Курс «Информационная безопасность и защита информации»

Задание 1.

Реализация алгоритма шифрования DES в режиме ECB

Задание

Создать приложение с графическим интерфейсом на языке программирования C++/C#, осуществляющее шифрование и расшифрование текста по алгоритму DES. Для простоты работы текст должен содержать символы, входящие в таблицу кодировки ASCII (латинские символы, цифры, знаки препинания, управляющие символы).

Приложение должно содержать три окна для ввода/вывода открытого текста (Plain Text) и шифротекста (Cipher Text) и ключа шифрования/расшифрования (Key).

Приложение должно работать в двух направлениях: из заданного открытого текста получать шифротекст и наоборот, из заданного шифротекста получать открытый текст. Должна быть предусмотрена возможность ввода текста в окна программы и загрузки текста из файла.

Ключ, состоящий из 64 бит (8 символов типа *unsigned char*), должен генерироваться случайно по соответствующей команде. При этом 56 бит ключа являются значащими, а биты с номерами 8, 16, 24, 32, 40, 48, 56, 64 должны быть установлены там образом, чтобы каждый байт ключа содержал нечётное число единиц.

При реализации шифрования/расшифрования исходный текст разбивается на блоки длиной 64 бит (8 символов). Если длина исходного текста не кратна размеру блока, его необходимо дополнить нулевыми битами (нуль-символами *'\0'*). В режиме «электронной цифровой книги» (ECB) каждый блок шифруется независимо от других, каждому блоку входного текста соответствует свой блок шифротекста, при этом блоки не перекрываются друг с другом, и каждый блок зашифровывается/расшифровывается с одним и тем же ключом.

При реализации алгоритма возможна работа с двумя типами данных:

- Исходный текст (массив *unsigned char*) преобразуется в массив бит, составляющих данный текст (массив *bool* или *std::vector<bool>*). Для этого код каждого символа исходного текста необходимо преобразовать в двоичную систему счисления, получив массив бит длиной 8 для данного конкретного символа. Далее работа ведется с получившимися массивами, каждый элемент которого представляет собой бит текста. После получения результата массив *bool* преобразуется обратно в массив *unsigned char* и выводится на экран или в файл.
- Работа непосредственно с исходным массивом *unsigned char*. При этом для операции с битами необходимо использовать битовые операции: !(отрицание), & (побитовое И), | (побитовое ИЛИ), ^ (побитовое исключающее ИЛИ), >> (сдвиг вправо), << (сдвиг влево).

Алгоритм шифрования данных DES

Алгоритм DES, используя комбинацию рядя подстановок и перестановок, осуществляет шифрование 64-битовых блоков данных с помощью 56-битового ключа k. Схема алгоритма DES изображена на рис. 1.

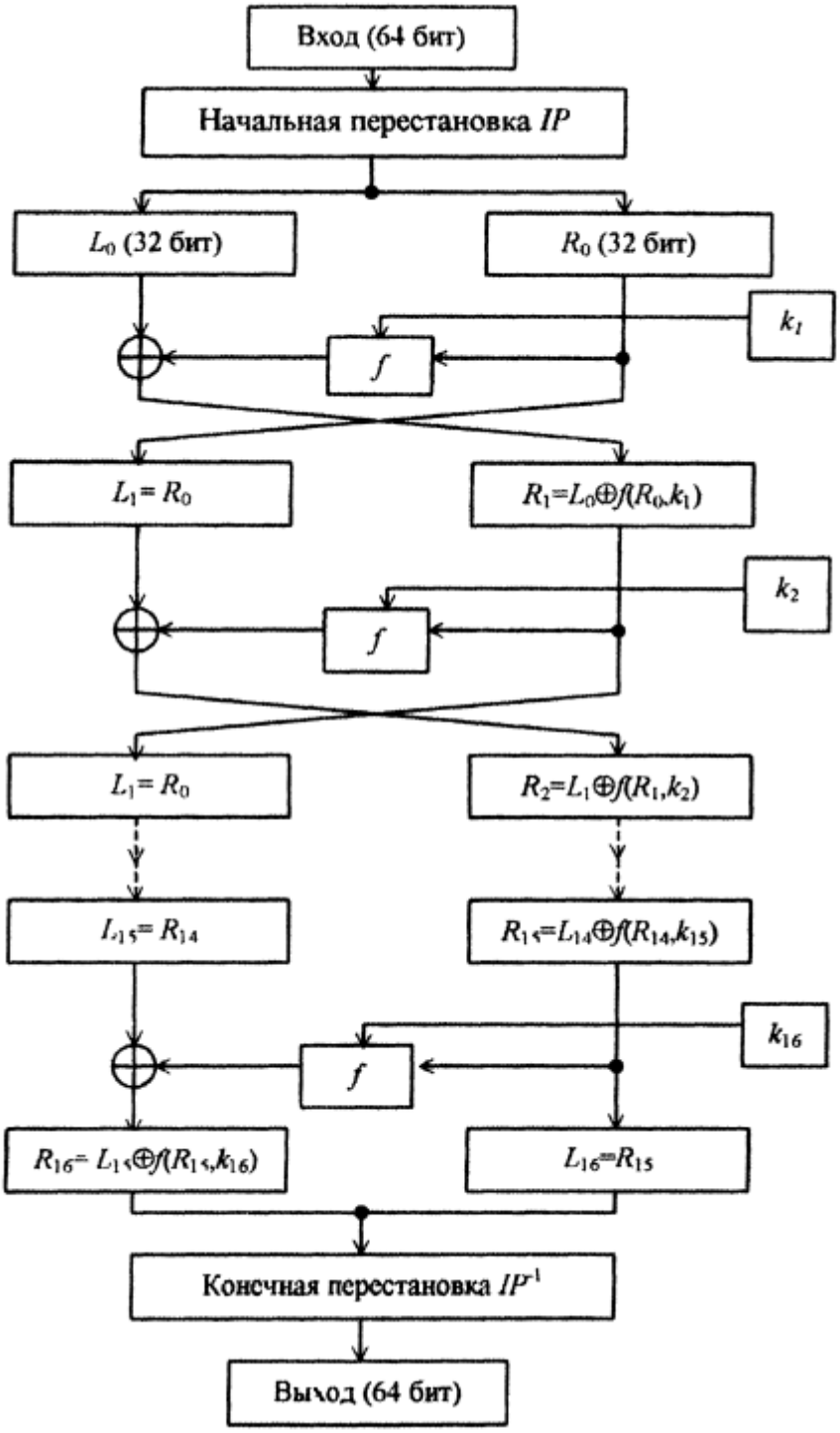

Рис. 1. Схема алгоритма DES

Процесс шифрования состоит в начальной перестановке битов входного блока, шестнадцати циклах шифрования и, наконец, конечной перестановке битов.

Отметим, что все приводимые ниже таблицы являются стандартными и должны использоваться при реализации алгоритма DES в неизменном виде. Все перестановки и

коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс вскрытия шифра путём подбора ключа.

В приводимом ниже описании алгоритма DES использованы следующие обозначения:

- L_i и R_i - левая и правая половины 64-битного блока $L_i R_i$;
- $\oplus$ - операция побитового сложения векторов-блоков по модулю 2;
- k_i - 48-битовые ключи;
- f - функция шифрования;
- IP - начальная перестановка степени 64.

При зашифровании очередного блока T его биты подвергаются начальной перестановке IP согласно табл.1.

Таблица 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

При этом бит 58 блока T становится битом 1, бит 50 – битом 2 и т.д. Полученный после перестановки блок $IP(T)$ разделяется на две половины: L_0, состоящую из 32 старших бит, и R_0, состоящую из 32 младших бит.

Затем выполняется итеративный процесс шифрования, состоящий из 16 циклов преобразования Фейстеля. Пусть $T_{i-1} = L_{i-1}R_{i-1}$ - результат $(i-1)$-й итерации.

Тогда результат i-й итерации $T_i = L_i R_i$ определяется формулами:

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i) \end{cases}, i = 1,..,16.$$

Функция f называется функцией шифрования. Её аргументами являются 32-битовый вектор R_{i-1} и 48-битовый ключ k_i, который является результатом преобразования 56-битового ключа шифра k. Результатом последней итерации является блок $T_{16} = R_{16}L_{16}$. По окончании шифрования осуществляется восстановление позиций битов применением к T_{16} обратной перестановки IP^{-1}.

При расшифровании данных все действия выполняются в обратном порядке, при этом используются соотношения:

$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus f(L_i, k_i) \end{cases}, i = 16,..,1.$$

Пользуясь данными соотношениями, можно «спуститься» от L_{16} и R_{16} к L_0 и R_0.

Схема вычисления значения функции шифрования $f(R_{i-1}, k_i)$ изображена на рис. 2.

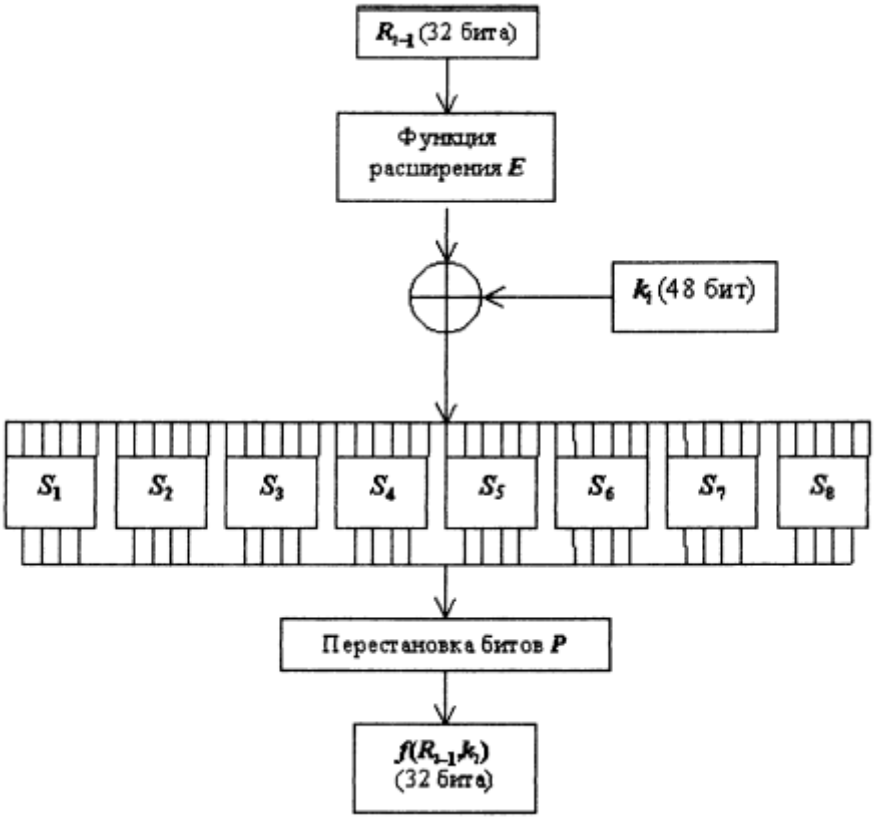

Рис.2. Схема вычисления значения функции шифрования

Для вычисления значения функции f используются: функция расширения E; преобразование S, составленное из восьми преобразований S-блоков $S_1, S_2, ..., S_8$; перестановка P. Аргументами функции f являются вектор R_{i-1} (32 бита) и вектор k_i (48 бит). Функция E «расширяет» 32-битовый вектор R_{i-1} до 48-битового вектора $E(R_{i-1})$ путём дублирования некоторых битов вектора R_{i-1}, при этом порядок следования битов в $E(R_{i-1})$ указан в табл. 2.

Таблица 2.

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 в $E(R_{i-1})$ - это соответственно биты 32, 1 и 2 вектора R_{i-1}, а последние три бита – это соответственно биты 31, 32 и 1 вектора R_{i-1}.

Полученный результат складывается побитно по модулю 2 (побитовое исключающее ИЛИ) с текущим значением ключа k_i и затем представляется в виде восьми последовательных 6-битовых блоков $B_1, ..., B_8$:

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

Далее каждый из блоков B_j трансформируется в 4-битовый блок B_j' с помощью подходящей таблицы S-блока S_j, список которых приведён в таблице 3.

Преобразование блока B_j в B_j' осуществляется следующим образом. Пусть, например, B_2 равен 111010. Первый и последний разряды B_2 являются двоичной записью числа $a, 0 \le a \le 3$. Аналогично средние 4 разряда представляют собой число $b, 0 \le b \le 15$. В нашем примере $a = 2, b = 13$.

Строки и столбцы таблицы S_2 пронумерованы числами a и b. Таким образом, пара (a,b) однозначно определяет число, находящееся на пересечении строки с номером a и столбца с номером b. В данном случае это число равно 3. Записывая его в двоичной форме, получаем B_2', равный 0011.

Значение $f(R_{i-1}, k_i)$ теперь получается применением перестановки битов P, заданной таблицей к результирующему 32-битовому блоку $B_1' B_2' ... B_8'$.

Таблица 3.

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

Таблица 4.

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

На каждой итерации используется текущее значение ключа k_i (48 бит), получаемое из исходного ключа k следующим образом.

Сначала пользователи выбирают сам ключ k, содержащий 56 случайных значащих битов. Восемь битов, находящихся в позициях 8,16,…,64, добавляются в ключ таким образом, чтобы каждый байт содержал нечётное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Значащие 56 бит ключа подвергаются перестановке, приведённой в таблице 5.

Таблица 5.

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

Эта перестановка определяется двумя блоками C_0 и D_0 по 28 бит в каждом (они занимают соответственно верхнюю и нижнюю половины таблицы). Так, первые три бита в C_0 есть соответственно 57, 49, 41 биты ключа. Затем индуктивно определяются блоки C_i и D_i, $i = 1,...,16$.

Если уже определены C_{i-1} и D_{i-1}, то C_i и D_i получаются из них одним или двумя левыми циклическими сдвигами согласно табл. 6.

Таблица 6.

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

Теперь определим ключи k_i, $1 \le i \le 16$. Ключ k_i состоит из 48 битов, выбираемых их битов блока $C_i D_i$ согласно таблице 7. Первыми тремя битами в k_i являются биты 14, 17, 11 из блока $C_i D_i$. Отметим, что 8 из 56 бит (с номерами 9, 18, 22, 25, 35, 38, 43, 54) из $C_i D_i$ отсутствуют в k_i.

Таблица 7.

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

Конечная перестановка IP^{-1} является обратной к начальной перестановке и действует на блок T_{16}. Перестановка определяется таблицей 8.

Таблица 8.

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