第 3 章　单向散列函数

本章导读：

随着以 Internet 为基础的电子商务技术的迅猛发展，以公钥密码术、数字签名等为代表的加密安全技术已成为研究的热点。单向散列函数是数字签名中的一个关键环节，可以大大缩短签名时间并提高安全性，另外在消息完整性检测、内存的散布分配、操作系统中账号口令的安全存储中，单向散列函数也有重要应用。

3.1　单向散列函数概述

单向散列函数(Hash Function，又称哈希函数、杂凑函数)是将任意长度的消息 M 映射成一个固定长度散列值 h（设长度为 m）的函数 H：

$$h=H(M)$$

散列函数要具有单向性，则必须满足如下特性：

(1) 给定 M，很容易计算 h。

(2) 给定 h，根据 $H(M)=h$ 反推 M 很难。

(3) 给定 M，要找到另一个消息 M' 并满足 $H(M)=H(M')$ 很难。

在某些应用中，单项散列函数还要满足抗碰撞(Collision) 的条件：要找到两个随机的消息 M 和 M'，使 $H(M)=H(M')$ 很难。

在实际应用中，单向散列函数是建立在压缩函数之上的，如图 3-1 所示。

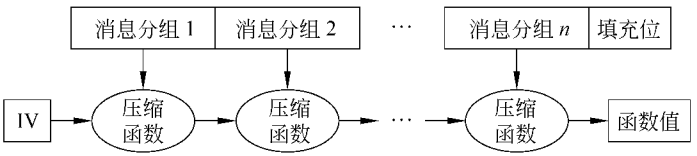

图 3-1　单向散列函数工作模式

给定一任意长度的消息输入，单向函数输出长度为 m 的散列值。压缩函数的输入是消息分组和前一分组的输出(对第一个压缩函数，其输入为消息分组 1 和初始化向量 IV)，输出是到该点的所有分组的散列，即分组 M_i 的散列值为

$$h_i=f(M_i,h_{i-1})$$

该散列值和下一轮的消息分组一起作为压缩函数下一轮的输入，最后一个分组的散列就是整个消息的散列。

单向散列函数是从全体消息集合到一个具有固定长度的消息摘要的变换，可分为两类：带密钥的单向散列函数和不带密钥的单向散列函数。带密钥的单向散列函数可用于认证、密钥共享和软件保护等方面。

3.2 MD5 算法

3.2.1 算法

MD 表示消息摘要(Message Digest),MD5 是 MD4 的改进版,它是 RSA 公钥密码算法的首位发明人 Ron Rivest 设计的。该算法对输入的任意长度消息产生 128 位(16 字节)长度的散列值(或称消息摘要)。MD5 算法如图 3-2 所示。

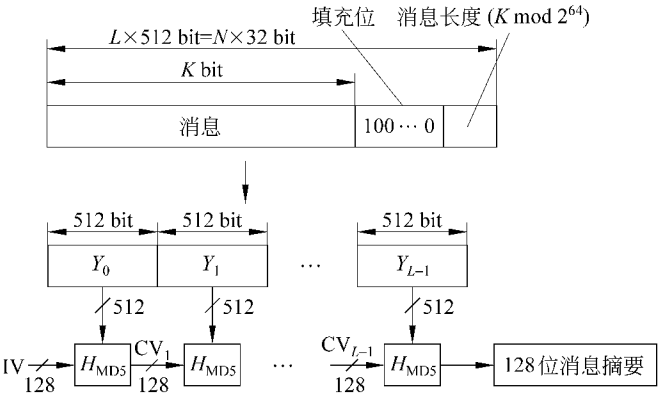

图 3-2 MD5 算法

由图 3-2 可知,MD5 算法包括以下 5 个步骤。

1. 附加填充位

首先填充消息,使其长度为一个比 512 的倍数小 64 位的数。填充方法是在消息后面填充一位 1,然后填充所需数量的 0。填充位的位数为 1～512。

2. 附加长度

将原消息长度的 64 位表示附加在填充后的消息后面,当原消息长度大于 2^{64} 时,用消息长度 mod 2^{64} 填充。这时,消息长度恰好是 512 的整数倍。令 $M[0\sim N-1]$ 为填充后消息的各个字(每字为 32 位),N 是 16 的倍数。

3. 初始化 MD 缓冲区

初始化用于计算消息摘要的 128 位缓冲区,这个缓冲区由 4 个 32 位寄存器 A、B、C、D 表示。寄存器的初始化值(按低位字节在前的顺序存放)为

A: 01 23 45 67
B: 89 ab cd ef
C: fe dc ba 98
D: 76 54 32 10

4. 按 512 位的分组处理输入消息

这一步为 MD5 的主循环,包括 4 轮,如图 3-3 所示。每个循环都以当前正在处理的 512 位分组 Y_q 和 128 位缓冲值 $ABCD$ 为输入,然后更新缓冲内容。

在图 3-3 中,4 轮的操作类似,每一轮进行 16 次操作,各轮的操作过程如图 3-4 所示。

4 轮操作的不同之处在于每轮使用的非线性函数不同,在第一轮操作之前,首先把 A、

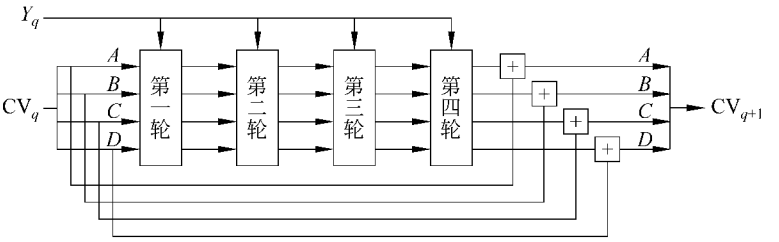

图 3-3　单个 512 位分组的 MD5 主循环处理

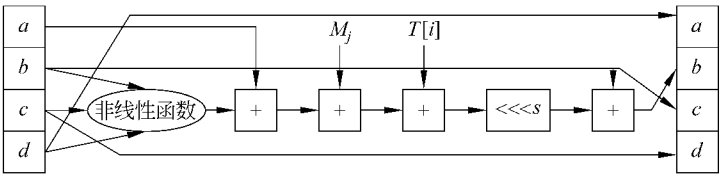

图 3-4　MD5 某一轮的执行过程

B、C、D 复制到另外的变量 a、b、c、d 中。这 4 个非线性函数分别为(其输入/输出均为 32 位字)：

$$F(X,Y,Z)=(X\wedge Y)\vee((\sim X)\wedge Z)$$
$$G(X,Y,Z)=(X\wedge Z)\vee(Y\wedge(\sim Z))$$
$$H(X,Y,Z)=X\oplus Y\oplus Z$$
$$I(X,Y,Z)=Y\oplus(X\vee(\sim Z))$$

其中，$\wedge$ 表示按位与；$\vee$ 表示按位或；$\sim$ 表示按位反；$\oplus$ 表示按位异或。

此外，由图 3-4 可知，这一步中还用到了一个有 64 个元素的表 $T[1\sim64]$，$T[i]=2^{32}\times\text{abs}(\sin(i))$，$i$ 的单位为弧度。

根据以上描述，将这一步骤的处理过程归纳如下：

```
    for i=0 to N/16-1 do     /* 每次循环处理 16 个字,即 512 字节的消息分组 */
                             /* 把第 i 个字块(512 位)分成 16 个 32 位子分组复制到 X 中 */
    for j=0 to 15 do
        set x[j] to M[i*16+j]
    end                      /* j 循环 */
                             /* 把 A 存为 AA,B 存为 BB,C 存为 CC,D 存为 DD */
    AA=A
    BB=B
    CC=C
    DD=D
/* 第一轮 */
/* 令[abcd k s i]表示操作
    a=b+((a+F(b,c,d)+X[k]+T[i])<<<s)
    其中, Y<<<s 表示 Y 循环左移 s 位 */
/* 完成下列 16 个操作 */
[ABCD  0  7  1] [DABC  1  12  2] [CDAB  2  17  3] [BCDA  3  22  4]
```

```
[ABCD  4  7  5] [DABC  5  12  6] [CDAB  6  17  7] [BCDA  7  22  8]
[ABCD  8  7  9] [DABC  9  12  10] [CDAB 10  17  11] [BCDA 11  22  12]
[ABCD 12  7  13] [DABC 13  12  14] [CDAB 14  17  15] [BCDA 15  22  16]
/*第二轮*/
/*令[abcd k s i]表示操作
    a=b+((a+G(b,c,d)+X[k]+T[i]) <<<s)*/
/*完成下列16个操作*/
[ABCD  1  5  17] [DABC  6  9  18] [CDAB 11  14  19] [BCDA  0  20  20]
[ABCD  5  5  21] [DABC 10  9  22] [CDAB 15  14  23] [BCDA  4  20  24]
[ABCD  9  5  25] [DABC 14  9  26] [CDAB  3  14  27] [BCDA  8  20  28]
[ABCD 13  5  29] [DABC  2  9  30] [CDAB  7  14  31] [BCDA 12  20  32]
/*第三轮*/
/*令[abcd k s t]表示操作
    a=b+((a+H(b,c,d)+X[k]+T[i])<<<s)*/
/*完成以下16个操作*/
[ABCD  5  4  33] [DABC  8  11  34] [CDAB 11  16  35] [BCDA 14  23  36]
[ABCD  1  4  37] [DABC  4  11  38] [CDAB  7  16  39] [BCDA 10  23  40]
[ABCD 13  4  41] [DABC  0  11  42] [CDAB  3  16  43] [BCDA  6  23  44]
[ABCD  9  4  45] [DABC 12  11  46] [CDAB 15  16  47] [BCDA  2  23  48]
/*第四轮*/
/*令[abcd k s t]表示操作
    a=b+((a+I(b,c,d)+X[k]+T[i]) <<<s) */
/*完成以下16个操作*/
[ABCD  0  6  49] [DABC  7  10  50] [CDAB 14  15  51] [BCDA  5  21  52]
[ABCD 12  6  53] [DABC  3  10  54] [CDAB 10  15  55] [BCDA  1  21  56]
[ABCD  8  6  57] [DABC 15  10  58] [CDAB  6  15  59] [BCDA 13  21  60]
[ABCD  4  6  61] [DABC 11  10  62] [CDAB  2  15  63] [BCDA  9  21  64]
A=A+AA
B=B+BB
C=C+CC
D=D+DD
end                          /*i循环*/
```

5. 输出

由A、B、C、D四个寄存器的输出按低位字节在前的顺序(即以A的低字节开始、D的高字节结束)得到128位的消息摘要。

以上就是对MD5算法的描述。MD5算法的运算均为基本运算,比较容易实现且速度很快,从网上可以找到实现MD5的源代码。

3.2.2 举例

以求字符串"abc"的MD5散列值为例来说明上面描述的过程。"abc"的二进制表示为01100001 01100010 01100011。

1. 填充消息

消息长 24，先填充 1 位 1，然后填充 423 位 0，再用消息长 24，即 0x00000000　00000018 填充，则：

$M[0]$=61626380　$M[1]$=00000000　$M[2]$=00000000　$M[3]$=00000000
$M[4]$=00000000　$M[5]$=00000000　$M[6]$=00000000　$M[7]$=00000000
$M[8]$=00000000　$M[9]$=00000000　$M[10]$=00000000　$M[11]$=00000000
$M[12]$=00000000　$M[13]$=00000000　$M[14]$=00000000　$M[15]$=00000018

2. 初始化

A：01　23　45　67
B：89　ab　cd　ef
C：fe　dc　ba　98
D：76　54　32　10

3. 主循环

利用 3.2.1 节中描述的过程对字块 1(本例只有一个字块)进行处理。变量 a、b、c、d 每一次计算后的中间值不再详细列出。

4. 输出

消息摘要=90015098　3cd24fb0　d6963f7d　28e17f72

3.3　SHA-1 算法

3.3.1　算法

SHA 是美国 NIST 和 NSA 共同设计的安全散列算法(Secure Hash Algorithm)，用于数字签名标准(Digital Signature Standard，DSS)。SHA 的修改版 SHA-1 于 1995 年作为美国联邦信息处理标准公告(FIPS PUB 180-1)发布。目前，SHA-1 与 MD5 是应用最广泛的两个算法。

SHA-1 产生消息摘要的过程类似 MD5，如图 3-5 所示。

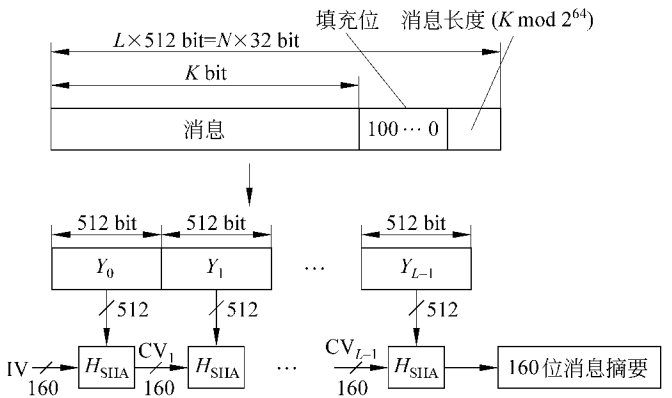

图 3-5　SHA-1 算法

SHA-1 的输入为长度小于 2^{64} 位的消息，输出为 160 位(20 字节)的消息摘要。具体过程如下。

1. 填充消息

首先将消息填充为 512 位的整数倍，填充方法和 MD5 完全相同：先填充一个 1，然后填充一定数量的 0，使其长度比 512 的倍数少 64 位；接下来用原消息长度的 64 位表示填充。这样，消息长度就成为 512 的整数倍。以 M_0、M_1、…、M_n 表示填充后消息的各个字块(每字块为 16 个 32 位字)。

2. 初始化缓冲区

在运算过程中，SHA-1 要用到两个缓冲区，两个缓冲区均有 5 个 32 位的寄存器。第一个缓冲区标记为 A、B、C、D、E；第二个缓冲区标记为 H_0、H_1、H_2、H_3、H_4。此外，运算过程中还用到一个标记为 W_0、W_1、…、W_{79} 的 80 个 32 位字序列和一个单字的缓冲区 TEMP。在运算之前，初始化$\{H_j\}$：

$$H_0 = 0\ \mathrm{x}\ 6\ 7\ 4\ 5\ 2\ 3\ 0\ 1$$
$$H_1 = 0\ \mathrm{x}\ \mathrm{E}\ \mathrm{F}\ \mathrm{C}\ \mathrm{D}\ \mathrm{A}\ \mathrm{B}\ 8\ 9$$
$$H_2 = 0\ \mathrm{x}\ 9\ 8\ \mathrm{B}\ \mathrm{A}\ \mathrm{D}\ \mathrm{C}\ \mathrm{F}\ \mathrm{E}$$
$$H_3 = 0\ \mathrm{x}\ 1\ 0\ 3\ 2\ 5\ 4\ 7\ 6$$
$$H_4 = 0\ \mathrm{x}\ \mathrm{C}\ 3\ \mathrm{D}\ 2\ \mathrm{E}\ \mathrm{I}\ \mathrm{F}\ 0$$

3. 按 512 位的分组处理输入消息

SHA-1 运算主循环包括 4 轮，每轮 20 次操作。SHA-1 用到一个逻辑函数序列 f_0、f_1、…、f_{79}。每个逻辑函数的输入为 3 个 32 位字，输出为一个 32 位字。定义如下(B、C、D 均为 32 位字)：

$$f_t(B,C,D) = (B \wedge C) \vee (\sim B \wedge D) \qquad (0 \leqslant t \leqslant 19)$$
$$f_t(B,C,D) = B \oplus C \oplus D \qquad (20 \leqslant t \leqslant 39)$$
$$f_t(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \qquad (40 \leqslant t \leqslant 59)$$
$$f_t(B,C,D) = B \oplus C \oplus D \qquad (60 \leqslant t \leqslant 79)$$

其中，运算符的定义与 3.1 节中 MD5 运算中的相同。

SHA-1 运算中还用到了常数字序列 K_0、K_1、…、K_{79}，其值为

$$K_t = 0\ \mathrm{x}\ 5\ \mathrm{A}\ 8\ 2\ 7\ 9\ 9\ 9 \qquad (0 \leqslant t \leqslant 19)$$
$$K_t = 0\ \mathrm{x}\ 6\ \mathrm{E}\ \mathrm{D}\ 9\ \mathrm{E}\ \mathrm{B}\ \mathrm{A}\ 1 \qquad (20 \leqslant t \leqslant 39)$$
$$K_t = 0\ \mathrm{x}\ 8\ \mathrm{F}\ 1\ \mathrm{B}\ \mathrm{B}\ \mathrm{C}\ \mathrm{D}\ \mathrm{C} \qquad (40 \leqslant t \leqslant 59)$$
$$K_t = 0\ \mathrm{x}\ \mathrm{C}\ \mathrm{A}\ 6\ 2\ \mathrm{C}\ 1\ \mathrm{D}\ 6 \qquad (60 \leqslant t \leqslant 79)$$

SHA-1 算法按如下步骤处理每个字块 M_j：

```
(1) 把 Mi 分为 16 个字 W0、W1、…、W15，其中，W0 为最左边的字。
(2) for t=16 to 79 do
        let Wt=(Wt-3 Wt-8 Wt-14 Wt-16)<<<1
(3) let A=H0, B=H1, C=H2, D=H3, E=H4
(4) for t=0 to 79 do
        TEMP=(A<<<5)+ft(B, C, D)+E+Wt+Kt
        E=D;D=C;C=(B<<<30);B=A;A=TEMP
```

(5) let $H_0=H_0+A$; $H_1=H_1+B$; $H_2=H_2+C$; $H_3=H_3+D$; $H_4=H_4+E$

4. 输出

处理完 M_n后，160 位的消息摘要为 H_0、H_1、H_2、H_3、H_4级联的结果。

3.3.2 举例

以求字符串“abc”的 SHA-1 散列值为例来说明上面描述的过程。“abc”的二进制表示为 01100001 01100010 01100011。

1. 填充消息

消息长 24，先填充 1 位 1，然后填充 423 位 0，再用消息长 24，即 0x00000000 00000018 填充。

2. 初始化

$$H_0=0x67452301$$
$$H_1=0xEFCDAB89$$
$$H_2=0x98BADCFE$$
$$H_3=0x10325476$$
$$H_4=0xC3D2E1F0$$

3. 主循环

处理消息字块 1(本例中只有 1 个字块)，分成 16 个字：

$W[0]=61626380$	$W[1]=00000000$	$W[2]=00000000$	$W[3]=00000000$
$W[4]=00000000$	$W[5]=00000000$	$W[6]=00000000$	$W[7]=00000000$
$W[8]=00000000$	$W[9]=00000000$	$W[10]=00000000$	$W[11]=00000000$
$W[12]=00000000$	$W[13]=00000000$	$W[14]=00000000$	$W[15]=00000018$

然后根据 3.3.1 节中描述的过程计算，其中，循环“for t=0 to 79”中，各步 A、B、C、D、E 的值如下：

	A	B	C	D	E
$t=0$: 0	116FC33	67452301	7BF36AE2	98BADCFE	10325476
$t=1$: 8	990536D	0126FC33	59D148C0	7BF36AE2	98BADCFE
$t=2$: A	1390F08	8990536D	C045BF0C	59D148C0	7BF36AE2
$t=3$: C	DD8E11B	A1390F08	626414DB	C045BF0C	59D148C0
$t=4$: C	FD499DE	CDD8E11B	284E43C2	626414DB	C045BF0C
$t=5$: 3	FC7CA40	CFD499DE	F3763846	284E43C2	626414DB
$t=6$: 9	93E30C1	3FC7CA40	B3F52677	F3763846	284E43C2
$t=7$: 9	B8C07D4	993E30C1	0FF1F290	B3F52677	F3763846
$t=8$: 4	B6AE328	9E8C07D4	664F8C30	0FF1F290	B3F52677
$t=9$: 8	351F929	4B6AE328	27A301F5	664F8C30	0FF1F290
$t=10$: F	BDA9E89	8351F929	12DAB8CA	27A301F5	664F8C30
$t=11$: 6	3188FE4	FBDA9E89	60D47E4A	12DAB8CA	27A301F5
$t=12$: 4	607B664	63188FE4	7EF6A7A2	60D47E4A	12DAB8CA

$t=13$：9	128F695	4607B664	18C623F9	7EF6A7A2	60D47E4A
$t=14$：1	96BEE77	9128F695	3181ED99	18C623F9	7EF6A7A2
$t=15$：2	0BDD62F	196BEE77	644A3DA5	1181ED99	18C623F9
$t=16$：4	E925823	20BDD62F	C65AFB9D	644A3DA5	1181ED99
$t=17$：8	2AA6728	4E925823	C82f758B	C65AFB9D	644A3DA5
$t=18$：D	C64901D	82AA6728	D3A49608	C82F758B	C65AFD9D
$t=19$：F	D9E1D7D	DC64901D	20AA99CA	D3A49608	C82F758B
$t=20$：1	A37B0CA	FD9E1D7D	77192407	20AA99CA	D3A49608
$t=21$：3	3A23BFC	1A37B0CA	7F67875F	77192407	20AA99CA
$t=22$：2	1283486	33A23BFC	868DEC32	7F67875F	77192407
$t=23$：D	541F12D	21283486	0CE88EFF	868DEC32	7F67875F
$t=24$：C	7567DC6	D541F12D	884A0D21	0CE88EFF	868DEC32
$t=25$：4	8413BA4	C7567DC6	75507C4B	884A0D21	0CE88EFF
$t=26$：B	E35FBD5	48413BA4	B1D59F71	75507C4B	884A0D21
$t=27$：4	AA84D97	BE35FBD5	12104EE9	B1D59F71	75507C4D
$t=28$：8	370B52E	4AA84D97	6F8D7EF5	12104EE9	B1D59F71
$t=29$：C	5F6AF5D	8370B52E	D2AA1365	6F8D7EF5	12104EE9
$t=30$：1	267B407	C5FBAF5D	A0DC2D4B	D2AA1365	6F8D7EF5
$t=31$：3	B845D33	1267B407	717EEBD7	A0DC2D4B	D2AA1365
$t=32$：0	46FAA0A	3B845D33	C499ED01	717EEBD7	A0DC2D4B
$t=33$：2	C0EBC11	046FAA0A	CEE1174C	C499ED01	717EEBD7
$t=34$：2	1796AD4	2C0EBC11	8116EA82	CEE174C	C499ED01
$t=35$：D	CBBB0CB	21796AD4	4B03AF04	811BEA82	CEE1174C
$t=36$：0	F11FD8	DCBBB0CB	085E5AB5	4B03AF04	811BEA82
$t=37$：D	C63973F	0F511FD8	F72EEC32	085E5AB5	4B03AF04
$t=38$：4	C986405	DC63973F	03D447F6	F72EEC32	085E5AB5
$t=39$：3	2DE1CBA	4C986405	F718E5CF	03D447F6	F72EEC32
$t=40$：F	C87DEDF	32DE1CDA	53261901	F718E5CF	03D447F6
$t=41$：9	70A0D5C	FC87DEDF	8CD7872E	53261901	F718E5CF
$t=42$：7	F193DC5	970A0D5C	FF21F7B7	8CB7872E	53261901
$t=43$：E	E1B1AAF	7F193DC5	25C28357	FF21F7B7	8CB7872E
$t=44$：4	0F28E09	EE1B1AAF	5FC64F71	25C28357	FF21F7B7
$t=45$：1	C51E1F2	40F28E09	FB86C6AB	5FC64F71	25C28357
$t=46$：A	01B846C	1C51E1F2	503CA382	FB86C6AB	5FC64F71
$t=47$：B	EAD02CA	A01B846C	8714787C	503CA382	FB86C6AB
$t=48$：B	AF39337	BEAD02CA	2806E11B	8714787C	503CA382
$t=49$：1	20731C5	BAF39337	AFAB40B2	2806E11B	8714787C
$t=50$：6	41DB2CE	120731C5	EEBCE4CD	AFAB40B2	2806E11B

$t=51$：3	847AD66	641DB2CE	4481CC71	EEBCE4CD	AFAB40B2
$t=52$：E	490436D	3847AD66	99076CB3	4481CC71	EEBCE4CD
$t=53$：2	7E9F1D8	E490436D	8E11EB59	99076CB3	4481CC71
$t=54$：7	B71F76D	27E9F1D8	792410DB	8E11EB59	99076CB3
$t=55$：5	E6456AF	7B71F76D	09FA7C76	792410DB	8E11EB59
$t=56$：C	846093F	5E6456AF	5EDC7DDB	09FA7C76	792410DB
$t=57$：D	262FF50	C846093F	D79915AB	5EDC7DDB	09FA7C76
$t=58$：0	9D785FD	D262FF50	F211824F	D79915AB	5EDC7DDB
$t=59$：3	F52DE5A	09D785FD	3498BFD4	F211824F	D79915AB
$t=60$：D	756C147	3F52DE5A	4275E17F	3498BFD4	F211824F
$t=61$：5	48C9CB2	D756C147	8FD4B796	4275E17F	3498BFD4
$t=62$：B	66C020B	548C9CB2	F5D5B051	8FD4B796	4275E17F
$t=63$：6	B61C9E1	B66C020B	9523272C	F5D5B051	8FD4B796
$t=64$：1	9DFA7AC	6B61C9E1	ED9B0082	9523272C	F5D5B051
$t=65$：1	01655F9	19DFA7AC	5ADB7278	D9B0082	9523272C
$t=66$：0	C3DF2B4	101655F9	0677E9EB	5AD87278	ED9B0082
$t=67$：7	8DD4D2B	0C3DF2B4	4405957E	0677E9EB	5AD87278
$t=68$：4	97093C0	78DD4D2B	030F7CAD	4405957E	0677E9EB
$t=69$：3	F2588C2	497093C0	DE37534A	030F7CAD	4405957E
$t=70$：C	199F8C7	3F2588C2	125C24F0	DE37534A	030F7CAD
$t=71$：3	9859DE7	C199F8C7	8FC96230	125C24F0	DE37534A
$t=72$：E	DB42DE4	39859DE7	F0667E31	8FC96230	125C24F0
$t=73$：1	1793F6F	EDB42DE4	CE616779	F0667E31	8FC96230
$t=74$：5	EE76897	11793F6F	3B6D0B79	CE616779	F0667E31
$t=75$：6	3F7DAB7	5EE76897	C45E4FDB	3B6D0B79	CE616779
$t=76$：A	079B7D9	63F7DAB7	D7B9DA25	C45E4FDB	3B6D0B79
$t=77$：8	60D21CC	A079B7D9	D8FDF6AD	D7B99A25	C45E4FDB
$t=78$：5	738D5E1	860D21CC	681E6DF6	D8FDF6AD	D7B9DA25
$t=79$：4	2541B35	5738D5E1	21834873	681E6DF6	D8FDF6AD

字块 1 处理完后，$\{H_j\}$的值为

H_0＝6 7 4 5 2 3 0 1＋4 2 5 4 1 B 3 5＝A 9 9 9 3 E 3 6

H_1＝E F C D A B 8 9＋5 7 3 8 D 5 E 1＝4 7 0 6 8 1 6 A

H_2＝9 8 B A D C F E＋2 1 8 3 4 8 7 3＝B A 3 E 2 5 7 1

H_3＝1 0 3 2 5 4 7 6＋6 8 1 E 6 D F 6＝7 8 5 0 C 2 6 C

H_4＝C 3 D 2 E 1 F 0＋D 8 F D F 6 A D＝9 C D 0 D 8 9 D

4. 输出

消息摘要：A9993E36　4706816A　BA3E2571　7850C26C　9CD0D89D

3.3.3 SHA-1 与 MD5 的比较

SHA-1 与 MD5 的比较如表 3-1 所示。

表 3-1 SHA-1 与 MD5 的比较

特 征 项	SHA-1	MD5
Hash 值长度	160 位	128 位
分组处理长度	512 位	512 位
步数	80(4×20)	64(4×16)
最大消息长度	$\leqslant 2^{64}$ 位	不限
非线性函数个性	3(第 2、4 轮相同)	4
常数个数	4	64

根据各项特征,简要地说明它们之间的不同。

(1) 安全性: SHA-1 所产生的摘要较 MD5 长 32 位。若两种散列函数在结构上没有任何问题的话,SHA-1 比 MD5 更安全。

(2) 速度: 两种方法都考虑了以 32 位处理器为基础的系统结构,但 SHA-1 的运算步骤较 MD5 多了 16 步,而且 SHA-1 记录单元的长度比 MD5 多了 32 位。因此若是以硬件来实现 SHA-1,其速度大约较 MD5 慢 25%。

(3) 简易性: 两种方法都相当的简单,在实现上不需要很复杂的程序或是大量的存储空间,然而总体上来讲,SHA-1 每一步的操作都比 MD5 简单。

3.4 消息认证码

与密钥相关的单向散列函数通常称为消息认证码(Message Authentication Code, MAC),用公式表示为

$$\mathrm{MAC}=C_K(M)$$

其中,M 为可变长的消息,K 为通信双方共享的密钥,C 为单向散列函数。

MAC 可为拥有共享密钥的双方在通信中验证消息的完整性,也可被单个用户用来验证其文件是否被改动,工作机制如图 3-6 所示。

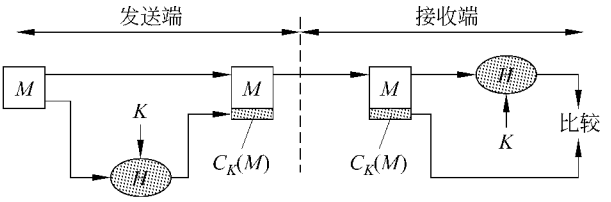

图 3-6 MAC 的工作机制

下面介绍由 RFC 2104 定义的 HMAC 算法, HMAC 全称为 Keyed-Hashing for