Задание №3

Стандарт шифрования данных DES (Data Encryption Standard)

Введение

Стандарт шифрования данных DES (Data Encryption Standard), который ANSI называет Алгоритмом шифрования данных DEA (Data Encryption Algorithm), а ISO - DEA-1, за 20 лет стал мировым стандартом. Хотя на нем и появился налет старости, он весьма прилично выдержал годы криптоанализа и все еще остается безопасным по отношению ко всем врагам, кроме, возможно, самых могущественных.

Описание DES

DES представляет собой блочный шифр, он шифрует данные 64-битовыми блоками. С одного конца алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрирования используются одинаковые алгоритм и ключ (за исключением небольших различий в использовании ключа).

Длина ключа равна 56 битам. (Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.) Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент времени. Ряд чисел считаются слабыми ключами, но их можно легко избежать. Безопасность полностью определяется ключом.

На простейшем уровне алгоритм не представляет ничего большего, чем комбинация двух основных методов шифрования: смещения и диффузии. Фундаментальным строительным блоком DES является применение к тексту единичной комбинации этих методов (подстановка, а за ней - перестановка), зависящей от ключа. Такой блок называется этапом. DES состоит из 16 этапов, одинаковая комбинация методов применяется к открытому тексту 16 раз (см. Рис. 1.).

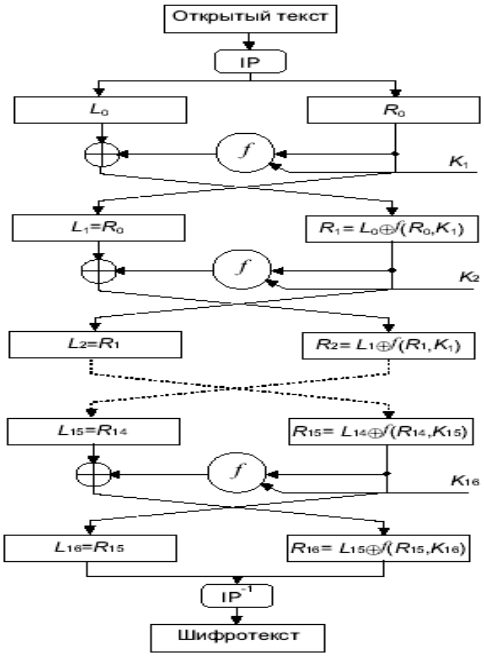


Рис. 1. DES.

Алгоритм использует только стандартную арифметику 64-битовых чисел и логические операции, поэтому он легко реализовывался в аппаратуре второй половины *10-х.* Изобилие повторений в алгоритме делает его идеальным для реализации в специализированной микросхеме. Первоначальные программные реализации были довольно неуклюжи, но сегодняшние программы намного лучше.

Схема алгоритма

DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 этапов одинаковых действий, называемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первойачальной).

На каждом этапе (см. 10-й) биты ключа сдвигаются, и затем из 56 битов ключа выбираются 48 битов. Правая половина данных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией f Затем результат функции f объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 этапов DES.

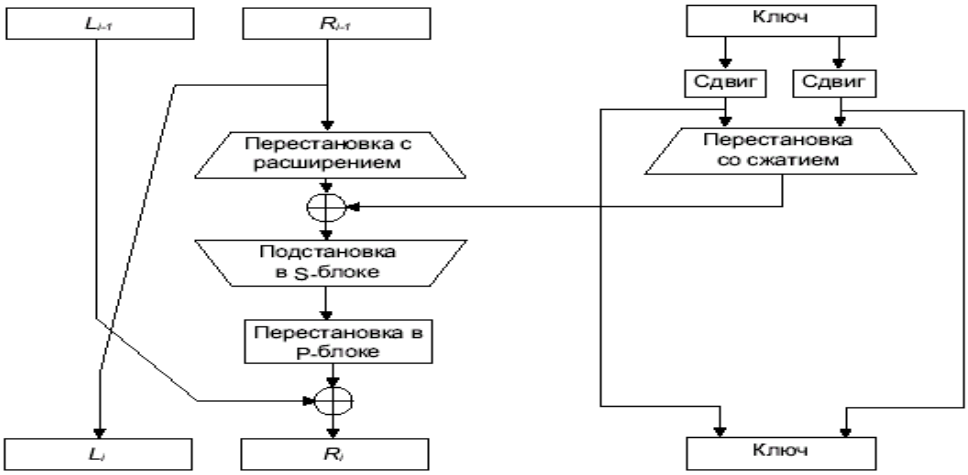


Рис. 2. Один этап DES.

Если *BJ* - это результат г-ой итерации, L_t и R_t - левая и правая половины B_t, K_t - 48-битовый ключ для этапа *i,* а f - это функция, выполняющие все подстановки, перестановки и XOR с ключом, то этап можно представить как начальная перестановка

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

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

Начальная перестановка выполняется еще до этапа 1, при этом входной блок переставляется, как показано в Табл. 1. Эту и все другие таблицы надо читать слева направо и сверху вниз. Например, начальная перестановка перемещает бит 58 в битовую позицию 1, бит 50 - в битовую позицию 2, бит 42 - в битовую позицию 3, и так далее.

Табл. 1. Начальная перестановка

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,	И,	3,
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

Начальная перестановка и соответствующая заключительная перестановка не влияют на безопасность DES. (Как можно легко заметить, эта перестановка в первую очередь служит для облегчения побайтной загрузки данных открытого текста и шифротекста в микросхему DES. Не забывайте, что DES появился раньше 16- и 32-битовых микропроцессорных шин.) Так как программная реализация этой многобитовой перестановки нелегка (в отличие от тривиальной аппаратной), во многих программных реализациях DES начальная и заключительные перестановки не используются. Хотя такой новый алгоритм не менее безопасен, чем DES, он не соответствует стандарту DES и, поэтому, не может называться DES.

Преобразования ключа

Сначала 64-битовый ключ DES уменьшается до 56-битового ключа отбрасыванием каждого восьмого бита. Эти биты используются только для контроля четности, позволяя проверять правильность ключа. После извлечения 56-битового ключа для каждого из 16 этапов DES генерируется новый 48-битовыйподключ**.** Эти подключи, *Ki,* определяются следующим образом.

Табл. 2. Перестановка ключа

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18,
10,	2	59,	51,	43,	35,	27,	19,	И,	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

Во первых, 56-битовый ключ делится на две 28-битовых половинки. Затем, половинки циклически сдвиг а-ются налево на один или два бита в зависимости от этапа. Этот сдвиг показан в 9-й.

Табл. 3. Число битов сдвига ключа в зависимости от этапа

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

После сдвига выбирается 48 из 56 битов. Так как при этом не только выбирается подмножество битов, но и изменяется их порядок, эта операция называется перестановка со сжатием**.** Ее результатом является набор из 48 битов. Перестановка со сжатием (также называемая переставленным выбором) определена в 8-й. Например, бит сдвинутого ключа в позиции 33 перемещается в позицию 35 результата, а 18-й бит сдвинутого ключа отбрасывается.

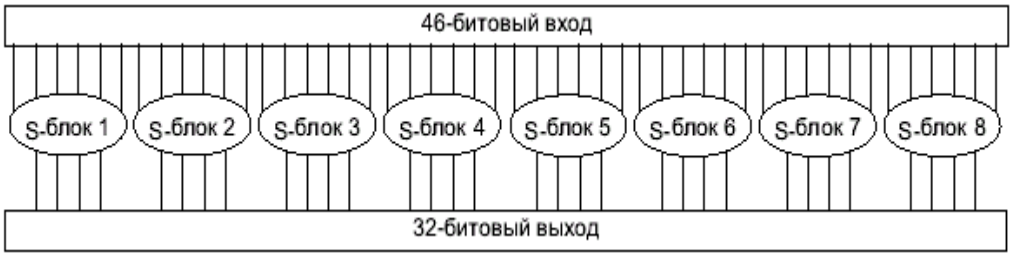
Табл. 4. Перестановка со сжатием

14,	17,	11,	24,	1,	5,	3,	28,	15,	6,	21,	Ю,
23,	19,	11,	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

Из-за сдвига для каждого подключа используется отличное подмножество битов ключа. Каждый бит используется приблизительно в 14 из 16 подключей, хотя не все биты используются в точности одинаковое число раз.

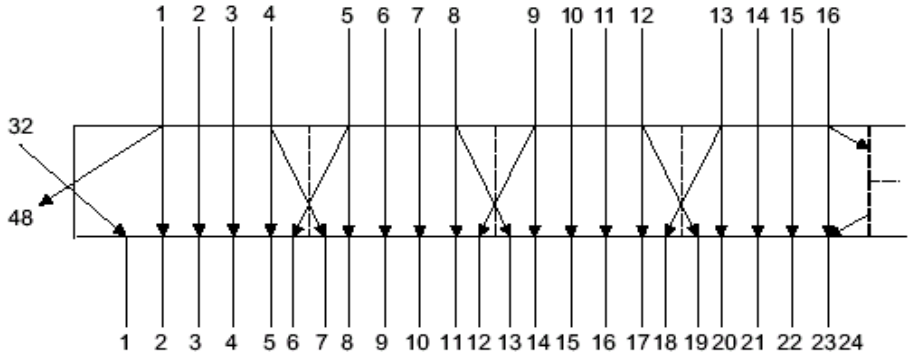
Перестановка с расширением

Эта операция расширяет правую половину данных, R_i, от 32 до 48 битов. Так как при этом не просто повторяются определенные биты, но и изменяется их порядок, эта операция называется перестановкой с расширением. У нее две задачи: привести размер правой половины в соответствие с ключом для операции XOR и получить более длинный результат,



который можно будет сжать в ходе операции подстановки. Однако главный криптографический смысл совсем в другом. За счет влияния одного бита на две подстановки быстрее возрастает зависимость битов результата от битов исходных данных. Это называется лавинным эффектом. DES спроектирован так, чтобы как можно быстрее добиться зависимости каждого бита шифротекста от каждого бита открытого текста и каждого бита ключа.

Перестановка с расширением показана на Рис. 3. Иногда она называется E-блоком (от expansion). Для каждого 4-битового входного блока первый и четвертый бит представляют собой два бита выходного блока, а второй и третий биты - один бит выходного блока. В 7-й показано, какие позиции результата соответствуют каким позициям исходных данных.



Например, бит входного блока в позиции 3 переместится в позицию 4 выходного блока, а бит входного блока в позиции 21 - в позиции 30 и 32 выходного блока.

Рис. 3. Перестановка с расширением.

Хотя выходной блок больше входного, каждый входной блок генерирует уникальный выходной блок.

Табл. 5. Перестановка с расширением

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

Подстановка с помощью S-блоков

После объединения сжатого блока с расширенным блоком с помощью XOR над 48-битовым результатом выполняется операция подстановки. Подстановки производятся в восьми блоках подстановки**,** или S-блоках (от substitution). У каждого S-блока 6-битовый вход и 4-битовый выход, всего используется восемь различных S-блоков. (Для восьми S-блоков DES потребуется 256 байтов памяти.) 48 битов делятся на восемь 6-битовых подблока. Каждый отдельный подблок обрабатывается отдельным S-блоком: первый подблок - S-блоком 1, вт о-рой - S-блоком 2, и так далее. См. Рис. 4.

Рис. 4. Подстановка - S-блоки.

Каждый S-блок представляет собой таблицу из 2 строк и 16 столбцов. Каждый элемент в блоке является 4-битовым числом. По 6 входным битам S-блока определяется, под какими номерами столбцов и строк искать выходное значение. Все восемь S-блоков показаны в Табл. 6.

Табл. 6. S-блоки

4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,
S-блок 2:															
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10,
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5,
0,	14,	7,	11,	10,	4,	13,	1,	5,	8,	12,	6,	9,	3,	2,	15,
13,	8,	10,	1,	3,	15,	4,	2,	11,	6,	7,	12,	0,	5,	14,	9,
S-блок 3:															
10,	0,	9,	14,	6,	3,	15,	5,	1,	13,	12,	7,	11,	4,	2,	8,
13,	7,	0,	9,	3,	4,	6,	10,	2,	8,	5,	14,	12,	11,	15,	1,
13,	6,	4,	9,	8,	15,	3,	0,	11,	1,	2,	12,	5,	10,	14,	7,
1,	10,	13,	0,	6,	9,	8,	7,	4,	15,	14,	3,	11,	5,	2,	12,
S-блок 4:															
7,	13,	14,	3,	0,	6,	9,	10,	1,	2,	8,	5,	11,	12,	4,	15,
13,	8,	11,	5,	6.	15,	0,	3,	4,	7,	2,	12,	1,	10,	14,	9,
10,	6,	9,	0,	12,	11,	7,	13,	15,	1,	3,	14,	5,	2,	8,	4,
3,	15,	0,	6,	10,	1,	13,	8,	9,	4,	5,	11,	12,	7,	2,	14,
S-блок 5:															
2,	12,	4,	1,	7,	10,	11,	6,	8,	5,	3,	15,	13,	0,	14,	9,
14,	11,	2,	12,	4,	7,	13,	1,	5,	0,	15,	10,	3,	9,	8,	6,
4,	2,	1,	11,	10,	13,	7,	8,	15,	9,	12,	5,	6,	3,	0,	14,
11,	8,	12,	7,	1,	14,	2,	13,	6,	15,	0,	9,	10,	4,	5,	3,
S-блок 6:															
12,	1,	10,	15,	9,	2,	6,	8,	0,	13,	3,	4,	14,	7,	5,	11,
10,	15,	4,	2,	7,	12,	9,	5,	6,	1,	13,	14,	0,	11,	3,	8,
9,	14,	15,	5,	2,	8,	12,	3,	7,	0,	4,	10,	1,	13,	11,	6,
4,	3,	2,	12,	9,	5,	15,	10,	11,	14,	1,	7,	6,	0,	8,	13,
S-блок 7:															
4,	11,	2,	14,	15,	0,	8,	13,	3,	12,	9,	7,	5,	10,	6,	1,
13,	0,	11,	7,	4,	9,	1,	10,	14,	3,	5,	12,	2,	15,	8,	6,
1,	4,	11,	13,	12,	3,	7,	14,	10,	15,	6,	8,	0,	5,	9,	2,
6,	11,	13,	8,	1,	4,	10,	7,	9,	5,	0,	15,	14,	2,	3,	12,
S-блок 8:															
13,	2,	8,	4,	6,	15,	11,	1,	10,	9,	3,	14,	5,	0,	12,	7,
1,	15,	13,	8,	10,	3,	7,	4,	12,	5,	6,	11,	0,	14,	9,	2,
7,	11,	4,	1,	9,	12,	14,	2,	0,	6,	10,	13,	15,	3,	5,	8,
2,	1,	14,	7,	4,	10,	8,	13,	15,	12,	9,	0,	3,	5,	6,	11

Входные биты особым образом определяют элемент S-блока. Рассмотрим 6-битовый вход S-блока: *Б\, Б2, Бз, bo,, bs* и *bf,*. Биты *b* и *bf,* объединяются, образуя 2-битовое число от 0 до 3, соответствующее строке таблицы. Средние 4 бита, с $Б_2$ по $Б_5$, объединяются, образуя 4-битовое число от 0 до 15, соответствующее столбцу таблицы.

Например, пусть на вход шестого S-блока (т.е., биты функции XOR с 31 по 36) попадает 110011. Первый и последний бит, объединяясь, образуют 11, что соответствует строке 3 шестого S-блока. Средние 4 бита образуют 1001, что соответствует столбцу 9 того же S-блока. Элемент S-блока 6, находящийся на пересечении строки 3 и столбца 9, - это 14. (Не забывайте, что строки и столбцы нумеруются с 0, а не с 1.) Вместо 110011 подставляется 1110.

Конечно же, намного легче реализовать S-блоки программно в виде массивов с 64 элементами. Для этого потребуется переупорядочить элементы, что не является трудной задачей. (Изменить индексы, не изменяя порядок элементов, недостаточно. S-блоки спроектированы очень тщательно.) Однако такой способ описания S-блоков помогает понять, как они работают. Каждый S-блок можно рассматривать как функцию подстановки 4-битового элемента: *Ъг* появляются входом, а некоторое 4-битовое число –
результатом. Биты Ъ\ и $Ь_6$ определяются соседними блоками, они определяют одну из четырех функций подстановки, возможных в данном S-блоке.

Подстановка с помощью S-блоков является ключевым этапом DES. Другие действия алгоритма линейны и легко поддаются анализу. S-блоки нелинейны, и именно они в большей степени, чем все остальное, обеспечивают безопасность DES.
В результате этого этапа подстановки получаются восемь 4-битовых блоков, которые вновь объединяются в единый 32-битовый блок. Этот блок поступает на вход следующего этапа - перестановки с помощью P-блоков.

Перестановка с помощью P-блоков

32-битовый выход подстановки с помощью S-блоков, перетасовываются в соответствии с P-блоком. Эта перестановка перемещает каждый входной бит в другую позицию, ни один бит не используется дважды, и ни один бит не игнорируется. Этот процесс называется прямой перестановкой или просто перестановкой. Позиции, в которые перемещаются биты, показаны в 5-й. Например, бит 21 перемещается в позицию 4, а бит 4 - в позицию 31.

Табл. 7. Перестановка с помощью P-блоков

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

Наконец, результат перестановки с помощью P-блока объединяется посредством XOR с левой половиной первоначального 64-битового блока. Затем левая и правая половины меняются местами, и начинается следующий этап.

Заключительная перестановка

Заключительная перестановка является обратной по отношению к начальной перестановки. Обратите внимание, что левая и правая половины не меняются местами после последнего этапа DES, вместо этого объединенный блок $Л_{16}£_{16}$ используется как вход заключительной перестановки. В этом нет ничего особенного, перестановка половинок с последующим циклическим сдвигом привела бы к точно такому же результату. Это сделано для того, чтобы алгоритм можно было использовать как для шифрования, так и для дешифрирования.

Табл. 8. Заключительная перестановка

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

Дешифрирование DES

После всех подстановок, перестановок, операций XOR и циклических сдвигов можно подумать, что алгоритм дешифрирования, резко отличаясь от алгоритма шифрования, точно также запутан. Напротив, различные компоненты DES были подобраны так, чтобы выполнялось очень полезное свойство: для шифрования и дешифрирования используется один и тот же алгоритм.

DES позволяет использовать для шифрования или дешифрирования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке. То есть, если на этапах шифрования использовались ключи KI, K_2, K_3, ..., K_{16}, то ключами дешифрирования будут K_{16}, K_{15}, K_{i4}, ..., KI. Алгоритм, который создает ключ для каждого этапа, также цикличен. Ключ сдвигается направо, а число поз и-ций сдвига равно 0, 1,2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Режимы DES

PIPS PUB 81 определяет четыре режима работы: ECB, CBC, OFB и CPB. Банковские стандарты ANSI определяют для шифрования ECB и CBC, а для проверки подлинности - CBC и n-битовый CPB .

В мире программного обеспечения сертификация обычно не важна. Из-за своей простоты в большинстве существующих коммерческих программ используется ECB, хотя этот режим наиболее чувствителен к вскрытию. CBC используется редко несмотря на то, что он лишь незначительно сложнее, чем ECB, и обеспечивает большую безопасность.

Безопасность DES

Специалисты давно интересуются безопасностью DES. Было много рассуждений о длине ключа, количестве итераций и схеме S-блоков. S-блоки были наиболее таинственными - какие-то константы, без видимого объяснения для чего и зачем они нужны. Хотя IBM утверждала, что работа алгоритма была результатом 17 человеко-лет интенсивного криптоанализа, были высказаны опасения, что NSA вставило в алгоритм лазейку, которая позволит агентству легко дешифрировать перехваченные сообщения.

Комитет по разведке Сената США чрезвычайно тщательно расследовал этот вопрос в 1978 году. Результаты работы комитета были засекречены, но в открытых итогах этого расследования с NSA были сняты все обвинения в неуместном вмешательстве в проектирование алгоритма. "Было сказано, что NSA убедило IBM в достаточности более короткого ключа, косвенно помогло разработать структуры S-блоков и подтвердило, что в окончательном варианте DES, с учетом всех знаний NSA, отсутствовали статистические или математические бреши ". Однако, так как правительство не опубликовало подробности расследования, убедить многих не удалось.

Слабые ключи

Из-за того, что первоначальный ключ изменяется при получении подключа для каждого этапа алгоритма, определенные первоначальные ключи являются слабыми. Вспомните, первоначальное значение расщепляется на две половины, каждая из которых сдвигается независимо. Если все биты каждой половины равны 0 или 1, то для всех этапов алгоритма используется один и тот же ключ. Это может произойти, если ключ состоит из одних 1, из одних 0, или если одна половина ключа состоит из одних 1, а другая - из одних 0. Кроме того, у два слабых ключа обладают другими свойствами, снижающими их безопасность.

Четыре слабых ключа показаны в шестнадцатиричном виде в Табл. 11. (Не забывайте, что каждый восьмой бит -это бит четности.).

Табл. 11. Слабые ключи DES

Значение слабого	ключа (с битами четности)	Действительный ключ

0101	0101	0101	0101	0000000 0000000
1F1F	1F1F	OEOE	OEOE	0000000 FFFFFFF
EOEO	EOEO	F1F1	F1F1	FFFFFFF 0000000
FEFE	FEFE	FEFE	FEFE	FFFFFFF FFFFFFF

Кроме того, некоторые пары ключей при шифровании переводят открытый текст в идентичный шифротекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары. Это происходит из-за метода, используемого DES для генерации подключей - вместо 16 различных подключей эти ключи генерируют только два различных подключа. В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называемые полуслабыми ключами**,** в шестнадцатиричном виде приведены в Табл. 12.

Табл. 12. Полуслабые пары ключей DES

01FE	01FE	01FE	01FE	и	FE01	FE01	FE01	FE01
1FEO	1FEO	OEF1	OEF1	и	E01F	E01F	F10E	F10E
01EO	01EO	01F1	01F1	и	E001	E001	F101	F101
1FFE	IEEE	OEFE	OEFE	и	FE1F	FE1F	FEOE	FEOE
01 IF	011F	010E	010E	и	1F01	1F01	OE01	OE01
EOFE	EOFE	FIFE	FIFE	и	FEEO	FEEO	FEE1	FEE1

Ряд ключей генерирует только четыре подключа, каждый из которых четыре раза используется в алгоритме.Эти возможно слабые ключи перечислены в Табл. 13.

Табл. 13. Возможно слабые ключи DES

IF	IF	01	01	OE	OE	01	01	EO	01	01	EO	Fl	01	01	Fl
01	IF	IF	01	01	OE	OE	01	FE	IF	01	EO	FE	OE	01	Fl
IF	01	01	IF	OE	01	01	OE	FE	01	IF	EO	FE	01	OE	Fl
01	01	IF	IF	01	01	OE	OE	EO	IF	IF	EO	Fl	OE	OE	Fl
EO	EO	01	01	Fl	Fl	01	01	FE	01	01	FE	FE	01	01	FE
FE	FE	01	01	FE	FE	01	01	EO	IF	01	FE	Fl	OE	01	FE
FE	EO	IF	01	FE	Fl	OE	01	EO	01	IF	FE	Fl	01	OE	FE
EO	FE	IF	01	Fl	FE	OE	01	FE	IF	IF	FE	FE	OE	OE	FE
FE	EO	01	IF	FE	Fl	01	OE	IF	FE	01	EO	OE	FE	01	Fl
EO	FE	01	IF	Fl	FE	01	OE	01	FE	IF	EO	01	FE	OE	Fl
EO	EO	IF	IF	Fl	Fl	OE	OE	IF	EO	01	FE	OE	Fl	01	FE
FE	FE	IF	IF	FE	FE	OE	OE	01	EO	IF	FE	01	Fl	OE	FE
FE	IF	EO	01	FE	OE	Fl	01	01	01	EO	EO	01	01	Fl	Fl
EO	IF	FE	01	Fl	OE	FE	01	IF	IF	EO	EO	OE	OE	Fl	Fl
FE	01	EO	IF	FE	01	Fl	OE	IF	01	FE	EO	OE	01	FE	Fl
EO	01	FE	IF	Fl	01	FE	OE	01	IF	FE	EO	01	OE	FE	Fl
01	EO	EO	01	01	Fl	Fl	01	IF	01	EO	FE	OE	01	Fl	FE
IF	FE	EO	01	OE	FE	FO	01	01	IF	EO	FE	01	OE	Fl	FE
IF	EO	FE	01	OE	Fl	FE	01	01	01	FE	FE	01	01	FE	FE
01	FE	FE	01	01	FE	FE	01	IF	IF	FE	FE	OE	OE	FE	FE
IF	EO	EO	IF	OE	Fl	Fl	OE	FE	FE	EO	EO	FE	FE	Fl	Fl
01	FE	EO	IF	01	FE	Fl	OE	EO	FE	FE	EO	Fl	FE	FE	Fl
01	EO	FE	IF	01	Fl	FE	OE	FE	EO	EO	FE	FE	Fl	Fl	FE
IF	FE	FE	IF	OE	FE	FE	OE	EO	EO	FE	FE	Fl	Fl	FE	FE

Прежде, чем порицать DES слабые ключи, обратите внимание на то, что эти 64 ключа - это крошечная часть полного набора из 72057594037927936 возможных ключей.

Если вы выбираете ключ случайно, вероятность выбрать один из слабых ключей пренебрежимо мала. Вы можете всегда проверять "на слабость" сгенерированный ключ. Некоторые думают, что нечего и беспокоиться на этот счет. Другие утверждают, что проверка очень легка, почему бы ее и не выполнить. Других слабых ключей в процессе и с-следований найдено не было.

Количество этапов

Почему 16 этапов? Почему не 32? После пяти этапов каждый бит шифротекста является функцией всех б и-тов открытого текста и всех битов ключа [1078, 1080], а после восьми этапов шифротекст по сути представляет собой случайную функцию всех битов открытого текста и всех битов ключа. (Это называется лавинным эффектом.) Так почему не остановиться после восьми этапов?

В течение многих лет версии DES с уменьшенным числом этапов успешно вскрывались. DES с тремя и четырьмя этапами был легко взломан в 1982 году. DES с шестью этапами пал несколькими годами позже. Дифференциальный криптоанализ Бихама и Шамира объяснил и это: DES с любым количеством этапов, меньшим 16, может быть взломан с помощью вскрытия с известным открытым текстом быстрее, чем с помощью вскрытия грубой силой. Конечно грубый взлом является более вероятным способом вскрытия, но интересен тот факт, что алгоритм содержит ровно 16 этапов.

Задание.

1. Ознакомиться с алгоритмом DES.
2. Установить программу DES.
3. Произвести шифрование контрольной фразы.
4. Оценить время шифрования и затраченные ресурсы компьтера.
5. Произвети расшифрование контрольной фразы.
6. Оценить затраченные ресурсы и время.