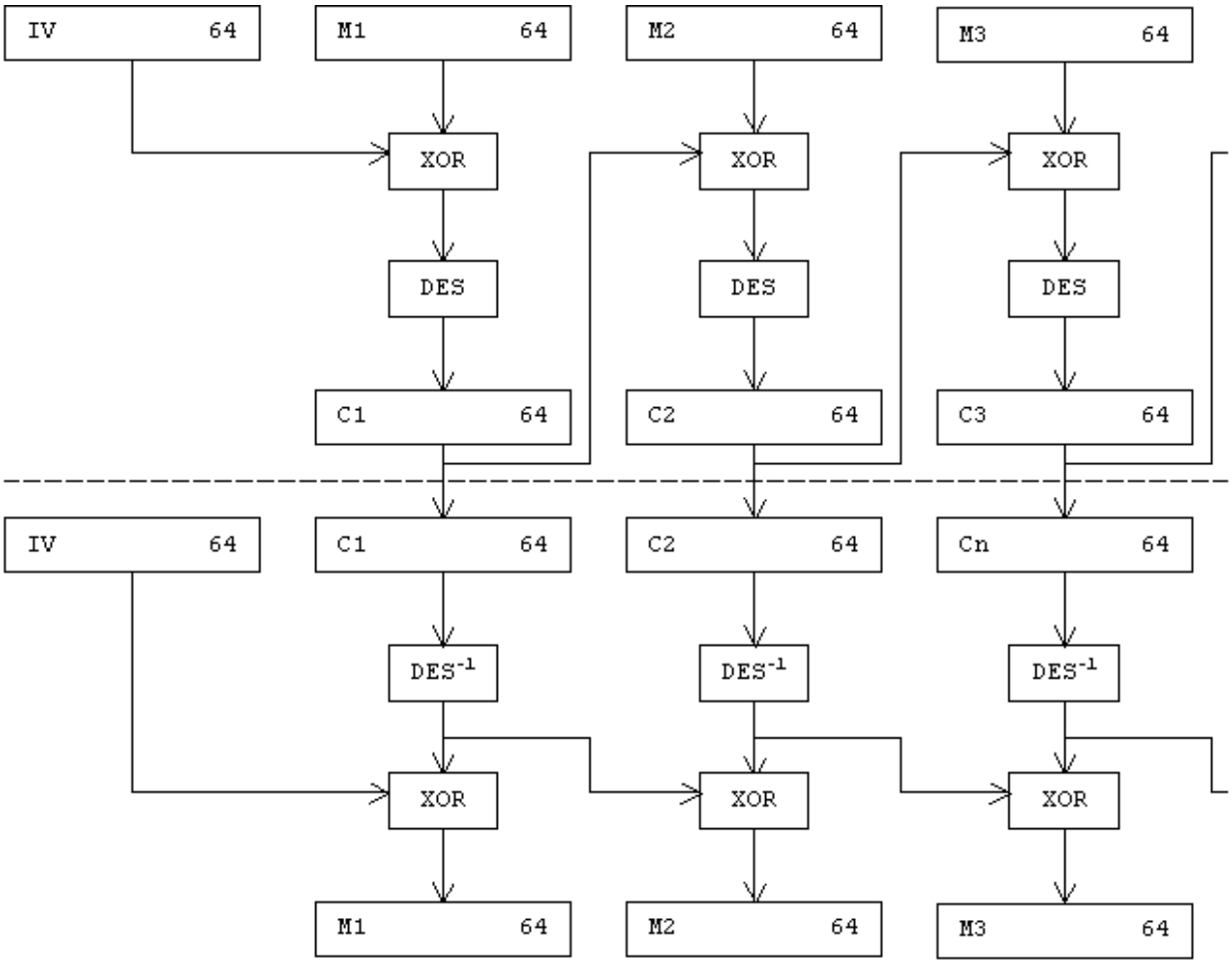

Рис.6. Работа алгоритма в режиме CBC

Таким образом для всех i = 1...n блок шифртекста C(i) определяется следующим образом:

C(i) = DES(M(i) xor C(i-1)),

C(0) = IV - начальное значение шифра, равное начальному вектору.

Расшифрование выполняется следующим образом:

M(i) = C(i-1) xor DES^{-1}(C(i)),

C(0) = IV - начальное значение шифра, равное начальному вектору.

Прелесть данного режима состоит в том, что он не позволяет накапливаться ошибкам при передаче. Блок M(i) является функцией только C(i-1) и C(i). Поэтому ошибка при передаче приведет к потере только двух блоков исходного текста.

DES-CFB

В этом режиме размер блока может отличаться от 64. Исходный файл M считывается последовательными t-битовыми блоками (t <= 64): M = M(1)M(2)...M(n) (остаток дописывается нулями или пробелами).

64-битовый сдвиговый регистр (входной блок) вначале содержит вектор инициализации IV, выравненный по правому краю. Для каждого сеанса шифрования используется новый IV.

Для всех i = 1...n блок шифртекста C(i) определяется следующим образом:

C(i) = M(i) xor P(i-1),

где P(i-1) - старшие t битов операции DES(C(i-1)), причем C(0)=IV.

Обновление сдвигового регистра осуществляется путем удаления его старших t битов и дописывания справа C(i).

Восстановление зашифрованных данных также не представляет труда: P(i-1) и C(i) вычисляются аналогичным образом и

M(i) = C(i) xor P(i-1).

Блок-схема режима CFB приведена на рис.7.

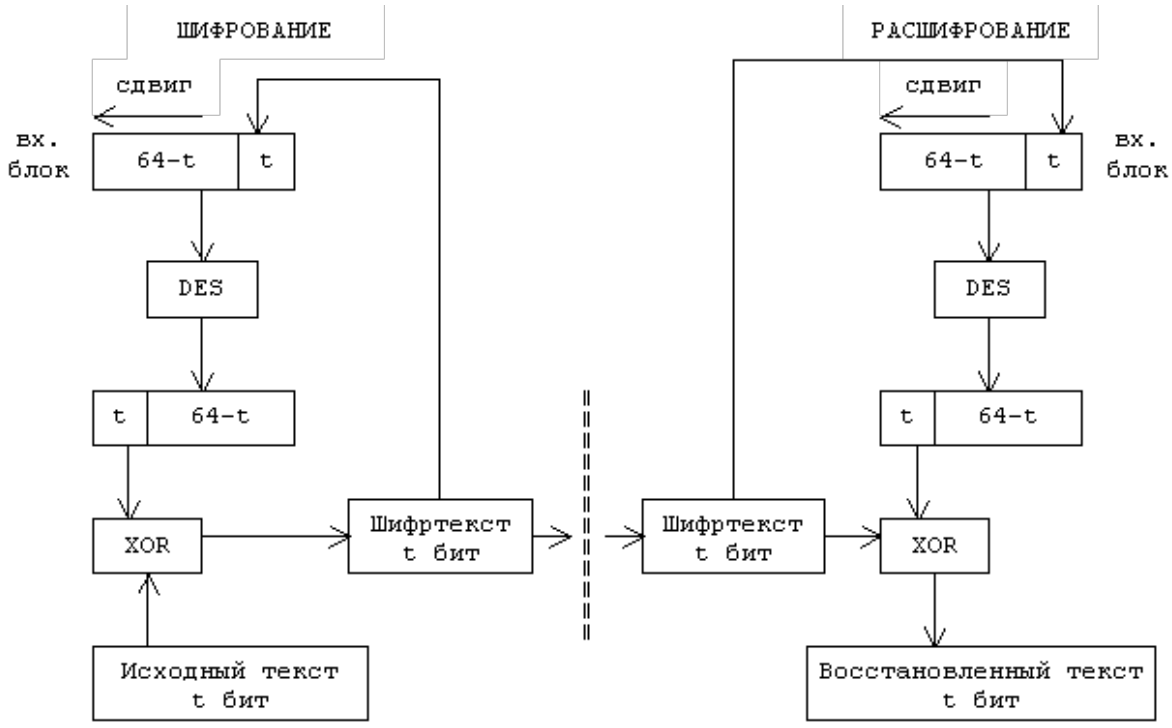

Рис.7. Работа алгоритма DES в режиме CFB

DES-OFB

Режим OFB очень похож на режим CFB.

Отличие от режима CFB состоит только в методе обновления сдвигового регистра. В данном случае это осуществляется путем удаления его старших t битов и дописывания справа P(i-1) (рис.8).

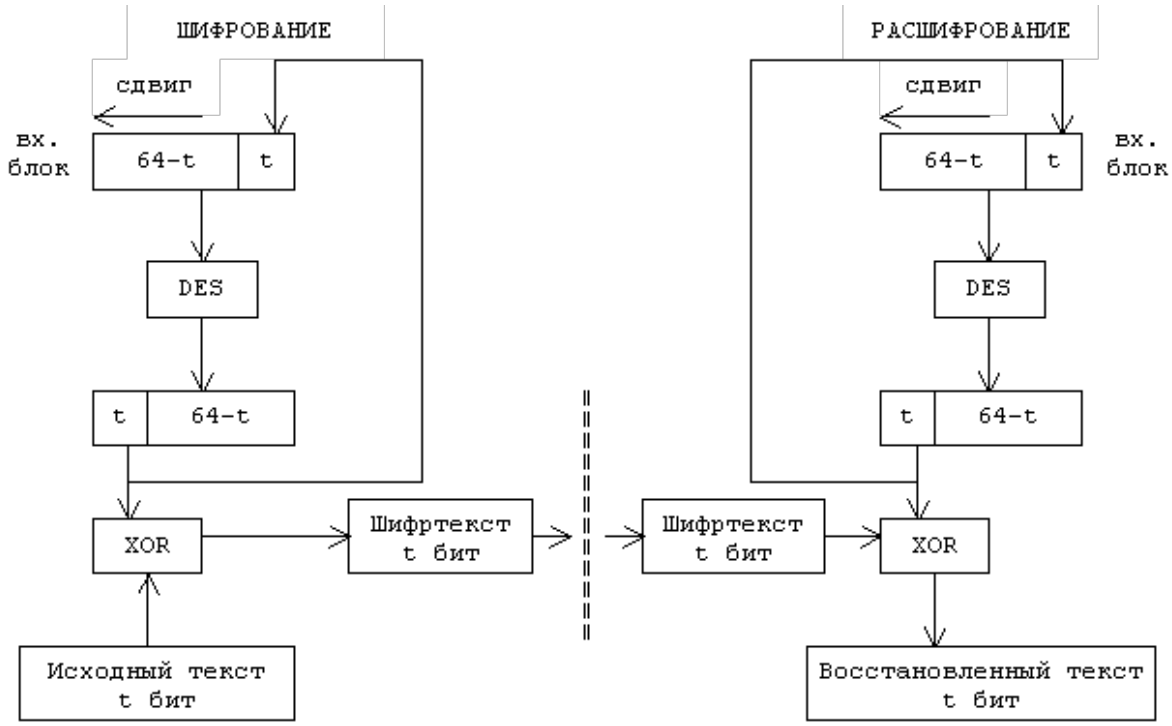

Рис.8. Блок-схема алгоритма DES в режиме OFB

Каждому из рассмотренных режимов свойственны свои достоинства и недостатки, что обусловливает области их применения.

Режим ECB хорошо подходит для шифрования ключей. Режимы CBC и CFB пригодны для аутентификации данных. Режим CFB, кроме того, предназначен для шифрования отдельных символов. Режим OFB нередко используется в спутниковых системах связи.
