# 第5章 chapter 5

# 对称加密

在当今数字化时代，信息安全至关重要。从古典密码学中的凯撒密码、维吉尼亚密码等简单置换加密算法，到现代密码学中的数据加密标准、高级加密标准等基于复杂数学运算的分组密码算法，其演进过程遵循着严格的数学逻辑和密码学原理。算法的设计基于数论、代数结构等数学原理，利用有限域上的运算、置换群的变换等方式，实现对明文的加密变换，同时确保加密的安全性和效率。

以 SSL/TLS 协议为例，在互联网通信中，通过对称加密算法对传输的数据进行加密，确保数据在网络传输过程中的机密性，防止数据被窃听和篡改。在金融领域，银行卡交易信息的加密传输也依赖于对称加密技术，以保障客户的资金安全。

## 5.1 对称加密算法的基本原理

对称加密算法的发展是一部与密码分析技术不断博弈的历史。对称加密作为保障信息机密性的基础技术，以高效快速的加解密性能、简洁的运行机制，在保障信息不被窃取和篡改中发挥着关键作用。

### 5.1.1 对称加密的概念

在历史上，密码学主要用于秘密通信，特别是使用密码学的加密术，使事先共享某种信息的两方能够秘密通信。此方式中，通信双方实现了共享某种秘密信息，现在称为秘密密钥（或对称密钥）。

对称加密又称为单密钥加密或密钥加密，是一种采用单钥密码系统的加密方法。在对称加密方式中，同一个密钥被用于加密和解密过程。发送方使用密钥加密（或“混合”）信息，接收方收到后使用相同的密钥解密（或“去混合”）恢复信息。消息本身称为明文（plaintext），从发送方传输到接收方的经加密的消息称为密文（ciphertext），如图 5.1 所示。共享的密钥用来从所有窃听者中区分通信方（假设通过公共信道传输）。

发送方的明文消息 $p=[p_1,p_2,\cdots,p_M]$，p 的 M 个元素是某个语言集中的字母，如 26 个英文字母，现在最常见的是二进制字母表{0,1}中元素组成的二进制串。加密之前先生成一个形如 $k=[k_1,k_2,\cdots,k_J]$ 的密钥作为密码变换的输入参数之一。该密钥可以由消息发送方生成，然后通过安全的渠道送到接收方；或者由可信的第三方生成，然后通过

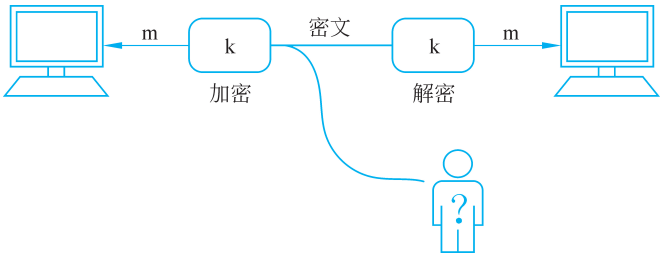


图 5.1 对称加密的基本情形

安全渠道分发给发送方和接收方。

发送方通过加密算法 E 根据输入的消息 p 和密钥 k 生成密文 $c=[c_1,c_2,\cdots,c_N]$，即

$$c=E_k(p)$$

接收方通过解密算法 D 根据输入的密文 c 和密钥 k 恢复明文 $p=[p_1,p_2,\cdots,p_M]$，即

$$p=D_k(c)$$

一个攻击者(密码分析者)能基于不安全的公开信道观察密文 c，但不能接触到明文 p 或密钥 k，他可以试图恢复明文或密钥。假设攻击者知道加密算法 E 和解密算法 D，如果只对当前这个特定的消息感兴趣，则努力的焦点是通过产生一个明文的估计值 $p'$来恢复明文 p。如果攻击者也对读取未来的消息感兴趣，则需要试图通过产生一个密钥的估计值 $k'$来恢复密钥 k，这是一个密码分析的过程。

对称加密算法的安全性主要取决于两个因素：①加密算法必须足够强大，使得不必为算法保密，仅根据密文就能破译出消息是不可行的；②密钥的安全性，密钥必须保密并保证有足够大的密钥空间，对称密码体制要求基于密文和加/解密算法的知识能破译出消息的做法是不可行的。

优点：加密、解密处理速度快，具有很高的数据吞吐率，硬件加密可达到每秒几百兆字节，软件也能达到每秒兆字节的量级，密钥相对较短。

缺点：

(1) 密钥是保密通信安全的关键，发信方需要安全、妥善地将密钥护送到收信方，不能泄露内容，密钥分发过程复杂、代价高；

(2) 多人通信时密钥组合数量爆炸式膨胀，使密钥分发更复杂；N 个人两两通信，总共需要密钥 $C_N^2=N(N-1)/2$ 个，且良好的密码使用习惯要求每次会话都要更换密钥；

(3) 通信双方必须统一密钥才能发送保密信息，若双方不相识，则无法发送秘密信息；

(4) 除密钥管理与分发问题以外，对称密码算法还存在数字签名困难的问题(通信双方拥有同样的消息，接收方可伪造签名，发送方可否认发送过某消息)。

### 5.1.2 加密和解密过程

对称密码体制又称为单钥体制，是加密和解密使用相同密钥的密码算法。采用单钥体制的系统的保密性主要取决于密钥的保密性，与算法的保密性无关，即由密文和加/解密算法不可能得到明文。算法不需要保密，需要保密的仅是密钥。对称密码体制加密和

解密过程如图 5.2 所示。

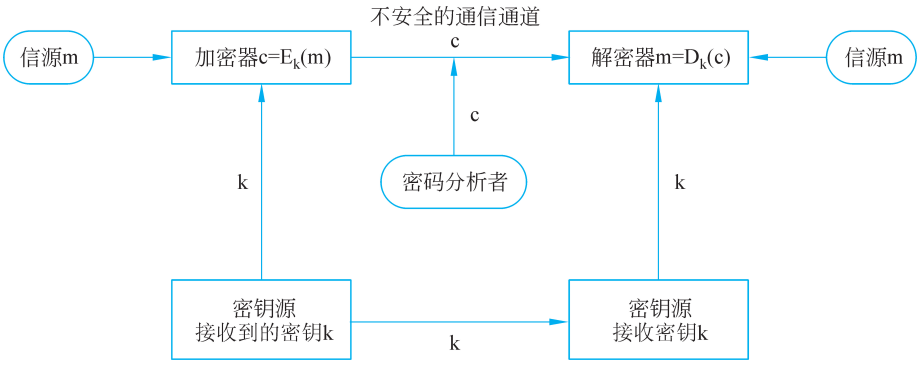


图 5.2 对称密码体制加密和解密过程

密钥可由发送方产生,然后经过一个安全可靠的途径(如信使递送)送至接收方,或由第三方产生后安全、可靠地分配给通信双方。

如何产生满足保密要求的密钥及如何将密钥安全、可靠地分配给通信双方是对称加/解密设计和实现的主要问题之一。密钥的产生、分配、存储、销毁等问题统称为密钥管理,是影响系统安全的关键因素。即使密码算法再好,但若密钥管理问题处理不好,也很难保证系统的安全保密。

对称密码体制对明文消息的加密有两种方式:一种是将明文消息按字符(如二元数字)逐位加密,此类密码体制称为序列密码或流密码;另一种是将明文消息分组(含有多个字符),逐组对其进行加密,此类密码体制称为分组算法或分组密码。

### 5.1.3 数学基础和运算原理

对称加密的理论是建立在数学的一些特殊领域之上的,它涉及群论、模运算、计算复杂性理论等多种数学知识。

#### 1. 群论

群(Group)由一个非空集合 G 组成,在集合 G 中定义了一个二元运算符“·”,并满足以下 4 个属性。

(1) 封闭性(Closure):对任意的 a,b∈G,有 a·b∈G。

(2) 结合律(Associativity):对任意的 a,b,c∈G,有 a·b·c=(a·b)·c=a·(b·c)。

(3) 单位元(Existence of identity):存在一个元素 i∈G(称为单位元),对任意元素,有 a·i=i·a=a。

(4) 逆元(Existence of inverse):对任意 a∈G,存在一个元素 $a^{-1}$∈G(称为逆元),使得 a·$a^{-1}$=$a^{-1}$·a=i(单位元)。

将满足上述性质的代数系统称为群,记为{G,·},如果一个群同时满足下面的交换律,则称之为交换群(或 Abel 群)。

(5) 交换律(Commutativity):对任意的 a,b∈G,有 a·b=b·a。

如果一个群的元素是有限的,则称该群为有限群,否则称之为无限群。有限群中的元素的幂运算如下:

$a^n = a \cdot a \cdot \cdots \cdot a$(n 个 a 作“·”运算)

$a^0 = i$(单位元)

$a^{-n} = (a^{-1})^n = (a^n)^{-1}$

如果群中每个元素都是其中某个元素 $g \in G$ 的某次幂 $g^k \in G$(k 为整数),则称该群是循环群。循环群总是交换群。在循环群中,认为元素 g 生成了群 G,或 g 是群 G 的生成元。一个群中元素 a 的阶就是使得 $a^n = i$(单位元)成立的最小的正整数 n。

群具有以下性质:

(1) 群中的单位元是唯一的;

(2) 群中每个元素的逆元是唯一的;

(3) 对任意的 $a,b,c \in G$,如果 $a \cdot b = a \cdot c$,则 $b = c$;同样,如果 $b \cdot a = c \cdot a$,则 $b = c$(消去律)。

**2. 模运算**

1) 带余除法

$\forall a \in Z^+$,可找出两个唯一确定的整数 q 和 r,使得 $a = qm + r, 0 \leqslant r < m$,q 和 r 分别称为用 m 除以 a 得到的商和余数(若 $r = 0$,则 $m \mid a$)。

**例 5.1** $a = 11; m = 7; 11 = 1 \times 7 + 4; r = 4$。

$a = -11; m = 7; -11 = (-2) \times 7 + 3; r = 3$。

如果 a 是一个整数,n 是一个正整数,则定义 a mod n 为 a 除以 n 的余数,因此对于任一整数 a,可表示为

$$a = [a/n] \times n + (a \bmod n)$$

其中,[X]表示不大于 X 的最大整数。

**例 5.2** $11 \bmod 7 = 4; -11 \bmod 7 = 3$。

2) 整数同余

**定义 5.1** 如果$(a \bmod n) = (b \bmod n)$,则称整数 a 和 b 模 n 同余,写作 $a \equiv b \pmod n$,整数 n 称为模数。

**例 5.3** $73 \bmod 23 = 4; 4 \bmod 23 = 4; 73 \equiv 4 \bmod 23$。

**注意:** 如果 $a \equiv 0 \bmod n$,则 $n \mid a$。

3) 模运算

a mod n 给出了 a 对模数 n 的余数,该运算称为模运算。

模运算具有如下性质。

(1) $m \mid (a-b) \rightarrow a \equiv b \pmod m$。

**证明:** $a - b = km$,k 为某一整数,由此可得 $a = b + km$,故 $a \bmod m = (b + km)$除以 m 的余数 $= b \pmod m$,即 a 和 b 分别除以 m 有相同的余数。“同余”二字的来源就在于此。

(2) 相对于某个固定模数 m 的同余关系,是整数间的一种等价关系。等价关系具有以下 3 点基本性质。

- 自反性:对任意整数 a,有 $a \equiv a \pmod m$。
- 对称性:如果 $a \equiv b \pmod m$,则 $b \equiv a \pmod m$。
- 传递性:如果 $a \equiv b \pmod m$,$b \equiv c \pmod m$,则 $a \equiv c \pmod m$。

(3) 模 m 运算将所有整数映射到整数集合$\{0,1,\cdots,(m-1)\}$,即模运算可实现对所

有整数进行分类,得到的整数集合称为剩余类集合;常用 $Z_m$ 表示所有整数模 m 后得到的剩余类集合,即 $Z_m=\{0,1,\cdots,(m-1)\}$。

**例 5.4** $Z_{10}=\{0,1,2,3,4,5,6,7,8,9\}$,即所有整数 mod 10 后得到的集合。

模运算就像普通的运算一样,它是可交换、可结合、可分配的。另外,对每个中间结果进行模 m 运算后进行四则运算,再进行模 m 运算,其作用与先进行全部普通运算,然后将所得结果再进行模 m 运算是一样的(以下 3 个式子是可逆的)。

(1) $[a(\bmod m)\pm b(\bmod m)]\bmod m=(a\pm b)(\bmod m)$。

(2) $[a(\bmod m)\times b(\bmod m)]\bmod m=(a\times b)(\bmod m)$。

(3) $[(a\times b)(\bmod m)+(a\times c)(\bmod m)](\bmod m)=[a\times(b+c)]\bmod m$。

**证明:** 假设 $a(\bmod m)=r_a$,$b(\bmod m)=r_b$,则可得 $a=r_a+jm$,j 为某一整数;同样,$b=r_b+km$,k 为某一整数。于是有

$$\begin{aligned}(a+b)\bmod m&=(r_a+jm+r_b+km)\bmod m\\&=[r_a+r_b+(k+j)m]\bmod m\\&=(r_a+r_b)\bmod m\\&=[a(\bmod m)+b(\bmod m)]\bmod m\end{aligned}$$

仅证明性质(1),其余性质请读者自行证明。

**例 5.5**

$11\bmod 8=3;15\bmod 8=7$

$[(11\bmod 8)+(15\bmod 8)]\bmod 8=(3+7)\bmod 8=2$

$(11+15)\bmod 8=26\bmod 8=2$

$[(11\bmod 8)-(15\bmod 8)]\bmod 8=(3-7)\bmod 8=-4\bmod 8=4$

$(11-15)\bmod 8=-4\bmod 8=4$

$[(11\bmod 8)\times(15\bmod 8)]\bmod 8=(3\times 7)\bmod 8=21\bmod 8=5$

$(11\times 15)\bmod 8=165\bmod 8=5$

指数模运算可以看作多次重复乘法,从而避免大的中间结果的出现。例如,为计算 $11^7\bmod 13$,可按如下方式计算:

$11^2=121\equiv 4\bmod 13$

$11^4=(11^2)^2\equiv 42\bmod 13\equiv 3\bmod 13$

$11^7=11\times 11^2\times 11^4\equiv(11\times 4\times 3)\bmod 13\equiv 132\bmod 13\equiv 2\bmod 13$

或可按如下方式计算:

$11^3=1331\equiv 5\bmod 13$

$11^7=11\times(11^3)^2\equiv 11\times 5^2\bmod 13\equiv 275\bmod 13\equiv 2\bmod 13$

### 3. 欧几里得算法

欧几里得算法(Euclidean Algorithm)是数论中的一项基本技术,它通过一个简单的过程确定两个正整数的最大公约数。欧几里得算法基于下面的定理。

对任何非负整数 a 和非负整数 b:

$$\gcd(a,b)=\gcd(b,a\bmod b)\quad(a\geqslant b)$$

可重复使用上式求出最大公约数。重复计算结束的条件是后一个整数等于 0,此时前一个数即为二者的最大公约数。

**例 5.6** gcd(18,12)=gcd(12,18 mod 12)=gcd(12,6)=gcd(6,12 mod 6)=gcd(6,0)=6。

欧几里得算法可描述为：假设整数 b>a>0，此处限制算法仅考虑正整数是可以接受的，因为 gcd(a,b)=gcd(|a|,|b|)。

EUCLID(a,b)：

(1) X←b;Y←a;

(2) 如果 Y=0，则返回 X=gcd(a,b);

(3) R=X mod Y;

(4) X←Y;

(5) Y←R;

(6) 返回步骤(2)。

### 4. 扩展的欧几里得算法

扩展的欧几里得算法(The Extended Euclidean Algorithm)不仅能确定两个正整数的最大公约数，如果这两个正整数互素，还能确定它们的逆元。

如果整数 n≥1，且 gcd(a,n)=1，那么 a 有一个模 n 的乘法逆元 $a^{-1}$。即对小于 n 的正整数 a，存在一个小于 n 的整数 $a^{-1}$，使得 $a\times a^{-1}\equiv 1 \bmod n$。

扩展的欧几里得算法可描述如下。

Extended EUCLID(a,n)：

(1) (X1,X2,X3)←(1,0,n);(Y1,Y2,Y3)←(0,1,a);

(2) 如果 Y3=0 返回 X3=gcd(a,n);无逆元;

(3) 如果 Y3=1 返回 Y3=gcd(a,n);Y2=a−1 mod n;

(4) Q=[X3/Y3];

(5) (T1,T2,T3)←(X1−Q·Y1,X2−Q·Y2,X3−Q·Y3);

(6) (X1,X2,X3)←(Y1,Y2,Y3);

(7) (Y1,Y2,Y3)←(T1,T2,T3);

(8) 返回步骤(2)。

### 5. 费马定理

费马(Fermat)定理：如果 p 是素数并且 a 是不能被 p 整除的正整数，那么

$$a^{p-1}\equiv 1 \bmod p$$

**例 5.7** a=7,p=19，求 $a^{p-1} \bmod p$。

解：$7^2=49\equiv 11 \bmod 19$

$7^4\equiv 11^2 \bmod 19\equiv 7 \bmod 19$

$7^8\equiv 7^2 \bmod 19\equiv 11 \bmod 19$

$7^{16}\equiv 11^2 \bmod 19\equiv 7 \bmod 19$

$a^{p-1}=7^{18}=7^{16}\times 7^2\equiv 7\times 11 \bmod 19\equiv 1 \bmod 19$

费马定理的另一种等价形式是：如果 p 是素数，a 是任意正整数，则对 gcd(a,p)=1，有

$$a^p\equiv a \bmod p$$

**例 5.8** p=5,a=3,$3^5=243\equiv 3 \bmod 5$。

## 5.2 常见的对称算法

对称加密算法犹如信息安全的坚固壁垒,为大量数据的快速加密与解密提供了高效且可靠的解决方案。从早期广泛应用的 DES,到如今在安全性和性能上更具优势的 AES,不仅见证了加密技术的发展历程,还持续为数据安全保驾护航,满足着不同行业和应用场景对数据保密性的严格要求。

### 5.2.1 数据加密标准

数据加密标准(Data Encryption Standard,DES)是由美国 IBM 公司于 20 世纪 70 年代中期的一个密码算法(LUCIFER 算法)发展而来的。1977 年 1 月 15 日,美国国家标准局(NBS)正式公布实施 DES,并得到了国际标准化组织(International Organization for Standardization,ISO)的认可。在国内,随着三金工程,尤其是金卡工程的启动,DES 算法在 POS、ATM、磁卡及智能卡(IC 卡)、加油站、高速公路收费站等领域被广泛应用,以实现关键数据的保密,如信用卡持卡人的 PIN 码加密传输,IC 卡与 POS 机之间的双向认证、金融交易数据包的 MAC 校验等,均用到了 DES 算法。

DES 算法的加密过程经过了 3 个阶段(图 5.3):首先,64 位的明文在一个初始置换 IP 后,比特重排产生了经过置换的输入,明文组被分成右半部分和左半部分,每部分各有 32 位,分别以 $L_0$ 和 $R_0$ 表示。后续阶段是由对同一个函数进行 16 次循环组成的,16 轮迭代称为乘积变换或函数 F,该函数既包含换位,又包含代替函数,将数据和密钥结合起来,最后一轮的输出由 64 位组成,其左边和右边两部分经过交换后就得到了预输出。最后阶段,预输出通过一个逆初始置换 $IP^{-1}$ 算法就生成了 64 位的密文结果。

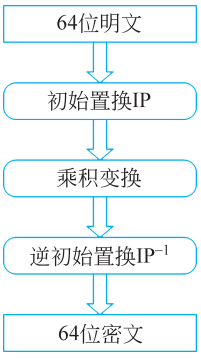

图 5.3 DES 加密处理略图

图 5.4 是 DES 全部 16 轮的加/解密结构图,其最上方的 64 位输入分组数据可能是明文,也可能是密文,视使用者要做加密或解密而定。而加密与解密的不同之处仅在于最右边的 16 个子密钥的使用顺序不同,加密的子密钥顺序为 $K_1,K_2,\cdots,K_{16}$,而解密的子密钥顺序正好相反,为 $K_{16},K_{15},\cdots,K_1$。

#### 1. 初始置换(IP)与逆初始置换($IP^{-1}$)

初始置换表和逆初始置换表分别如图 5.5 和图 5.6 所示,表中的数字代表初始置换或逆初始置换时 64 位输入分组的位序号,表中的位置代表置换后输出的位顺序。初始置换和逆初始置换都是基于普通型 P 盒的换位操作,这两个置换是彼此互逆的,如经过初始置换后,输入消息的第 1 位被置换到第 40 位的位置输出,再经过逆初始置换后,第 40 位又回到第 1 位的位置。

初始置换(IP)表中的位序号表现出这样的特征:整个 64 位按 8 行、8 列排列;最右边一列按 2、4、6、8 和 1、3、5、7 的次序排列;往左边各列的位序号依次为紧邻其右边一列各位的序号加 8。

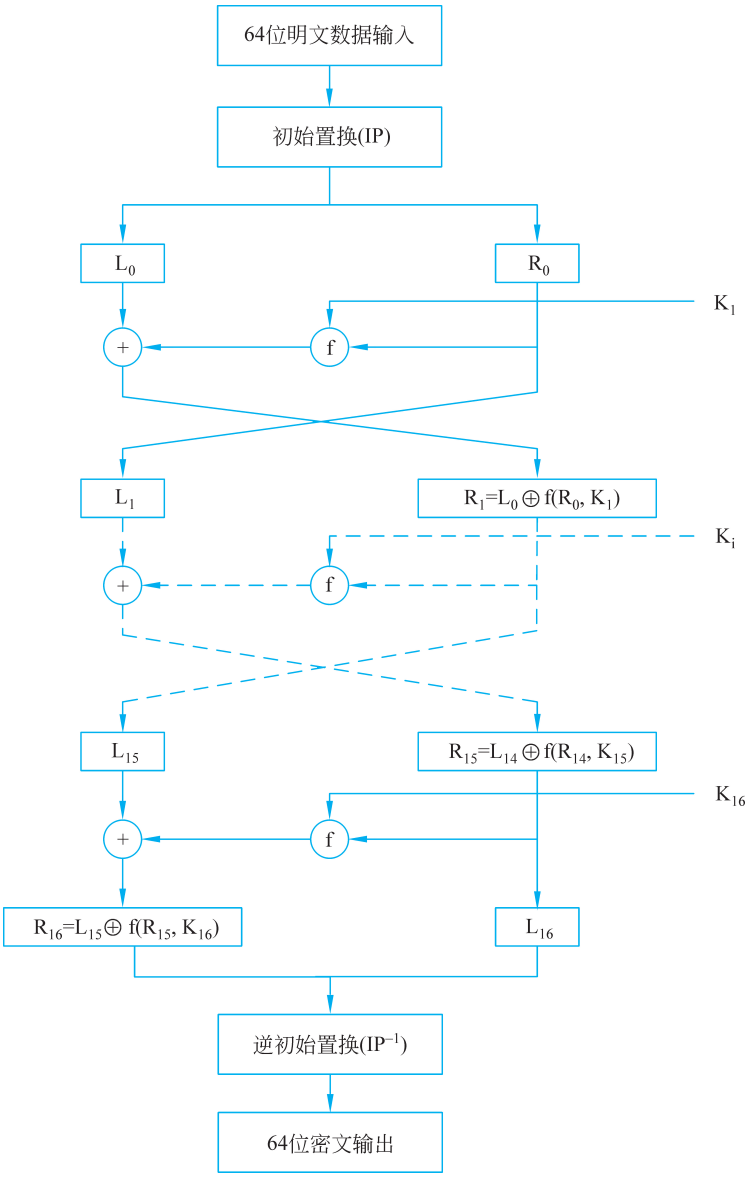


图 5.4 DES 加/解密流程

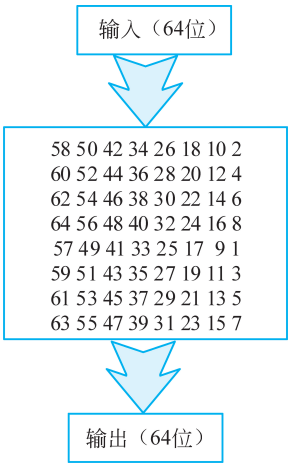


图 5.5 初始置换(IP)

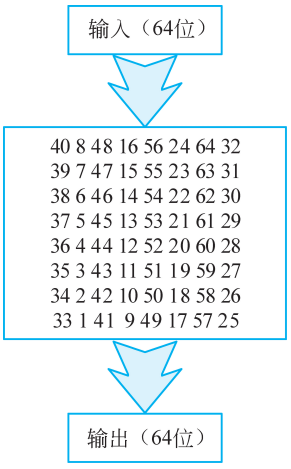


图 5.6 逆初始置换($IP^{-1}$)

逆初始置换($IP^{-1}$)则是初始置换的逆过程。相应地，表中位序号表现出这样的特征：整个 64 位依然按 8 行、8 列排列；左边第二列按 8、7、6、5、4、3、2、1 的次序排列；往右边隔一列的位序号依次为当前列各位的序号加 8；认为最右边一列的隔列为最左边一列。

### 2. 每个循环的详细过程

图 5.7 所示为一个循环的内部结构。每个 64 位的中间结果的左、右两部分被当成两个独立的 32 位数据处理。每轮变换的逻辑关系为

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

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

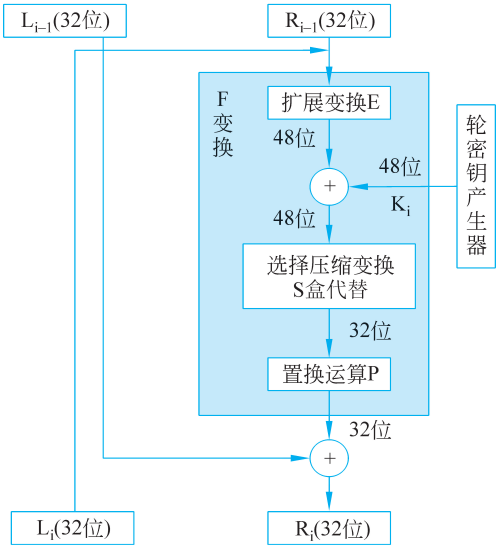


图 5.7 DES 算法的一轮迭代处理过程

在这个循环中使用的轮密钥 $K_i$ 的长度是 48 位，输入的 $R_{i-1}$ 是 32 位，先被扩展到 48 位，扩展操作的定义由图 5.8 决定。由图 5.8 可知：扩展变换 E 将 32 位输入扩展为 48 位输出，实际上 32 位输入中有 16 位被重用；将 48 位输出按 8 行 6 列的次序排列；排列时，将输入位序号按 32，1，2，…，31，32 的次序依次排列，但上一行的后两位依次

在下一行的前两位得到重用，如第一行的最后两位“4”“5”同时出现在第二行的前两位；则最后一行的下一行是第一行。扩展后得到的 48 位结果再与各 $K_i$ 进行异或，由此得到的 48 位结果再经过一个代替函数 S(S 变换)产生 32 位的输出，最后按照图 5.9 进行置换(P 置换)。

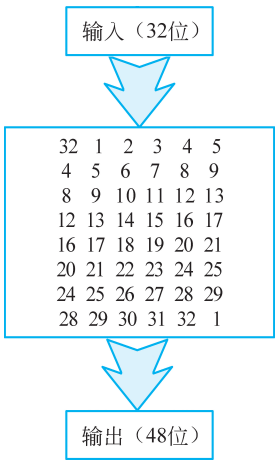


图 5.8 扩展变换 E

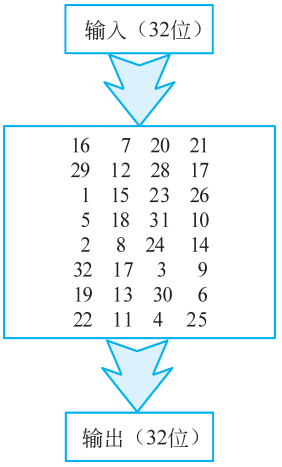


图 5.9 P 置换

S 盒在函数 F 中的作用如图 5.10 所示，代替(S 变换)由一组共 8 个 S 盒完成，其中每个 S 盒都接收 6 位的输入，并产生 4 位的输出，对应的变换由 8 个表定义，见表 5-1 至表 5-8。

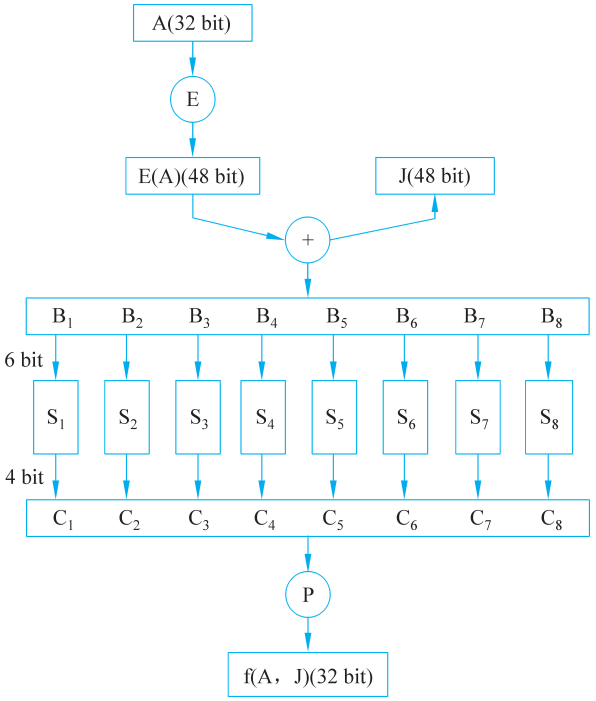


图 5.10 $F(R_{i-1}, K_i)$ 函数的计算