第3章　信息认证技术

3.1　概　　述

在当前开放式的网络环境中，任何在网络上的通信都可能遭到黑客的攻击，窃听机密消息，伪造、复制、删除和修改消息等攻击越来越多。所有的攻击都可能对正常通信造成破坏性的影响。因此，一个真实可靠的通信环境成为能够有效进行网络通信的基本前提。认证技术作为信息安全中的一个重要组成部分也显得尤为重要。

为了防止通信中的消息被非授权使用者攻击，有效的方法就是要对发送或接收到的消息具有鉴别能力，能鉴别消息的真伪和通信对方的真实身份。实现这样功能的过程称为认证。

一个安全的认证系统应满足以下条件：

(1) 合法的接收者能够检验所接收消息的合法性和真实性；

(2) 合法的发送方对所发送的消息无法进行否认；

(3) 除了合法的发送方之外，任何人都无法伪造、篡改消息。

通常情况下，一个完整的身份认证系统中除了有消息发送方和接收方外，还要有一个可信任的第三方，负责密钥分发、证书的颁发、管理某些机密信息等工作，当通信双方遇到争执、纠纷时，还充当仲裁者的角色。

通信双方进行认证的目的是进行真实而安全的通信。所谓认证就是在通信过程中，通信一方验证另一方所声称的某种属性。信息安全中的认证技术主要有两种：消息认证与身份认证。

如果验证的是消息的某种属性，则该认证方式称为消息认证。消息认证用于保证信息的完整性与不可抵赖性，验证消息在传送和存储过程中是否遭到篡改、重放等攻击。若认证的属性是关于通信中某一方或双方身份的话，则该认证过程称为身份认证。身份认证主要用于鉴别用户身份，是用户向对方出示自己身份的证明过程，通常是确认通信的对方是否拥有进入某个系统或使用系统中某项服务的合法权利的第一道关卡，确认消息的发送者和接收者是否合法。

3.2　哈 希 函 数

哈希函数也叫单向散列函数，是信息安全领域广泛使用的一种密码技术，它主要用于提供消息的完整性验证。哈希函数以任一长度的消息 M 为输入，产生固定长度的数据输出。这个定长输出称为消息 M 的散列值或消息摘要。由于哈希函数具有单向的特性，因此该散列值也称为数据的“指纹”。

3.2.1 哈希函数概述

由于哈希(Hash)函数通过产生定长的散列值作为数据的特征“指纹”,因此用于消息认证的哈希函数必须具有下列性质:

(1) 哈希函数的输入可以是任意长度的数据块 M,产生固定长度的散列值 h。

(2) 给定消息 M,很容易计算散列值 h。

(3) 给定散列值 h,根据 $H(M)=h$ 推导出 M 很难,这个性质称为单向性。单向性要求根据报文计算散列值很简单,但反过来根据散列值计算出原始报文十分困难。

(4) 已知消息 M,通过同一个 $H(\cdot)$,计算出不同的 h 是很困难的。

(5) 给定消息 M,要找到另一消息 M',满足 $H(M)=H(M')$,在计算上是不可行的,这条性质称为弱抗碰撞性。该性质是保证无法找到一个替代报文,否则就可能破坏使用哈希函数进行封装或者签名的各种协议的安全性。哈希函数的重要之处就是赋予 M 唯一的“指纹”。

(6) 对于任意两个不同的消息 $M\neq M'$,它们的散列值不可能相同,这条性质被称为强抗碰撞性。强抗碰撞性对于消息的哈希函数安全性要求更高,这条性质保证了对生日攻击的防御能力。

碰撞性是指对两个不同的消息 M 和 M',如果它们的散列值相同,则发生了碰撞。我们需要处理的消息是无限的,但可能的散列值却是有限的。不同的消息可能会产生同一散列值,因此碰撞是存在的。但是,哈希函数要求用户不能按既定的需要找到一个碰撞,意外的碰撞更是不太可能的。显然,从安全性的角度来看,哈希函数输出的位越长,抗碰撞的安全强度越大。

在信息认证技术中,哈希函数扮演着非常重要的角色,在通信安全中起着重要的作用,同时也是许多密码协议的基本模块。哈希函数的散列值也被称为哈希值、消息摘要、数字指纹、密码校验和、信息完整性检验码、操作检验码等。如果对明文进行轻微的改动,哪怕只是一个字母或一个标点符号,其对应的散列值也会有很大的不同。

哈希函数的设计是建立在压缩函数的思想上。压缩函数的输入是消息分组和文本前一分组的输出,即分组 M_j 的散列值为:

$$h_j = f(M_j, h_{j-1})$$

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值就成为整个消息的散列值。

目前常用的哈希函数有 MD5、SHA-1 和 RIPEMD-160 等,本章重点介绍 MD5 和 SHA-1 算法的基本原理。

3.2.2 MD5

MD5 算法是由麻省理工学院的 Ron Rivest 提出的,是一种常用的哈希函数,它可将任意长度的消息经过变换得到一个 128 位的散列值,MD5 被广泛用于各种软件的密码认证和密钥识别上。

对 MD5 算法简要的叙述可以概括为:MD5 以 512 位分组来处理输入的信息,且每一分组又被划分为 16 个 32 位子分组,经过了一系列的处理后,算法的输出由 4 个 32 位分组

组成，将这 4 个 32 位分组级联后将生成一个 128 位散列值。

MD5 是经 MD2、MD3 和 MD4 发展而来的。它的作用是让大容量信息在数字签名前被“压缩”成一种保密的格式（就是把一个任意长度的字串变换成一定长的大整数）。不管是 MD2、MD4 还是 MD5，它们都需要获得一个随机长度的信息并产生一个 128 位的信息摘要。虽然这些算法的结构或多或少有些相似，但 MD2 的设计与 MD4 和 MD5 完全不同，那是因为 MD2 是为 8 位机器做设计优化的，而 MD4 和 MD5 却是面向 32 位计算机的。

MD5 的算法步骤如下：

(1) 数据填充与分组。

将输入信息 M 按顺序进行分组，每 512 位为一组，即 $M=M_1,M_2,\cdots,M_{n-1},M_n$。将 M_n 的长度填充为 448 位，当 M_n 的长度 L 小于 448 位时，在信息 M_n 后加一个“1”，然后再填充若干个“0”，使得最后 M_n 的长度为 448 位。当 M_n 的长度大于 448 位时，在信息 M_n 后添加一个“1”，然后再填充 $512-L+448$ 个“0”，使最后信息 M_n 的长度为 512 位，M_{n+1} 的长度为 448 位。现在填充后的信息 M 的长度恰好是一个比 512 的倍数少 64 位的数，也就是信息长度等于 $512\times(n-1)+448$，其中 n 为某个整数。然后将填充前的信息 M 的长度 L 转换成 64 位二进制数，如果原信息长度 L 超过 64 位所能表示的范围，则只保留最后 64 位。最后再将该 64 位二进制数增加到填充后的信息 M 的 M_n 后面，使得最后一块的长度为 512 位。经过这些处理，现在信息的位长为 $(n-1)\times512+448+64=n\times512$，即长度恰好是 512 的整数倍。这样做的原因是为了满足后面处理中对信息长度的要求。

(2) 初始化散列值。

MD5 算法的中间结果和最终结果保存在 128 位的缓冲区中，缓冲区用 4 个 32 位的变量表示，这些变量被称做链接变量(Chaining Variable)，初始化为：

$A=$0x01234567

$B=$0x89abcdef

$C=$0xfedcba98

$D=$0x76543210

当设置好这 4 个链接变量后，就开始进入算法的 4 轮循环运算。循环的总次数是信息中 512 位信息分组的数目。

(3) 计算散列值。

将上面 4 个链接变量复制到另外 4 个变量中，即 A 到 a，B 到 b，C 到 c，D 到 d。

主循环有 4 轮（MD4 只有 3 轮），每轮循环都很相似，主循环如图 3.1 所示。第一轮进行 16 次操作。每次操作对 a、b、c 和 d 中的三个作一次非线性函数运算，然后将所得结果加上第 4 个变量，文本的一个子分组和一个常数。再将所得结果循环左移某个位数，并加上 a、b、c 或 d 中之一。最后用该结果取代 a、b、c 或 d 中之一。

每轮运算使用的非线性函数都是一个基本的逻辑函数，其输入是三个 32 位的信息，输出是一个 32 位的信息，按位进行逻辑运算，这 4 个函数分别为：

$$F(X,Y,Z)=(X\wedge Y)\vee((\neg X)\wedge Z)$$

$$G(X,Y,Z)=(X\wedge Z)\vee(Y\wedge(\neg Z))$$

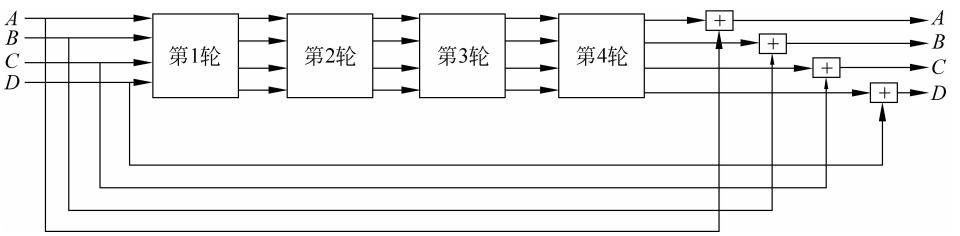

图 3.1 MD5 主循环

$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$$

其中，$\wedge$ 是与，$\vee$ 是或，$\neg$ 是非，$\oplus$ 是异或。如果 X、Y 和 Z 的对应位是独立和均匀的，那么这 4 个函数结果的每一位也应是独立和均匀的。

MD5 的基本操作过程如图 3.2 所示。

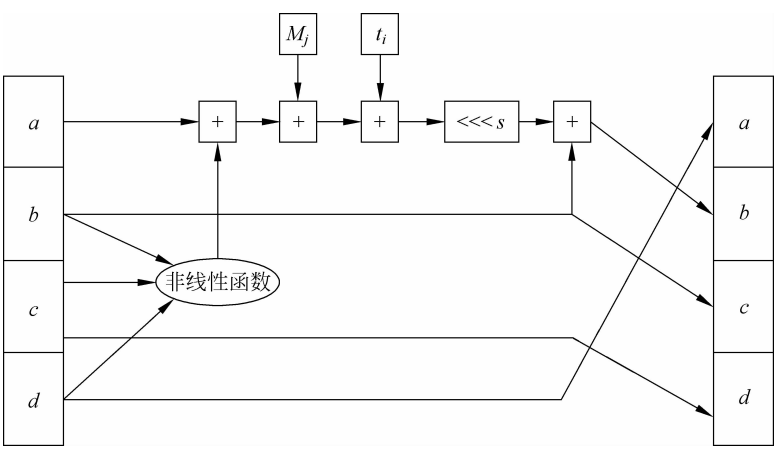

图 3.2 MD5 的基本操作过程

4 轮的迭代操作分别为：

FF(a,b,c,d,M_j,s,t_i) 表示 $a = b + ((a + F(b,c,d) + M_j + t_i) <<< s)$

GG(a,b,c,d,M_j,s,t_i) 表示 $a = b + ((a + G(b,c,d) + M_j + t_i) <<< s)$

HH(a,b,c,d,M_j,s,t_i) 表示 $a = b + ((a + H(b,c,d) + M_j + t_i) <<< s)$

II(a,b,c,d,M_j,s,t_i) 表示 $a = b + ((a + I(b,c,d) + M_j + t_i) <<< s)$

其中，M_j 表示消息的第 j 个子分组（从 0 到 15），常数 $t_i = 2^{32} \times \text{abs}(\sin(i))$ 的整数部分，$i = 1,2,\cdots,64$，i 单位是弧度。$+$ 为模 2^{32} 加法，$<<<s$ 表示循环左移 s 位。

这 4 轮共 64 步如下：

第 1 轮：

FF(a,b,c,d,M_0,7,0xd76aa478)
FF(d,a,b,c,M_1,12,0xe8c7b756)
FF(c,d,a,b,M_2,17,0x242070db)
FF(b,c,d,a,M_3,22,0xc1bdceee)

FF(a,b,c,d,M_4,7,0xf57c0faf)
FF(d,a,b,c,M_5,12,0x4787c62a)
FF(c,d,a,b,M_6,17,0xa8304613)
FF(b,c,d,a,M_7,22,0xfd469501)
FF(a,b,c,d,M_8,7,0x698098d8)
FF(d,a,b,c,M_9,12,0x8b44f7af)
FF(c,d,a,b,M_{10},17,0xffff5bb1)
FF(b,c,d,a,M_{11},22,0x895cd7be)
FF(a,b,c,d,M_{12},7,0x6b901122)
FF(d,a,b,c,M_{13},12,0xfd987193)
FF(c,d,a,b,M_{14},17,0xa679438e)
FF(b,c,d,a,M_{15},22,0x49b40821)

第 2 轮：

GG(a,b,c,d,M_1,5,0xf61e2562)
GG(d,a,b,c,M_6,9,0xc040b340)
GG(c,d,a,b,M_{11},14,0x265e5a51)
GG(b,c,d,a,M_0,20,0xe9b6c7aa)
GG(a,b,c,d,M_5,5,0xd62f105d)
GG(d,a,b,c,M_{10},9,0x02441453)
GG(c,d,a,b,M_{15},14,0xd8a1e681)
GG(b,c,d,a,M_4,20,0xe7d3fbc8)
GG(a,b,c,d,M_9,5,0x21e1cde6)
GG(d,a,b,c,M_{14},9,0xc33707d6)
GG(c,d,a,b,M_3,14,0xf4d50d87)
GG(b,c,d,a,M_8,20,0x455a14ed)
GG(a,b,c,d,M_{13},5,0xa9e3e905)
GG(d,a,b,c,M_2,9,0xfcefa3f8)
GG(c,d,a,b,M_7,14,0x676f02d9)
GG(b,c,d,a,M_{12},20,0x8d2a4c8a)

第 3 轮：

HH(a,b,c,d,M_5,4,0xfffa3942)
HH(d,a,b,c,M_8,11,0x8771f681)
HH(c,d,a,b,M_{11},16,0x6d9d6122)
HH(b,c,d,a,M_{14},23,0xfde5380c)
HH(a,b,c,d,M_1,4,0xa4beea44)
HH(d,a,b,c,M_4,11,0x4bdecfa9)
HH(c,d,a,b,M_7,16,0xf6bb4b60)
HH(b,c,d,a,M_{10},23,0xbebfbc70)
HH(a,b,c,d,M_{13},4,0x289b7ec6)
HH(d,a,b,c,M_0,11,0xeaa127fa)
HH(c,d,a,b,M_3,16,0xd4ef3085)
HH(b,c,d,a,M_6,23,0x04881d05)
HH(a,b,c,d,M_9,4,0xd9d4d039)
HH(d,a,b,c,M_{12},11,0xe6db99e5)
HH(c,d,a,b,M_{15},16,0x1fa27cf8)
HH(b,c,d,a,M_2,23,0xc4ac5665)

第 4 轮：

II(a,b,c,d,M_0,6,0xf4292244)
II(d,a,b,c,M_7,10,0x432aff97)
II(c,d,a,b,M_{14},15,0xab9423a7)
II(b,c,d,a,M_5,21,0xfc93a039)
II(a,b,c,d,M_{12},6,0x655b59c3)
II(d,a,b,c,M_3,10,0x8f0ccc92)
II(c,d,a,b,M_{10},15,0xffeff47d)
II(b,c,d,a,M_1,21,0x85845dd1)
II(a,b,c,d,M_8,6,0x6fa87e4f)
II(d,a,b,c,M_{15},10,0xfe2ce6e0)
II(c,d,a,b,M_6,15,0xa3014314)
II(b,c,d,a,M_{13},21,0x4e0811a1)
II(a,b,c,d,M_4,6,0xf7537e82)
II(d,a,b,c,M_{11},10,0xbd3af235)
II(c,d,a,b,M_2,15,0x2ad7d2bb)
II(b,c,d,a,M_9,21,0xeb86d391)

4 轮循环操作完成之后，将 A、B、C、D 分别加上 a、b、c、d，即

$$A = A + a$$
$$B = B + b$$
$$C = C + c$$
$$D = D + d$$

这里的加法是模 2^{32} 加法。然后用下一分组数据继续运行算法。

(4) 输出

对每个分组都作相应的处理以后，最后的输出就是 A、B、C 和 D 的级联，即 A 作为低位，D 作为高位，共计 128 位输出。

MD5 被广泛用于加密和解密技术中，在很多操作系统中(如 Linux 等)，用户的密码是以 MD5 值的方式保存的。用户在登录验证时，系统首先把用户输入的密码计算成 MD5 值，然后再和系统中保存的密码 MD5 值进行比较。在该操作过程中，系统在不知道用户密码的情况下就可以确定用户登录系统的合法性。这不但可以避免用户的密码被具有系统管理员权限的用户知道，而且还在一定程度上增加了密码被破解的难度。

3.2.3 SHA-1

安全散列算法(Secure Hash Algorithm，SHA)是由美国国家标准与技术研究院提出的，并于 1993 年作为联邦信息处理标准(FIPS 180)发布，1995 年又发布了其修订版(FIPS 180-1)，通常称为 SHA-1。SHA 算法也是建立在 MD4 算法之上的，其设计是在 MD4 的基础上改进而成的。

SHA-1 算法的输入是长度小于 2^{64} 位的消息，输出是 160 位的散列值，输入消息以 512 位的分组为单位进行处理。

与 MD5 算法相同，SHA 算法首先也需要对消息进行填充补位。补位是这样进行的：先添加一个 1，然后再添加若干个 0，使得消息长度满足对 512 取模后余数是 448。以“abc”为例显示补位的过程。

原始信息：

01100001 01100010 01100011

补位第一步：

01100001 01100010 01100011 1

首先补一个“1”，然后补 423 个“0”。

补位第二步：

01100001 01100010 01100011 10…0

可以把最后补位完成后的数据用十六进制写成下面的样子：

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000

现在，数据的长度是 448 了，可以进行下一步操作。

填充完信息后，需要将原始信息的长度补到已经进行了补位操作的信息后面。通常用一个 64 位的数据来表示原始信息的长度。如果信息长度不大于 2^{64}，那么第一个字就是 0。在进行了补长度的操作以后，整个信息就变成下面的样子(十六进制格式)：

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018

如果原始的信息长度超过了 512，需要将它补成 512 的倍数。然后把整个信息分成几个 512 位的数据块，分别处理每一个数据块，从而得到散列值。

与 MD5 算法不同，SHA 的中间结果和最终结果保存在 160 位的缓冲区中，缓冲区用 5 个 32 位的变量表示，这些变量初始化为：

A＝0x67452301
B＝0xefcdab89
C＝0x98badcfe
D＝0x10325476
E＝0xc3d2e1f0

在进入主循环函数处理前，将上面 5 个变量复制到 5 个变量中：A 到 a，B 到 b，C 到 c，D 到 d，E 到 e。

当设置好这 5 个变量后，就开始进入 4 轮，每轮 20 步的循环运算，循环的总次数是信息中 512 位信息分组的数目，主循环结构如图 3.3 所示。每一步操作都使用一个非线性的逻辑函数对 a、b、c、d、e 中的 3 个变量进行一次按位的逻辑运算。

这几个非线性函数定义为：

$$f_t(X,Y,Z)=\begin{cases}(X\wedge Y)\vee((\neg X)\wedge Z) & 0\leqslant t\leqslant 19\\ X\oplus Y\oplus Z & 20\leqslant t\leqslant 39\\ (X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z) & 30\leqslant t\leqslant 59\\ X\oplus Y\oplus Z & 40\leqslant t\leqslant 79\end{cases}$$

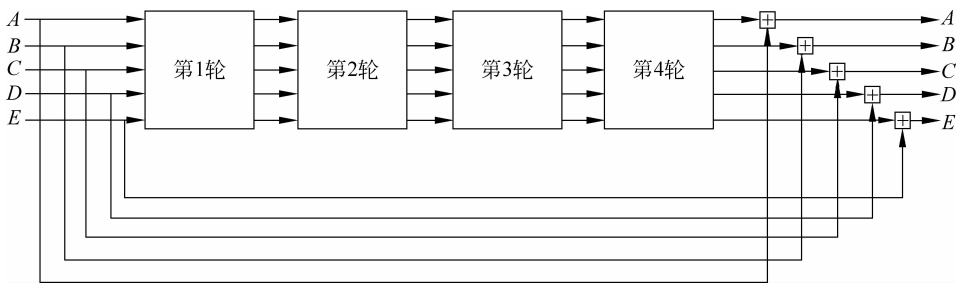

图 3.3 SHA-1 的主循环

其中，∧、∨、¬和⊕分别是与、或、非和异或运算。

与此同时，每一步操作也都使用一个加法常量 K_t，K_t 定义如下：

$$K_t = \begin{cases} 0\text{x}5\text{a}827999 & 0 \leqslant t \leqslant 19 \\ 0\text{x}6\text{ed}9\text{eba}1 & 20 \leqslant t \leqslant 39 \\ 0\text{x}8\text{f}1\text{bbcdc} & 40 \leqslant t \leqslant 59 \\ 0\text{xca}62\text{c}1\text{d}6 & 60 \leqslant t \leqslant 79 \end{cases}$$

这些数的取值来自于 $0\text{x}5\text{a}827999 = 2^{30} \times \sqrt{2}$，$0\text{x}6\text{ed}9\text{eba}1 = 2^{30} \times \sqrt{3}$，$0\text{x}8\text{f}1\text{bbcdc} = 2^{30} \times \sqrt{5}$，$0\text{xca}62\text{c}1\text{d}6 = 2^{30} \times \sqrt{10}$。

接着对 512 位的信息进行处理，将其从 16 个 32 位的信息分组(M_0，…，M_{15})变成 80 个 32 位的信息分组(W_0，…，W_{79})。W_t 定义如下：

$$W_t = \begin{cases} M_t & \text{当 } t = 0,1,\cdots,15 \\ (M_{t-3} \oplus M_{t-8} \oplus M_{t-14} \oplus M_{t-16}) <<< 1 & \text{当 } t = 16,17,\cdots,79 \end{cases}$$

设 t 是操作序号($t=0,\cdots,79$)，$<<<s$ 表示循环左移 s 位，则 SHA-1 中的每一步操作可表示为：

$TEMP = (a <<< 5) + f_t(b,c,d) + e + W_t + K_t$

$e = d$

$d = c$

$c = b <<< 30$

$b = a$

$a = TEMP$

图 3.4 给出了 SHA-1 的一次基本操作的运算过程。在 4 轮循环结束后，将进行：

$A = A + a$

$B = B + b$

$C = C + c$

$D = D + d$

$E = E + e$

这里的加法是模 2^{32} 加法。

然后用同样的方法对下一个分组进行运算，直到所有分组都处理完毕为止，最后将 A、B、C、D、E 输出，就得到 SHA 的散列值。

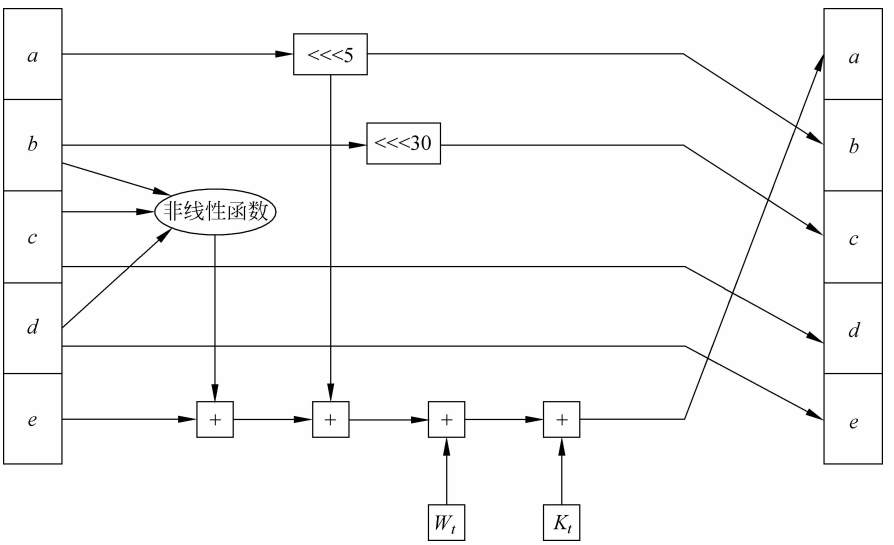

图 3.4　SHA-1 的基本操作过程

从上面的原理可以看出，通过使用不同移位、不同逻辑函数和不同初始变量，SHA 和 MD5 都实现了同样的功能。

1981 年，NIST 发布 FIPS 180-2，新增了三个哈希函数，分别为 SHA-256、SHA-384 和 SHA-512，其散列值的长度分别为 256、384 和 512。同时，NIST 指出 FIPS 180-2 的目的是要与使用 AES 而增加的安全性相适应。SHA 性质对比如表 3.1 所示。

表 3.1　SHA 性质对比

性　　质	SHA-1	SHA-256	SHA-384	SHA-512
散列值长度	160	256	384	512
信息长度	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
分组大小	512	512	1024	1024
字长	32	32	64	64
步数	80	80	80	80

3.3　消息认证技术

消息认证是指使合法的接收方能够检验消息是否真实的过程。检验内容包括验证通信的双方和验证消息内容是否伪造或遭到篡改。消息认证技术主要通过密码学的方法来实现，对通信双方的验证可采用数字签名和身份认证技术，对消息内容是否伪造或遭篡改通常使用的方式是在消息中加入一个认证码，并加密后发送给接收方，接收方通过对认证码的比较来确认消息的完整性。

本节首先对消息认证技术进行概述，然后介绍基于密码学的各种认证方法。

3.3.1 概述

随着因特网技术的发展，对网络传输过程中信息的保密性提出了更高的要求，这些要求主要包括：

(1) 对敏感的信息进行加密，即使别人截取信息也无法得到其内容。

(2) 保证数据的完整性，防止截获人在信息中加入其他信息。

(3) 对数据和信息的来源进行验证，以确保发信人的身份。

现在业界普遍采用加密技术来实现以上要求，实现消息的安全认证。消息认证就是验证所收到的消息确实是来自真正的发送方且未被修改的消息，也可以验证消息的顺序和及时性。

消息认证实际上是对消息产生一个指纹信息——MAC(消息认证码)，消息认证码是利用密钥对待认证消息产生的新数据块，并对该数据块加密得到的。它对待保护的信息来说是唯一的，因此可以有效地保证消息的完整性，以及实现发送消息方的不可抵赖和不能伪造性。

消息认证技术可以防止数据伪造和篡改，以及证实消息来源的有效性，已广泛应用于当今的信息网络环境中。随着密码技术与计算机计算能力的提高，消息认证码的实现方法也在不断地改进和更新，实现方式的多样化为安全的消息认证提供了保障。

3.3.2 消息认证方法

消息认证主要使用密码技术来实现。在实际使用中，通过消息认证函数 f 产生用于鉴别的消息认证码，将其用于某个身份认证协议，发送方和接收方通过消息认证码对其进行相应的认证。

由此可见，在消息认证中，认证函数 f 是认证系统的一个重要组成部分。常见的认证函数主要有以下三种：

(1) 消息加密：将整个消息的密文作为认证码。

(2) 哈希函数：通过哈希函数产生定长的散列值作为认证码。

(3) 消息认证码(Message Authentication Code，MAC)：将消息与密钥一起产生定长值作为认证码。

1. 基于加密方法的消息认证

就加密的方式而言，加密可以基于对称加密的消息认证和基于非对称加密的消息认证。

基于对称加密方式的消息认证简单明了，如图 3.5 所示，假设 K 是通信双方共同拥有的会话密钥，发送方 A 只需使用 K 对消息 M 进行加密，将密文 C 发送给接收方 B 即可。由于密钥 K 只有 A 和 B 共同拥有，因此能够保证消息的机密性。此外，由于 A 是除 B 外唯一

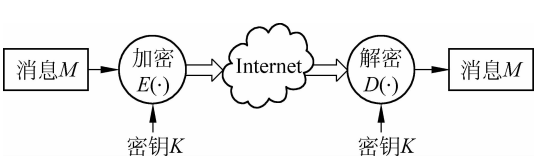

图 3.5 基于对称加密方式的消息认证过程