ЗАЩИТА ИНФОРМАЦИИ

Методические указания к лабораторной работе

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1
“ШИФРОВАНИЕ ДАННЫХ МЕТОДОМ ПОДСТАНОВКИ”

Цель работы

Целью работы является знакомство с классическим криптографическим алгоритмом - алгоритмом шифрования данных при помощи подстановки.

Основные сведения

В современной криптографии рассматриваются два типа криптографических алгоритмов. Это классические криптографические алгоритмы, основанные на использовании секретных ключей, и новые криптографические алгоритмы с открытым ключом, основанные на использовании ключей двух типов: секретного (закрытого) и открытого.

В классической криптографии ("криптографии с секретным ключом" или "одноключевой криптографии") используется только одна единица секретной информации - ключ, знание которого позволяет отправителю зашифровать информацию в шифртекст, а получателю - расшифровать его. Операция шифрования/дешифрования с большой вероятностью невыполнима без знания секретного ключа. Поскольку при использовании классических криптографических алгоритмов ключ шифрования и ключ дешифрования совпадают и такие криптосистемы называются симметричными.

Подстановочное шифрование основывается на использовании некоторой взаимно однозначной функции $C_V : V^m \to V^m$, где V - алфавит шифруемых сообщений, m - длина блока открытого текста и блока шифрограммы. В процессе шифрования открытый текст X разбивается на m-символьные блоки x_1, x_2, ..., x_l, каждый из которых заменяется m-символьным блоком

$y_i = C_V(x_i),\ i = \overline{1,l}$. Дешифрование сводится к обратной замене m-символьных блоков y на m-символьные блоки $C_V^{-1}(y_i),\ i = \overline{1,m}$.

Например, пусть алфавит V = {0,1, .., 9, <пробел>, A, B, .., Z, a, b, .., z, А, Б, .., Я, а, б, .., я}, длина блока шифрограммы и блока открытого текста m = 3. Допустим, что необходимо зашифровать открытый текст X = “Произвольный блок открытого текста”. Разобьем открытый текст X на m-символьные (трехсимвольные в нашем примере) блоки: “Про”, “изв”, “оль”, “ный”, “_бл”, “ок_“, “отк”, “рыт”, “ого”, “_те”, “кст”, “а__”. Пробелы обозначены символом _, при необходимости последний блок может быть дополнен с правой стороны необходимым количеством пробелов. Если пробел не входит в алфавит языка, то его функцию (функцию разделительного элемента) может выполнять любой другой символ алфавита, если стороны, обменивающиеся сообщениями, достигли соответствующей договоренности.

Для шифрования необходимо иметь функцию C_V, ставящую каждому трехсимвольному блоку открытого текста трехсимвольный блок шифртекста. Такая функция может быть задана, например, при помощи таблицы:

x_i	Про	изв	оль	ный	_бл	ок_	отк	рыт	ого	_те	кст	а__	…
$C_V(x_i)$	Атр	ф7ы	нрв	св_	ркк	ыт0	мкф	ц_й	1ся	щн_	ы34	вхш	…

Каждый блок открытого текста заменяется при помощи функции C_V соответствующим блоком шифртекста. Таким образом, для рассматриваемого примера шифртекст будет выглядеть следующим образом: “Атрф7ынрвсв_рккыт0мкфц_й1сящн_ы34вхш”.

Поскольку функция C_V является взаимно однозначной, эта же таблица используется и для дешифрации шифртекста.

Очевидно, что приведенные в этом примере алфавит и принятый размер блока открытого текста требуют очень большой таблицы, задающей функцию шифрования: эта таблица должна задавать все возможные трехсимвольные сочетания из русских и латинских букв, а также цифр. Если в качестве алфавита рассматривать двоичный алфавит {0, 1}, а размер блока открытого текста принять равным 7 (как это имеет место в случае обычного ASCII-кода), то для задания функции шифрования требуется таблица со 128 столбцами. В общем случае, требуется определить $|V|^m$ значений функции, где $|V|$ - мощность множества V, то есть количество элементов алфавита. Разумеется, если заранее известно, что некоторые комбинации символов открытого текста являются недопустимыми, то указанное значение может быть уменьшено.

Индивидуальные задания

Задания выбираются студентами из нижеприведенной таблицы в соответствии со своими номерами по списку (по модулю 10).

Задание			Для информации	
№ п.п.	V	m	$\|V\|$	$\|V\|^m$
1	{0,1}	5	2	32
2	{0,1,2}	3	3	27
3	{0,1,2,3,4}	2	5	25
4	{0,1,2,3,4,5}	2	6	36
5	{A,B,..,Z}	1	26	26
6	{А,Б,..,Я}	1	33	33
7	{x,y}	5	2	32
8	{x,y,z}	3	3	27
9	{a,b,c,d,e}	2	5	25
10	{а,б,в,г,д,е}	2	6	36

Порядок выполнения

1. Изучить основы шифрования данных методом подстановки.

2. В соответствии с индивидуальным заданием разработать алгоритм и написать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

Примечание. Функцию шифрования, основываясь на данных индивидуального задания, определить самостоятельно.

Содержание отчета

1. Цель работы.
2. Индивидуальное задание.
3. Функция шифрования.
4. Текст программы, реализующей индивидуальное задание.
5. Пример открытого текста и соответствующей ему шифрограммы.
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №2
“ШИФРОВАНИЕ ДАННЫХ МЕТОДОМ ПЕРЕСТАНОВКИ”

Цель работы

Целью работы является знакомство с классическим криптографическим алгоритмом - алгоритмом шифрования данных при помощи перестановки.

Основные сведения

Перестановочное шифрование, также как и подстановочное, относится к классической криптографии (см. лабораторную работу №1): здесь тоже используется только одна единица секретной информации - ключ, знание которого позволяет отправителю зашифровать информацию в шифртекст, а получателю - расшифровать его.

Перестановочное шифрование основывается на использовании некоторой взаимно однозначной функции $C_N : M \to M$, где M={1,...,m} - множество номеров позиций символов в m-символьном блоке открытого текста. В процессе шифрования блок открытого текста, имеющий вид $x=v_1v_2..v_m$, где v_k, k=1..m, означает символ алфавита V, стоящий на k-ой позиции, заменяется блоком $y=w_1w_2\ ..w_m$, где $w_j=v_q$, $q = C_N(j)$, $j = \overline{1,m}$. Дешифрование состоит в обратной замене блока $y=w_1w_2\ ..w_m$ на $v_1v_2\ ..v_r\ ..v_m$, где $v_r=w_j$, $j = C_N^{-1}(j)$.

Например, пусть алфавит V = {0,1, .., 9, <пробел>, A, B, .., Z, a, b, .., z, А, Б, .., Я, а, б, .., я}, длина блока шифрограммы и блока открытого текста m = 15. Допустим, что необходимо зашифровать открытый текст X = “Произвольный блок открытого текста”. Разобьем открытый текст X на m-символьные (пятнадцатисимвольные в нашем примере) блоки:

“Произвольный_бл”, ”ок_открытого_те”, “кста_____”. Пробелы обозначены символом _, при необходимости последний блок может быть дополнен с правой стороны необходимым количеством пробелов. Если пробел не входит в алфавит языка, то его функцию (функцию разделительного элемента) может выполнять любой другой символ алфавита, если стороны, обменивающиеся сообщениями достигли соответствующей договоренности.

Для шифрования необходимо иметь функцию C_N, ставящую i-й позиции символа в блоке открытого текста $C_N(i)$-ю позицию в блоке шифртекста. Такая функция может быть задана, например, при помощи таблицы:

$C_N(i)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i	13	5	1	8	10	12	2	14	4	6	3	15	7	11	9

В каждом пятнадцатисимвольном блоке шифртекста, в нашем примере, на первой позиции будет стоять символ, занимающий 13-ю позицию в блоке открытого текста, на второй позиции - символ, занимающий 5-ю позицию в блоке открытого текста, и т.д. Таким образом, наш открытый текст даст следующие блоки шифртекста: “_зПлнйрбиволоыь”, “_тоыоокток_ергт”, “__к___с_а_т____”, или целиком: “_зПлнйрбиволоыь_тоыоокток_ергт__к___с_а_т____”

Поскольку функция C_N является взаимно однозначной, эта же таблица используется и для дешифрации шифртекста.

Очевидно, что размер таблицы, задающей функцию шифрования, определяется принятым размером блока открытого текста: всего необходимо определить m значений функции.

Индивидуальные задания

Задания выбираются студентами из нижеприведенной таблицы в соответствии со своими номерами по списку (по модулю 10).

Задание		
№ п.п.	*V*	*m*
1	{ а,б,..,я,a,b,..,z}	15
2	{А,Б,..,Я,а,б,..,я}	16
3	{A,B,..,Z,a,b,..,z}	18
4	{0,1,.,9,A,B,C,D,E,F}	16
5	{A,B,..,Z,<пробел>}	15
6	{А,Б,..,Я,<пробел>}	17
7	{a,b,c,d,e,0,1,..9}	15
8	{а,б,в,г,д,е,0,1,..9}	18
9	{a,b,c,d,e,#,б,в,г,д}	16
10	{а,б,в,г,д,е,@,#,$,%}	17

Порядок выполнения

1. Изучить основы шифрования данных методом перестановки.

2. В соответствии с индивидуальным заданием разработать алгоритм и написать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

Примечание. Функцию шифрования, основываясь на данных индивидуального задания, определить самостоятельно.

Содержание отчета

1. Цель работы.

2. Индивидуальное задание.

3. Функция шифрования.

4. Текст программы, реализующей индивидуальное задание.

5. Пример открытого текста и соответствующей ему шифрограммы.

6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №3
“ЛИНЕЙНОЕ ШИФРОВАНИЕ ДАННЫХ (ГАММИРОВАНИЕ)”

Цель работы

Целью работы является знакомство с классическим криптографическим алгоритмом - алгоритмом линейного шифрования данных (шифрования гаммированием).

Основные сведения

Если изученные в двух предыдущих лабораторных работах криптографические алгоритмы - подстановка и перестановка - основаны на блочном способе шифрования, то гаммирование, называемое также линейным шифрованием, основывается на поточном способе шифрования.

Шифрование гаммированием состоит в наложении на открытый текст X некоторой псевдослучайной последовательности K ("гаммы") символов того же алфавита. Для двоичного алфавита $V=\{0,1\}$ наложение $K=k_1..k_m$ на $X=v_1..v_m$ состоит в замене v_i на $v_i \oplus k_i$, где символом $\oplus$ обозначена операция сложения по модулю 2. Дешифрование сводится к наложению той же гаммы на шифртекст. Поскольку $(v_i \oplus k_i) \oplus k_i = v_i$, результатом повторного выполнения той же операции будет действительно открытый текст. Для алфавита, включающего $n>2$ символов, где n - размер алфавита, при шифровании вместо операции $\oplus$ используется операция сложения по модулю n. При дешифровании происходит вычитание гаммы из шифртекста, а при получении отрицательной величины следует прибавить к ней величину n.

Чтобы обеспечить высокую криптостойкость шифртекста, последовательность K должна быть достаточно длинной, а именно - превосходить длину открытого текста X. Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются датчики псевдослучайных чисел (ПСЧ). Одним из таких датчиков, зарекомендовавшим свою эффективность, является так называемый линейный конгруэнтный датчик ПСЧ. Он вырабатывает последовательности псевдослучайных чисел *T(i)*, описываемые соотношением:

$$T(i+1)=(A*T(i)+C) \bmod B,$$

где *A* и *C* - константы, *T(0)* - исходная величина, выбранная в качестве порождающего числа, а значение числа *B* устанавливается равным 2^b, где *b* - длина слова ЭВМ в битах. Такой датчик генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений *A* и *C*. Датчик имеет максимальный период *M* до того, как генерируемая последовательность чисел начнет повторяться. Для достижения максимальной криптостойкости, доступной данному методу шифрования, необходимо выбирать числа *A* и *C* таким образом, чтобы период *M* был максимальным. Как было показано Д.Кнутом, линейный конгруэнтный датчик ПСЧ имеет максимальную длину тогда и только тогда, когда *C* - нечетное и $A \bmod 4=1$.

Очевидно, что, поскольку гамма определяется датчиком ПСЧ, то совокупность значений *A*, *C*, *T(0)* и *B* фактически является тем секретным ключом, который и обеспечивает защиту передаваемой или хранимой информации.

Рассмотрим пример. Пусть задан алфавит *V* = {0,1, .., 9, <пробел>, A, B, .., Z, a, b, .., z, А, Б, .., Я, а, б, .., я}, состоящий из 129 элементов (10 цифр, пробел, по 26 строчных и прописных букв латинского алфавита, по 33 строчных и прописных букв русскаго алфавита). Вместо обычного ASCII-кода будем использовать в качестве кода символа его позицию в приведенном выше

перечислении: цифры будут иметь коды от 0 до 9, пробел - 10, латинская “A” - 11, латинская “a” - 37, и т.д. до русской “я” с кодом 128. В качестве параметров датчика ПСЧ выберем нечетное C=13 и A=17 (такое, что $A \bmod 4 = 1$). Выберем произвольное порождающее число $T(0)$, например, 22. Выберем B=129; разумеется, получаемая последовательность будет иметь период повторения существенно меньший, чем у последовательности, получаемой при $B=2^{32}$, однако для удобства и в демонстрационных целях мы примем именно такое значение.

Допустим, что необходимо зашифровать открытый текст X = “Произвольный блок открытого текста”. Используя указанную выше кодировку, получим следующую последовательность:

```
79 113 111 105 104 98 111 108 125 110 124 106 10
97 108 111 107 10 111 115 107 113 124 115 111 99
111 10 115 101 107 114 115 96,
```

где код 79 соответствует символу “П”, код 113 - символу “р”, и т.д. Длина этого открытого текста составляет 34 символа.

Используя порождающее число 22 и указанные параметры датчика ПСЧ, получим 34 элемента гаммы:

```
22 0  13 105 121 6  115 33 58 96 97 114 16 27 85
39 31 24 34 75 127 108 43 99 19 78 49 72 76  15 10
54 28 102
```

Теперь наложим полученную гамму на открытый текст, используя операцию сложения по модулю 129 (так как в нашем алфавите 129 элементов). Другими словами, если сумма i-го символа открытого текста и i-го элемента гаммы будет превосходить 128, то из полученного числа следует вычесть модуль - 129. В результате этой операции мы получим следующий шифртекст:

101 113 124 81 96 104 97 12 54 77 92 91 26 124 64 21 9 34 16 61 105 92 38 85 1 48 31 82 62 116 117 39 14 69,

или: “ерыСазбBrНЬЫРыБК9XFуиЬbX1lUTzуфcDЁ”.

Для дешифрации надо вычесть ту же гамму из шифртекста; при этом, если разность *i*-го символа шифртекста и *i*-го элемента гаммы даст отрицательное число, то к нему следует прибавить 129:

79 113 111 105 104 98 111 108 125 110 124 106 10 97 108 111 107 10 111 115 107 113 124 115 111 99 111 10 115 101 107 114 115 96,

что соответствует исходному тексту.

Слабым местом линейного шифрования является простота определения гаммы, а затем, если необходимо, то и закономерностей ее генерации, по открытому тексту и криптограмме: очевидно, что из Y=X⊕K вытекает K=X⊕Y.

Индивидуальные задания

Для всех вариантов алфавит задается при помощи расширенной таблицы ASCII (размер - 256 элементов, с кодами от 0 до 255), параметр *B*=256. Параметры датчика ПСЧ выбираются студентами из нижеприведенной таблицы в соответствии со своими номерами по списку (по модулю 10).

Задание			
№ п.п.	*A*	*C*	*T(0)*
1	5	51	13
2	9	49	24
3	13	43	37
4	17	39	41
5	21	33	20

Задание			
№ п.п.	*A*	*C*	*T(0)*
6	25	37	7
7	29	27	16
8	33	23	21
9	37	11	30
10	41	13	11

Порядок выполнения

1. Изучить основы шифрования данных методом перестановки.

2. В соответствии с индивидуальным заданием разработать алгоритм и написать программу, реализующую датчик ПСЧ, обеспечивающую ввод произвольного открытого текста, вывод шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы. Кроме того, программа должна осуществлять вывод гаммы в виде последовательности ASCII-кодов элементов гаммы.

Содержание отчета

1. Цель работы.

2. Индивидуальное задание.

3. Текст программы, реализующей индивидуальное задание.

4. Пример открытого текста длиной не менее 50 символов, полученную шифрограмму и сгенерированную последовательность (гамму) соответствующей длины.

5. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №4
“КЛАССИЧЕСКИЙ КРИПТОГРАФИЧЕСКИЙ АЛГОРИТМ DES”

Цель работы

Целью работы является знакомство с классическим криптографическим алгоритмом DES и его работой в режиме ECB.

Основные сведения

Ни один из изученных в предыдущих лабораторных работах методов шифрования не отвечает требованиям, предъявляемым к серьезным криптоалгоритмам. В связи с этим для достижения потребного уровня криптостойкости реальные алгоритмы шифрования, применяемые в информационно-вычислительных сетях общего пользования, включают многократно повторяемые шаги перестановки (транспозиции), гаммирования и нелинейной подстановки. Классическим примером такого подхода является криптографический алгоритм DES (Data Encryption Standard).

Существуют четыре режима работы DES:

– ECB (Electronic Code Book), электронный кодоблокнот;

– CBC (CipherBlock Chaining), сцепление блоков шифра;

– CFB (Cipher FeedBack), обратная связь по шифртексту;

– OFB (Output FeedBack), обратная связь по выходу.

Настоящая лабораторная работа посвящена изучению работы криптоалгоритма DES в режиме ECB.

Режим ECB является базовым. Для него характерны разбиение открытого текста на блоки по 64 бита и их независимое шифрование при помощи одного и того же 64-битового ключа. Блок шифртекста, как и соответствующий ему блок открытого текста, имеет длину 64 бита. Для шифрования используются 56 бит ключа из 64; оставшиеся 8 бит используются для контроля.

На рис.1 приведена обобщенная структурная схема алгоритма шифрования DES.

Результатом шифрования 64-битного блока открытого текста ОТ при помощи ключа K является 64-битный шифртекст ШТ, что принято обозначать равенством ШТ=DES(ОТ, K).

Алгоритм DES функционирует следующим образом. Вначале выполняется так называемая начальная перестановка IP входного блока ОТ. Логика этой перестановки задается при помощи таблицы (см. табл. 1), где $ОТ_i$ обозначает *i*-й бит входного блока ОТ, а $IP(ОТ_i)$ - номер позиции, в которую поступает *i*-й бит входного блока в результате перестановки.

Таблица 1

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

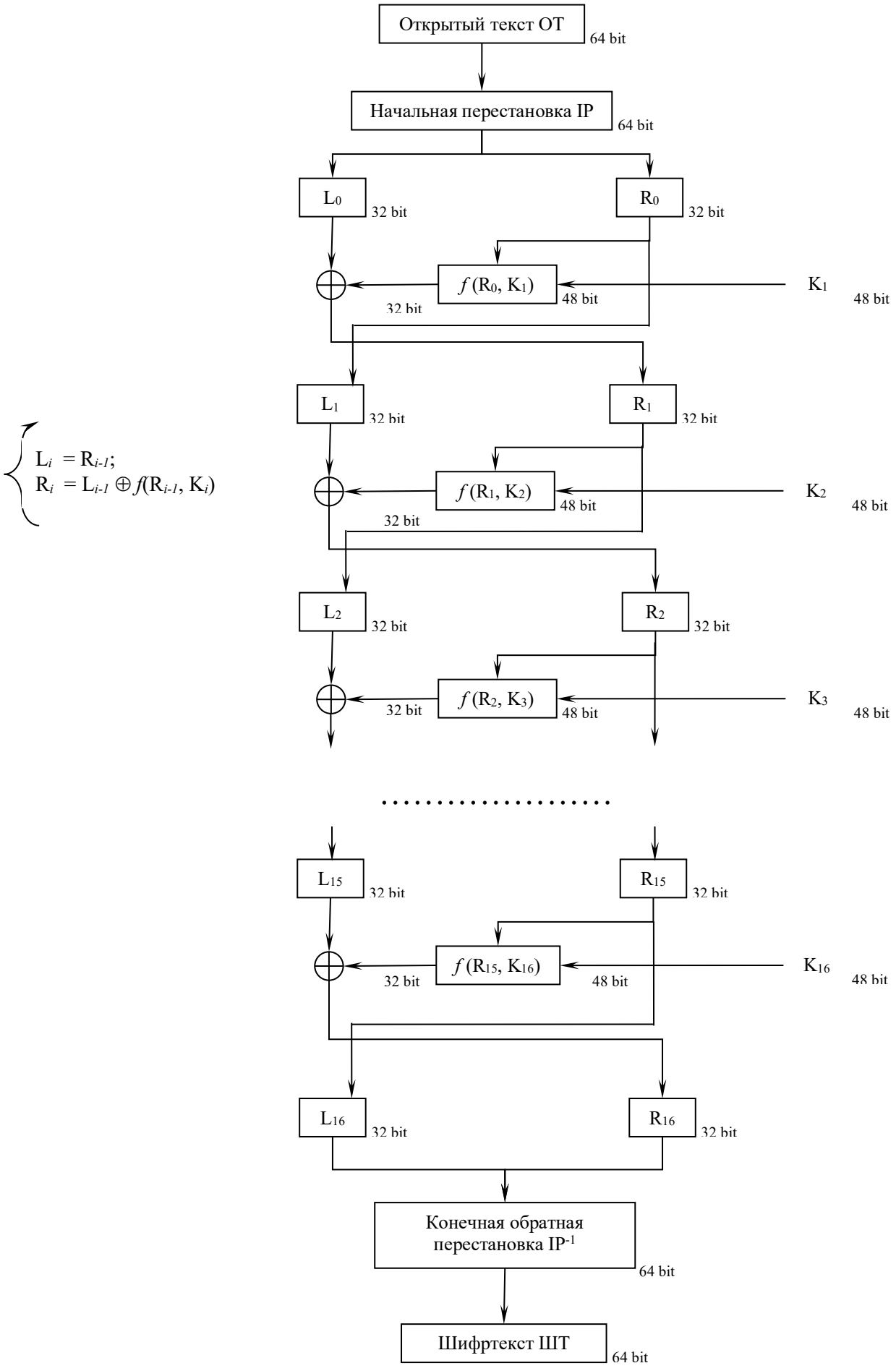

Рис. 1 Структурная схема алгоритма шифрования DES.

После начальной перестановки выполняются 16 циклов шифрующих преобразований, подробно описываемых ниже, после чего следует завершающая обратная перестановка, логика которой представлена в таблице 2, где B_i обозначает *i*-й бит шифртекста, полученного в результате выполнения упомянутых 16-ти циклов, а $IP^{-1}(B_i)$ - номер позиции, в которую поступает *i*-й бит указанного блока в результате перестановки. Блок $ШТ=IP^{-1}(B_i)$ есть не что иное, как DES(OT,K).

Таблица 2

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

Перейдем к рассмотрению шифрующих преобразований, выполняемых внутри каждого из 16 циклов. Через L_{i-1} и R_{i-1}, i=1..16, обозначаются, соответственно, левые и правые 32-битные полублоки блока, являющегося входным для *i*-го цикла. При этом L_i и R_i определяются по следующим правилам:

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

где $f(R_{i-1}, K_i)$ есть 32-битный полублок, а операция сложения по модулю 2 выполняется над битами, находящихся в идентичных позициях обоих операндов.

Логика вычисления значений $f(R_{i-1}, K_i)$ представлена в виде структурной схемы, приведенной на рис. 2.

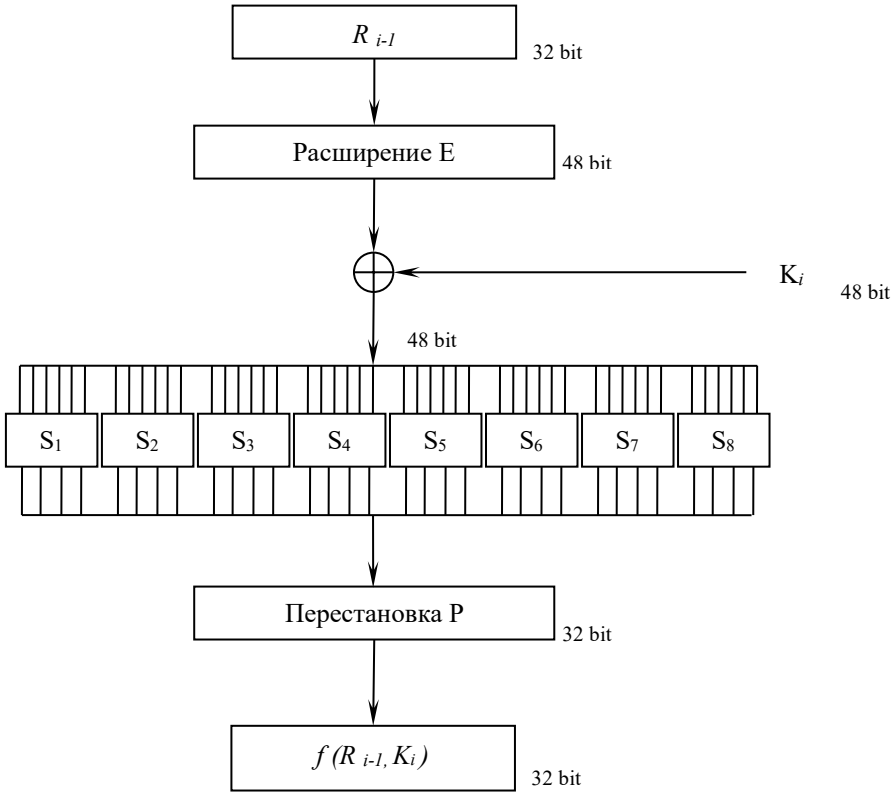

Рис. 2 Схема вычислений значений $f(R_{i-1}, K_i)$.

Вычисление начинается с так называемого расширения E, преобразующего 32-битный полублок в 48-битный вектор в соответствии с таблицей E (табл. 3), в которой *i* обозначает номер бита в 32-битном полублоке R, а E(*i*) - номер позиции в 48-битном векторе, куда поступает *i*-й бит полублока R. В случае, когда *i*-й бит "размножается" в две позиции, их номера указываются в соответствующей ячейке таблицы через запятую.

Таблица 3

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E(i)	2,48	3	4	5,7	6,8	9	10	11,13	12,14	15	16	17,19	18,20	21	22	23,25

i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
E(i)	24,26	27	28	29,31	30,32	33	34	35,37	36,38	39	40	41,43	42,44	45	46	1,47

Полученные 48 бит гаммируются с 48-битным вектором K, логика формирования которого рассматривается ниже.

Результат гаммирования разбивается на восемь 6-битовых векторов, поступающих на входы так называемых S-блоков, $S_1..S_8$. Каждый S-блок выдает 4-битовый вектор, причем взаимосвязь между входными и выходными данными определяется содержимым приводимых ниже таблиц (табл. 4) и правилами их использования, существо которых состоит в следующем.

Пусть $b_1..b_6$ - 6-битовый вектор, поступающий на вход некоторого S-блока S_i. Образуем два десятичных числа k и l, двоичная запись первого из которых есть b_1b_6, а второго - $b_2b_3b_4b_5$.

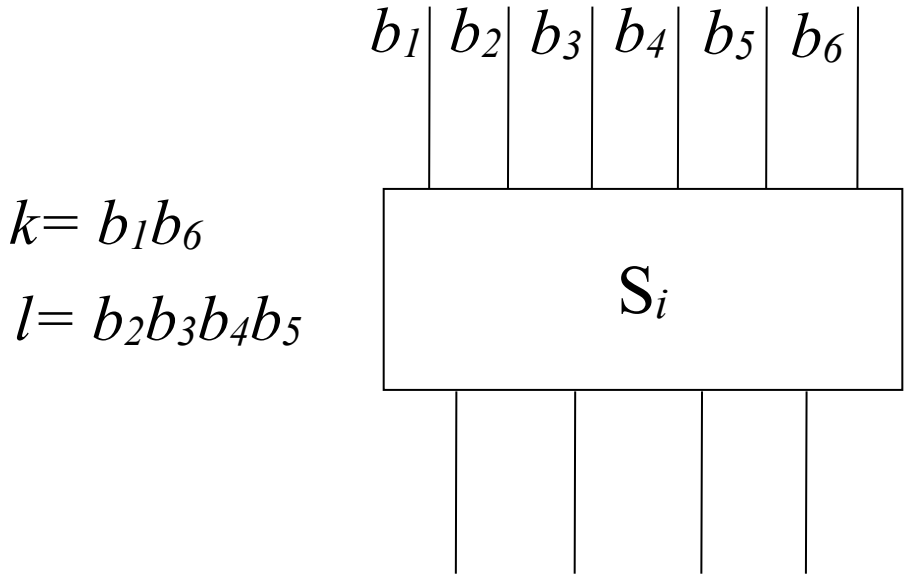

Рис. 3 S-блок.

Например, применительно к вектору 000110B: k=0, l=3. Тогда результатом работы S-блока S_i, соответствующим входу $b_1..b_6$, является 4-битовая двоичная запись десятичного числа, стоящего на пересечении k-й строки и l-го столбца i-й матрицы (см. табл. 3.2.4). Например, при поступлении вектора 000110B на вход блока S_4 результатом будет 3D=0011B; результатом работы блока S_6 при поступлении вектора 110001B, k=3, l=8, будет 11D=1011B.

Таблица 4.

$S_i\downarrow$	$l\rightarrow$ / $k\downarrow$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S_1	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
S_2	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
	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
S_3	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
	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
S_4	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
	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
S_5	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
	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
S_6	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
	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
S_7	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
	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
S_8	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
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	13	5	8
	3	2	1	14	7	4	10	8	13	5	12	9	0	3	5	5	11

Восемь 4-битовых векторов, поступающих с выходов S-блоков, образуют 32-битовый полублок, который подвергается перестановке, определяемый

таблицей P (табл. 5), в которой по аналогии с ранее приведенными таблицами, P(*i*) есть номер позиции, в которую попадает *i*-й бит входного полублока.

Таблица 5

P(i)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I	16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10

P(i)	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
I	2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Рассмотрим теперь логику формирования ключей K_i, i=1..16, используемых для гаммирования в ходе выполнения соответствующих циклов шифрования.

Исходный 64-битовый ключ K разбивается на восемь 8-битовых блоков. Восьмой бит каждого блока служит для контроля четности, который позволяет выявлять ошибки типа искажения нечетного числа битов (подобные ошибки могут возникнуть при хранении или непосредственно в процессе формирования ключей). При этом значение указанного восьмого бита представляет собой результат сложения по модулю 2 предшествующих семи битов. Контрольные биты в процессе формирования ключей не используются.

Структурная схема алгоритма формирования ключей приведена на рис. 4.

Вначале выполняется операция перестановки выбора битов исходного ключа K, существо которой определяется таблицей 6, при этом контрольные биты ключа K в формируемый 56-битовый ключ не переносятся, а биты, используемые для шифрования, переставляются в соответствии с указанной таблицей, где $PC_1(j)$ означает номер позиции, в которую поступает j-й бит исходного ключа.

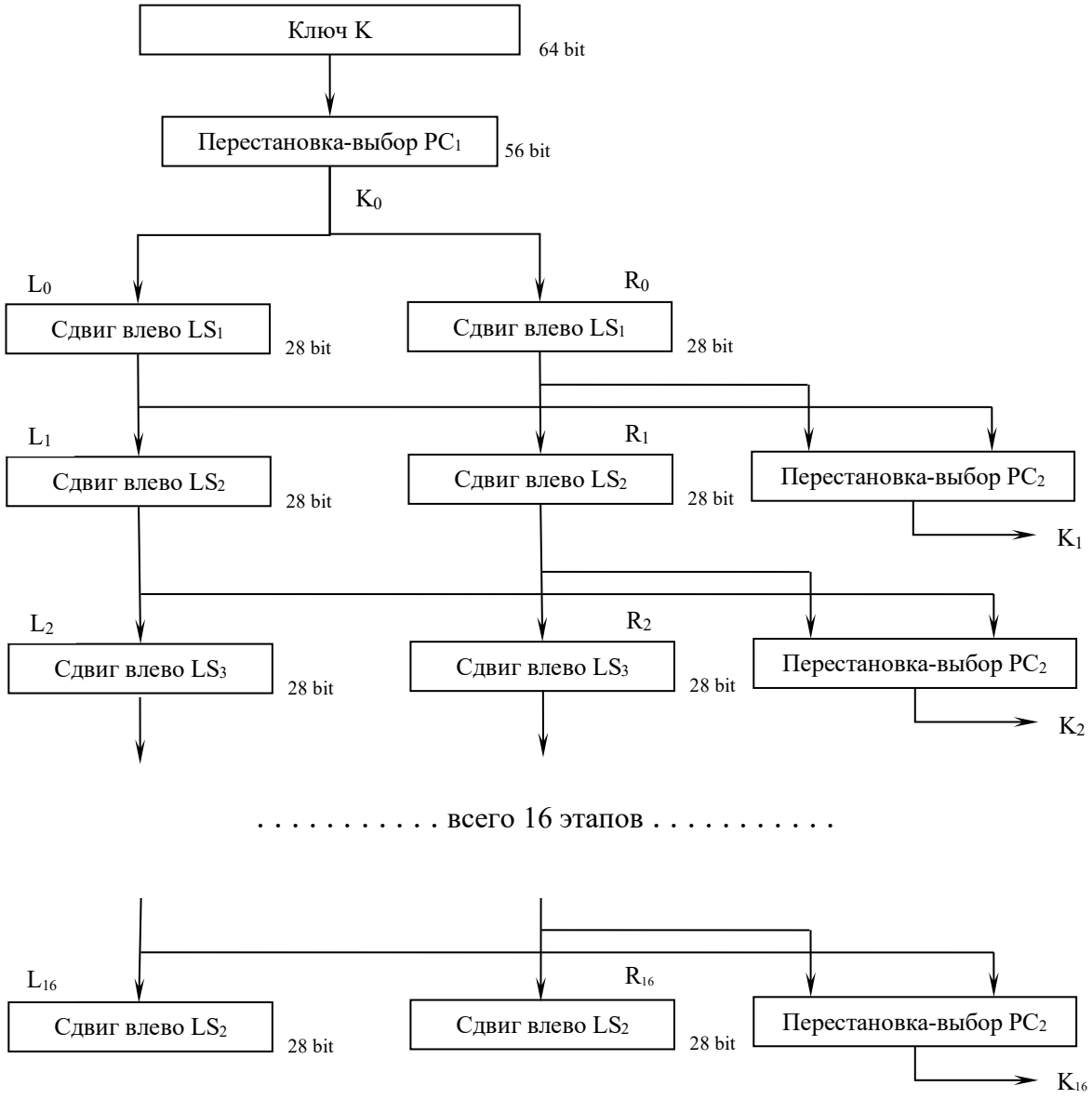

Рис. 4 Схема формирования ключей.

После выполнения перестановки-выбора PC_1 полученный 56-битовый вектор K разбивается на два 28-битовых вектора L_0 и R_0. Далее в каждом из 16 циклов над 28-битовыми векторами L_{i-1} и R_{i-1}, полученными на предыдущем цикле, выполняется операция циклического сдвига на LS_i позиций влево; значения LS_i приведены в таблице 7.

Таблица 6

$PC_1(j)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14

j	57	49	41	33	25	17	9	1	58	50	42	34	26	18

$PC_1(j)$	15	16	17	18	19	20	21	22	23	24	25	26	27	28
j	10	2	59	51	43	35	27	19	11	3	60	52	44	36

$PC_1(j)$	29	30	31	32	33	34	35	36	37	38	39	40	41	42
j	63	55	47	39	31	23	15	7	62	54	46	38	30	22

$PC_1(j)$	43	44	45	46	47	48	49	50	51	52	53	54	55	56
j	14	6	61	53	45	37	29	21	13	5	28	20	12	4

Таблица 7

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

Полученные в результате выполнения указанной операции 28-битовые векторы L_i и R_i при помощи конкатенации объединяются в 56-битовый вектор, который подвергается перестановке-выбору PC_2. В результате перестановки PC_2 j-й бит 56-битового вектора поступает в $PC_2(j)$-ую позицию формируемого 48-битового вектора, который и есть ключ K_i, используемый при шифровании (рис. 1). Логика перестановки-выбора PC_2 определяется таблицей 8.

Таблица 8

$PC_2(j)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j	14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4

$PC_2(j)$	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
j	26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40

$PC_2(j)$	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
J	51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

Алгоритм дешифрования двоичных блоков, зашифрованных при помощи DES, обозначаемый через DES^{-1}, достаточно очевиден: достаточно “прогнать” DES с тем же ключом в обратном направлении. При этом:

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

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

$i = 16, .., 1$

Таковы основные особенности алгоритма шифрования DES. Следует отметить, что криптостойкость DES определяется, главным образом, нелинейностью S-блоков. Однако логика, в соответствии с которой разработаны S-блоки, авторами DES не раскрыта и до сих пор неизвестна пользователям. В то же время исследование статистических свойств указанных блоков показывает, что они не являются полностью случайными. В связи с этим, среди специалистов распространена точка зрения о возможности существования “лазеек”, так называемых «слабых ключей», разрушающих DES.

Слабыми ключами, применительно к DES, называются такие ключи, при которых повторное шифрование шифртекста с тем же ключом дает в результате исходный открытый текст: **DES(K, DES(K, OT)) = OT**.

Всего известны 4 слабых ключа, они приведены в таблице 9.

Таблица 9

Слабые ключи (hex)	L_0	R_0
0101 0101 0101 0101	$[0]_{28}$	$[0]_{28}$
FEFE FEFE FEFE FEFE	$[1]_{28}$	$[1]_{28}$
1F1F 1F1F 0E0E 0E0E	$[0]_{28}$	$[1]_{28}$
E0E0 E0E0 F1F1 F1F1	$[1]_{28}$	$[0]_{28}$

Обозначения $[0]_{28}$ и $[1]_{28}$ означают векторы из 28 нулей и единиц, соответственно. L_0 и R_0 – левый и правый 28-битовые полублоки, получаемые после перестановки-выбора PC_1 (см. схему формирования ключей на рис. 4).

Для каждого случайного ключа существуют 2^{32} «неподвижные точки», для которых **DES(K, OT) = OT**.

Известны также шесть пар частично-слабых ключей. Частично-слабыми (в англоязычной литературе применяется термин *semi-weak*, *полуслабые*)

ключами называются такие пары ключей (K1, K2), для которых **DES(K1, DES(K2, OT) = OT**.

Другим объектом критики DES является относительно малая длина ключа (56 бит), которая, в принципе, делает реальной так называемую “силовую атаку” на DES путем организации полного перебора всего пространства ключей. Тем не менее, вычислительных ресурсов современных ЭВМ, в том числе структур с высокой степенью параллельности (систолических массивов, конвейеров, транспьютеров и т.п.), для реализации подобной атаки за практически приемлемое время пока недостаточно.

Рассмотренный нами режим ECB обладает тем недостатком, что результаты шифрования одинаковых блоков при помощи одного и того же ключа совпадают. С учетом того, что длина блока криптограммы постоянна и относительно невелика, наличие в шифртексте идентичных блоков служит явным указанием на присутствие совпадающих блоков и в соответствующих фрагментах открытого текста, что, в общем случае, может существенно облегчить криптоанализ.

С другой стороны, наблюдение за информационным обменом между двумя абонентами применительно к ситуации, когда отправитель вторично посылает открытый текст, зашифрованный одним и тем же ключом, позволяет имитировать возврат ему той же криптограммы-ответа, который был передан получателем в первом случае. Как видно, при этом DES не обеспечивает имитостойкости.

Наконец, алгоритм DES в режиме ECB обладает свойством так называемого размножения ошибок, которое состоит в том, что искажение одного бита криптограммы приводит к искажению нескольких битов в открытом тексте, полученном в результате дешифрования.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы шифрования и дешифрования DES в режиме ECB. Различные индивидуальные задания не предусматриваются. Язык программирования студент выбирает самостоятельно.

Порядок выполнения

1. Изучить основы шифрования данных при помощи алгоритма DES в режиме ECB.

2. Разработать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

3. Продемонстрировать недостаток режима ECB, заключающийся в размножении ошибок, следующим образом. Изменить в полученной шифрограмме 1 бит. {*Это можно сделать, например, заменой какого-либо символа на соседний, по таблице ASCII, символ. Например, символ "S" имеет ASCII-код* 83_{10} *(0x53, или 01010011 в двоичной системе счисления). Соседний символ "R" имеет код* 82_{10} *(0x52, или в двоичной системе 01010010). Разумеется, можно изменить любой один бит криптограммы.*} Выполнить дешифрацию и сравнить полученный открытый текст с первоначальным открытым текстом.

4. Продемонстрировать работу алгоритма на слабых ключах.

Содержание отчета

1. Цель работы.
2. Текст программы.
3. Пример открытого текста и соответствующей ему шифрограммы.
4. Шифрограмма с искаженным битом и соответствующий искаженный открытый текст.
5. Пример открытого текста и соответствующей шифрограммы, полученной при использовании слабого ключа.
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №5
“РАБОТА АЛГОРИТМА DES В РЕЖИМЕ CBC”

Цель работы

Целью работы является знакомство с особенностями работы алгоритма DES в режиме CBC.

Основные сведения

Для нейтрализации присущих режиму ECB недостатков, изученных в предыдущей лабораторной работе, применяется специальный алгоритм использования DES, называемый CBC (CipherBlock Chaining, сцепление блоков шифра). Обобщенная схема функционирования DES в режиме CBC приведена на рис. 5.

Примем, что открытый текст разбит на n 64-битовых блоков $M_1..M_n$. Через C_0 обозначим некоторый начальный 64-битовый вектор, загружаемый на начальных шагах шифрования и дешифрования в соответствующие буферы.

Далее для любого $i = 0..n$ логика шифрования определяется соотношением:

$$C_{i+1} = DES(K, M_{i+1} \oplus C_i),$$

а логика дешифрования - соотношением:

$$M_{i+1} \oplus C_i = DES^{-1}(K, C_{i+1}),$$

или, что то же самое:

$$M_{i+1} = C_i \oplus DES^{-1}(K, C_{i+1}).$$

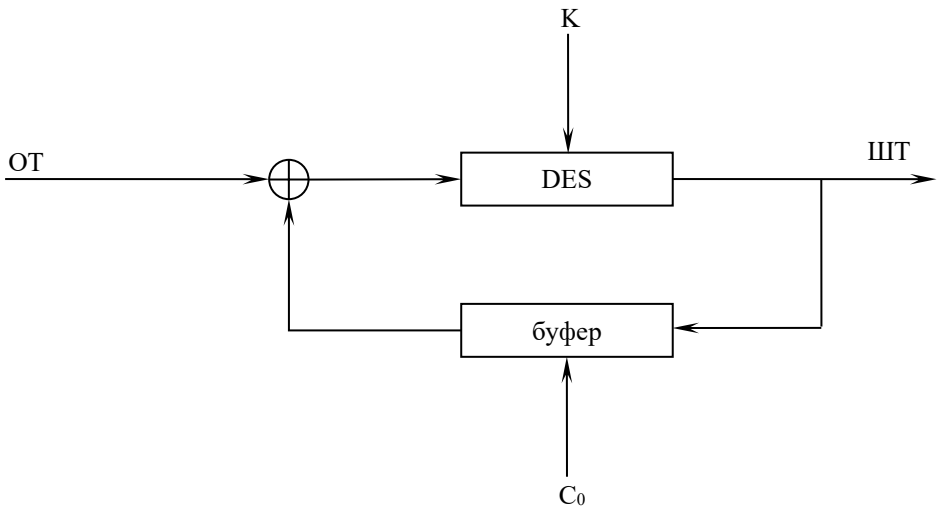

Рис.5 а) Шифрование DES в режиме CBC.

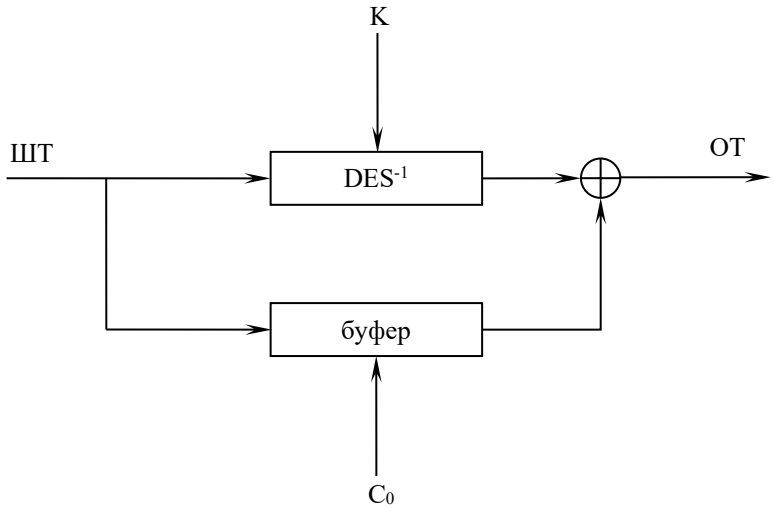

Рис. 5 б) Дешифрование DES в режиме CBC.

Как следует из приведенных соотношений, каждый очередной блок криптограммы является функцией предшествующего. Поэтому искажение одного бита в блоке шифртекста искажает два блока, полученных в результате дешифрования. Однако поскольку искаженный блок криптограммы в процессе дешифрования гаммируется со следующим блоком открытого текста, число искажений в следующем дешифруемом блоке равно числу искажений в шифртексте.

Что касается криптостойкости, то режим CBC считается более стойким, нежели ECB, по двум причинам. Во-первых, шифртекст является функцией не только открытого текста и ключа, но и, в конечном итоге, начального вектора C_0. Это, естественно, усложняет криптоанализ. С другой стороны, идентичным блокам открытого текста в общем случае соответствуют различные блоки криптограммы, что нейтрализует основной недостаток ECB.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы шифрования и дешифрования DES в режиме CBC. 64-битовый начальный вектор C_0 следует сгенерировать, используя датчик ПСЧ, разработанный в лабораторной работе №3.

Порядок выполнения

1. Изучить основы шифрования данных при помощи алгоритма DES в режиме CBC.

2. Разработать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

3. Изменить в полученной шифрограмме 1 бит. Выполнить дешифрацию и сравнить полученный открытый текст с первоначальным открытым текстом.

4. Подать на вход алгоритма открытый текст, содержащий два идентичных 64-битовых блока. Сравнить соответствующие блоки шифрограммы.

Содержание отчета

1. Цель работы.

2. Текст программы.

3. Пример открытого текста и соответствующей ему шифрограммы.

4. Шифрограмма с искаженным битом и соответствующий искаженный открытый текст.

5. Открытый текст с идентичными блоками и соответствующая шифрограмма. (Возможно совмещение этого пункта с п.3.)

6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №6
“РАБОТА АЛГОРИТМА DES В РЕЖИМЕ CFB”

Цель работы

Целью работы является знакомство с особенностями работы алгоритма DES в режиме CFB.

Основные сведения

Наряду с режимами ECB и CBC, алгоритм DES используется еще в двух режимах с так называемой обратной связью.

Обобщенная структурная схема использования алгоритма DES в режиме обратной связи по шифртексту (CFB, Cipher FeedBack) с *k*-битовыми блоками приведена на рис. 6.

В отличие от режимов ECB и CBC, оперирующих 64-битовыми блоками, режимы CFB и OFB оперируют *k*-битовыми блоками ($1<k<63$). Это, в частности, позволяет шифровать текстовые данные посимвольно (для кода EBCDIC достаточно положить $k=8$, для кода ASCII - $k=7$).

Шифрование в режиме CFB происходит следующим образом. Вначале на вход поступает случайным образом сгенерированный 64-битовый вектор C_0, который шифруется алгоритмом DES с некоторым ключом K.

Левые *k* бит результата шифрования гаммируются с левыми *k* битами открытого текста и полученный таким образом *k*-битовый шифртекст поступает на дальнейшую обработку (передачу по каналу связи, хранение и т.п.).

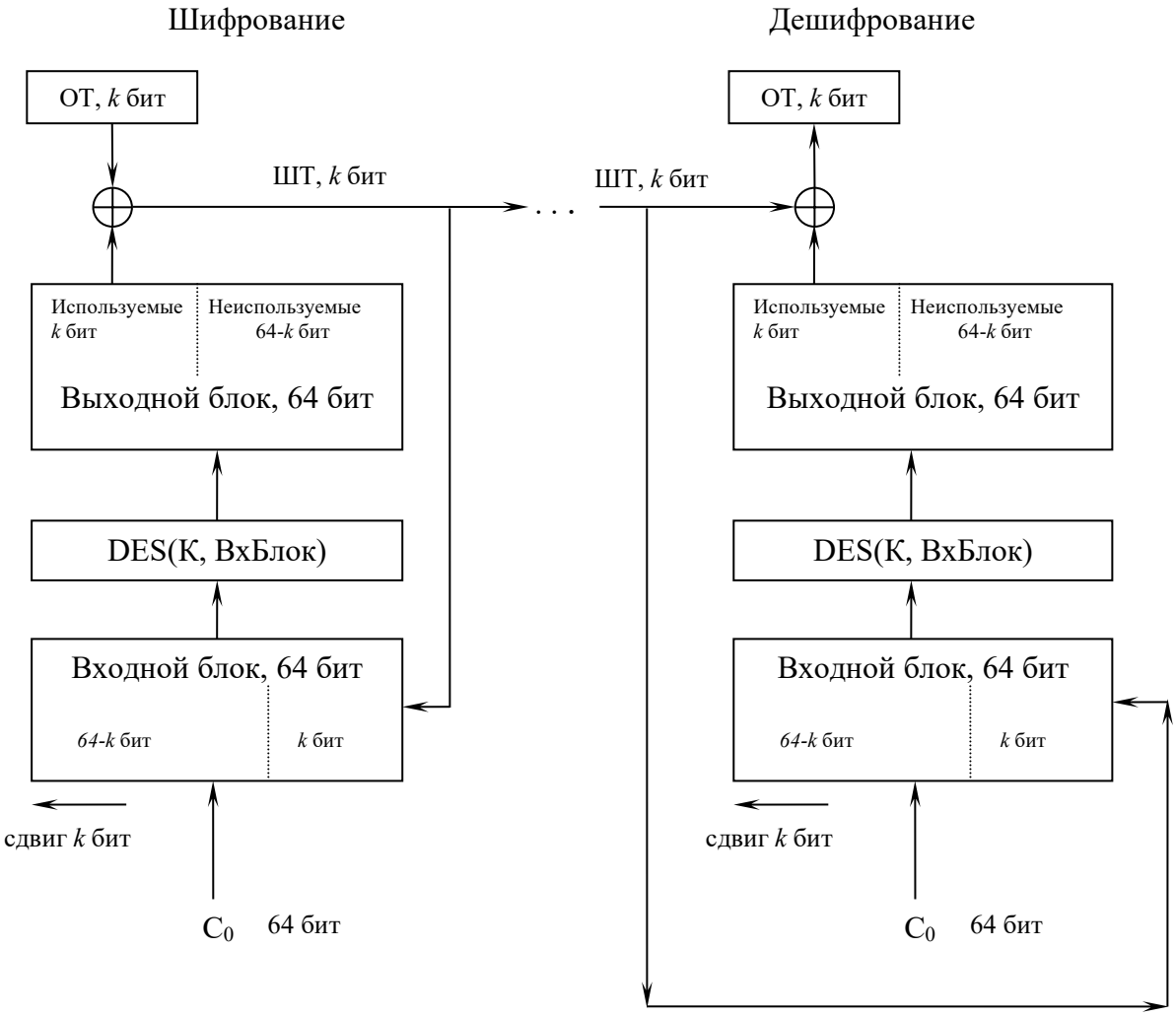

Рис. 6 Шифрование DES в режиме CFB.

Существенно, что k-битовый вектор, гаммируемый с k битами открытого текста, загружается в k правых блоков входного блока посредством сдвига его содержимого на k позиций влево, после чего описанные операции повторяются применительно к следующим k битам открытого текста и т.д. Обратим внимание на то, что упомянутый сдвиг содержимого входного блока не является циклическим, в связи с чем левые k бит этого содержимого на каждом шаге “выталкиваются” за пределы блока.

Процесс дешифрования в режиме CFB аналогичен описанному и поэтому очевиден из правой половины рис. 6.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы шифрования и дешифрования DES в режиме CFB. 64-битовый начальный вектор C_0 следует сгенерировать, используя датчик ПСЧ, разработанный в лабораторной работе №3.

Порядок выполнения

1. Изучить основы шифрования данных при помощи алгоритма DES в режиме CFB.

2. Разработать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

3. Изменить в полученной шифрограмме 1 бит. Выполнить дешифрацию и сравнить полученный открытый текст с первоначальным открытым текстом.

4. Подать на вход алгоритма открытый текст, содержащий два идентичных 64-битовых блока. Сравнить соответствующие блоки шифрограммы.

Содержание отчета

1. Цель работы.
2. Текст программы.
3. Пример открытого текста и соответствующей ему шифрограммы.
4. Шифрограмма с искаженным битом и соответствующий искаженный открытый текст.
5. Открытый текст с идентичными блоками и соответствующая шифрограмма. (Возможно совмещение этого пункта с п.3.)
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №7
“РАБОТА АЛГОРИТМА DES В РЕЖИМЕ OFB”

Цель работы

Целью работы является знакомство с особенностями работы алгоритма DES в режиме OFB.

Основные сведения

Наряду с режимами ECB и CBC, алгоритм DES используется еще в двух режимах с так называемой обратной связью.

Обобщенная структурная схема использования алгоритма DES в режиме обратной связи по выходу (OFB, Output FeedBack) с k-битовыми блоками приведена на рис. 7.

В отличие от режимов ECB и CBC, оперирующих 64-битовыми блоками, режимы CFB и OFB оперируют k-битовыми блоками ($1<k<63$). Это, в частности, позволяет шифровать текстовые данные посимвольно (для кода EBCDIC достаточно положить $k=8$, для кода ASCII - $k=7$).

Шифрование в режиме OFB происходит во многом аналогично шифрованию в режиме CFB. Отличие режима OFB от режима CFB заключается в том, что k-битовый вектор обратной связи поступает в k правых битов входного блока не после гаммирования, а до него.

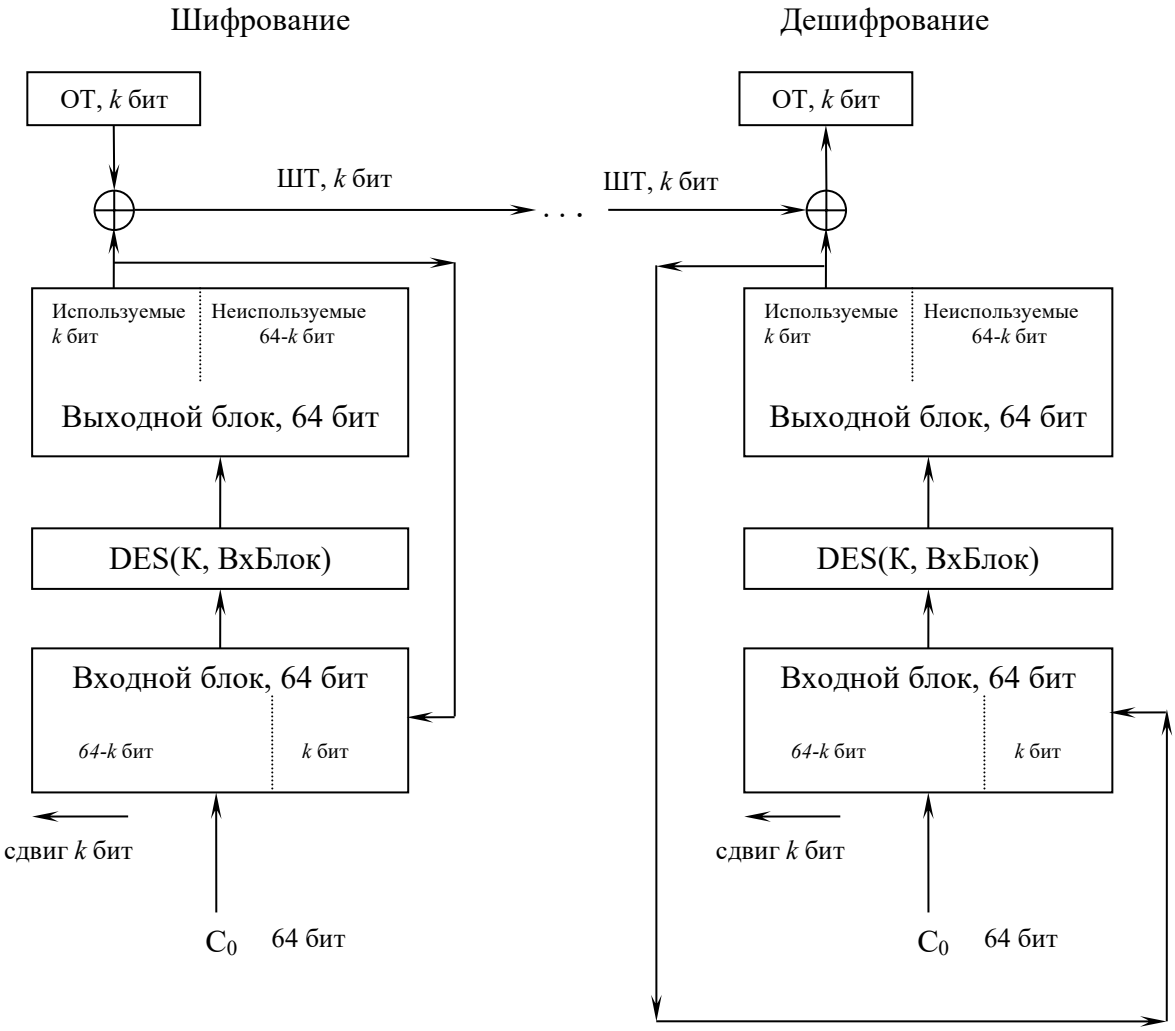

Рис. 7 Шифрование DES в режиме OFB.

Иными словами, указанный вектор еще не является блоком шифртекста и, таким образом, не зависит от открытого текста. При этом DES как бы “зацикливается” сам на себя, генерируя *k*-битовые ключи для гаммирования с соответствующими блоками открытого текста. Это обстоятельство, как считается, снижает криптостойкость OFB в сравнении с CFB. В самом деле, в режиме OFB алгоритм DES используется, по существу, в качестве генератора псевдослучайных последовательностей 64-битовых двоичных чисел. Общеизвестно, что любой такой генератор имеет период повторения N, такой, что N-ым вектором на его выходе будет вектор C_0, с которого началась генерация. После этого генератор будет циклически выдавать ту же

последовательность ключей, что была использована для шифрования первых N k-битовых блоков открытого текста. Впрочем, теоретически максимальный период повторения для DES-генератора, равный 2^{64}, применительно к доступным в настоящее время вычислительным ресурсам, по-видимому, достаточен, чтобы обеспечить приемлемую криптостойкость. Однако реальный период повторения подобных генераторов, как правило, существенно меньше максимального (в частности, по некоторым оценкам, этот период для DES находится в районе 2^{32}).

Процесс дешифрования в режиме OFB аналогичен описанному и поэтому очевиден из правой половины рис. 7.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы шифрования и дешифрования DES в режиме OFB. 64-битовый начальный вектор C_0 следует сгенерировать, используя датчик ПСЧ, разработанный в лабораторной работе №3.

Порядок выполнения

1. Изучить основы шифрования данных при помощи алгоритма DES в режиме OFB.

2. Разработать программу, обеспечивающую ввод произвольного открытого текста и выдачу шифрограммы, полученную изучаемым методом, а также дешифрацию - получение открытого текста из шифрограммы.

3. Изменить в полученной шифрограмме 1 бит. Выполнить дешифрацию и сравнить полученный открытый текст с первоначальным открытым текстом.

4. Подать на вход алгоритма открытый текст, содержащий два идентичных 64-битовых блока. Сравнить соответствующие блоки шифрограммы.

Содержание отчета

1. Цель работы.
2. Текст программы.
3. Пример открытого текста и соответствующей ему шифрограммы.
4. Шифрограмма с искаженным битом и соответствующий искаженный открытый текст.
5. Открытый текст с идентичными блоками и соответствующая шифрограмма. (Возможно совмещение этого пункта с п.3.)
6. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №8
“МЕТОД ИЗМЕНЕНИЯ ИНТЕРВАЛА МЕЖДУ ПРЕДЛОЖЕНИЯМИ”

Цель работы

Целью работы является знакомство со стеганографическим методом скрытия информации путем изменения интервала между предложениями.

Основные сведения

Основные понятия стеганографии и обобщенная структурная схема стеганографической системы приведены в тексте лекций.

Метод изменения интервала между предложениями основан на размещении одного или двух пробелов после каждого символа завершения предложения. При этом, обычно, единичным пробелом кодируется бит «1», а двойным пробелом – бит «0». Признаком завершения предложения можно считать точку, вопросительный и/или восклицательный знаки, после которых следует пробел. Скрываемое сообщение в двоичном формате встраивается в контейнер-текст путем размещения соответствующего каждому биту числа пробелов после каждого предложения.

Поскольку контейнер имеет существенно больший объем, чем сообщение, протокол стеганографического обмена должен предусматривать некоторый маркер завершения сообщения. Этот маркер дописывается в конец сообщения на этапе предварительной обработки, до поступления сообщения в стеганокодер. При извлечении сообщения из контейнера, обнаружение маркера позволяет прервать обработку и не анализировать оставшуюся часть контейнера.

В качестве примера представим себе контейнер следующего вида:

С	л	о	в	а		ф	р	а	з	ы		1	.		С	л	о	в	а		ф	р	а	з
ы		2	.		С	л	о	в	а		ф	р	а	з	ы		3	.		С	л	о	в	а
	ф	р	а	з	ы		4	.		С	л	о	в	а		ф	р	а	з	ы		5	.	
С	л	о	в	а		ф	р	а	з	ы		6	!		С	л	о	в	а		ф	р	а	з
ы		7	.		С	л	о	в	а		ф	р	а	з	ы		8	?		С	л	о	в	а
	ф	р	а	з	ы		9	.		С	л	о	в	а		ф	р	а	з	ы		А	.	

Этот контейнер состоит из десяти предложений и, следовательно, может быть использован для размещения десяти бит скрываемого сообщения. Сообщение представляется в двоичном формате; каждый пробел после очередного предложения контейнера заменяется *двумя* пробелами, если очередной бит сообщения равен нулю, или *одним* пробелом (то есть, остается без изменений), если очередной бит сообщения равен единице. Пусть в нашем примере символ сообщения имеет значение **"M"**, в двоичной форме код символа **"M"** будет представлен битами ASCII("M") = 140_d = 10001100_b. Тогда заполненный контейнер будет выглядеть следующим образом:

С	л	о	в	а		ф	р	а	з	ы		1	.		С	л	о	в	а		ф	р	а	з
ы		2	.			С	л	о	в	а		ф	р	а	з	ы		3	.			С	л	о
в	а		ф	р	а	з	ы		4	.			С	л	о	в	а		ф	р	а	з	ы	
5	.		С	л	о	в	а		ф	р	а	з	ы		6	!		С	л	о	в	а		ф
р	а	з	ы		7	.			С	л	о	в	а		ф	р	а	з	ы		8	?		
С	л	о	в	а		ф	р	а	з	ы		9	.		С	л	о	в	а		ф	р	а	з
ы		А	.																					

Затемнением выделены пробелы после окончания предложений, кодирующие биты скрываемого сообщения. Эта иллюстрация также более наглядно демонстрирует необходимость добавления маркера окончания сообщения до начала стеганографического преобразования: в противном случае на конце сообщения у нас ошибочно «декодировались» бы два единичных бита из той части контейнера, которая не понадобилась для скрытия сообщения.

На практике, стеганографические системы, реализующие рассмотренный и подобные методы, осуществляют предварительную обработку пустого контейнера. В ходе предобработки из текста контейнера удаляются «лишние» пробелы, которые могли оказаться в конце предложений на этапе создания контейнера: это может произойти из-за особенностей конкретного текстового редактора, системы распознавания сканированного текста, или действий человека, набиравшего текст.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы прямого и обратного стеганографического преобразования.

Пустой и заполненный текстовые контейнеры должны задаваться пользователем в виде файлов либо через графический интерфейс – стандартный диалог выбора файла и/или поле ввода имени файла, либо через параметры командной строки. Выбор вида интерфейса (графический или командной строки) – на усмотрение студента, выполняющего работу. Аналогичным образом задается скрываемое сообщение.

Перед прямым стеганографическим преобразованием должна выполняться предварительная обработка пустого контейнера для замены двух и более пробелов на один пробел после окончания предложения, а также добавление к сообщению маркера окончания. Маркер окончания выбирается студентом самостоятельно таким образом, чтобы выбранный маркер не являлся частью исходного сообщения. До начала стеганографического преобразования предобработчик должен выводить пользователю информацию о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере.

Для простоты реализации можно принять, что сообщение начинает размещаться в контейнере с первой возможной позиции (то есть, после первого

же предложения) и размещается последовательно; при этом стеганографический ключ отсутствует.

Порядок выполнения

1. Изучить метод изменения интервала между предложениями.

2. Разработать программу, обеспечивающую реализацию поставленного задания на лабораторную работу.

3. Продемонстрировать практически корректную реализацию прямого и обратного стеганографических преобразований на тестовых примерах.

Содержание отчета

1. Цель работы.

2. Текст фрагментов программы, реализующих прямое и обратное стеганографические преобразования (полный листинг приводить не нужно).

3. Текст скрываемого сообщения и выбранный маркер окончания – в виде строки символов и в виде битовых строк.

4. Результат работы предварительного обработчика на тестовом примере: вывод информации о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере.

5. Фрагменты открытого и заполненного контейнеров (суммарно – не более одной страницы).

6. Выводы по работе

ЛАБОРАТОРНАЯ РАБОТА №9
“МЕТОД ИЗМЕНЕНИЯ КОЛИЧЕСТВА ПРОБЕЛОВ В КОНЦЕ ТЕКСТОВЫХ СТРОК”

Цель работы

Целью работы является знакомство со стеганографическим методом скрытия информации путем изменения количества пробелов в конце текстовых строк.

Основные сведения

Основные понятия стеганографии и обобщенная структурная схема стеганографической системы приведены в тексте лекций.

Изучаемый в данной лабораторной работе метод основан на добавлении пробелов в конец каждой текстовой строки. Конец строки индентифицируется, в зависимости от операционной системы, парой символов CR/LF или символом CR.

Аналогично рассмотренному в предыдущей лабораторной работе методу изменения интервала между предложениями, сообщение представляется в виде битовой последовательности; каждый бит сообщения определяет количество пробелов, дописываемых в конец каждой строки контейнера (один пробел для единичного бита, два пробела – для нулевого бита).

Приведем пример контейнера, аналогичного рассмотренному ранее, но с небольшой модификацией – в новом контейнере каждая строка завершается символом CR:

С	л	о	в	а		ф	р	а	з	ы		1	.		С	л	о	в	а	**CR**
ф	р	а	з	ы		2	.		С	л	о	в	а		ф	р	а	з	ы	**CR**
3	.		С	л	о	в	а		ф	р	а	з	ы		4	.	**CR**			
С	л	о	в	а		ф	р	а	з	ы		5	.		С	л	о	в	а	**CR**
ф	р	а	з	ы		6	.		С	л	о	в	а		ф	р	а	з	ы	**CR**
7	.		С	л	о	в	а		ф	р	а	з	ы		8	.	**CR**			
С	л	о	в	а		ф	р	а	з	ы		9	.		С	л	о	в	а	**CR**
ф	р	а	з	ы		А	.		С	л	о	в	а		ф	р	а	з	ы	**CR**

Пусть в данном примере, как и в ранее рассмотренном, символ сообщения имеет значение **"M"**, в двоичной форме код символа **"M"** будет представлен битами 10001100_b. Тогда заполненный контейнер будет выглядеть следующим образом:

С	л	о	в	а		ф	р	а	з	ы		1	.		С	л	о	в	а		**CR**	
ф	р	а	з	ы		2	.		С	л	о	в	а		ф	р	а	з	ы			**CR**
3	.		С	л	о	в	а		ф	р	а	з	ы		4	.			**CR**			
С	л	о	в	а		ф	р	а	з	ы		5	.		С	л	о	в	а			**CR**
ф	р	а	з	ы		6	.		С	л	о	в	а		ф	р	а	з	ы		**CR**	
7	.		С	л	о	в	а		ф	р	а	з	ы		8	.		**CR**				
С	л	о	в	а		ф	р	а	з	ы		9	.		С	л	о	в	а			**CR**
ф	р	а	з	ы		А	.		С	л	о	в	а		ф	р	а	з	ы			**CR**

Затемнением выделены пробелы после окончания строк, кодирующие биты скрываемого сообщения.

Очевидно, что данный метод по сравнению с ранее рассмотренным будет характеризоваться более эффективным использованием контейнера в случае

коротких строк и длинных предложений, и менее эффективным – в случае длинных строк и коротких предложений. В любом случае, данный метод обладает большей скрытностью по сравнению с предыдущим, поскольку добавляемые невидимые пробелы находятся в стороне от основного текста и не влияют на расстояние между элементами текста. Недостатком этого метода (как, впрочем, и предыдущего), является то, что некоторые редакторы текста могут автоматически убирать концевые пробелы.

Существует более совершенная модификация этого метода, основанная на применении двух разных символов пробелов. Как известно, в таблице кодов ASCII существует так называемый «обычный» пробел с десятичным кодом 32, а также «неразрывный» пробел с десятичным кодом «160». Визуально неразрывный пробел выглядит точно так же, как и обычный – то есть, пустое место. Однако, многие текстовые редакторы обрабатывают факт наличия неразрывного пробела таким образом, что по такому пробелу запрещен перенос предложения на другую строку. Другими словами, если между двумя альфанумерическими символами находится неразрывный пробел, то *оба* эти символа, вместе с пробелом между ними, должны быть расположены на одной строке.

Таким образом, биты скрываемого сообщения можно кодировать не одним или двумя (обычными) пробелами, а, например, нулевой бит – обычным пробелом, и единичный бит – неразрывным пробелом. Тогда сторонам стеганографического обмена необходимо заранее условиться о том, сколько бит сообщения будет кодироваться на одной строке контейнера.

Вернемся к ранее рассмотренному примеру, но теперь будем использовать обычный и неразрывный пробелы, и условимся, что мы будем кодировать по два бита скрываемого сообщения на строку. Тогда заполненный контейнер будет выглядеть следующим образом (напомним, что $M = 10001100_b$):

С	л	о	в	а		ф	р	а	з	ы		1	.		С	л	о	в	а	160	32	CR
ф	р	а	з	ы		2	.		С	л	о	в	а		ф	р	а	з	ы	32	32	CR
3	.		С	л	о	в	а		ф	р	а	з	ы		4	.	160	160	CR			
С	л	о	в	а		ф	р	а	з	ы		5	.		С	л	о	в	а	32	32	CR
ф	р	а	з	ы		6	.		С	л	о	в	а		ф	р	а	з	ы	CR		
7	.		С	л	о	в	а		ф	р	а	з	ы		8	.	CR					
С	л	о	в	а		ф	р	а	з	ы		9	.		С	л	о	в	а	CR		
ф	р	а	з	ы		А	.		С	л	о	в	а		ф	р	а	з	ы	CR		

Такой вариант метода позволяет существенно более эффективно использовать контейнер и скрывать в нем сравнительно больший объем конфиденциальных данных. Разумеется, необходимо руководствоваться здравым смыслом и чувством меры: если на одну строку контейнера в 80 символов будет приходиться по 1 Кб пробелов, то большой размер файла контейнера, явно не соответствующий видимому тексту, будет вызывать подозрения.

Напомним, что текстовый контейнер должен быть подвергнут предобработке, удаляющей «лишние» пробелы на концах строк. В отличие от метода изменения интервала между предложениями, использование маркера окончания сообщения в случае одиночного сообщения в контейнере не будет обязательным.

Задание на работу

В данной лабораторной работе необходимо программно реализовать алгоритмы прямого и обратного стеганографического преобразования.

Пустой и заполненный текстовые контейнеры должны задаваться пользователем в виде файлов либо через графический интерфейс – стандартный диалог выбора файла и/или поле ввода имени файла, либо через параметры командной строки. Выбор вида интерфейса (графический или командной строки) – на усмотрение студента, выполняющего работу. Аналогичным

образом задается скрываемое сообщение. Кроме того, интерфейс пользователя должен задавать режим работы алгоритма: с одним или двумя пробелами, или с «обычным» и «неразрывным» пробелом. Во втором случае должно также задаваться количество кодируемых бит на строку.

Перед прямым стеганографическим преобразованием должна выполняться предварительная обработка пустого контейнера для удаления пробелов перед окончаниями строк. До начала стеганографического преобразования предобработчик должен выводить пользователю информацию о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере при выбранном режиме работы алгоритма.

Для простоты реализации можно принять, что сообщение начинает размещаться в контейнере с первой возможной позиции (то есть, после первой же строки) и размещается последовательно; при этом стеганографический ключ отсутствует.

Порядок выполнения

1. Изучить метод скрытия информации путем изменения количества пробелов в конце текстовых строк.

2. Разработать программу, обеспечивающую реализацию поставленного задания на лабораторную работу.

3. Продемонстрировать практически корректную реализацию прямого и обратного стеганографических преобразований на тестовых примерах в режимах с кодированием битов сообщения одним и двумя пробелами, и «обычным» и «неразрывным» пробелами.

Содержание отчета

1. Цель работы.

2. Текст фрагментов программы, реализующих прямое и обратное стеганографические преобразования (полный листинг приводить не нужно).

3. Текст скрываемого сообщения – в виде строки символов и в виде битовой строки.

4. Результат работы предварительного обработчика на тестовом примере: вывод информации о максимальной емкости контейнера и возможности размещения заданного сообщения в выбранном контейнере, для обоих режимов работы.

5. Фрагменты открытого и заполненного контейнеров (суммарно – не более одной страницы), для обоих режимов работы.

6. Выводы по работе