第 5 章　散列函数、消息摘要与消息认证码

随着数字签名技术越来越被广泛地理解和使用，世界许多国家都在竞相开发自己的签名标准，以便使数字签名的使用与应用更标准化。电子商务中的一些电子应用，如电子邮件和金融交易都需要使用数字签名。当传输敏感信息或需要某些安全服务时（如对发送方的身份验证、消息完整性和不可否认性），都需要对电子邮件进行数字签名。在金融交易中，要么是直接转账，要么是服务和货物的交换，这些金融交易都受益于数字签名。对消息摘要签名而非对消息本身签名，通常能提高处理的效率，因为消息摘要通常要比消息小得多。

在电子商务中，常常需要通信双方互相验证彼此的身份。一种实用的身份验证方法是使用单向散列函数的加密验证协议。通过填充全零，其包括 1 位标志位、十六进制的原始消息长度，将可变长度的消息映射到合适的位值来实现按固定长度对消息进行分组。为了能方便地将消息按某一固定长度进行分组，则必须进行一些适当地填充。为了使用散列函数计算消息摘要，本章将介绍几种与之相关的算法。本章所涉及的散列函数有：**类 DES 消息摘要计算**（DES-like Message Digest Computation，DMDC）（1994）、MD5（1992）和 SHA-1（1995）。

5.1　DMDC 算法

DMDC 使用**数据加密标准**（Data Encryption Standard，DES）变种作为单向散列函数。1994 年，CDMA 蜂窝移动通信系统引入这一方案，用来计算 18 位的认证数据。DMDC 将消息按 64 位进行分组。DMDC 散列函数能生成可变大小的消息摘要，这些消息摘要可以是 18 位、32 位 64 位或 128 位。DMDC 方案非常适用于数字签名，因此，采用它可以增强 Internet 的安全性。

先将要签名的消息按 64 位进行分组，得到的分组序列为：

$$M_1, M_2, \cdots, M_t$$

为了能方便地分割消息，需要制定适当的填充规则。相邻的消息分组与自生成的密钥散列在一起。更好的方法是使正确消息的一个分组长度（64 位）正好等于密钥的长度。

图 5.1 给出了 M＝192 位，使用 DMDC 计算散列码的典型方案。

5.1.1　密钥调度

在 CDMA 移动系统中，一个认证问题是如何通过移动台与基站之间的交换信息，来确认该移动台的身份。当消息接入参数的认证字段被设置为“01”时，移动台通过在接入信道发送注册请求消息来试着注册，并执行认证过程。为了计算移动台注册的认证数据，消息值应为 152 位，包括 RAND（32 位）、ESN（32 位）、MIN（24 位）和 SSD-A（64 位）：

- RAND：认证随机挑战值。

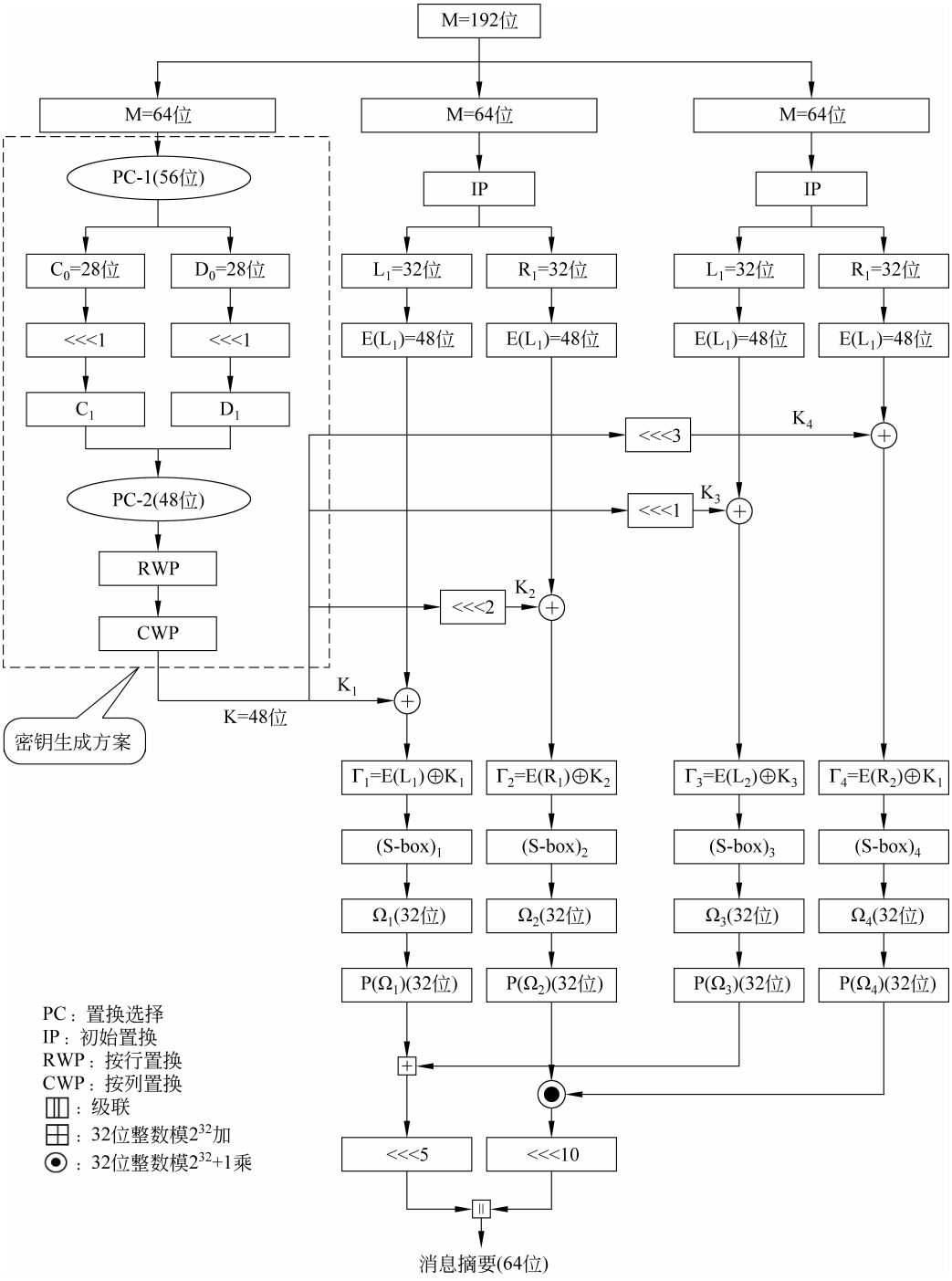


图 5.1　M＝192 位的 DMDC 算法

- ESN：电子序列号。
- MIN：移动台识别号。
- SSD-A：支持认证过程的共享机密数据。

192 位的值是由长度为 152 位的消息和 40 位的填充组成。假设 M_1、M_2 和 M_3 是 192 位填充消息的分解。M_1=64 位将作为图 5.1 中密钥生成方案的输入。置换选择 2 运算将产生 48 位密钥,排列成如下所示的 6×8 数组:

输入(按列)

⇓

1	7	13	19	25	31	37	43
2	8	14	20	26	32	38	44
3	9	15	21	27	33	39	45
4	10	16	22	28	34	40	46
5	11	17	23	29	35	41	47
6	12	18	24	30	36	42	48

按行置换

5	11	17	23	29	35	41	47
1	7	13	19	25	31	37	43
3	9	15	21	27	33	39	45
6	12	18	24	30	36	42	48
2	8	14	20	26	32	38	44
4	10	16	22	28	34	40	46

按列置换

11	35	5	47	17	41	29	23
7	31	1	43	13	37	25	19
9	33	3	45	16	39	27	21
12	36	6	48	18	42	30	24
8	32	2	44	14	38	26	20
10	34	4	46	16	40	28	22

→输出(按行)

因此,表 5.1 给出了根据M_1 计算生成的 48 位密钥。

表 5.1 利用行/列置换生成的 48 位密钥

11	35	5	47	17	41	29	23	7	31	1	43	13	37	25	19
9	33	3	45	15	39	27	21	12	36	6	48	18	42	30	24
8	32	2	44	14	3/8	26	20	10	34	4	46	16	40	28	22

例 5.1 假设将 192 位填充消息按 64 位进行分组,所得到的分组如下:

$$M_1 = 7a138b2524af17c3$$

$$M_2 = 17b439a12f51c5a8$$

$$M_3 = 51cb360000000000$$

注意,在这个示例中,没有插入 1 位标志位和十六进制的消息长度。下面给出了按行/列置换生成的 48 位密钥。假定将第一个数据分组 M_1 作为密钥输入。利用表 3.1(PC-1),将 M_1

分成两个分组：

$$C_0 = a481394, \quad D_0 = e778253$$

如表 3.2 所示，将C_0和D_0分别向左移动 1 位，得到C_1 和D_1：

$$C_1 = 4\ 902\ 729, \quad D_1 = cef04a7$$

利用表 3.3(PC-2)，计算所得的 48 位压缩密钥为：

$K_0 = 058c4517a7a2$。

最后，使用表 5.1，按行/列置换计算所得的 48 位密钥为：

K=5458c42bcc07

这是为 M_2 和 M_3 提供的密钥分组，如例 5.2 所示。

例 5.2　参考图 5.1，处理 $M_2 = 17b439a12f51c5a8$，$M_3 = 51cb360000000000$ 的过程如下所示：

利用表 3.4，将 M_2 和 M_3 分成：

$$L_1 = 6027537d \quad R_1 = ca9e9411$$

和

$$L_2 = 03050403 \quad R_2 = 02040206$$

使用表 3.5 对这四个数据分组进行扩展，得到：

$$E(L_1) = b0010eaa6bfa$$
$$E(R_1) = e554fd4a80a3$$
$$E(L_2) = 80680a808006$$
$$E(R_2) = 00400800400c$$

通过行/列置换得到的 48 位密钥 K＝5458c42bcc07，应向左移动 0 位、2 位、1 位和 3 位，使得

$K_1 = 5458c42bcc07$(移动 0 位)
$K_2 = a8b18857970e$(移动 2 位)
$K_3 = 516310af301d$(移动 1 位)
$K_4 = a2c6215e603a$(移动 3 位)

将这四个密钥与扩展分组进行异或运算，得到

$$\Gamma_1 = E(L_1) \oplus K_1 = c459ca81a7fd$$
$$\Gamma_2 = E(R_1) \oplus K_2 = b437ede5b0be$$
$$\Gamma_3 = E(L_2) \oplus K_3 = 28d982d71808$$
$$\Gamma_4 = E(R_2) \oplus K_4 = a286295e2036$$

将这 4 个 $\Gamma_i (1 \leqslant i \leqslant 4)$分别作为 S-盒的输入。利用表 3.6，计算所得的 S 盒输出 Ω_i 为：

$$\Omega_1 = a4064766$$
$$\Omega_2 = 1d1dabb8$$
$$\Omega_3 = f89d0b16$$
$$\Omega_4 = dabaae4d$$

对每个 Ω_i 应用表 3.7 的运算，得到：

$$P(\Omega_1) = 00f63638$$
$$P(\Omega_2) = 9f2874d3$$
$$P(\Omega_3) = 96aab362$$
$$P(\Omega_4) = 5df889ee$$

从表 3.7 所得的这 4 个数据分组将用于计算消息摘要(或散列码),如例 5.3 所示。

5.1.2 消息摘要计算

例 5.3 计算散列码。

32 位散列码的计算

图 5.2 给出了 32 位散列码的计算处理方案。在该图中,使用了下面的符号。

⊙:16 位整数模 $2^{16}+1=65\ 537$ 乘。

⊞:16 位整数模 $2^{16}=655\ 360$ 加。

⊕:16 位子分组逐位进行异或运算。

‖级联。

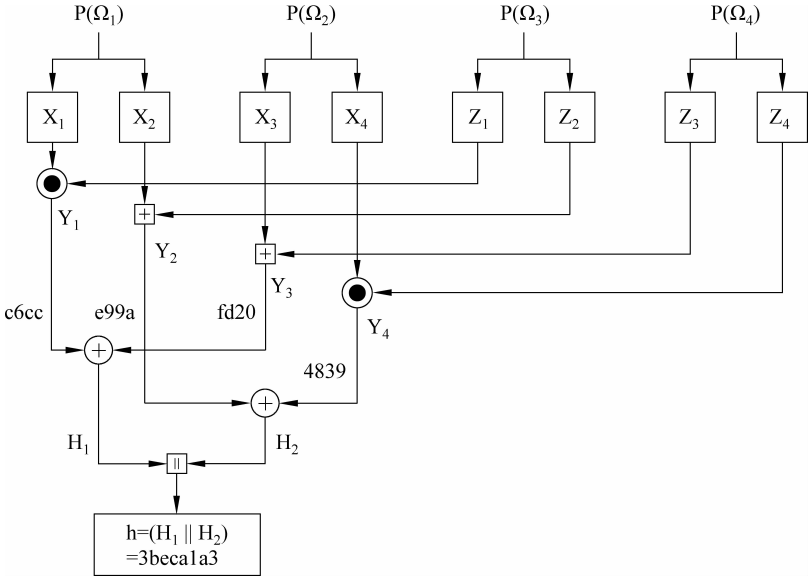


图 5.2 32 位散列码的计算方案

由于在例 5.2 中,我们已计算出 $P(\Omega_i)$,所以根据图 5.2,就能计算出 32 位的消息摘要:

$$Y_1 = c6cc$$
$$Y_2 = e99a$$
$$Y_3 = fd20$$
$$Y_4 = 4839$$
$$H_1 = 3bec$$
$$H_2 = a1a3$$

级联 H_1 与 H_2,得到 32 位的散列码 h,使得

$$h=(H_1||H_2)=3beca1a3$$

64 位散列码的计算

参照图 5.3，计算 64 位消息摘要的过程如下：

$$Y_1 = 97a0e99a$$

$$Y_2 = 371d4fc8$$

$$H_1 = f41d3352$$

$$H_2 = 753f20dc$$

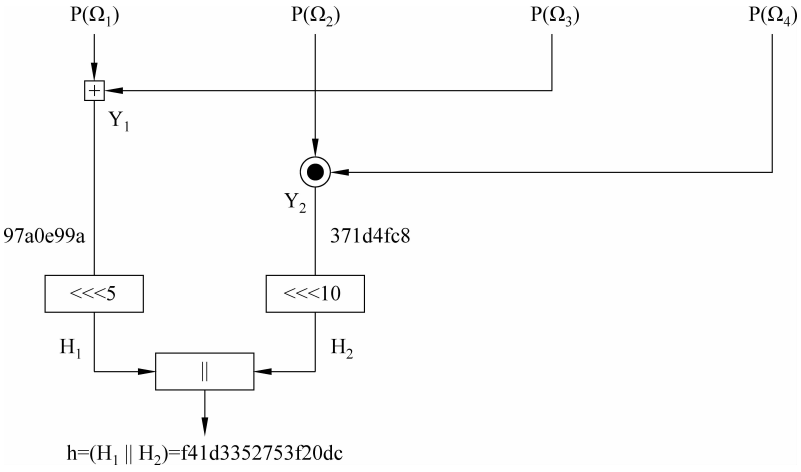


图 5.3　64 位散列码的计算方案

因此，计算所得的 64 位散列码为：

$$h=(H_1||H_2)=f41d3352753f20dc$$

注意：

⊙：32 位分组模 $2^{32}+1=4\ 294\ 967\ 297$ 乘。

⊞：32 位分组模 $2^{32}2=4\ 294\ 967\ 296$ 加。

$\lll m$：向左移动 m 位。

18 位散列码的计算

利用上面得到的 64 位消息摘要，根据图 5.4，从抽取处理中计算 18 位散列码。

$$h=f41d3352753f20dc(64\ 位)$$

抽取的规则是将 64 位的消息 h 两端丢弃 6 位，然后每 3 位抽选 1 位，使得

$$h=001110011101110001(18\ 位)$$

128 位散列码的计算（使用左移）

参考图 5.5，将每个 $P(\Omega_i)$ 左移 m 位。然后，级联它们产生 128 位消息摘要：

$$H_1 = 7b1b1c00$$

$$H_2 = a1d34e7c$$

$$H_3 = 59b14b55$$

$$H_4 = bf113dcb$$

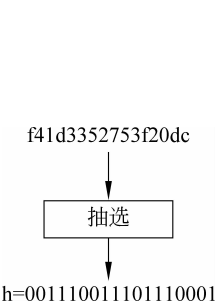


图 5.4 18 位散列码的计算方案

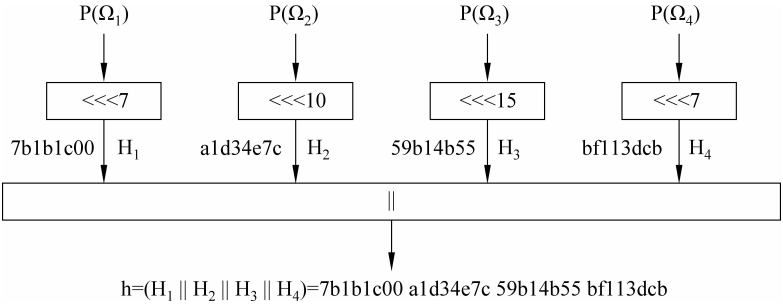


图 5.5 采用左移的 128 位散列码的计算

因此,128 位散列码为:

$$h = (H_1 \parallel H_2 \parallel H_3 \parallel H_4) = 7b1b1c00a1d34e7c59b14b55bf113dcb$$

128 位散列码的计算(使用逆)

根据图 5.6,下面计算另外一个 128 位消息摘要:

$$X_1 = 00f6 \quad X_2 = 3638 \quad X_3 = 9f28 \quad X_4 = 74d3$$

$$X_1^{-1} = 9b24 \quad -X_2 = c9c8 \quad -X_3 = 60d8 \quad X_4^{-1} = 8e12$$

$$Z_1 = 96aa \quad Z_2 = b362 \quad Z_3 = 5df8 \quad Z_4 = 89ee$$

$$Z_1^{-1} = bf34 \quad -Z_2 = 4c9e \quad -Z_3 = a208 \quad Z_4^{-1} = b652$$

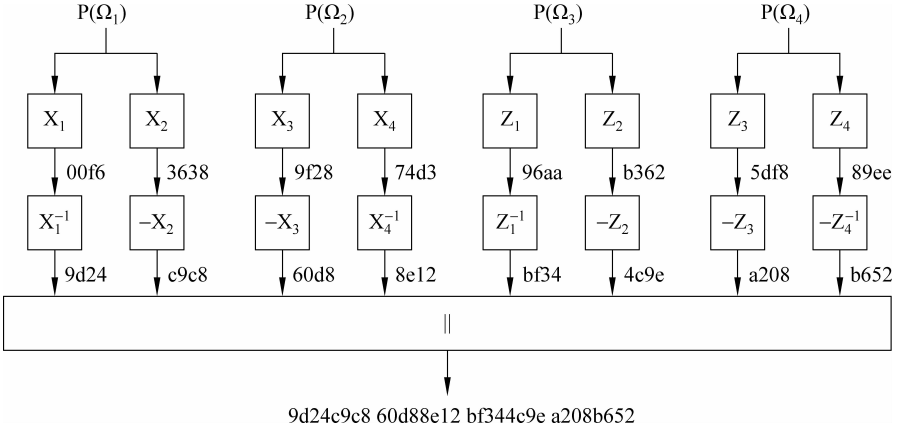


图 5.6 使用逆运算计算 128 位散列码

因此,从逆值的联结中计算 128 位散列码:

$$h = (X_1^{-1} \parallel -X_2 \parallel -X_3 \parallel X_4^{-1} \parallel Z_1^{-1} \parallel -Z_2 \parallel -Z_3 \parallel Z_4^{-1})$$

$$= 9d24c9cB60d88e12bf344c9ea208b652$$

128 位散列码计算(使用加和乘)

纵观图 5.7,计算 128 位消息摘要的过程如下:

$$P(\Omega_1) \boxplus P(\Omega_3) = 97a0e99a \lll 5 = f41d3352$$

$$P(\Omega_2) \odot P(\Omega_4) = 371d4fc8 \lll 10 = 753f20de$$

$$P(\Omega_1)\odot P(\Omega_3) = 56c9017f \lll 10 = 2405fd5b$$

$$P(\Omega_2)\boxplus P(\Omega_4) = fd20fec1 \lll 5 = a41fd83f$$

$$h = (P(\Omega_1)\boxplus P(\Omega_2)) \lll 5 \parallel (P(\Omega_2)\odot P(\Omega_4)) \lll 10 \parallel$$
$$(P(\Omega_2)\boxplus P(\Omega_4)) \lll 5 \parallel (P(\Omega_2)\odot P(\Omega_3)) \lll 10$$
$$= f41d3352\ 753f20dc\ a41fd83f\ 2405fd5b(128\ 位)$$

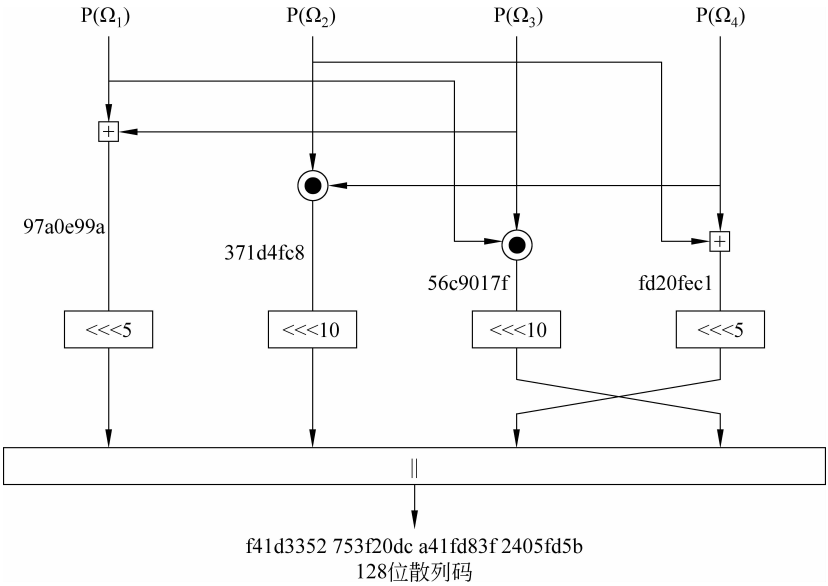


图 5.7　使用加和乘的 128 位散列码计算

这是得到的 128 位散列码。到目前为止，我们已经讨论了 DMDC 的计算，此 DMDC 没有追加 1 位标志和十六进制消息长度。

5.2　高级 DMDC 算法

本节介绍安全的 DMDC 算法，其能提供令人满意的安全级别。

5.2.1　密钥调度

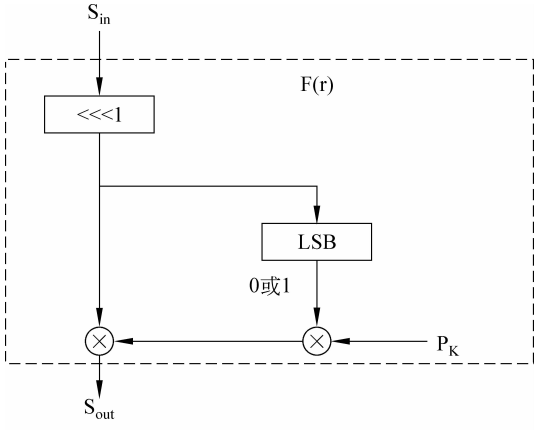
图 5.9 给出了新设计的密钥生成方案。利用表 3.1(PC-1)，将 64 位输入密钥重构为 56 位密钥序列。将 56 位密钥加载到两个 28 位寄存器(C_0，D_0)。这两个寄存器的值分别向左移动 S_r^L 和 S_r^R 位。S_r^L 和 S_r^R 由状态转换函数 F(r)产生，如图 5.8 和图 5.9 所示。在图 5.9 中，64 位的输入密钥被分成两个 32 位密钥。每个 32 位密钥都作为 F(r)的输入 S_{in}。且是从 S_{out} (mod 32)来计算 S_r^L 和 S_r^R。在图 5.10 中，LFSR 是生成伪随机二进制序列(pseudorandom binary sequence，PRBS)的设备，其特征函数为：

$$f(x) = x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1,\quad 周期为\ 2^{32} - 1。$$

假定 64 位输入密钥为 7a138b2524af17c5。使用图 5.11，计算出所有的轮密钥，如表 5.2 所示。

表 5.2 对应于(S_r^L 和 S_r^R)而生成的轮密钥

第 r 轮	(S_r^L,S_r^R)	K_r(第 r 轮密钥)
1	(2, 21)	36320340397a
2	(14,19)	9394d0aac24c
3	(0, 15)	91c2c6fcd01e
4	(7, 7)	fcf6701c06a4
5	(21,13)	c38496e8c45e
6	(1, 20)	12f64d47235d
7	(7, 17)	174a16a3c335
⋮	⋮	⋮
332	(21, 2)	17320b413872
333	(19, 17)	9ad8226cd646
334	(1, 11)	961203c1315b
335	(2, 18)	125ec46f8a55
336	(2, 13)	cd8d4610f0c4
337	(19, 9)	5e40db051358
338	(15, 8)	0414fc86b547



LSB:输入值的最低有效位
⊕: 异或
⊗: 乘
P_K: 32位常量(如, 0x000000AE)

图 5.8 生成 PRBS 的状态转换函数 F(r)

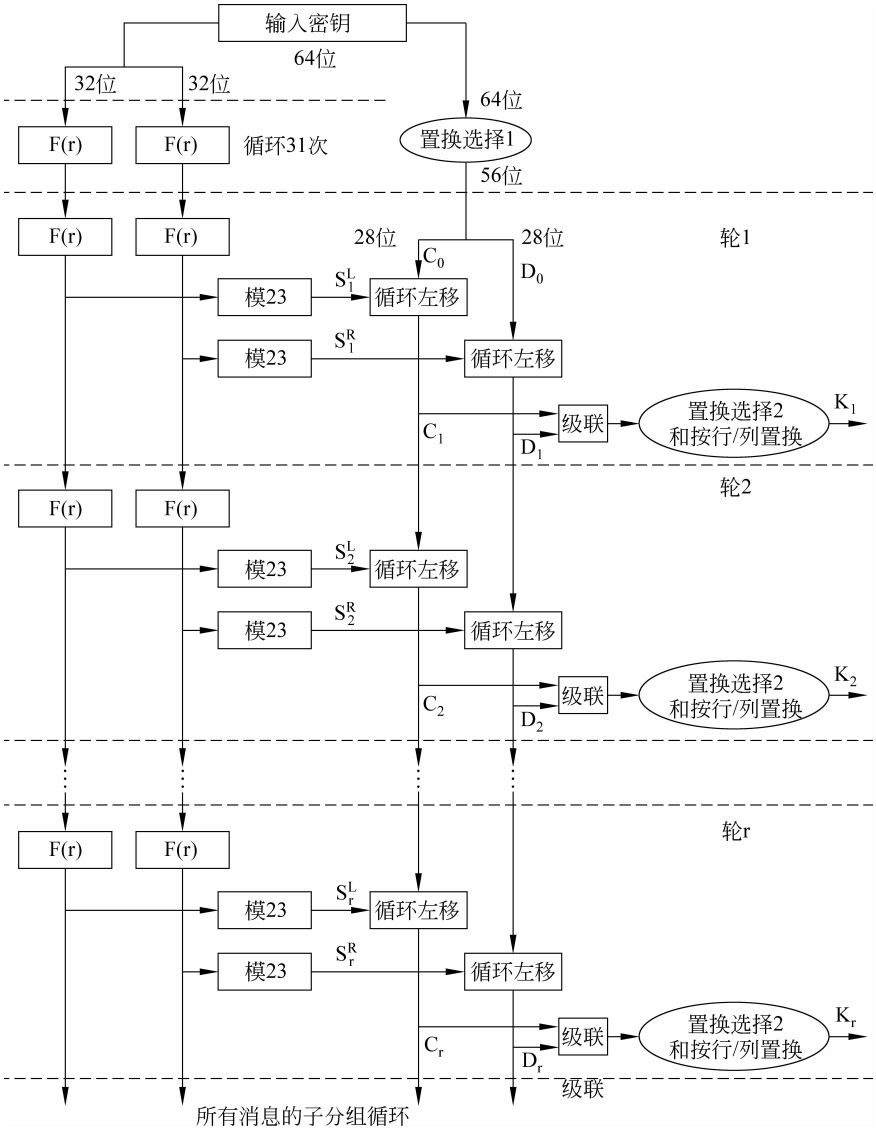


图 5.9　新设计的 DMDC 密钥生成方案

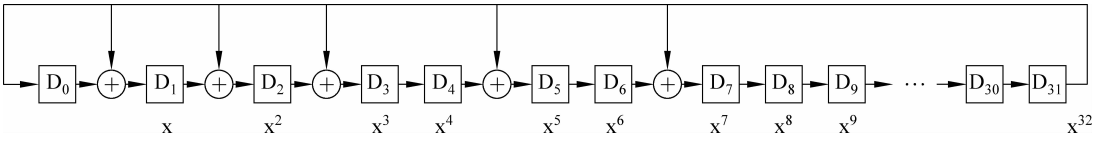


图 5.10　生成 PRBS，LFSR 所用的本原多项式