第3章 对称密码体制

作为现代密码学，虽然算法更加复杂，但原理还是没变。只不过现代密码学的算法是针对比特而不是针对字母进行变换，实际上这只是字母表长度上的改变：从26个元素变为2个元素。大多数优秀算法的主要组成部分仍然是代换和置换的组合，例如DES算法。

前面已经提到，对称密码体制就是在加密和解密时用到的密钥相同，或者加密密钥和解密密钥之间存在着确定的转换关系。其实质是设计一种算法，能在密钥控制下，把n比特明文简单而又迅速地置换成唯一的n比特密文，并且这种变换是可逆的(解密)。

根据不同的加密方式，对称密码体制又有两种不同的实现方式，即分组密码和序列密码(或称流密码)。其中，分组密码是先把明文划分为许多分组，每个明文分组被当做一个整体来产生一个等长(通常情况下)的密文分组，通常使用的是64位或128位分组大小。而序列密码则每次加密数据流中的一位或一个字节。

3.1 分组密码

分组密码体制是目前商业领域中比较重要而流行的一种加密体制，它广泛地应用于数据的保密传输、加密存储等应用场合。分组密码对明文进行加密时，首先需要对明文进行分组，每组的长度都相同，然后对每组明文分别加密得到等长的密文。分组密码的安全性主要依赖于密钥，而不依赖于对加密算法和解密算法的保密，因此，分组密码的加密和解密算法可以公开。

1. 要求

分组密码算法实际上就是在密钥的控制下，简单而迅速地找到一个置换，用来对明文分组进行加密变换，一般情况下对密码算法的要求是：

(1) 分组长度m足够大。当分组长度m较小时，分组密码很类似于某些古典密码，如维吉尼亚密码、希尔密码和置换密码，它仍然有效地保留了明文中的统计信息，这种统计信息将给攻击者留下可乘之机，攻击者可以有效地穷举明文空间，得到密码变换本身。

(2) 密钥空间足够大。分组密码的密钥所确定的密码变换只是所有置换中极小的一部分。如果这一部分足够小，攻击者可以有效地通过穷举密钥，确定所有的置换。到时，攻击者就可以对密文进行解密，以得到有意义的明文。

(3) 密码变换必须足够复杂。使攻击者除了穷举法攻击以外,找不到其他简洁的数学破译方法。

2. 基本思想

为了便于实现和分析,在设计分组密码时通常遵循以下两个基本思想:

(1) 扩散(diffusion)。将明文及密钥的影响尽可能迅速地散布到较多个输出的密文中。产生扩散的最简单方法是通过置换(比如重新排列字符)。

(2) 混淆(confusion)。其目的在于使作用于明文的密钥和密文之间的关系复杂化,使明文和密文之间、密文和密钥之间的统计相关特性极小化,从而使统计分析攻击不能奏效。通常的方法是代换。

3. 技术

具体来说,可以综合采用以下两种技术:

(1) 将大的明文分组再分成几个小段,分别完成各个小段的加密置换,最后进行并行操作。这样使得总的分组长度足够大,有利于对密码的实际分析和评测,以保证密码算法的强度。

(2) 采用乘积密码技术。乘积密码就是以某种方式连续执行两个或多个密码变换。例如,设有两个子密码变换 $E1$ 和 $E2$,则先以 $E1$ 对明文进行加密,然后再以 $E2$ 对所得结果进行加密。其中 $E1$ 的密文空间与 $E2$ 的明文空间相同。如果使用得当的话,乘积密码可以有效地掩盖密码变换的弱点,构成比其中任意一个密码变换强度更高的密码系统。

典型的分组密码有 DES、AES。

3.2　数据加密标准 DES

3.2.1　DES 简介

数据加密标准(Data Encryption Standard,DES)是一种对计算机数据进行密码保护的数学算法,它的产生被认为是 20 世纪 70 年代信息加密技术发展史上的里程碑之一。由于 20 世纪 60 年代计算机得到了迅猛的发展,大量的数据资料被集中储存在大型计算机数据库中并在计算机通信网中进行传输,其中有些通信具有高度的机密性,有些数据具有极为重要的价值,因此对计算机通信及计算机数据进行保护的需求日益增长。当时的美国虽然已经制定了数据保密措施,但是人们普遍认为这些保密措施只对业余人员有效,对于职业人员来说,要截获通信信号、绕过报警系统并接通通信设备,取出、复制、删除、插入或更改信息是不成问题的。针对这种情况,有人提出了两种对数据进行保护的方法,一种方法是对数据进行物理保护,即把重要的数据存放到安全的地方,如银行的地下室中;另一种方法是对数据进行密码保护。

由于普遍认为加密算法如果足够复杂的话,密码是一种有效的措施。所以美国国家标准局 NBS 于 1973 年 5 月发出通告,公开征求一种标准算法用于对计算机数据在传输和存储期间实现加密保护的密码算法。1975 年美国国家标准局接受了美国国际商业机器公司

IBM 推荐的一种密码算法并向全国公布，征求对采用该算法作为美国信息加密标准的意见。

经过两年的激烈争论，美国国家标准局于 1977 年 7 月正式采用该算法作为美国数据加密标准。1980 年 12 月美国国家标准协会 ANSI 正式采用这个算法作为美国的商用加密算法。

DES 是一种对称密码体制，它所使用的加密和解密密钥是相同的，是一种典型的按分组方式工作的密码。其基本思想是将二进制序列的明文分成每 64 比特一组，用长为 64 比特的密钥对其进行 16 轮代换和置换加密，最后形成密文。

DES 的巧妙之处在于除了密钥输入顺序，其加密和解密的步骤完全相同，这就使得在制作 DES 芯片时易于做到标准化和通用化。这一点尤其适合现代通信的需要，在 DES 出现以后，经过许多专家学者的分析论证证明它是一种性能良好的数据加密算法，不仅随机特性好，线性复杂度高，而且易于实现，加上能够标准化和通用化，因此 DES 在国际得到了广泛的应用。

3.2.2 DES 加密解密原理

DES 是典型的传统密码体制，它利用传统的换位和置换等加密方法，现介绍算法如下：

加密前，先将明文分成 64 比特的分组，然后将 64 比特二进制码输入到密码器中，密码器对输入的 64 比特码首先进行初始置换，然后在 64 比特主密钥产生的 16 个子密钥控制下进行 16 轮乘积变换，接着再进行逆初始置换就得到 64 比特已加密的密文。

算法的主要步骤如图 3-1 所示。

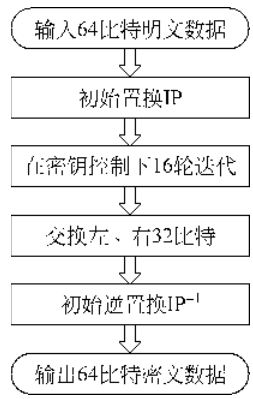

图 3-1 DES 算法的主要步骤

假定信息空间都是由{0,1}组成的字符串，信息被分成 64 比特的块，密钥是 56 比特。经过 DES 加密的密文也是 64 比特的块。设 m 是一个 64 比特的信息块，k 为 56 比特的密钥，即

$$m = m_1\, m_2 \cdots m_{64} \quad m_i = 1,2,\cdots,64$$

$$k = k_1 k_2 \cdots k_{64} \quad k_i = 1,2,\cdots,64$$

其中，k_8，k_{16}，k_{24}，k_{32}，k_{40}，k_{48}，k_{56}，k_{64} 是奇偶校验位，真正起作用的密钥仅 56 位。

下面介绍一个分组的加密过程。

1. 初始置换 IP

将 64 个明文比特的位置进行置换，得到一个乱序的 64 比特明文组，然后分成左右两段，每段为 32 比特以 L 和 R 表示，如图 3-2 所示。

2. 迭代变换

它是 DES 算法的核心部分。如图 3-3 所示，将经过 IP 置换后的数据分成 32 比特左右两组，在迭代过程中彼此左右交换位置，每次迭代只对右边的 32 比特进行一系列的加密变

换。在此轮迭代即将结束时，把左边的 32 比特与右边的 32 比特诸位模 2 相加，作为下一轮迭代时右边的段，并将原来右边的未经变换的段直接送到左边的寄存器中作为下一轮迭代时左边的段。在每一轮迭代时，右边段要经过选择扩展运算 E、密钥加密运算、选择压缩运算 S、置换运算 P 和左右混合运算。

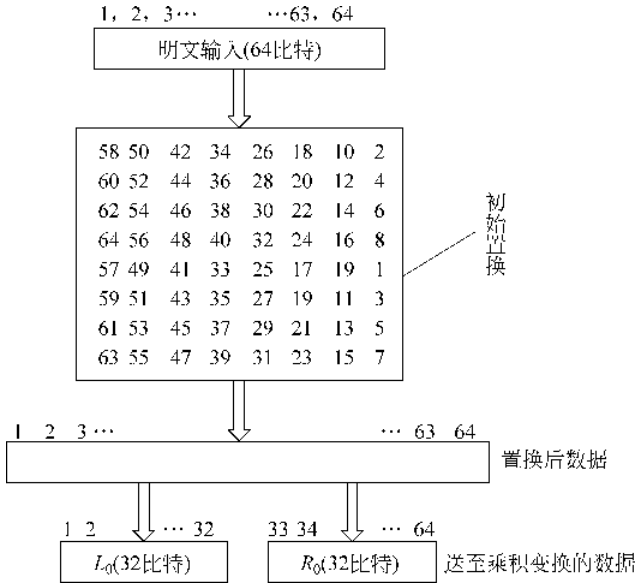

图 3-2　初始置换

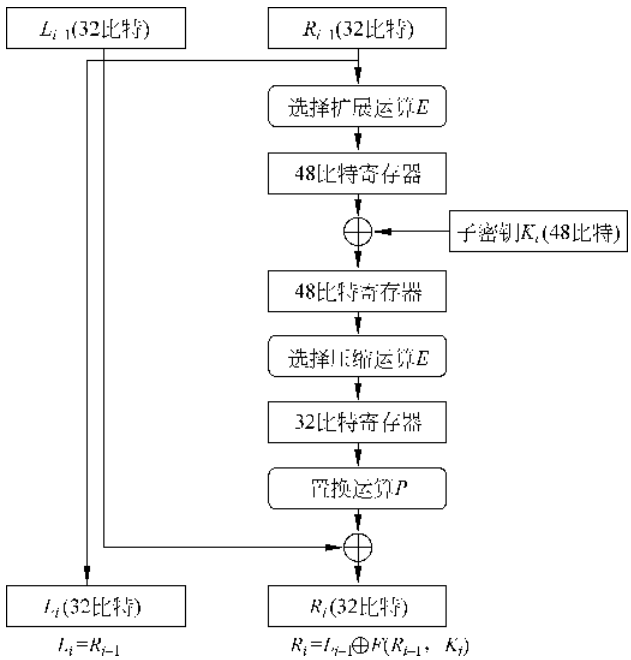

图 3-3　乘积变换

这样的迭代共进行 16 轮，结束后，再将所得的左、右长度相等的 L_{16} 和 R_{16} 进行交换得到 64 比特数据。

1）选择扩展运算 E

将输入的 32 比特 R 扩展成 48 比特输出，其变换表如图 3-4 所示。

令 s 表示 E 的输入的下标，则 E 的输出将是对原下标 $s=0$ 或 $1(\mathrm{mod}\ 4)$ 的各比特重复一次得到的，即对原第 32 1 4 5 8 9 12 13 16 17 20 21 24 25 28 29 各位重复一次得到数据扩展。将表中数据按行读出即得到 48 比特输出。

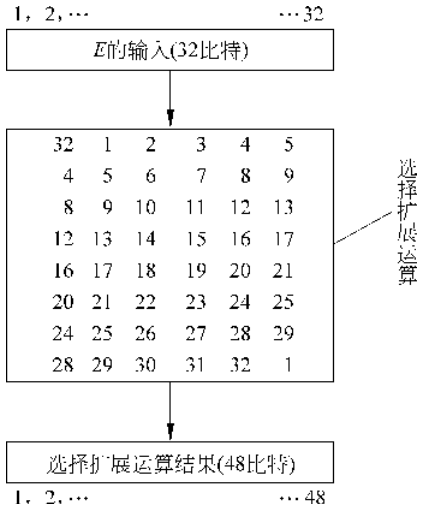

图 3-4　选择扩展运算 E

2）密钥加密运算

将子密钥产生器输出的 48 比特子密钥 k 与选择扩展运算 E 输出的 48 比特数据按位模 2 相加（子密钥如何产生请见后文）。

3）选择压缩运算 S

将前面送来的 48 比特数据自左至右分成 8 组，每组 6 比特。然后并行送入 8 个 S 盒，每个 S 盒为一非线性代换网络，有 4 个输出。盒 S_1 至 S_8 的选择函数关系如表 3-1 所示，运算 S 的框图如图 3-5 所示。

8 个 S 盒是将 6 比特的输入映射为 4 比特的输出。

以 S 盒为例说明它们的功能如下：

若输入为 $b_1b_2b_3b_4b_5b_6$ 其中 b_1b_6 两位二进制数表达了 0～3 之间的数。$b_2b_3b_4b_5$ 为四位二进制数，表达 0～15 之间的某个数。

在 S_1 表中的 b_1b_6 行 $b_2b_3b_4b_5$ 列找到一数 m（$0\leqslant m\leqslant 15$），若用二进制表示为 $m_1m_2m_3m_4$，则 $m_1m_2m_3m_4$ 便是它的 4 比特输出。

例如输入为 001111，$b_1b_6=01=1$，$b_2b_3b_4b_5=0111=7$，即在 S_1 盒中的第 1 行第 7 列求得数 1，所以它的 4 比特输出为 0001。

又如对于 S_2 盒输入为 101011，则 S_2 盒中 3 行 5 列元素为 15，故输出为 1111。

表 3-1　DES 的 *S* 盒定义

行＼列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	S_1
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	
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	S_2
3	13	8	10	1	3	15	4	2	11	6	7	42	0	5	14	9	
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	S_3
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	
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	S_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	
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	S_5
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	
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	S_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	
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	S_7
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	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	1	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	S_8
3	2	1	14	7	1	10	7	13	15	12	9	0	3	5	6	11	

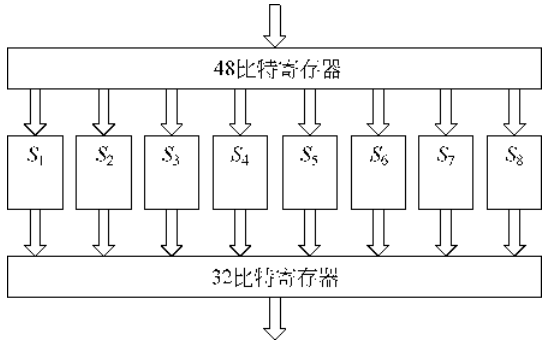

图 3-5　*S* 盒的结构

S 盒是 DES 的核心，也是 DES 算法最敏感的部分，其设计原理至今仍讳莫如深，显得非常神秘。所有的替换都是固定的，但是又没有明显的理由说明为什么要这样，有许多密码学家担心美国国家安全局设计 S 盒时隐藏了某些陷门，使得只有他们才可以破译算法，但研究中并没有找到弱点。

美国国家安全局曾透露了 S 盒的几条设计准则：

(1) 所有的 S 盒都不是它输入的线性仿射函数，换句话说，就是没有一个线性方程能将四个输出比特表示成 6 个比特输入的函数。

(2) 改变 S 盒的 1 位输入，输出至少改变 2 位，这意味着 S 盒是经过精心设计的，它最大程度上增大了扩散量。

(3) S 盒的任意一位输出保持不变时 0 和 1 个数之差极小，即如果保持一位不变而改变其他五位，那么其输出 0 和 1 的个数不应相差太多。

4) 置换运算

置换运算 P 对 S_1 至 S_8 盒输出的 32 比特数据进行坐标变换，如图 3-6 所示，置换 P 输出的 32 比特数据与左边 32 比特即 R_{i-1} 诸位模 2 相加所得到的 32 比特作为下一轮迭代用的右边的数字段，并将 R_{i-1} 并行送到左边的寄存器作为下一轮迭代用的左边的数字段。

3. 子密钥产生器

将 64 比特初始密钥经过置换选择 PC-1，循环移位置换、置换选择 PC-2，给出每次迭代加密用的子密钥 k_i，如图 3-7 所示。

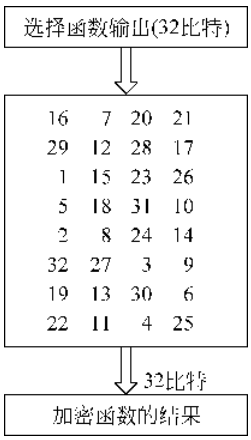

图 3-6 置换运算 P

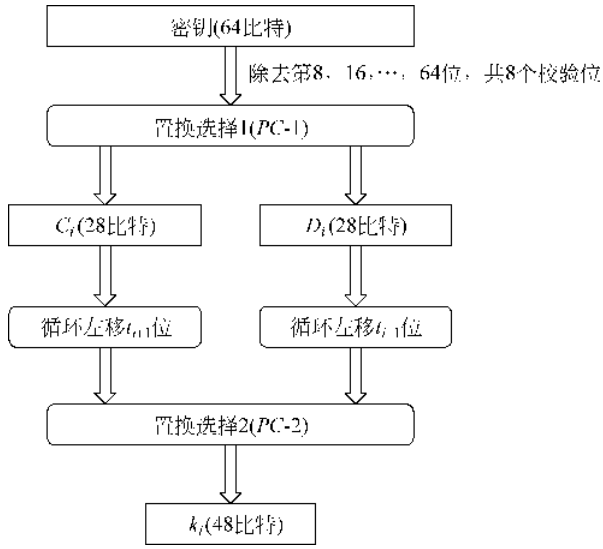

图 3-7 子密钥产生器

在 64 比特初始密钥中有 8 位为校验位，其位置号为 8 16 24 32 48 56 和 64，其余 56 位用于子密钥计算，将这 56 位送入置换选择 PC-1。置换后分为两组，每组为 28 比特，分别送入 C 寄存器和 D 寄存器中，如图 3-8 所示。

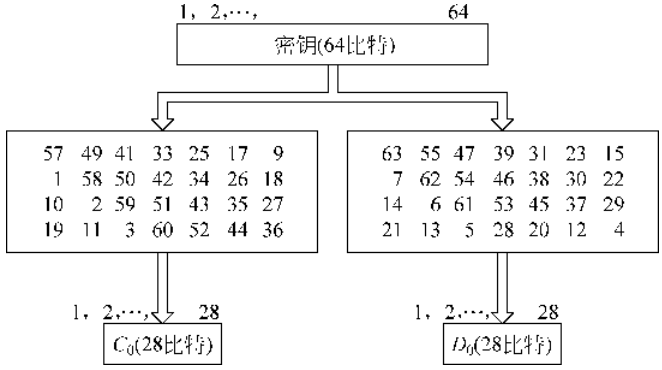

图 3-8　置换选择 *PC*-1

在各次迭代中 C 和 D 寄存器分别将存数进行左循环移位置换，移位次数如表 3-2 所示。每次移位后将 C 和 D 寄存器的存数送给置换选择 *PC*-2。

表 3-2　移位次数表

第 i 次迭代	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
循环左移次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

置换选择 *PC*-2 将 C 中第 9 18 22 25 位和 D 中第 7 9 15 26 位删去，并将其余数字置换位置后送出 48 比特数字作为第 i 次迭代时所用的子密钥 k_i，如图 3-9 所示。

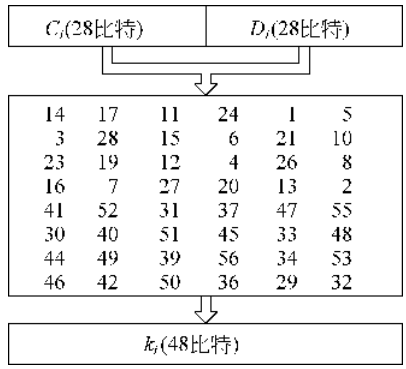

图 3-9　置换选择 *PC*-2

例如，$k=k_1k_2\cdots k_{64}$，则 $C_0=k_{57}k_{49}\cdots k_{44}k_{36}$，$C_0=k_{63}k_{55}\cdots k_{12}k_4$。

下面介绍如何从 C_i、D_i 求 C_{i+1}、D_{i+1}，$i=0,1,2,\cdots,15$。

设 $C_i=c_1c_2\cdots c_{28}$，$D_i=d_1d_2\cdots d_{28}$。首先要做左移运算，左移的位数如表 3-2 所示。例如，设 $C_i=c_1c_2\cdots c_{28}$，$D_i=d_1d_2\cdots d_{28}$，则 $C_2=c_2c_3\cdots c_{28}\ c_1$，$D_2=d_2d_3\cdots d_{28}\ d_1$。

C_3 和 D_3 是由 C_2、D_2 左移 1 位而得到的，C_4 和 D_4 是由 C_3、D_3 左移 2 位而得到的，所以 $C_4=c_5c_6\cdots c_{28}\ c_1\ c_2\ c_3\ c_4$，$D_4=d_5d_6\cdots d_{28}\ d_1\ d_2\ d_3\ d_4$。

因此，$C_i\ D_i=b_1b_2\cdots b_{56}$，则 $k_i=b_{14}\ b_{17}\ b_{11}\ b_{24}\cdots b_{36}\ b_{29}\ b_{32}$。

4. 逆初始置换 IP^{-1}

将 16 轮迭代后给出的 64 比特组进行置换得到输出的密文组，如图 3-10 所示，输出为阵中元素按行读的结果。注意 IP 中的第 58 位正好是 1，也就是说在 IP 的置换下第 58 位换为第 1 位。同样，在 IP 的置换下应将第 1 位换回第 58 位，依此类推，由此可见输入组 m 和 $IP^{-1}(IP(m))$是一样的，IP 和 IP^{-1}在密码上的意义不大，它的作用在于打乱原来输入 m 的 ASCII 码字划分关系。

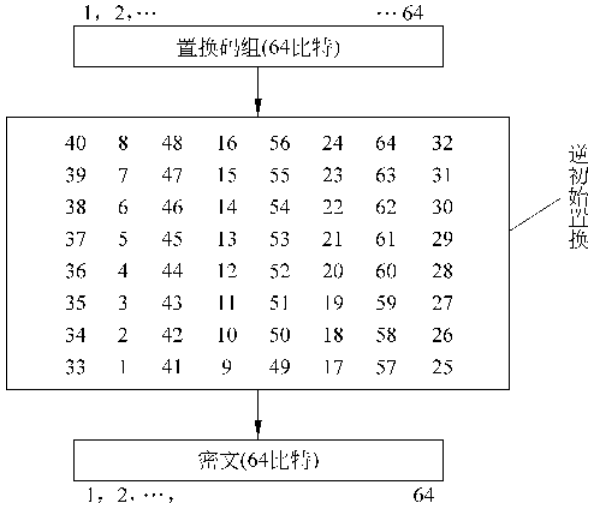

图 3-10　逆初始置换 IP^{-1}

逆初始置换后得到的 64 位数据分组，即为加密后得到的密文。

5. DES 的解密

解密算法与加密算法相同，只是子密钥的使用次序相反。把 64 位密文当做输入，第一次解密迭代使用子密钥 k_{16}，第二次解密迭代使用子密钥 k_{15}，第 16 次解密迭代使用子密钥 k_1，最后的输出便是 64 位的明文。

3.2.3　DES 的安全性

DES 的出现是密码学史上的一个创举，以前任何设计者对于密码体制及其设计细节都是严加保密的。而 DES 公开发表，任人研究和分析。无须经过许可就可以制作 DES 的芯片和以 DES 为基础的保密设备。DES 的安全性完全依赖于所用的密钥。DES 在 20 多年的应用实践中，没有发现严重的安全缺陷，在世界范围内得到了广泛的应用，为确保信息安全做出了不可磨灭的贡献。

然而，出于安全方面的考虑，在 DES 被采纳为标准之前，曾受到过激烈的批评，其实直到今天都未平息。批评主要集中在两个方面。其一，DES 的前身是 IBM 公司的 LUCIFER 算法，该算法采用的密钥长度是 128 位，而 DES 却只使用了 56 位的密钥，减少了 72 位。批

评者担心密钥太短而无法抗击穷举攻击。其二，DES的内部结构，即S盒的设计标准被列入官方机密。所以，用户不能确信DES的内部结构是没有弱点的，美国国家安全局(NSA)有可能利用这些弱点在没有密钥的情况下来解密。不过近年来的差分分析方面的研究表明，DES的内部结构是强健的。而且按照IBM的参与者所说，原始算法中唯一做过改动的就是S盒，这是由NSA建议的，它去掉了测试过程中发现的一些算法脆弱性。

自从DES问世至今已经有近30年了，尽管一开始人们就对它有颇多的担心和争议，这些年来许许多多的人对它进行各种各样的研究攻击。从目前的成果来看，除了穷举攻击之外，就没有更好的方法破译DES了。

从应用实践来看，DES具有良好的“雪崩效应”。所谓雪崩效应，就是明文或密钥的微小改变将对密文产生很大的影响。特别地，明文或密钥的某一位发生变化，会导致密文的很多位发生变化。

DES显示了很强的雪崩效应，例如通过实验可以发现，两条仅有一位不同的明文，使用相同的密钥，仅经过3轮迭代，所得两段准密文就有21位不同；一条明文，使用两个仅一位不同的密钥加密，经过数轮变换之后，有半数的位都不相同。

当然，DES也存在着一些有可能被利用的弱点，如：

(1) 密钥较短：56位密钥空间约为7.2×10^{16}。1997年6月Rocke Verser小组通过因特网利用数万台微机历时4个月破译了DES；1998年7月，EFF用一台25万美元的机器，历时56小时破译了DES。

(2) 存在弱密钥：有的密钥产生的16个子密钥中有重复者。

(3) 互补对称性：$C=\mathrm{DES}(M,K)$，则$C'=\mathrm{DES}(M',K')$。其中，M'，C'，K'是M，C，K的非。

20多年来对DES进行的大量研究，主要成果集中在以下几个方面。

1. 弱密钥

DES算法在每次迭代时都有一个子密钥供加密用，如果一个外部密钥所产生的所有子密钥都是一样的，则这个密钥就称为弱密钥(weak key)。

若k为弱密钥，则有

$$\mathrm{DES}_k(\mathrm{DES}_k(x))=x$$

$$\mathrm{DES}_k^{-1}(\mathrm{DES}_k^{-1}(x))=x$$

即以k对x加密或解密两次都可以恢复出明文，其加密运算和解密运算没有区别。

如果随机选取密钥，在总数2^{56}个密钥中，弱密钥所占比例很小，加以注意就可以避开，对DES的安全性影响不大。

2. 密文与明文、密文与密钥的相关性

有人详细研究了DES的输入，表明每个密文比特都是所有明文比特和所有密钥比特的复合函数，并且指出达到这一要求所需的迭代次数最少为5，迭代8次以后输出和输入就可认为是不相关的了。

3. 密钥搜索机

对DES的安全性的意见中，较为一致的看法是DES的密钥短了些，密钥长度是56比

特，密钥量为 $2^{56}\approx7.2\times10^{16}$ 个，选择长密钥时会使成本提高，运行速度降低，若要对 DES 进行密钥搜索破译，分析者在得到一组明文-密文对的情况下，可对明文进行不同的密钥加密，直到得到的密文与已知的密文-明文对中的相符就可确定所用的密钥了。

1977 年 Differ 和 Hellman 认为利用 100 万个超大规模集成电路块所组成的一台专门用于破译 DES 的并行计算机能在一天中穷举搜索所有 2^{56} 个密钥，每个集成块每微秒检查一个密钥，则每天可以检查 8.64×10^{10} 个密钥。如果用一个集成块检查全部密钥，则几乎要花费 8.33×10^{5} 天，约 2283 年。

但是如果用 100 万个集成块，则在一天时间内可以检查整个密钥空间。这样一台机器在 1977 年需耗资约 2000 万美元。Differ 和 Hellman 据此指出，除了像美国国家安全局那样的机构外，任何人不可能破译 DES。但他们预测到 1990 年制造和破译 DES 专用机的成本将要大幅度下降，那时 DES 将完全是不安全的。

事实证明，他们的预测是有道理的，在过去相当长的一段时间里，人们找不到比穷举搜索更有效的方法攻击 DES。也没有能力对 56 比特的密钥进行穷举搜索，因而在过去，DES 是安全的，但目前的事实证明这个历史已成为过去，DES 不能经受住穷举攻击。

3.2.4 多重 DES

DES 在穷举攻击下相对比较脆弱，因此需要用某种算法来替代它，有两种解决方法。其一，设计全新的算法；其二，用 DES 进行多次加密，且使用多个密钥，即多重 DES。这种方法能够保护用于 DES 加密的已有软件和硬件继续使用。

1. 二重 DES

二重 DES 的加密与解密过程如图 3-11 所示。给定明文 P 和两个加密密钥 k_1 和 k_2，采用 DES 对 P 进行加密 E，有

$$\text{密文 } C = E_{k2}(E_{k1}(P))$$

对 C 进行解密 D，有

$$\text{明文 } P = D_{k1}(D_{k2}(C))$$

显然，二重 DES 比单一的 DES 要安全，因为这里使用了两个密钥(均为 56 比特)，设 $k=k_1 \parallel k_2$，则 k 的密钥空间数量为 2^{112}。

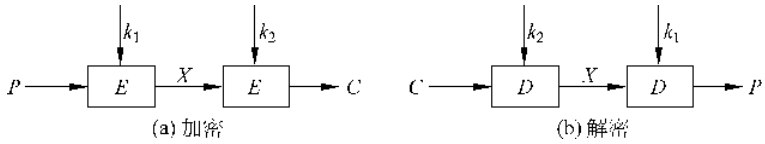

图 3-11 二重 DES

这里要注意，如果采用普通穷举法来攻击二重 DES，其攻击代价并不是 2^{112}。这是因为一个分组为 64 比特，明文空间为 2^{64}，而密钥空间为 2^{112}。因此对于某个明文 P，可产生密文 C 的密钥平均个数为 $2^{112}/2^{64}=2^{48}$ 个。换句话说，大约有 2^{48} 个不同的密钥，对明文 P 加密后得到的密文相同且都等于 C。这就需要另一个 (P,C) 对，以从这 2^{48} 个不同的密钥中确定真