CHAPTER 3

第3章 网络安全基础

3.1 加密算法

加密是网络安全的基础，加密过程如图 3.1 所示，发送端通过加密算法 E 和加密密钥 K_1 将明文 P 转换成密文 $Y(Y=E_{K_1}(P))$，接收端通过解密算法 D 和解密密钥 K_2 重新将密文还原成明文 $P(P=D_{K_2}(Y)=D_{K_2}(E_{K_1}(P)))$。密文 Y 虽然和明文 P 相关联，在不知道解密算法 D 和解密密钥 K_2 的情况下，是无法通过密文 Y 解析出明文 P，因此，只要不让黑客获悉解密算法 D 和解密密钥 K_2，即使让黑客截获密文 Y，黑客也无法获得明文 P，因而也无法实现窃取信息的攻击目标。如果加密算法所使用的加密密钥 K_1 等于解密密钥 K_2，这种加密算法被称为对称密钥加密算法，否则，被称为不对称密钥加密算法，目前，常见的对称密钥加密算法有数据加密标准(Data Encryption Standard，DES)、高级加密标准(Advanced Encryption Standard，AES)，不对称密钥加密算法有 RSA(Rivest-Shamir-Adleman)公开密钥加密算法。

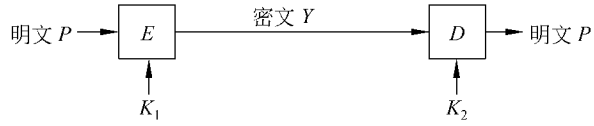


图 3.1 加密解密过程

3.1.1 对称密钥加密算法

对称密钥加密算法由五个元素组成：明文 P、密文 Y、加密算法 E、解密算法 D 和密钥 K(加密密钥 K_1=解密密钥 $K_2=K$)。由于加密算法和解密算法的运算过程是互逆的，因此，很容易从一种操作过程推出另一种操作过程，这样，密文 Y 的安全性就取决于加密算法和密钥的安全性。由于黑客可以截获到密文 Y，甚至可以得到一部分密文 Y 及对应的明文 P，对称密钥加密算法必须保证在黑客即使获得一部分密文Y及对应的明文

P 的条件下，也无法推导出加密算法和密钥。如果加密算法和密钥长期不变的话，黑客获得加密算法和密钥只是时间问题，为保证对称密钥加密算法的安全性，或者定期更换密钥 K，或者定期更换加密算法 E 及对应的解密算法 D。在目前互联网时代，大量用户之间需要实现动态安全传输，为了实现动态安全传输，两个用户之间必须动态同步加密算法和密钥，在用软件实现加密、解密运算过程的情况下，不同的加密、解密算法需要不同的用于实现对应运算过程的程序块，而密钥只是实现加密、解密算法的程序块的输入参数，因此，实现密钥动态同步比实现加密、解密算法动态同步要简单得多。基于这样的理由，往往采用公开的、标准的加密、解密算法，而把密文 Y 的安全性完全基于密钥的安全性。

加密体制的 Kerckhoff's 原则是：所有加密、解密算法都是公开的，保密的只是密钥。

一旦加密、解密算法公开，在黑客能够获得一部分密文 Y 及对应的明文 P 的条件下仍然保证密钥安全性的前提是：或者黑客无法在知道加密、解密算法的情况下，通过有限的密文 Y 及对应的明文 P 推导出密钥 K。或者每一个密钥只进行一次加密运算，而且每一个密钥都是从一个足够大的密钥集中随机产生，密钥之间没有任何相关性。第一种情况要求足够复杂的加密、解密运算过程，而且这种运算过程必须经过广泛测试，保证黑客无法破解，即无法通过有限的密文和明文对解析出密钥。第二种情况要求一次一密钥，而且密钥必须在足够大的密钥集中随机产生，确保密钥之间没有相关性，黑客无法从已知的有限密钥序列推导出下一次用于加密运算的密钥，但对加密、解密算法的复杂性没有要求。

1. 流密码体制

流密码体制就是一次一密钥的加密运算过程，如图 3.2 所示，发送端在密钥集中随机产生一个与明文 P 相同长度的密钥 K，密钥 K 和明文 P 进行异或运算后得到密文 Y。接收端用同样的密钥 K 和密文 Y 进行异或运算，还原出明文 P。如果密钥集足够大，每一次加密运算的密钥不同，且这些密钥之间不存在相关性，这种密码体制是最安全的。但一是密钥集总是有限的，二是计算机很难真正在密钥集中随机产生密钥，密钥之间无法做到没有任何相关性，三是发送端和接收端必须同步密钥。在这些限制下，流密码体制的安全性就打了折扣。图 3.3 是无线局域网 WEP 安全机制使用的加密、解密运算过程，用于发送端加密运算的密钥由伪随机数生成器产生，原始密钥 K 和初始向量 IV 作为伪随机数种子，为了使接收端产生相同的密钥，必须使接收端的伪随机数生成器输入相同的伪随机数种子，为了做到一次一密钥，每一次产生密钥时，必须对伪随机数生成器输入不同的随机数种子。在无线局域网 WEP 安全机制中，原始密钥 K 是发送端和接收端共同约定的，安全传输过程中是不变的，双方需要同步的只是初始向量 IV，发送端每一次必须选择不

图 3.2 流密码体制加密、解密过程

同的初始向量作为伪随机数种子，并以明文方式将该次加密运算选择的初始向量发送给接收端。由于一是用伪随机数生成器来产生密钥，无法保证这些密钥之间没有任何相关性。二是作为伪随机数种子一部分的原始密钥 K 是不变的，增加了这些密钥之间的相关性。三是初始向量的长度只有 24 位，密钥集中的密钥数$\leqslant 2^{24}$，如果原始密钥较长一段时间保持不变，加密用的密钥很容易重复。

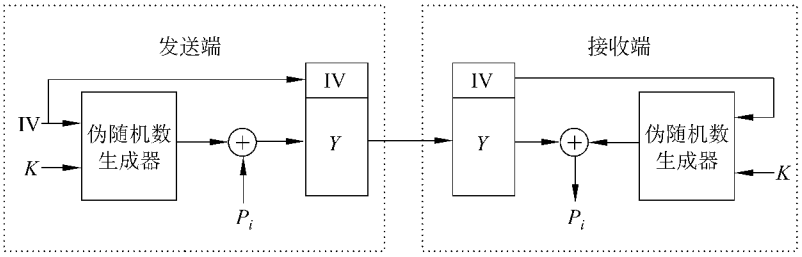


图 3.3　WEP 安全体制加密、解密过程

2. 分组密码体制

分组密码体制下，同一密钥用于多次加密运算过程，黑客可能截获这些用同一密钥加密后的密文，甚至可能获得了一部分密文对应的明文，加密、解密算法必须保证黑客无法通过密文，甚至有限的密文、明文对推导出密钥。这就要求分组密码体制下的加密、解密算法足够复杂，不能是流密码体制下简单的运算过程，因为类似图 3.3 所示的加密操作(异或运算)很容易让黑客根据密文、明文对推导出密钥($Y=P\oplus K$. $P\oplus Y=P\oplus P\oplus K=K$)。分组密码体制下的加密运算过程，首先将明文分割成固定长度的数据组，然后单独对每一组数据进行加密运算，产生和数据组长度相同的密文，密文序列和明文分组后产生的数据组序列一一对应。解密运算过程就是将密文还原为对应数据组的过程。将数据组变成密文的加密过程通过多级操作完成，而每一级操作所进行的运算通常是替代与置换。

1) 替代运算

替代是将数据组中的二进制数分段，每一段二进制数用对应的编码代替。假定 8 位数据组为 11010010，如果采用下述替代规则{00,110},{01,010},{10,000},{11,100}，进行替代运算后的二进制位流为 100 010 110 000。当然，也可以用同样的替代规则完成反向替代运算，但必须指出：用二位二进制编码替代三位二进制编码的前提是三位二进制编码集包含的编码数量小于等于 4(二位二进制编码集的最大编码数量)。

2) 置换运算

置换运算是按照置换规则重新排列数据组中二进制数的顺序，假定 8 位数据组为 11010010，如果置换规则为{0,2,5,7,3,1,6,4}，则数据组的置换运算过程如图 3.4 所示。置换规则中最高位对应值为 0，表明原来 8 位数据组中最高位被置换到第 0 位，第 6 位对应值为 2，表明原来 8 位数据组中第 6 位被置换到第 2 位，以此类推。

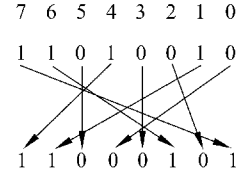


图 3.4　置换运算过程

3）Feistel 分组密码结构

Feistel 分组密码结构给出了分组密码体制下的加密运算过程，如图 3.5 所示，它的输入是分割明文后产生的长度固定为 $2W$ 位的数据组和密钥 K，输出是 $2W$ 位的密文。整个加密运算过程由 n 次迭代完成，上一次迭代运算的结果作为下一次迭代运算的输入，以密钥 K 为原始密钥经过子密钥生成运算产生各次迭代运算需要的子密钥集$\{K_1, K_2, \cdots, K_i, \cdots, K_n\}$。在第一次迭代运算中，数据组中的数据分成长度各为 W 位的左右两部分：L_0 和 R_0，R_0 和子密钥 K_1 作为迭代函数 F 的输入，迭代函数 F 的输出和 L_0 进行异或运算，运算的结果成为下一次迭代运算的 R_1，而 R_0 成为下一次迭代运算的 L_1，迭代函数 F 的功能通过多次替代和置换运算实现。第 n 次迭代运算后产生的结果 L_n 和 R_n 分别成为构成密文的 R_{n+1} 和 L_{n+1}。解密运算是图 3.5 所示运算过程的逆过程，从构成密文的 R_{n+1} 和 L_{n+1} 导出 L_n 和 R_n，L_n 和 K_n 作为迭代函数 F 的输入，迭代函数 F 的输出和 R_n 进行异或运算，结果作为下一次迭代运算的 L_{n-1}，L_n 作为下一次迭代运算的 R_{n-1}。

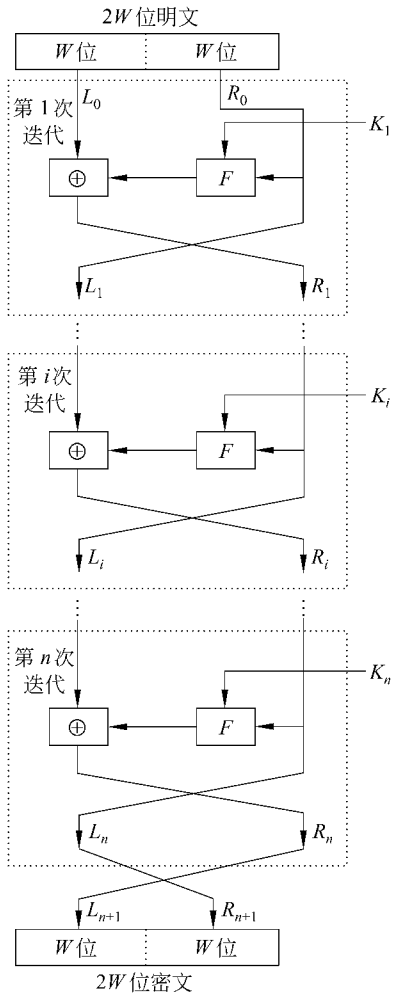


图 3.5 Feistel 分组密码结构

图 3.5 所示的分组密码加密运算过程的安全性取决于以下几个因素。

- 数据组长度：增加数据组的长度，有利于提高加密算法的安全性（不容易通过明文、密文对解析出密钥），但增加运算复杂性。
- 密钥长度：增加密钥的长度，有利于提高加密算法的安全性，但增加运算复杂性。
- 迭代次数：增加迭代次数，有利于提高加密算法的安全性，但增加运算复杂性。目前选择 16 次迭代次数就是综合考虑加密算法安全性和运算复杂性的结果。
- 子密钥序列生成算法：采用复杂的子密钥序列生成算法，有利于提高加密算法的安全性。
- 迭代函数：采用复杂的迭代函数，有利于提高加密算法的安全性。目前采用的迭代函数由多级替代和置换运算实现。

3. 对称密钥加密算法举例

1）数据加密标准

数据加密标准（Data Encryption Standard，DES）采用图 3.5 所示的分组密码结构，明文被分割成固定 64 位长度的数据组 X，数据组 X 在开始图 3.5 所示的迭代运算前，先进行初始置换运算（IP），即 $\mathrm{IP}(X)=(L_0, R_0)$。加密运算过程中：

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

解密运算是加密运算的逆过程。

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

迭代函数的运算过程如图 3.6 所示，首先通过一个扩展函数 E 将右半部分(R_{i-1})32 位二进制数扩展为 48 位二进制数，由于 48 位二进制数是由 32 位二进制数扩展后产生的，因此，它的编码数量不是 2^{48}，而是 2^{32}，这样的编码集可以由 32 位二进制数的编码进行替代。将 32 位右半部分(R_{i-1})扩展为 48 位的目的是为了和 48 位的子密钥 K_i 进行异或运算。和子密钥异或运算后产生的 48 位结果被分成 8 组，每组 6 位，每一组结果单独进行一次替代运算 S_i，用 4 位编码替代每组 6 位结果。这里能够用 4 位编码替代 6 位结果的原因是 48 位的结果只组成 2^{32}个编码的编码集。对 8 组 6 位结果进行替代运算后产生 8 组 4 位结果，并由它们构成 32 位的数据组，对这 32 位数据组根据置换规则 P 进行置换运算，最终产生迭代运算结果 $F(R_{i-1}, K_i)$。

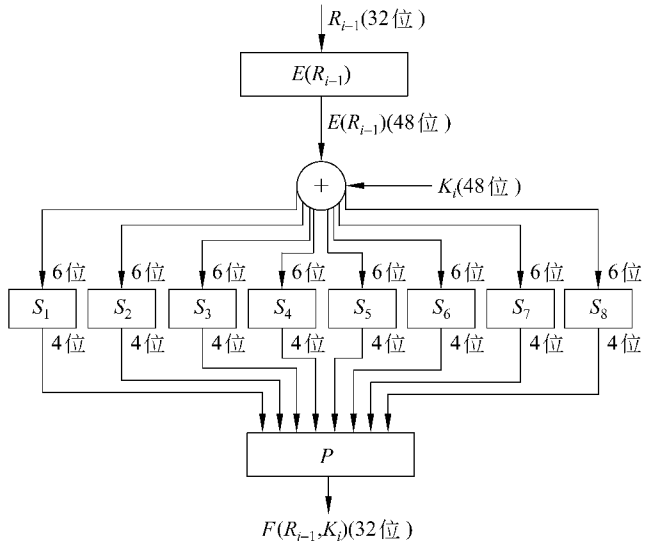


图 3.6　迭代函数运算过程

参与每一次迭代运算的子密钥是不同的，它由密钥 K 和子密钥生成算法产生。由密钥 K 产生子密钥序列的运算过程如图 3.7 所示。64 位密钥由 8 字节组成，每一个字节中，只有 7 位是密钥，另 1 位是其他 7 位的奇偶校验位，因此，64 位密钥 K 中真正作为密钥的只有 56 位，提取操作所完成的功能就是从 64 位密钥 K 中提取出真正用作密钥的 56 位二进制数。56 位密钥根据置换规则 P_1 进行置换运算，得到的结果被分成左右两部分，每部分 28 位，这两部分结果分别左移 n_1 位。产生不同的子密钥时，左移的位数不同，分别由 $n_1, n_2, \cdots, n_{16}$表示。如图 3.7 所示，左移后的结果作为计算下一个子密钥时的左移寄存器的输入。从左移后得到的 56 位结果中提取 48 位，对提取出的 48 位根据置换规

则 P_2 进行置换运算，置换运算结果就是本次的子密钥 K_i。

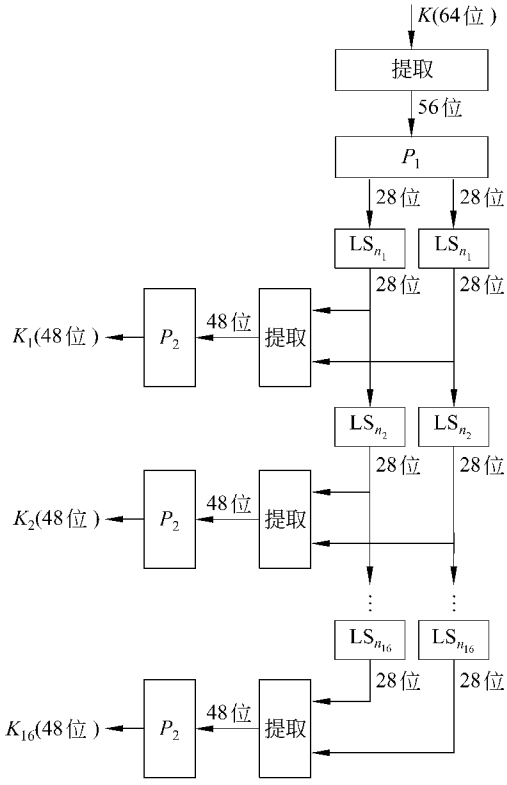


图 3.7 子密钥序列生成过程

由于 56 位长度的密钥只能产生 2^{56} 个不同的密钥，因此，在目前由高性能计算机进行密钥破解的情况下，即使采用穷举法这样简单、传统的密钥破解方法，可以在几个小时内根据密文和明文对推导出密钥 K，因此，采用 56 位密钥的 DES 加密算法并不是一种安全的对称密钥加密算法，目前实际采用的 DES 加密算法是三重 DES，加密解密过程如图 3.8 所示，密文 $Y=E_{K_3}(D_{K_2}(E_{K_1}(P)))$，明文 $P=D_{K_1}(E_{K_2}(D_{K_3}(Y)))=D_{K_1}(E_{K_2}(D_{K_3}(E_{K_3}(D_{K_2}(E_{K_1}(P))))))=D_{K_1}(E_{K_2}(D_{K_2}(E_{K_1}(P))))=D_{K_1}(E_{K_1}(P))=P$。这种加密过程相当于采用了3×56 位密钥的 DES 加密算法。对于安全性要求不是特别高的应用环境，可以采用两组 56 位密钥的三重 DES 加密算法，这种情况下，图 3.8 的密钥 K_3 由密钥 K_1 代替。

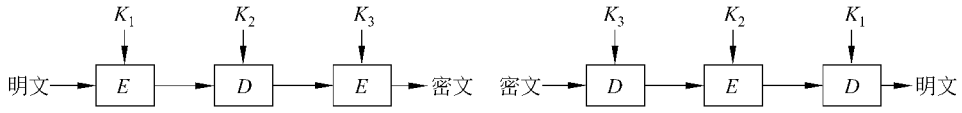


图 3.8 三重 DES

2）高级加密标准

高级加密标准(Advanced Encryption Standard，AES)将明文分割成固定 128 位长度

的数据组，密钥可以是 128 位、192 位或者 256 位（这里只讨论 128 位密钥的情况），和 DES 不同，AES 不再将数据组分成等长的两部分，而是以 128 位为单位进行迭代运算，整个加密运算过程如图 3.9 所示。128 位密钥经过扩展运算成为 11×128 位，构成子密钥序列$\{K_0, K_1, \cdots, K_{10}\}$，每一个子密钥和数据组长度相同，为 128 位。AES 中子密钥序列由一组 32 位的字组成，$W[i]$，$0 \leqslant i \leqslant 43$，在这一组字中，子密钥 K_0 对应的 4 个字为 $W[i]$（$0 \leqslant i \leqslant 3$），以此类推，子密钥 K_j 对应的 4 个字为字 $W[i]$（$j \times 4 \leqslant i \leqslant j \times 4 + 3$）。在开始第 1 次迭代运算前，数据组先和子密钥 K_0 进行异或运算，运算结果作为第 1 次迭代运算的输入。除第 10 次迭代运算外，每一次迭代运算过程包括逐字节替代运算、逐行置换运算、逐列变换运算和与子密钥 K_i 的异或运算。为了实现这些运算，128 位的数据组被分成 16 字节，16 字节又被组织成 4×4 的矩阵，如图 3.10 所示，A_i 是构成 128 位数据组中的某个字节，$S_{r,c}$是对应的 A_i 在矩阵中的表示，其中，r 是行号，c 是列号。构成子密钥 K_i 的每一个字对应矩阵中的一列，4 个字对应矩阵中的 4 列。

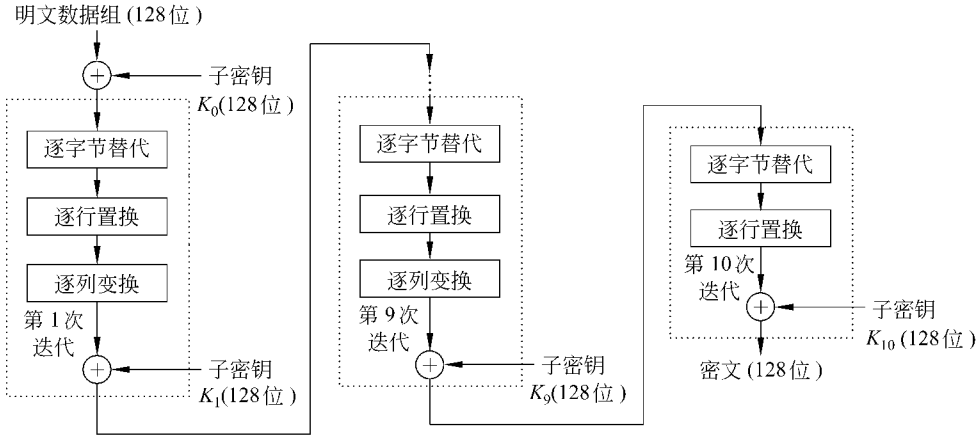


图 3.9　AES 加密运算过程

$$\begin{bmatrix} A_0 & A_4 & A_8 & A_{12} \\ A_1 & A_5 & A_9 & A_{13} \\ A_2 & A_6 & A_{10} & A_{14} \\ A_3 & A_7 & A_{11} & A_{15} \end{bmatrix} \Rightarrow \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}$$

图 3.10　矩阵表示的 128 位数据组

逐字节替代运算是指矩阵中每一个字节独立进行替代运算，矩阵中每一个字节以该字节值为索引，检索替代表，找到对应的 8 位替代值予以替代，如原值 00H 用 63H 替代，原值 FFH 用 16H 替代。

逐行置换运算过程如图 3.11 所示，置换规则是：$S'_{r,c} = S_{r,c+\text{shift}(r) \bmod 4}$，其中 shift(0)＝0，shift(1)＝1，shift(2)＝2，shift(3)＝3，因此，

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \xrightarrow{\text{逐行置换运算}} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

图 3.11 逐行置换运算过程

$$\begin{aligned} S'_{0,c} &= S_{0,c} \\ S'_{1,c} &= S_{0,c+1 \bmod 4} \\ S'_{2,c} &= S_{0,c+2 \bmod 4} \\ S'_{3,c} &= S_{0,c+3 \bmod 4} \end{aligned}$$

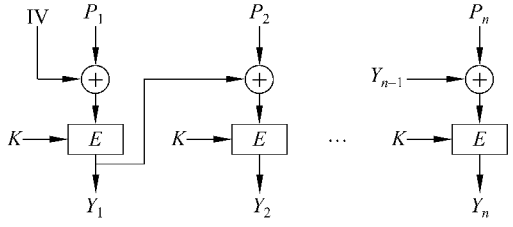
逐列变换运算过程如图 3.12 所示。

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

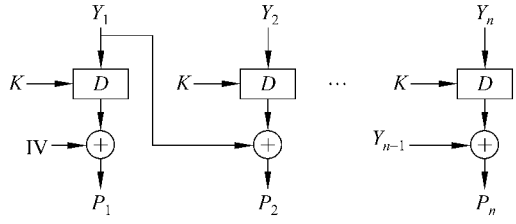
图 3.12 逐列变换运算过程

3）加密分组链接

分组密码体制对由明文分割产生的数据组独立进行加密运算，如果两组数据组相同，且这两组数据组加密运算时使用的密钥也相同，这两组数据组将生成相同的密文。对于对称密钥加密算法，很难对不同的数据组使用不同的密钥，因此，当明文有规则重复时，加密后的密文也同样有规则重复，这样有利于破解密文，降低加密算法的安全性，为了避免发生相同数据组加密后产生相同密文的情况，采用图 3.13 所示的加密分组链接(Cipher-Block Chaining，CBC)的方法。



(a) 加密运算过程



(b) 解密运算过程

图 3.13 加密分组链接模式

在图 3.13 所示的加密分组链接模式中，加密运算模块的输入不是分割明文后产生的数据组 P_i，而是数据组 P_i 和前一次加密运算后的结果 Y_{i-1} 异或运算后的结果。

$$Y_i = E_K(Y_{i-1} \oplus P_i)$$

第 1 组数据组和初始向量 IV 的异或运算结果作为加密运算模块的输入。

$$Y_1 = E_K(\mathrm{IV} \oplus P_1)$$

在接收端，为了还原数据组 P_i，必须进行以下运算过程。

$$D_K(Y_i) = D_K(E_K(Y_{i-1} \oplus P_i) = Y_{i-1} \oplus P_i$$

$$Y_{i-1} \oplus D_K(Y_i) = Y_{i-1} \oplus Y_{i-1} \oplus P_i = P_i$$

因此，数据组 P_i 是对应密文 Y_i 解密运算后的结果和前一组数据组对应密文异或运算后的结果。同样，

$$P_1 = \mathrm{IV} \oplus D_K(Y_1)$$

因此，发送端和接收端必须具有相同的初始向量 IV。

由于加密分组链接模式中每一组数据组和前一组数据组对应的密文异或运算后作为加密运算模块的输入，因此，即使密钥相同，两组相同数据组加密运算后产生的密文也不会相同，增加了加密算法的安全性，因此，加密分组链接模式在分组密码体制中得到了广泛应用。

4. 对称密钥加密算法的密钥分配过程

由于加密、解密算法是标准的、公开的，安全性就完全基于密钥的保护上。由于通信双方都需要拥有共同的密钥，而且，如果长期使用同一密钥的话，泄露密钥的可能性就会增加，因此，如何分配密钥就成了一大难题。可靠的办法是让信使携带密封的密钥给互相通信的各个用户，但在网络如此发达的今天，仍然采用这种笨方法有点不合时宜，而且，如果经常变换通信对象，为了安全，也定期更换密钥的话，这种方法也很难操作。因此，需要采用通过网络分配密钥的方法。

目前常用的通过网络分配密钥的方法是建立一个密钥分配中心(Key Distribution Center，KDC)，由 KDC 负责为用户分配密钥。需要 KDC 分配密钥的用户先注册到 KDC，获得和 KDC 通信时使用的密钥。当某个注册用户 A 希望和另一个注册用户 B 用密文通信时，通过向 KDC 申请，获得这一次通信所使用的密钥，如图 3.14 所示。

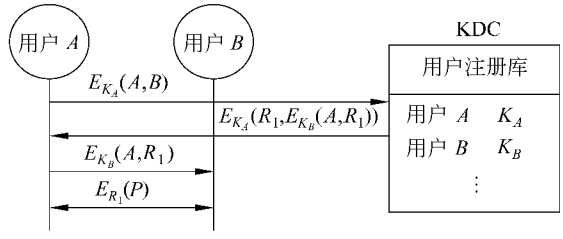


图 3.14　密钥分配过程

(1) 用户 A 将希望和用户 B 通信的请求用 KDC 分配给自己的密钥 K_A 进行加密，然后将加密后的请求：$E_{K_A}(A,B)$ 发送给 KDC。

(2) KDC 用随机数生成算法产生一个密钥 R_1 供用户 A 和用户 B 通信时使用，这种

密钥只用于这一次通信，称之为“一次一密”。KDC 将密钥 R_1 用分配给用户 A 的密钥加密后发送给用户 A，除了密钥 R_1，KDC 还将用分配给用户 B 的密钥 K_B 加密用户 A 希望和用户 B 通信的请求及密钥 R_1，并将加密后的内容包含在用 K_A 加密的数据中。因此，当用户 A 用 KDC 分配给它的密钥 K_A 解密 KDC 发送给它的密文后，得到由 KDC 生成并分配给这一次用户 A 和用户 B 通信时使用的密钥 R_1，同时得到用 KDC 分配给用户 B 的密钥加密的用户 A 希望和用户 B 通信的请求及密钥 R_1。由于是用分配给用户 B 的密钥加密的密文，用户 A 无法解密，向用户 B 转发该密文。

(3) 用户 B 用 KDC 分配给它的密钥 K_B 解密该密文，获知用户 A 希望和它通信并使用 KDC 分配的一次性密钥 R_1，双方用 R_1 作为密钥，对相互传输的数据进行加密、解密操作。

为了提高安全性，KDC 分配给注册用户的密钥如 K_A、K_B 也需要经常更换。

3.1.2 公开密钥加密算法

1. 公开密钥加密算法原理

公开密钥加密算法使用不同的加密密钥和解密密钥，它的加密、解密过程如图 3.15 所示，发送者用加密算法 E 和密钥 P_K 对明文 P 进行加密，接收者用解密算法 D 和密钥 S_K 对密文 Y 进行解密。加密密钥 P_K 是公开的，而解密密钥 S_K 是保密的，只有接收者知道，用于解密用公开密钥加密的密文，习惯上将加密密钥称为公钥，而将解密密钥称为私钥。

$$Y=E_{P_K}(P)$$

$$D_{S_K}(Y)=D_{S_K}(E_{P_K}(P))=P$$

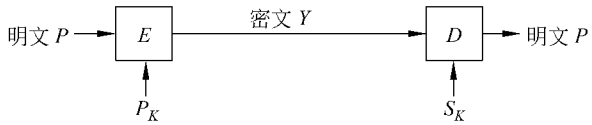


图 3.15 公开密钥加密算法的加密、解密过程

公开密钥加密算法的原则是：

(1) 容易成对生成密钥 P_K 和 S_K。

(2) 加密和解密算法是公开的，而且可以对调：

$$D_{S_K}(E_{P_K}(P))=E_{P_K}(D_{S_K}(P))=P$$

(3) 加密和解密过程容易实现。

(4) 从计算可行性讲，无法根据 P_K 推导出 S_K。

(5) 从计算可行性讲，无法根据 P_K 和密文 Y 推导出明文 P。

2. 公开密钥加密算法举例

目前使用最广泛的公开密钥加密算法是 RSA 和 Diffie-Hellman 公开密钥加密算法。

1) RSA 公开密钥加密算法

RSA(Rivest-Shamir-Adleman)公开密钥加密算法也是一种分组密码算法，每一组数