视频讲解

第5章 安全认证技术

认证分为消息认证和身份认证。消息认证是指接收方对收到的消息进行检验，检验内容包括：消息的源地址，目的地址，消息的内容是否受到篡改，消息的有效生存时间，等等。消息认证可以是实时的，也可以是非实时的。例如，网上银行对消息的认证属于实时认证，而电子邮件中对消息的认证属于非实时认证。

消息认证的基本方法有采用哈希（Hash）函数和采用消息认证码（Message Authentication Code，MAC）。这两种方法的区别在于是否需要密钥的参与。消息认证码是指消息被一密钥控制的公开函数作用后产生的固定长度的、用作认证符的数值。

身份认证可以说是信息系统安全的基础，访问主体要访问信息系统的资源首先需要通过身份认证。身份认证是指系统对网络主体进行验证的过程，用户必须向系统证明他是谁。系统验证通信实体的一个或多个参数的真实性和有效性，以判断他的身份是否和他所声称的身份一致。身份认证的目的是在不可信的网络上建立通信实体之间的信任关系。在网络安全中，身份认证的地位非常重要，它是最基本的安全服务之一，也是信息安全的第一道防线。在具有安全机制的系统中，任何一个想要访问系统资源的人都必须首先向系统证实自己的合法身份，然后才能得到相应的权限。例如，轻量级目录访问协议（Lightweight Directory Access Protocol，LDAP）、远程认证拨号用户服务（Remote Authentication Dial in User Service，RADIUS）、安全断言标记语言（Security Assertion Markup Language，SAML）、OpenID Connect、线上快速认证（Fast Identity Online，FIDO）、Kerberos 以及生物识别技术等常用的身份认证方案。

任何认证在功能上基本都有上、下两层：下层中一定有某种产生认证标识的函数，认证标识是一个用来认证消息的值；上层协议中将该函数作为原语使接收方可以验证消息的真实性。用来产生认证标识的函数可以分为如下三类。

（1）消息加密：对整个消息加密后的密文作为认证。消息加密本身提供了一种认证手段。对称密码和非对称密码体制中对消息加密的分析是不相同的。发送方 A 用 A 和接收方 B 共享的密钥 K 对发送到接收方 B 的消息 M 加密，如果没有其他方知道该密钥，那么可提供保密性，因为任何其他方均不能恢复出消息明文。此外，接收方 B 可确信该消息是由发送方 A 产生的因为除接收方 B 外只有发送方 A 拥有 K，发送方 A 能产生出可用 K 解密的密文，所以该消息一定来自发送方 A。由于攻击方不知道密钥，也就不知如何改变密文中的信息位才能在明文中产生预期的改变，因此若接收方 B 可以恢复出明文，则接收方 B 可以认为 M 中的每一位都未被改变。

（2）Hash 函数：将任意长的消息映射为定长的散列值的函数，以该散列值作为认证。

（3）MAC：消息和密钥的函数，它产生定长的值，以该值作为认证。

对用户的身份认证手段可以按照不同的分类标准进行分类。若仅通过一个条件的符合来证明一个人的身份，则称为单因子认证。由于仅使用一种条件判断用户的身份容易被仿冒，可以通过组合两种不同条件来证明一个人的身份，称为双因子认证。根据身份认证技术是否使用硬件可以分为软件认证和硬件认证，根据认证需要验证的条件，可以分为单因子认证和双因子认证，根据认证信息可以分为静态认证和动态认证。身份认

证技术的发展经历了从软件认证到硬件认证，从单因子认证到双因子认证，从静态认证到动态认证的过程。

用户名/静态口令是最简单，也是最常用的身份认证方法，它是基于“你知道什么”的验证手段。静态口令由用户自己设定，只有用户自己才知道，因此只要能够正确输入口令，计算机认为就是这个用户。实际上，许多用户防止忘记口令，经常采用如自己或家人的生日、电话号码等容易被他人猜测到的字符串作为口令，或者把口令抄在自己认为安全的地方，这都存在着许多安全隐患，极易造成口令泄露。即使能保证用户口令不被泄露，由于口令是静态的数据，并且在验证过程中需要在计算机内存中和网络中传输，而每次验证过程使用的验证信息都是相同的，很容易驻留在计算机内存中的木马程序或网络中的监听设备截获。因此，静态口令方式一直是极不安全的身份认证方式。虽然也存在定期更改静态口令、大小写字母和数字混排的口令、加入特殊字符的口令等增强口令安全性的策略，但这些并不被大多数用户所接受。

动态口令技术是一种让用户的密码按照时间或使用次数不断动态变化，每个密码只使用一次的技术。用于支持认证“某人拥有某东西”的认证。它采用动态令牌的专用硬件，内置电源、密码生成芯片和显示屏，密码生成芯片运行专门的密码算法，根据当前时间或使用次数生成当前密码并在显示屏上显示。认证服务器采用相同的算法计算当前的有效密码。用户使用时只需要将动态令牌上显示的当前密码输入客户端计算机，即可实现身份的确认。由于每次使用的密码必须由动态令牌来产生，只有合法用户才持有该硬件，所以只要密码验证通过就可以认为该用户身份是可靠的。而用户每次使用的密码都不相同，即使黑客截获了一次密码，也无法利用这个密码来仿冒合法用户的身份。动态口令认证相比静态口令认证安全性方面提高了不少，但是也不能满足可信网络的需要。动态口令技术采用一次一密的方法，有效地保证了用户身份的安全性。如果客户端硬件与服务器端程序的时间或次数不能保持良好的同步，就可能发生合法用户无法登录的问题。用户每次登录时还需要通过键盘输入一长串无规律的密码，一旦看错或输错就要重新输入，使用非常不方便。

5.1 Hash 函数

分组密码、流密码和 Hash 函数是密码学的三个重要分支，分组密码在数据加密和 MAC 等领域有着广泛的应用，流密码应用领域主要是在军事、外交等部门，Hash 函数在数字签名、身份认证、消息认证等很多领域有着广泛的应用。

1989 年，Rivest 设计出 MD2 算法，该算法是针对 8 位计算机设计的，所以填充后的消息长度是 16 的倍数，每次处理的消息块为 16 位。1990 年，Rivest 设计出了 MD4 算法，该算法是针对 32 位计算机设计的，该算法首先将信息填充，确保填充后的消息长度模 512 余 448。然后，在末尾添加一个以 64 位二进制表示的最初长度信息，消息的总长度将是 512 的倍数，将消息划分成 512 个比特块分别处理。1992 年，Rivest 在 MD4 算法基础上设计出了更为安全的 MD5 算法。MD5 算法的信息摘要大小和填充准则与 MD4 完全相同。MD5 在 MD4 的基础上增加了一轮，所以速度比 MD4 稍慢，却更安全。

1993年，美国RSA公司在MD5基础上作出改进，提出SHA-0算法，该算法被作为美国国家标准使用。SHA-0算法继承了MD4算法结构清晰、速度快和运算简单的优点。但是提出后不久发现，SHA-0算法在消息扩展过程中存在一些漏洞，于是RSA公司在1994年改进消息扩展的方式，改进后的算法称为SHA-1。2002年，美国国家标准与技术研究院(NIST)又根据实际情况在SHA-1的基础上增加了输出长度，形成SHA-256、SHA-384、SHA-512算法，它们统称为SHA-2。

5.1.1 Hash函数性质

Hash函数在现代密码学中占有很重要的地位。Hash函数是可以将任意长度的字符串压缩成固定长度字符串的函数。Hash函数的输出结果称为散列值，也称为消息摘要。通俗地讲，Hash函数用于压缩消息，将任意长的消息映射为固定长度的散列值。但是，对于给定的散列值计算出其原始消息在计算上是不可行的。Hash函数主要应用在数字签名、完整性检验、身份认证、密钥交换和伪随机数产生等领域。为了实现对数据的认证，Hash函数应满足下列性质。

(1) 为一个给定的输出找出能映射到该输出的一个输入在计算上是困难的。

(2) 为一个给定的输入找出能映射到同一输出的另一个输入在计算上是困难的。

(3) 要发现不同的输入映射到同一输出在计算上是困难的。

(4) H 可应用于任意大小的数据块。

(5) H 产生定长的输出。

其中：性质(1)给出了Hash函数单向性的概念；性质(2)和(3)给出了Hash函数无碰撞性的概念。

Hash函数必须满足以下条件。

(1) 单向性：设 H 是一个Hash函数，如果对于任意给定的 z，寻找满足 $H(x)=z$ 的消息 x 在计算上是不可行的，那么 H 是单向的。

(2) 弱无碰撞：设 H 是一个Hash函数，给定一个消息 x，如果寻找另外一个与 x 不同的消息 x' 使得在计算上是不可行的，那么 H 关于消息 x 是弱无碰撞的。

(3) 强无碰撞：设 H 是一个Hash函数，如果寻找两个不同的消息 x 和 x' 使得在计算上是不可行的，那么 H 是强无碰撞的。

(4) 有碰撞：如果Hash函数对不同的输入可产生相同的输出，那么称该函数具有碰撞性。

可以证明，如果一个Hash函数 H 不是单向的，那么 H 一定不是强无碰撞的；如果一个HASH函数 H 是强无碰撞的，那么 H 一定是单向的。

(5) 效率：对于任意给定的输入 x，计算 $y=H(x)$ 要相对容易。并且，随着输入 x 长度的增加，虽然计算 $y=H(x)$ 的工作量会增加，但增加的量不会太快。

(6) 压缩：对于任意给定的输入 x，都会输出固定长度的 $y=H(x)$，且 y 要比 x 小得多。

在Hash函数中任何输入字符串中单个位的变化，将会导致输出位串中大约一半的

位发生变化。图 5-1 是 Hash 函数结构，该结构称为迭代 Hash 函数，目前广泛应用的 MD5、SHA、Ripend160、Whirlpool 等算法的实现都遵循该结构。f 为压缩函数，用于对分组进行迭代处理。Hash 函数重复使用压缩函数 f，它的输入是前一步得出的 n 位输出（称为链接值）和一个 b 位的消息分组，输出为一个 n 位分组。链接值（CV）的初始值（IV）由算法在开始时指定，而最后的输出值即为 Hash 函数值。基于图 5-1 所示的迭代 Hash 函数在算法实现可以归纳如下。

$$CV_0 = IV = n \text{ 位初始值}$$

$$CV_i = f(CV_{i-1}, Y_{i-1}), \quad 1 \leqslant i \leqslant L$$

$$H(M) = CV_L$$

其中：Hash 函数的输入为消息 M，经填充后的消息分成 $Y_0, Y_1, \cdots, Y_{L-1}$，共 L 个分组。

通常，Hash 函数中，其输入消息被划分成 L 个固定长度的分组，每一分组长为 b 位，最后一个分组不足 b 位时需填充为 b 位，最后一个分组包含输入的总长度。由于输入中包含长度，所以攻击方必须找出具有相同散列值且长度相等的两条消息，或者找出两条长度不等但加入消息长度信息后散列值相同的消息，从而增加了攻击的难度。Merkle 和 Damgard 发现，如果压缩函数具有抗碰撞能力，那么迭代 Hash 函数也具有抗碰撞能力，因此 Hash 函数常使用迭代结构，这种结构可用于对任意长度的消息产生安全 Hash 函数。

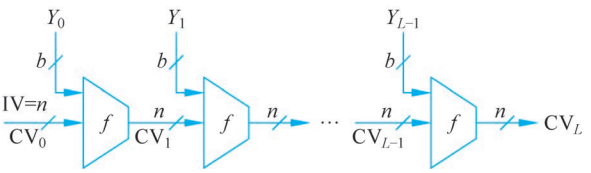

图 5-1　Hash 函数结构

注：IV—初始值；CV—链接值；Y_i—第 i 个输入分组；f—压缩函数；L—输入分组数；n—Hash 函数值的位长；b—输入分组的位长。

5.1.2　Hash 函数分类

常用的算法主要有 MD5、SHA-1、SHA-256、SHA-512。MD5 是对 MD4 的改进版本。其将输入以 512 位进行分组。其输出是 4 个 32 位字的级联，与 MD4 相同。MD5 比 MD4 来得复杂，并且速度较要慢，但更安全，在抗分析和抗差分方面表现更好。

SHA-1 是与数字签名算法（DSA）一起使用的，其对长度小于 264 位的输入，可以产生长度为 160bit 的散列值，因此抗穷举性更好。SHA-1 由 SHAO 升级而来，设计思路上源于 MD4。SHA-1 最大输入消息长度为 264 位。SHA-256、SHA-512 的输入为任意长度，输出散列值长度分别为 256 位和 512 位。几种 SHA 算法主要区别在于所提供的安全级别不同。Hash 函数的分类有很多种，按照 Hash 函数设计方法可以分为三类。

1. 基于分组密码设计的 Hash 函数

基于分组密码构造 Hash 函数仅仅局限于对分组密码输入、输出模式加以变换构造压缩函数，不包括利用分组密码组件来构造 Hash 函数。例如，Tiger 算法的设计者继承

了部分分组密码的设计思想，使用了S盒，但它是一个专用的Hash函数，不是一个基于分组密码构造的Hash函数。

对初始消息 m 进行填充，填充后消息总长度是单个分组长度 n 的整数倍，这样，可以将初始消息分成 t 个分组 $M_i(i=1,2,\cdots,t)$，每个分组的长度为 n。h_0 是初始向量，$h_i=f(M_i,H_{i-1})(i=1,2,\cdots,t)$，$E_K$ 是以 K 为密钥分组长度为 n 的分组加密算法，m 的Hash函数值 $H(m)=H_t$。

利用分组密码的方法来构造Hash函数有以下优点：相对于Hash函数，分组密码的发展较为前沿，设计理论也很成熟，因此很多分组密码的研究成果可以用于Hash函数。而其劣势：一是分组密码构造的Hash函数的效率将比分组密码低很多，原因在于分组密码的密钥一般是不变的，而由分组密码构造的Hash函数是由初始消息转化为密钥，密钥的转化过程需要很多时间；二是对分组密码不构成安全威胁的漏洞在Hash函数上可能会带来很大安全隐患。

2. 标准Hash函数

标准的Hash函数主要分为：MDx系列和SHA系列，MDx系列主要包括MD4、MD5、HAVAL、RIPEMD、RIPEMD-160等，SHA系列主要包括SHA-0、SHA-1、SHA-224、SHA-256、SHA-384、SHA-512。

标准Hash函数就是直接构造的Hash函数，这类方法构造的Hash函数在实现速度上要比利用分组密码方法构造Hash函数快得多，但算法的扩散性一般不如采用分组密码方法构造的Hash函数。在2004年国际密码学会议上，王小云等宣布了对一系列Hash函数的碰撞结果，包括MD4、MD5、HAVAL-128和REPEMD等各种算法的碰撞攻击。此后，国际上也陆续出现了一系列的攻击算法，给使用中的Hash函数带来了重创。随着MDx和SHA系列算法的安全性受到质疑，如何改进MD迭代结构，或者设计更好的迭代结构，如何构造和评价一个安全的Hash函数，这些问题变得迫在眉睫。

3. 基于离散对数构造Hash函数

基于某些困难数学问题，如离散对数问题、因子分解问题、背包问题等构造Hash函数，这些Hash函数的安全性依赖相应数学问题的困难性。最具代表性的一种算法是由Chaum、Heijst和Pfitzmann在1992年提出的Chaum-Heijst-PfitzmannHash算法。该算法在实际运行中速度不是很快，但在合理的条件下可以证明它是安全的。基于离散对数的Hash函数大多采用比特的逻辑运算，即与、或、异或。这些运算必须加入其他的运算才能引入非线性，但是其效率不高，安全性较脆弱，所以应用并不广泛。

Hash函数可以按是否有密钥参与运算分为不带密钥的Hash函数和带密钥的Hash函数。

不带密钥的Hash函数在运算过程中没有密钥参与。不带密钥的Hash函数的散列值只是消息输入的函数，无需密钥就可以计算。因此，这种类型的Hash函数不具有身份认证功能，它仅提供数据完整性检测，如篡改检测码(MDC)。按照所具有的性质，MDC又可分为弱单向Hash函数(OWHF)和强单向Hash函数(CRHF)。

带密钥的Hash函数在消息送算过程中有密钥参与。这类Hash函数需要满足各种

安全性要求，其散列值同时与密钥和消息输入相关，只有拥有密钥的人才能计算出相应的散列值。不带密钥的 Hash 函数不仅能够检验数据完整性，而且能提供身份认证功能，称为消息认证码(MAC)。消息认证码的性质保证了只有拥有秘密密钥 Hash 函数的人才能产生正确的消息——MAC 对。

5.1.3 MD5 算法

麻省理工学院 Ron Rivest 提出，可将任意长度的消息经过变换得到一个 128 位的散列值。MD5 以 512 位分组来处理输入的信息，每一分组又划分为 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 位分组级联后生成 128 位散列值。MD5 算法原理如图 5-2 所示。

MD5 算法的步骤如下。

1. *数据填充与分组*

(1) 将输入信息 M 按顺序每 512 位一组进行分组，$M=M_1,M_2,\cdots,M_{n-1},M_n$。

(2) 将信息 M 的 M_n 长度填充为 448 位。

① 当 M_n 长度 $L<448$ 时，在信息 M_n 后加一个“1”，再填充 $447-L$ 个“0”，使最后的信息 M_n 长度为 448 位。

② 当 M_n 长度 $L>448$ 时，在信息 M_n 后加一个“1”，再填充 $512-L+447$ 个“0”，使最后的信息 M_n 长度为 512 位，M_{n+1} 长度为 448 位。

2. *初始化散列值*

在 MD5 算法中要用到 4 个 32 位变量，分别为

$$A=0x01234567$$
$$B=0x89abcdef$$
$$C=0xfedcba98$$
$$D=0x76543210$$

在 MD5 算法过程中，4 个 32 位变量称为链接变量，它们始终参与运算并形成最终的散列值。

3. *计算散列值*

(1) 将填充后的信息按每 512 位分为一块，每块按 32 位为一组划分成 16 个分组，即 $M_i=M_{i0},M_{i2},\cdots,M_{i15},i=1\sim n$。

(2) 分别对每一块信息进行 4 轮计算(主循环)。每轮定义一个非线性函数：

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

(3) 将 A、B、C、D 变量分别复制到变量 a、b、c、d 中。

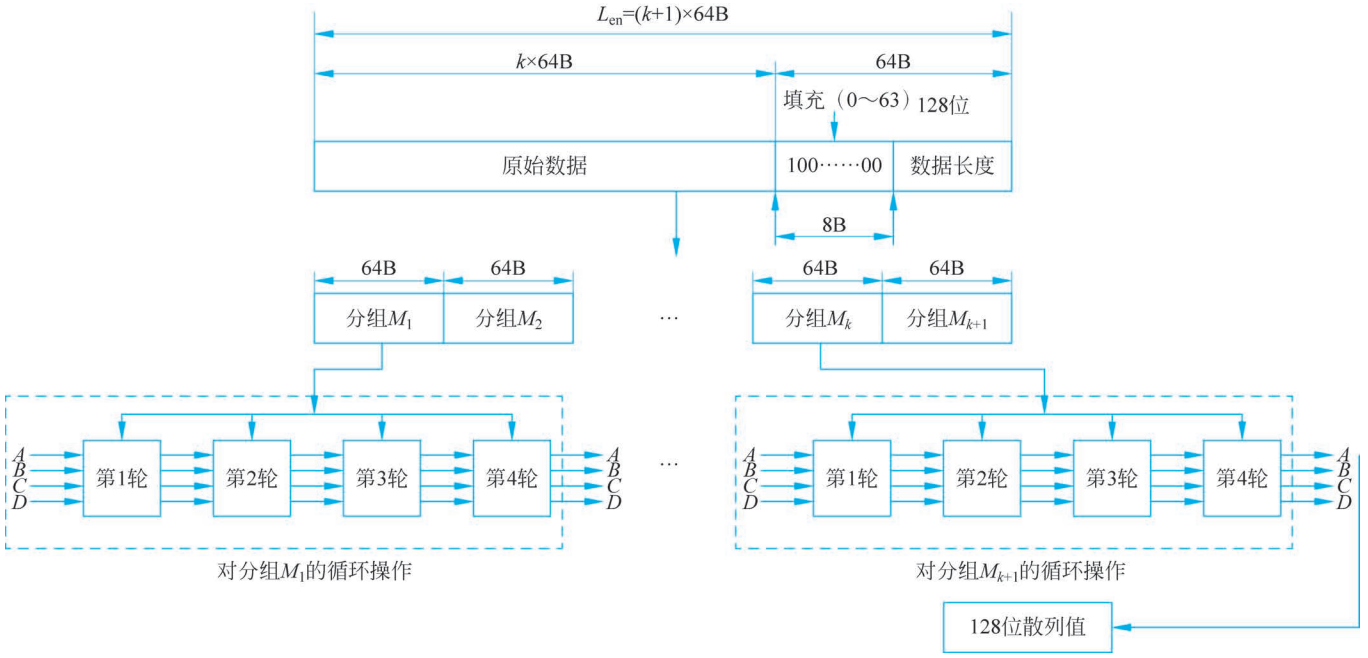
L_{en}=(k+1)×64B
k×64B
64B
填充（0～63）128位
原始数据
100……00
数据长度
8B
64B
64B
64B
64B
分组M_1
分组M_2
分组M_k
分组M_{k+1}
A
B
C
D
第1轮
第2轮
第3轮
第4轮
对分组M_1的循环操作
对分组M_{k+1}的循环操作
128位散列值

图 5-2　MD5 算法原理

(4) 每一轮进行 16 次操作，每次操作对 a、b、c、d 中的 3 个变量做一次非线性函数运算，然后将所得的结果与第 4 个变量、信息的一个分组 M_j 和一个常数 t_i 相加。再将所得的结果循环左移一个不定数 s，并加上 a、b、c、d 中的一个变量。

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

在第 i 步中，常数 t_i 取值为 $2^{32}\times abs(\sin(i))$的整数部分。这样就可以得到 4 轮共 64 步操作。

5.1.4 SHA-1 算法

MD5 目前的应用已经很广泛。另一个应用较为广泛的标准是由 NIST 提出的安全散列算法(Secure Hash Algorithm，SHA)。安全散列标准(SHS)于 1992 年 1 月 31 日在美国联邦记录中公布，1993 年，美国 RSA 公司在 MD5 基础上做出改进，提出 SHA-0 算法，该算法被作为美国国家标准使用。该算法通过四圈压缩函数，对 512 位的输入产生 160 位的散列值。SHA-0 算法继承了 MD4 算法结构清晰、速度快和运算简单的优点。但是提出后不久发现，SHA-0 算法在消息扩展过程中存在一些漏洞，于是在 1995 年 NIST 改进 SHA-0 算法，改进后的算法称为 SHA-1。SHA-1 与 SHA-0 仅在消息扩展上存在差异。SHA-1 目前已广泛应用于各种密码方案。这些算法的主要不同在于操作数长度、初始向量值、常量值和最后产生信息摘要的长度。表 5-1 列出了 SHA-1 和 SHA-2 系列算法的主要区别。

表 5-1 SHA-1 和 SHA-2 系列算法特性对比

SHA 标准	信息块大小/位	循环次数	字长/位	散列值/位
SHA-1	<264	80	32	160
SHA-224	<264	64	32	224
SHA-256	<264	64	32	256
SHA-384	<2128	80	64	384
SHA-512	<2128	80	64	512

SHA-1 函数的输入消息长度不超过 264 位，输出长度 160 位，SHA-1 的输入在填充后被分割成 512 位的消息块，对每一个消息块，经过 4 轮迭代，每轮 20 步操作，反复压缩迭代，最终输出消息的散列值，该算法详细步骤如下。

(1) 对初始消息进行填充，按照“第一位是 1、后面位全部是 0”的原则进行填充，填充后的长度要满足模 512 后余 448，这样可以保证附加 64 位的初始消息长度后是 512 的倍数。将填充后的消息分组，每组 512 位，消息分组为 $mt(t=1,2,\cdots,N)$，共 N 组。

(2) 初始化消息摘要缓存器，SHA-1 需要 A、B、C、D、E 5 个 32 位的寄存器。这些寄存器用于存放算法的中间结果和最终的散列值。寄存器 A、B、C、D、E 的初始值依次为 0x67452301、0xEFCDAB89、0x98BADCFE、0x10325476、0xC3D2E1F0。

(3) 处理每一个 512 位的消息分组。每个消息分组要经过这 4 个圈循环,每个圈循环有 20 步迭代压缩,在每个圈循环中使用一个非线性的函数,4 个非线性函数如下(f_i 为第 i 圈的非线性函数):

$$f_1(X,Y,Z)=(X \wedge Y) \oplus (-X \wedge Z)$$

$$f_2(X,Y,Z)=X \oplus Y \oplus Z$$

$$f_3(X,Y,Z)=(X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$$

$$f_4(X,Y,Z)=X \oplus Y \oplus Z$$

用到 4 个常量分别为 $K_1=0\text{x}5\text{A}827999$,$K_2=0\text{x}6\text{ED}9\text{EBA}1$,$K_3=0\text{x}8\text{F}1\text{BBCDC}$,$K_4=0\text{xCA}62\text{C}1\text{D}6$。

每个消息分组 m_t,长度 512 位,将这 512 位分成 16 个 32 位字 $M_0,M_1,M_2,\cdots,M_{15}$,利用这 16 个 32 位字扩展成 80 个字长的扩展报文 $W_0,W_1,W_2,\cdots,W_{79}$。扩展算法如下:

$$W_i=M_i, \quad i=0,1,\cdots,15$$

$$W_i=(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16})<<<1, \quad i=16,17,\cdots,79$$

SHA-0 的消息扩展算法如下:

$$W_i=M_i, \quad i=0,1,\cdots,15$$

$$W_i=W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \quad i=16,17,\cdots,79$$

消息扩展算法不同也是 SHA-1 与 SHA-0 唯一不同的地方。SHA-1 算法每一步的迭代流程如图 5-3 所示。

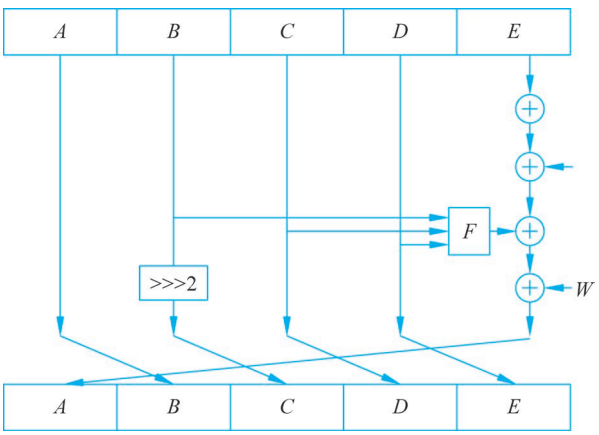

图 5-3 SHA-1 算法迭代流程图

(4) 当所有分组都处理完成后,最后产生的 A、B、C、D、E 连接构成 160 位的散列值。

5.1.5 SHA-512 算法

SHA-512 算法中规定的输入消息最大不超过 2128 位,输出为 512 位固定长度的散列值。如图 5-4 所示,SHA-512 算法的实现与 MD5 类似,其实现过程主要有以下步骤。

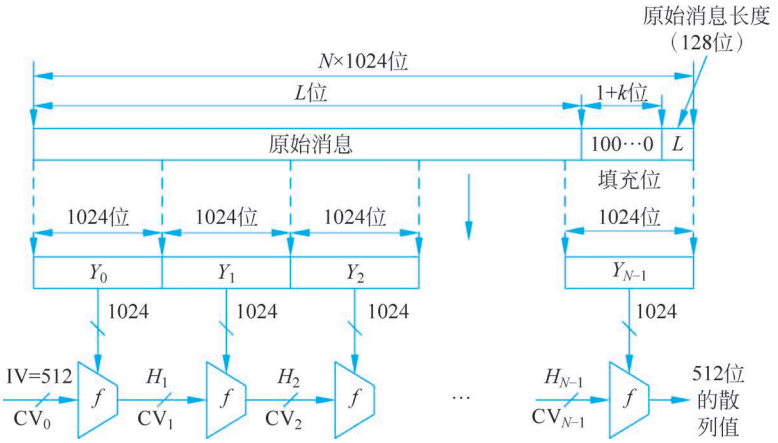

图 5-4　SHA-512 算法实现

(1) 消息填充。SHA-512 算法规定,输入消息以 1024 位的分组为组织进行处理。为此,在算法开始时首先要对原始消息进行填充,使其长度是 1024 位的整数倍。由于在消息的最后还要添加消息长度信息,所以即使是原始消息正好是 1024 位的整数位,仍然需要进行填充,此时的填充位在 1～1024 位之间。

具体填充方法:假设消息的长度为 L 位,首先将"1"添加到消息的末尾,再添加 k 个"0",满足条件 $L+1+K+128=N\times1024$,其中,N 为正整数,128 位用于表示原始消息的长度。通过消息填充操作,扩展后的消息被表示为一串长度为 1024 位的消息分组 y_0,y_1,…,y_{N-1},扩展后消息的总长度为 $N\times1024$ 位。

(2) 初始化 Hash 函数缓冲区。Hash 函数计算的中间结果和最终结果都保存在 512 位的缓冲区中,分别用 8 个 64 位的寄存器(A,B,C,D,E,F,G,H)表示,并将这些寄存器初始化为下列 64 位的整数(十六进制值):

A＝6A09E667F3BCC908,　E＝510E527FADE682D1

B＝BB67AE8584CAA73B,　F＝9B05688C2B3E6C1F

C＝3C6EF372FE94F82B,　G＝1F83D9ABFB41BD6B

D＝A54FF53A5F1D36F1,　H＝5BE0CD19137E2179

这些值以高端格式存储,即字的最高有效字节存于低地址字节位置(最左边)。

(3) 以 1024 位分组(16 个字)为组织处理消息。该操作的核心是中模块 f,该模块要进行 80 轮的运算。图 5-5 给出了 SHA-512 算法对每个 1024 位分组的处理。

① 每一轮处理中都把步骤(2)形成的 512 位 Hash 函数缓冲区中初始化值 A、B、C、D、E、F、G、H 作为输入,并更新缓冲区的值。

② 在进行第一轮操作时,缓冲区的值是中间的 Hash 函数值 H_{i-1}。

③ 每一轮使用一个 64 位的值 W_t($0\leqslant t\leqslant79$),该值由当前被处理的 1024 位消息分组 Y_i 导出,导出算法采用消息调度算法。

④ 每一轮还将使用常数 K_t($0\leqslant t\leqslant79$),这些常数的获取方法是前 80 个素数取 3 次根,再取小数部分的前 64 位。这些常数提供了 64 位随机串集合,可以消除输入数据中

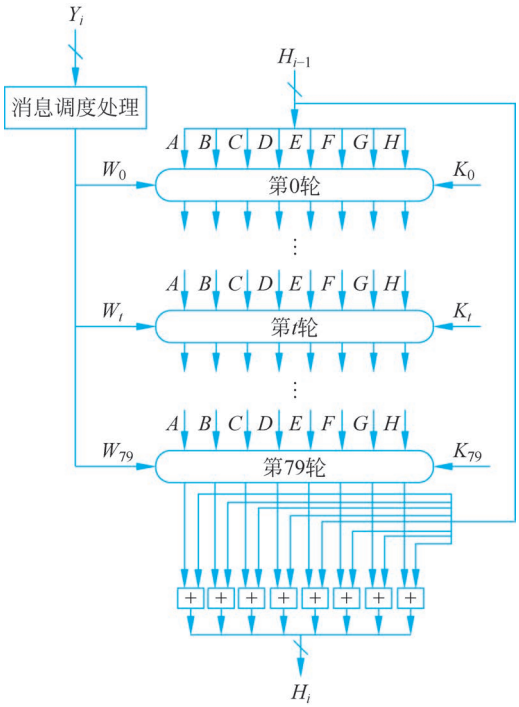

图 5-5 SHA-512 对每个 1024 位分组的处理

存在的任何规则性。

⑤ 最后一轮的输出和第一轮的输入 H_{i-1} 相加产生 H_i。缓冲区里的 8 个字和 H_{i-1} 里的相应字独立进行模 2^{64} 的加法运算。

(4) 输出。所有的 N 个 1024 个分组都处理结束后,最后输出的是 512 位散列值。

SHA-512 运算总结如下:

$$H_0 = \mathrm{IV}$$

$$H_I = \mathrm{SUM}_{64}(H_{i-1}, ABCDEFGH_i)$$

$$\mathrm{MD} = H_N$$

其中:HV 为上述操作中步骤(2)定义的 A,B,C,D,E,F,G,H 缓冲区的初始值;$\mathrm{ABCDEFGH}_i$:第 i 个消息分组处理的最后一轮的输出;N 为消息(包含填充和 128 位消息长度)的 1024 位分组数;SUM_{64} 为对输入对中的每一个字进行独立的模 2^{64} 加运算;MD 为最后的散列值。

5.1.6 SM3 算法

SM3 算法是中国国家密码管理局于 2010 年公布的中国商用密码杂凑算法标准。该算法由王小云等设计,消息分组 512 位,输出 256 位散列值,采用 Merkle-Damgard 结构。SM3 算法的压缩函数与 SHA-256 的压缩函数具有相似的结构,但 SM3 算法的压缩函数的结构和消息拓展过程的设计更加复杂,比如压缩函数的每一轮都使用 2 个消息字,消

息拓展过程的每一轮都使用 5 个消息字等。

1. SM3 算法中的常量与函数

初始值：

$$\mathrm{IV}=7380166\mathrm{f};\ 4914\ \mathrm{b2}\ \mathrm{b9};\ 172442\ \mathrm{d7};\ \mathrm{da8a0600};$$
$$\mathrm{a96f30bc};\ 163138\mathrm{aa};\ \mathrm{e38dee4d};\ \mathrm{b0fb0e4e}$$

常量：

$$T_j=\begin{cases}79\mathrm{cc}\ 4519, & 0\leqslant j\leqslant 15\\ 7\mathrm{a879}\ \mathrm{d8a}, & 16\leqslant j\leqslant 63\end{cases}$$

布尔函数：

$$\mathrm{FF}_j(X;Y;Z)=\begin{cases}X\oplus Y\oplus Z, & 0\leqslant j\leqslant 15\\ (X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z), & 16\leqslant j\leqslant 63\end{cases}$$

$$\mathrm{GG}_j(X;Y;Z)=\begin{cases}X\oplus Y\oplus Z, & 0\leqslant j\leqslant 15\\ (X\wedge Y)\vee(\neg X\wedge Z), & 16\leqslant j\leqslant 63\end{cases}$$

式中：X、Y、Z 为 32 位字。

置换函数：

$$P_0(X)=X\oplus(X<<<9)\oplus(X<<<17)$$
$$P_1(X)=X\oplus(X<<<15)\oplus(X<<<23)$$

式中：X 为 32 位的字。

2. SM3 算法描述

对于长度为 $l(l<2^{64})$ 位的消息 M，SM3 算法经过消息填充和迭代压缩，产生散列值，散列值的长度为 256 位。

1）消息填充

假定消息输入的长度为 $l(l<2^{64})$ 位。首先将比特“1”添加到消息的末尾；其次添加 k 个“0”，k 是满足 $l+1+k\equiv 448 \bmod 512$ 的最小的非负整数；然后添加一个 64 位比特串，该比特串是长度 l 的二进制表示。填充后的消息 M' 的比特长度为 512 的倍数。

例如：对消息 01100001 01100010 01100011，其长度 $l=24$，经填充得到的比特串如下：

$$01100001\ 01100010\ 011000111\ \overbrace{0\cdots 0}^{423\ \mathrm{b}}\overbrace{0\cdots 011000}^{64\ \mathrm{b}}$$
$$\underbrace{\qquad\qquad}_{l\text{的二进制表示}}$$

2）迭代压缩

迭代压缩是 SM3 算法的主体操作，此步骤产生最终散列值。迭代压缩过程如下：

（1）将消息填充后的消息 M' 按 512 位进行消息分组，$M'=B^{(0)},B^{(1)},\cdots,B^{(n-1)}$，其中，$n=(k+1+1+64)/512$。

（2）对 M' 按照如下过程迭代：

$$\mathrm{FOR}\ i=0\ \mathrm{to}\ n-1\ \mathrm{do}$$
$$V^{(i+1)}=\mathrm{CF}(V^{(i+1)};B^{(i+1)})$$

ENDFOR

其中，CF 是压缩函数；$V^{(0)}$ 为 256 位初始值 IV；$B^{(i)}$ 为填充后的消息分组；迭代压缩的结果为 $V^{(n)}$，同时也是消息 M 的散列值。

(3) 消息拓展

将消息分组 $B^{(i)}$ 按以下方法扩展生成 132 个字 $W_0, W_1, \cdots, W_{67}, W'_0, W'_1, \cdots, W'_{63}$ 用于压缩函数 CF：

将消息分组 $B^{(i)}$ 划分为 16 个字 $W_0, W_1, \cdots, W_{15}$；

FOR $j=16$ to 67 do

$W_j = P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} <<< 15)) \oplus (W_{j-13} <<< 7) \oplus W_{j-4}$

ENDFOR

FOR $j=0$ to 63 do

$$W'_j = W_j \oplus W_{j+4}$$

ENDFOR

(4) 压缩函数

令 A、B、C、D、E、F、G、H 为字寄存器，SS1、SS2、TT1、TT2 为中间变量；压缩函数 $V^{(i+1)} = \mathrm{CF}(V^{(i+1)}, B^{(i+1)})$，$0 \leqslant i \leqslant n-1$。

计算过程如下：

$$ABCDEFGH \leftarrow V^{(i)}$$

FOR $j=0$ to 63

$\mathrm{SS1} \leftarrow ((A <<< 12) + E + (T_j <<< j)) <<< 7$

$\mathrm{SS2} \leftarrow \mathrm{SS1} \oplus (A <<< 12)$

$\mathrm{TT1} \leftarrow \mathrm{FF}_j(A; B; C) + D + \mathrm{SS2} + W'_j$

$\mathrm{TT2} \leftarrow \mathrm{GG}_j(E; F; G) + H + \mathrm{SS1} + W_j$

$D \leftarrow C; C \leftarrow B <<< 9$

$B \leftarrow A; A \leftarrow \mathrm{TT1}$

$H \leftarrow G; G \leftarrow F <<< 19$

$F \leftarrow E; E \leftarrow P_0(\mathrm{TT2})$

ENDFOR

$$V^{(i+1)} \leftarrow ABCDEFGH \oplus V^{(i)}$$

其中字的存储为大端(big-endian)格式。

(5) 散列值

$$ABCDEFGH \leftarrow v^{(n)}$$

输出 256 位的散列值 $y = ABCDEFGH$。

5.1.7 Hash 攻击

随着 Hash 函数设计的发展，越来越多的学者或者密码工作者也在不断分析各种已出现的 Hash 函数。在近十多年，对 Hash 函数的分析也取得了许多成果。密码学中，对

密码算法的分析和攻击是两个等同的概念，以下对分析和攻击等同视之。

1996 年，Dbbertin 给出了对 MD4 算法的攻击，该攻击以 2^{-22} 的概率找到全部轮数的 MD4 算法的一个碰撞。同时，Dbbertin 还给出了如何找到有意义消息碰撞的方法。1998 年，Dbbertin 证明了 MD4 算法的前 2 轮不是单向的，这一结论表明，对于 MD4 算法而言，寻找原根和第二原根存在着有效的方法。Dbbertin 还以 2^{31} 的复杂度找到了 2 轮 RIPEMD 的碰撞。

1993 年，Boer 和 Bosselaers 提出了 MD5 算法的一种伪碰撞，即在不同的初始值下，同一消息产生的散列值相同。在 1996 年欧洲密码学会议上，Dbbertin 提出了 MDS 算法的另一种形式的伪碰撞，即两个不同初始值下的不同消息产生的散列值相同。1998 年，Chabaud 和 Joux 证明了用差分攻击的方法可以以 2^{-61} 的概率找到 SHA-0 的一个碰撞。

一个安全的 Hash 算法需要满足抗原像性、抗第二原像性和抗碰撞性。近几年，人们在评估 Hash 算法安全性时，不仅考虑这三个经典的安全要求，同时还会考虑其在近似碰撞、差分区分器、反弹攻击及飞去来器（Boomerang）区分器等攻击方面的抵抗性能，并认为，若给定 Hash 算法表现不同于期望的随机函数，则被视为该 Hash 函数的一个安全弱点。

（1）合法的签名方对于其认为合法的消息愿意使用自己的私钥对该消息生成的 m 位的散列值进行数字签名。

（2）攻击方为了伪造一份有（1）中的签名方签名的消息，首先产生一份签名方将会同意签名的消息，再产生出该消息的 22 种不同的变化，且每一种变化表达相同的意义（如在文字中加入空格、换行字符）。然后，攻击方伪造一条具有不同意义的新的消息，并产生出该伪造消息的 22 种变化。

（3）攻击方在上述两个消息集合中找出可以产生相同散列值的一对消息。根据“生日悖论”理论，能找到这样一对消息的概率是非常大的。如果找不到这样的消息，攻击方将再产生一条有效的消息和伪造的消息，并增加每组中的明文数目，直至成功为止。

（4）攻击方用第一组中找到的明文提供给签名方要求签名，这个签名就可以被用来伪造第二组中找到的明文的数字签名。这样，即使攻击方不知道签名私钥也能伪造签名。生日攻击表明，散列值的长度必须达到一定的值，过短，容易受到穷举攻击。例如，一个 40 位的散列值只需要穷举 100 万次。一般建议散列值需要 160 位，SHA-1 的最初选择是 128 位，后来改为 160 位，这就是为了防止利用生日攻击原理穷举散列值。

（5）模差分是一种精确差分，它不同于一般差分攻击使用的异或差分，能够准确地表达整数模减差分和异或差分这两种信息。利用模差分方法对 Hash 算法进行分析包括 4 个步骤：①选择合适的消息差分，它决定了攻击成功的概率；②针对选择的消息差分寻找可行的差分路线，这是模差分分析关键一步，也是最难的一步，它需要聪明的分析、熟练的技术、持久的耐心；③推导出保证差分路线可行的充分条件，在寻找差分路线的过程中确定链接变量的条件，一个可行的差分路线就意味着从路线上推导出来的所有链接变量的条件相互之间没有冲突；④使用消息修改技术，使被修改的消息满足尽可能多的充分条件。

5.2 消息认证

消息认证也称“报文认证”或“报文鉴别”，是一个证实收到的报文来自可信任的信息源且未被篡改的过程。消息认证也可用于证实报文的序列编号和及时性，因此利用消息认证方式可以避免以下现象的发生。

（1）伪造消息。攻击方伪造消息发送给目标端，却声称该消息源来自一个已授权的实体（如计算机或用户），或攻击方以接收方的名义伪造假的确认报文。

（2）内容篡改。以插入、删除、调换或修改等方式篡改消息。

（3）序号篡改。在 TCP 等依赖报文序列号的通信协议中，对通信双方的报文序号进行修改，包括插入、删除、重排序号等。这在目前的网络攻击事件中很常见。

（4）计时篡改。篡改报文的时间戳以达到报文延迟或重传的目的。

实现消息认证的手段可分为基于对称密码体制、基于非对称密码体制、利用散列函数和基于消息认证码四类。

5.2.1 基于对称密码体制

发送方 A 和接收方 B 事先共享一个密钥 k。A 用密钥 k 对消息 M 加密后通过公开信道传送给 B，B 接收到密文消息后，通过是否能用密钥 k 将其恢复成合法明文来判断消息是否来自 A，信息是否完整，如图 5-6 所示。

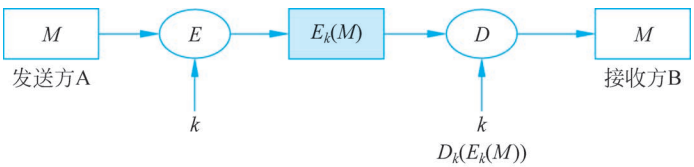

图 5-6 基于对称加密实现消息鉴别

这种方法局限于需要接收方有某种方法能判定解密出来的明文是否合法。因此，在处理中可以规定合法的明文只能是属于在可能位模式上有微小差异的一个小子集，这使得任何伪造密文解密恢复出来后能成为合法明文的概率非常小。

在实际中这是很容易实现的，可以假定明文是有意义的语句，而不是杂乱无章的字符串。例如，将一条有意义的明文加密后（无论使用什么算法加密），它都会以极大的概率变成一段杂乱无章的字符串，而几乎没有可能变成另一条有意义的语句。因此，如果发送方不知道密钥，用不正确的密钥 k' 对明文加密，接收方收到后用正确的密钥 k 对密文解密，就相当于对密文再加密了一次，这样得到的是两次加密后的密文，有极大的概率仍然会是一段杂乱的字符串。所以，当接收方解密后发现明文是有意义的语句，即使不知道明文是什么内容，也极大概率相信发送方是用正确的密钥加密的。

利用对称密码体制实现消息认证有如下三个特点。

（1）能提供认证。可确认消息只可能来自 A，传输途中未被更改。

（2）提供保密性。因为只有 A 和 B 知道密钥 k。

（3）不能提供数字签名。接收方可以伪造消息，发送方可以抵赖消息的发送。

可见，认证双方共享一个秘密就可以相互认证，这是最简单，也是最常用的认证机制。例如，现实生活中如果两人知道某个共同的秘密（并且只有他们知道），就能依靠这个秘密相互认证。该机制的原理很简单，实现时却要解决诸多问题，例如，如何让认证双方能够共享一个秘密，如何保证该秘密在传输过程中不会被他人窃取或利用等。

5.2.2 基于非对称密码体制

1. 提供消息认证

发送方 A 用自己的私钥 SK_A 对消息进行加密运算，再通过公开信道传送给接收方 B；接收方 B 用 A 的公钥 PK_A、对得到的消息进行解密运算并完成鉴别，如图 5-7 所示。

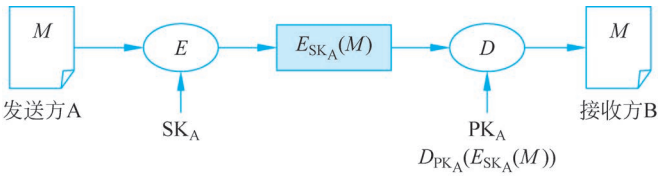

图 5-7 基于非对称加密实现消息鉴别

因为只有发送方 A 才能产生用公钥 PK_A，可解密的密文，所以消息一定来自拥有私钥 SK_A 的发送方 A。这种机制也要求明文具有某种内部结构使接收方能易于确定得到的明文是正确的。这种方法能提供认证和数字签名功能，但不能提供保密性，因为任何人都能用 A 的公钥解密查看消息。

2. 提供消息认证和机密性保护

发送方 A 用自己的私钥 SK_A 进行加密运算（数字签名）之后，再用接收方 B 的公钥 PK_B。进行加密，从而实现机密性。这种方法能提供机密性、数字签名和鉴别。其缺点是一次完整的通信需要执行公钥算法的加密、解密操作各两次，如图 5-8 所示。

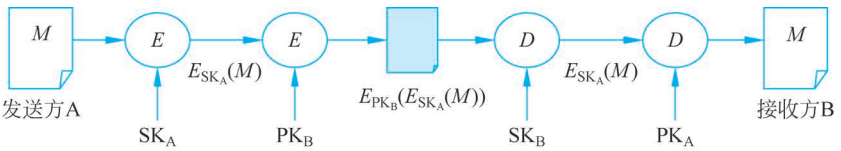

图 5-8 用公钥密码体制实现签名、加密和鉴别

通常情况下，都是先对消息进行签名再加密（因为被签名的消息应该能够理解）。若将消息加密之后再签名，则不符合常理（因为一般不会对一个看不懂的文件进行签名）。当然，上述原则也不是绝对的，有时候也需要先加密再传给别人签名，即盲签名。

5.2.3 基于 Hash 函数

消息认证中使用 Hash 函数的本质：发送方根据待发送的消息使用该函数计算一组散列值，然后将散列值和消息一起发送过去。接收方收到后对于消息执行同样的 Hash 计算，并将结果与收到的散列值进行对比，如果不匹配，则接收方推断出消息（也可能是散列值）遭受了篡改。

Hash 函数运算结果必须通过安全的方式传输。Hash 函数得到保护后，攻击方在篡改或替换消息的同时，不能轻易地修改散列值以蒙骗接收方。如果中间人拦截了传输消息时附上数据的散列值，并篡改或替换其中的数据，重新计算散列值并附在后面，接收方收到篡改后的数据块以及新的散列值，未能发现消息已经被篡改。

散列值能够通过不同的方法用于提供消息认证。图 5-9 展示了使用对称密码算法 E 加密值消息和散列值。

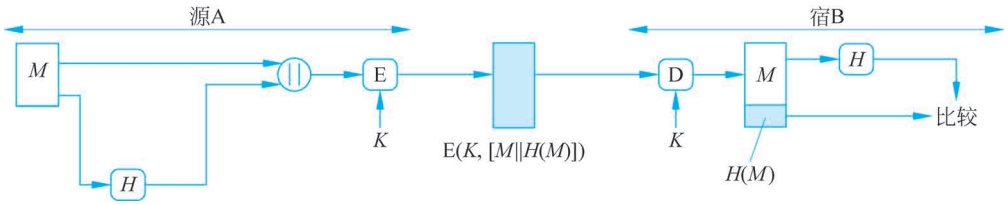

图 5-9 使用对称密码算法 E 加密消息和 Hash 码

因为只有 A 和 B 共享密钥 K，所以消息必然是发自 A 处，并且未被更改过。散列值提供了实现认证功能的结构，对整个消息以及散列值都进行加密，同时也提供了保密性，图 5-10 展示了使用对称密码算法 E 只对散列值进行加密。

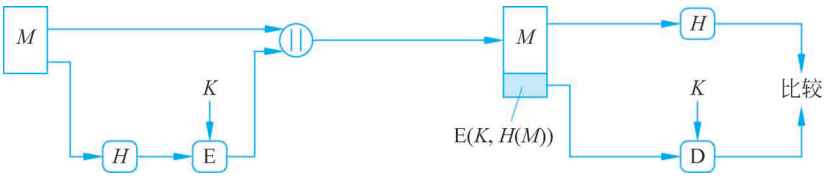

图 5-10 使用对称密码算法 E 只对散列值进行加密

对于无需保密性的应用，使用对称密码算法 E 只对散列值进行加密减轻了加解密操作的负担。图 5-11 展示了仅使用 Hash 函数实现消息认证。

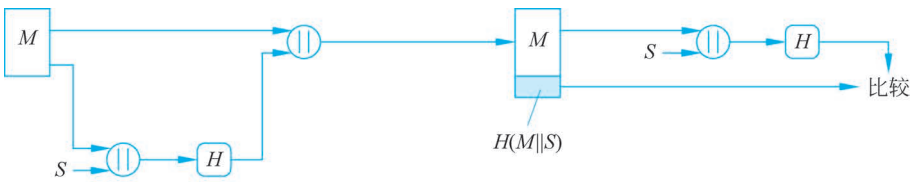

图 5-11 仅使用 Hash 函数实现消息认证

该方案假设通信双方共享相同的秘密值 S。发送方 A 将消息 M 和秘密值 S 串联后计算其散列值，并将得到的散列值附在消息 M 后发送。因为接收方 B 同时掌握 S，所以

能够重新计算该散列值进行验证。由于秘密值 S 本身没有在信道传送，攻击方不能对在信道上拦截的消息进行修改，也不能制作假消息。图 5-12 展示了通过将整个消息和散列值加密。

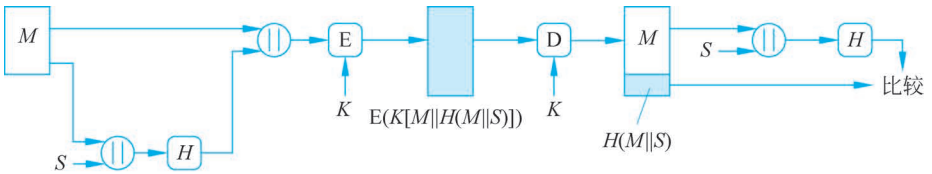

图 5-12　通过将整个消息和散列值加密

通过将整个消息和散列值加密，能够在仅使用 Hash 函数基础上提供保密性。

目前，人们越来越对不含加密函数消息认证的方法感兴趣，理由如下。

(1) 加密软件速度慢。即使每条消息需要加密的数据量不大，也总有消息串需要通过加密系统输入或输出。

(2) 加密硬件成本不容忽视。尽管已有实现 DES 的低成本芯片，但是若网络中所有节点都必须有该硬件，则总成本很大。

(3) 加密硬件的优化通常是针对大数据块的。对于小数据块，大比例的时间开销在初始化/调用上。加密算法受专利保护，这也会增加成本。

更一般地，消息认证是通过使用 MAC 实现的，即带密钥的 Hash 函数。通常情况下，通信双方基于共享的同一密钥来认证彼此交互的信息时，就会使用 MAC。MAC 函数将密钥和数据块作为输入，产生散列值作为 MAC 码，然后将 MAC 码和受保护的消息一起传递或存储。需要检查消息的完整性时，使用 MAC 函数对消息重新计算，并将计算结果与存储的 MAC 码对比。攻击方能对消息进行篡改，但在不知道密钥的情况下不能够计算出与篡改后的消息相匹配的 MAC 值。注意，这里验证方也知道发送方是谁，因为除了通信双发之外其他人不知道密钥。

总体来看，MAC 是 Hash 函数和加密函数操作的结合，即对于函数 $E(K, H(M))$，长度可变的消息 M 和密钥 K 是函数的输入，输出是固定长度的值。MAC 提供安全保护，用于抵抗不知道密钥的攻击方的攻击。在实现中，往往使用比加密算法效率更高的特殊设计的 MAC 算法。

5.2.4　基于消息认证码

消息认证码是用于提供数据原发认证和数据完整性保证的密码校验值。MAC 是消息被一个密钥控制的公开 Hash 函数作用后产生的、用作认证符的固定长度的数值，此时需要通信双方 A 和 B 共享一个密钥 k。它由如下形式的函数产生：

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

式中：M 为变长的消息；k 为收发双方共享的密钥；$H_k(\cdot)$为密钥 k 控制下的公开 Hash 函数。

MAC需要使用密钥 k ，这类似于加密，但其区别是MAC函数不可逆，因为它使用的是带密钥的Hash函数作为 $H_k(\cdot)$ 来实现MAC。另外，由于收发双方使用的是相同的密钥，因此单纯使用MAC是无法提供数字签名的。

对称加密和公钥加密都可以提供认证，为什么还要使用单独的MAC认证？其原因：一是机密性和真实性的概念不同，从根本上讲，信息加密提供的是机密性而非真实性，而且加密运算的代价很大，公钥算法的代价更大；二是证函数与加密函数的分离有利于提供功能上的灵活性，可以把加密和认证功能独立地实现在通信的不同传输层次；三是某些信息只需要真实性而不需要机密性，比如，广播的信息，信息量大，难以实现加密，政府的公告等信息只需要保证真实性。因此，在大多数场合MAC更适合用来专门提供认证功能。

1. 用MAC实现消息认证

如图5-13所示，设发送方A欲发送给接收方B的消息是 M，发送方A首先计算 $\mathrm{MAC}=H_k(M)$，然后向接收方B发送 $M'=M \parallel \mathrm{MAC}$，接收方B收到后做与发送方A相同的计算，求得一新MAC′，并与收到的MAC做比较，如果二者相等，由于只有发送方A和接收方B知道密钥 k，故可进行以下判断。

（1）确认消息的完整性，即该消息在传输过程中没有被篡改。因为攻击方篡改了该消息，也必须同时篡改对应的MAC值。在攻击方不知道密钥 k 的前提下，是无法对MAC值进行修改的。

（2）确认消息源的正确性，即接收方B可以确认该消息来自到发送方A。在无法获得密钥 k 的前提下，攻击方是无法生成正确的MAC值的，所以攻击方也无法冒充成消息的发送方A。

（3）确认序列号的正确性，即接收方B可以确认接收到的消息的顺序是正确的。

通过以上应用可以看出，MAC函数与加密函数类似，在生成MAC和验证MAC时需要共享密钥，但加密算法必须是可逆的，而MAC函数不需要。

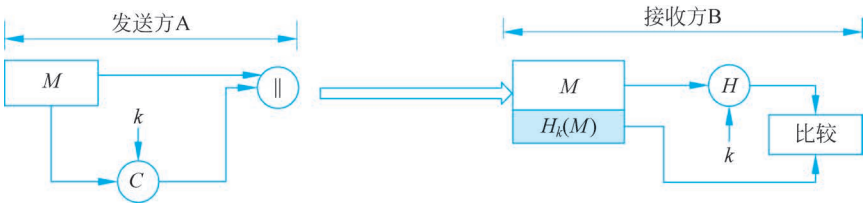

图5-13 消息认证码使用

2. 提供消息鉴别与机密性

在实际应用中，MAC可以与加密算法一起提供消息认证和保密性，在生成MAC之前或之后使用加密机制，获得机密性。这两种方法生成的MAC基于明文或密文，因此相应的鉴别与明文或密文有关，如图5-14和图5-15所示。一般来说，基于明文生成MAC的方法在实际应用中会更方便一些。

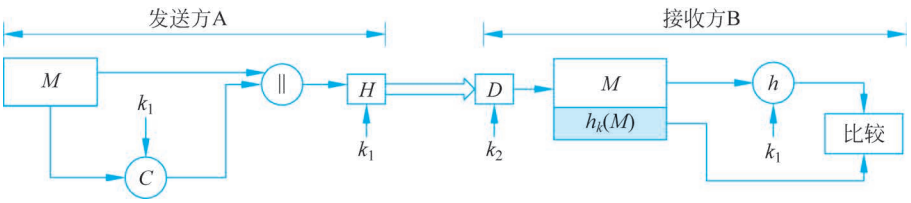

图 5-14　提供消息鉴别与机密性(与明文相关)

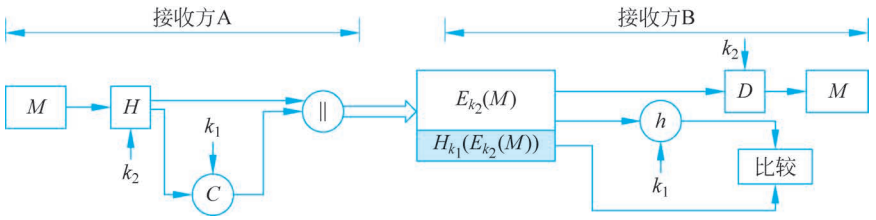

图 5-15　提供消息鉴别与机密性(与密文相关)

5.3 数字签名

在 ISO/IEC 7498-2《基本参考模型-安全体系结构》中,数字签名定义为"附加在数据单元上的一些数据,或是对数据单元所做的密码变换。这种数据和变换允许数据单元的接收方用以确认数据单元来源和数据单元的完整性,并保护数据,防止被人(如接收方)进行伪造"。

数字签名是实现安全认证的重要工具和手段,它能够提供身份认证、数据完整性、不可抵赖等安全服务。

(1) 防冒充(伪造)。其他人不能伪造对消息的签名,因为私有密钥只有签名者自己知道和拥有,所以其他人不可能构造出正确的签名数据。

(2) 可鉴别身份。接收方使用发送方的公开密钥对签名报文进行解密运算,并证明对方身份是真实的。

(3) 防篡改。即防止破坏信息的完整性。签名数据和原有文件经过加密处理已形成了一个密文数据,不可能被篡改,从而保证了数据的完整性。

(4) 防抵赖。数字签名可以鉴别身份,不可能冒充伪造。

数字签名是附加在报文(数据或消息)上并随报文一起传送的一串代码,与传统的亲笔签名和印章一样,目的是让接收方相信报文的真实性,必要时还可以对真实性进行鉴别。现在已有多种数字签名的实现方法,采用较多的是技术非常成熟的数据加密技术,既可以采用对称加密也可以采用非对称加密,非对称加密比对称加密更容易实现和管理。

数字签名用来保证信息传输过程中完整性、提供信息发送方的身份认证和不可抵赖性。使用公开密钥算法是实现数字签名的主要技术。

鉴别文件或书信真伪的传统做法是亲笔签名或盖章。签名起到认证、核准、生效的作用。网络系统中要求对电子文档进行辨认和验证,因而产生数字签名。数字签名主要

是保证信息完整性和提供信息发送方的身份认证。数字签名与传统签名的区别如下。

(1) 需要将签名与消息绑定在一起。

(2) 通常任何人都可验证。

(3) 需要考虑防止签名被复制、重用。

信息发送方使用公钥密码算法技术产生他人无法伪造的一段数字串。发送者用自己的私有密钥加密数据传给接收方,接收方用发送者的公钥解密后,就可确定消息来自谁,同时是对发送方发送的信息的真实性的一个证明。此时,发送方也无法对所发信息抵赖。

数字签名技术由公钥密码发展而来,它在身份认证、数据完整性、不可否认性和匿名性等安全方面发挥着非常重要的作用,目前已成为数字化社会的重要安全保障之一。

经典的传统数字签名算法包括以下几种。

(1) RSA 数字签名算法:该算法是目前计算机密码学中经典的算法之一,也是截至目前使用最广泛的数字签名算法,在信息安全和认证领域都发挥了很大的作用。值得注意的是,RSA 数字签名算法的密钥和 RSA 加密算法的密钥实现方式一样,因此统称为 RSA 算法。该算法的安全性依赖数论中的大数分解困难问题,即两个大素数相乘非常容易得到一个大整数,但是将一个大整数分解为两个大素数非常困难。

(2) Elgamal 数字签名算法:1985 年,斯坦福大学的 Tather Elgamal 利用 Elgamal 公钥密码体制提出了 Elgamal 数字签名算法,是经典的数字签名算法之一,它的安全性依赖计算有限域上的离散对数困难问题。目前很多的数字签名算法就是根据该算法扩展或者改进而来的,实用性较高。

(3) Schnorr 数字签名算法:1989 年,C. Schnorr 在 Elgamal 数字签名算法的基础上提出了 Schnorr 数字签名算法,其安全性也基于计算有限域上的离散对数困难问题。

(4) 数字签名算法:1991 年,NIST 提出了数字签名算法,该算法是 Elgamal 算法的变种,其安全性也依赖求解离散对数的困难性。1994 年 5 月,NIST 提出的数字签名标准 DSS 采用的就是数字签名算法 DSA。

(5) 椭圆曲线数字签名算法(Elliptic Curve Digital Signature Algorithm,ECDSA):1992 年,Scott 和 VLstrne 首次提出了 ECDSA,该算法是 ECC 椭圆曲线密码和 DSA 签名算法的结合,具有密钥存储空间小、安全性高的特点。1999 年,ECDSA 成为 ANSI 的标准,并于 2000 年成为 IEEE 和 NIST 的标准。目前,比特币一般利用 ECDSA 生成交易用户的密钥对,并对交易中的数据信息的消息摘要进行签名,利用交易账户的私钥进行签名认证。

(6) 盲签名算法:盲签名于 1982 年首次被 Chaum 提出,主要用于需要匿名的电子投票或者电子支付系统中。该签名方案保证了签名方案的匿名性和不可追踪性。签名者只能够进行签名操作,但是无法知道被签名消息的具体内容,消息－签名被公开之后,签名者也无法获得消息和签名过程间的关系。1992 年,Okamoto 提出了一种基于大数分解和离散对数的盲签名方案。基于 Schnorrl5 和 Guillou Quisquater 的协议,Pointcheval 和 Stern 提出了一种可证明安全的盲签名方案。2009 年,Overbeck 提出了首个基于编码的盲签名方案。

(7) 群签名算法：1991 年，Chaum 等首次提出了群签名方案。群签名的匿名性表现为任何一个群成员都能够代表所属的群，对消息进行匿名的签名操作；验证者只能够确定群中的某一个群成员生成了签名，但是无法确定是某个群成员进行了签名操作。群签名的可追踪性表现为在发生争议时群管理员可以通过签名确定具体执行签名的签名者，签名者无法否认自身产生的签名。除了群管理员，所有人无法确定不同的群签名是否由相同的群成员产生。

(8) 环签名算法：环签名最初由 Rivest 和 Tau-man 根据群签名而提出的，所以具有群签名的一些特性。环签名方案允许签名者在一组成员中保持匿名；环签名中不需要群管理者，没有群成员预设机制，没有更改和删除群的机制。签名者直接指定任意环，然后在不经过其他的成员许可或协助的情况下进行签名操作。如果要生成有效的环签名，签名者需要知道其私钥和其他成员的公钥。环签名具有匿名性和不可伪造性。

5.3.1 利用对称加密方式实现数字签名

对称加密在通信过程中密钥交换比较困难。在对称加密中，由于加密密钥和解密密钥是相同的，若将其用于数字签名，则要求消息的发送方和接收方都要使用相同的密钥，发送方用密钥对消息进行加密处理(签名)生成密文，接收方对接收到的密文利用同一个密钥进行解密。在这一过程中违反了数字签名的原则防抵赖。由于在加密(签名)和解密(鉴别身份)过程中参与者只有消息的发送方和接收方而没有第三方，一旦出现签名的抵赖，就无法进行判别。

为解决这一问题，在利用对称加密方式实现数字签名的过程中需要一个大家共同依赖的权威机构作为第三方。数字签名的用户都要向该权威机构申请一个密钥，这个密钥在该系统中是唯一的，即唯一标识了某一个用户。当权威机构向用户分配了密钥后，将该密钥的副本保存在该机构的数据库中，用以识别用户的真实性。

假设用户 A 和用户 B 之间要实现数字签名，具体过程如图 5-16 所示。

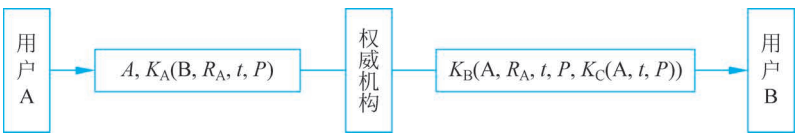

图 5-16 利用对称加密方式实现数字签名过程

(1) 用户 A 对要发送的明文消息 P 进行签名处理，生成 $K_A(B, R_A, t, P)$。其中，K_A 是用户 A 的加密密钥，即从权威机构申请到的密钥；B 是用户 B 的标识，在网络上是公开的；R_A 是用户 A 选择的一个随机数，以防止用户 B 收到重复的签名消息；t 是一个时间戳，用于保证该消息是最新的；P 是用户 A 要发送的明文消息。

(2) 用户 A 将利用自己的密钥加密生成的签名消息 $K_A(B, R_A, t, P)$ 发送出去，当权威机构接收到该消息后，通过数据库中用户 A 的密钥副本 K_A 知道该消息是用户 A 发送的，所以将利用密钥副本 K_A 进行解密处理，得到用户 A 发送的明文 P，并根据用户的标识符知道该消息是发送给用户 B 的。

(3) 权威机构利用用户 B 的密钥副本 K_B 生成消息 $K_B(A,R_A,t,P,K_C(A,t,P))$。其中 $K_C(A,t,P)$是一条由权威机构经过签名的消息，一旦将来出现抵赖，就可以通过该消息来证明。

(4) 用户 B 在接收到消息 $K_B(A,R_A,t,P,K_C(A,t,P))$后，利用自己的密钥 K_B 解密得到用户 A 发送的明文。

在图 5-16 所示的数字签名方式中，系统的安全性主要决定于两方面：一是用户密钥的保存中的安全性；二是权威机构的可信赖性。

5.3.2 利用非对称加密方式实现数字签名

在利用对称加密实现的数字签名中，用户必须依赖第三方的权威机构，所以权威机构的可信任度是决定该方式能否正常使用的关键。非对称加密解决了这一问题。利用非对称加密方式实现数字签名，主要是基于在加密和解密过程中 $D(E(P))=P$ 和 $E(D(P))=P$ 两种方式的同时实现，其中 RSA 就具有此功能。

利用非对称加密方式实现数字签名过程如图 5-17 所示，首先发送方利用自己的私有密钥对消息进行加密(实现签名)，接着对经过签名的消息利用接收方的公开密钥再进行加密(保证消息传送的安全性)，经过双重加密后的消息(密文)通过网络传送到接收方。接收方在接收到密文后，首先利用接收方的私有密钥进行第一次解密(保证数据的完全性)，接着用发送方的公开密钥进行第二次解密(鉴别签名的真实性)，最后得到明文。

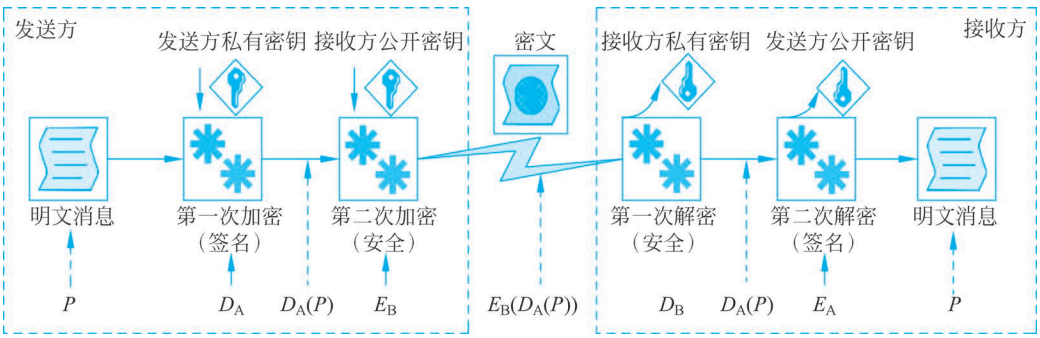

图 5-17 利用非对称加密方式实现数字签名过程

假设发送方否认自己给接收方发送过消息 P，接收方只需要同时提供 P 和 $D_A(P)$。第三方可对接收方提供的 $D_A(P)$利用 E_A 进行解密，即 $E_A(D_A(P))$。由于 $D_A(P)$是由发送方使用自己的私有密钥签名的，而 E_A 是发送方的公开密钥，第三方很容易得到且不需要发送方的许可。若 $E_A(D_A(P))=P$，则说明该消息是发送方发送的，因为只有发送方才有签名密钥 D_A。

5.3.3 SM2 算法

SM2 算法包括数字签名算法、密钥交换协议、公钥加密算法和系统参数四部分。

ECC 的系统参数是有限域上的椭圆曲线，包括：有限域 F_q 的规模 q；定义椭圆曲线 $E(F_q)$ 方程的两个元素 $a,b\in F_q$；$E(F_q)$ 上的基点 $G=(x_G,y_G)(G\neq O)$，其中 x_G 和 y_G 是 F_q 中的两个元素；G 的阶 n 及其他可选项（如 n 的余因子 h 等）。记 SM2 算法中使用的密码 Hash 算法为 $H_v(\cdot)$，其输出是位长恰为 v 的散列值，SM2 算法目前版本中 v 只取 256。SM2 算法的系统参数为 256 位素数域上的椭圆曲线。

数字签名算法由签名者对数据产生数字签名，并由验证者验证签名的可靠性。每个签名者有一个公钥和一个私钥，其中私钥用于产生签名，验证者用签名者的公钥验证签名。SM2 数字签名算法中，签名者用户 A 的密钥对包括其私钥 d_A 和公钥 $P_A=[d_A]G=(x_A,y_A)$，用户 A 具有位长为 entlen A 的可辨别标 2 字节数据，记为 $\mathrm{ENTL_A}$，签名者和验证者都需要用密码 Hash 算法求得用户 A 的散列值 $Z_A=H_{256}(\mathrm{ENTL_A}\|\mathrm{ID_A}\|a\|b\|x_G\|y_G\|x_A\|y_A)$。SM2 数字签名算法规定 H_{256} 为 SM3 密码 Hash 算法。

1. SM2 数字签名算法

设待签名的消息为 M，为了获取消息 M 的数字签名 (r,s)，签名者用户 A 应实现以下运算步骤。

(1) 置 $\overline{M}=Z_A\|M$。

(2) 计算 $e=H_v(\overline{M})$，将 e 的数据类型转换为整数。

(3) 用随机数发生器产生随机数 $k\in[1,n-1]$。

(4) 计算椭圆曲线点 $(x_1,y_1)=[k]G$，将 x_1 的数据类型转换为整数。

(5) 计算 $r=(e+x_1)\bmod n$，若 $r=0$ 或 $r+k=n$，则返回步骤(3)。

(6) 计算 $s=((1+d_A)^{-1}\cdot(k-r\cdot d_A))\bmod n$，若 $s=0$，则返回步骤(3)。

(7) 将 r、s 的数据类型转换为字节串，消息 M 的签名为 (r,s)。

SM2 数字签名算法如图 5-18 所示。

2. 数字签名的验证算法

为了检验收到的消息 M' 及其数字签名 (r',s')，验证者用户 B 应实现以下运算步骤。

(1) 检验 $r'\in[1,n-1]$ 是否成立，若不成立，则验证不通过。

(2) 检验 $s'\in[1,n-1]$ 是否成立，若不成立，则验证不通过。

(3) 置 $\overline{M}'=Z_A\|M'$。

(4) 计算 $e'=H_v(\overline{M}')$，将 e' 的数据类型转换为整数。

(5) 将 r'、s' 的数据类型转换为整数，计算 $t=(r'+s')\bmod n$，若 $t=0$，则验证不通过。

(6) 计算椭圆曲线点 $(x_1',y_1')=[s']G+[t]P_A$。

(7) 将 x_1' 的数据类型转换为整数，计算 $R=(e'+x_1')\bmod n$，检验 $R=r'$ 是否成立，若成立则验证通过，否则验证不通过。

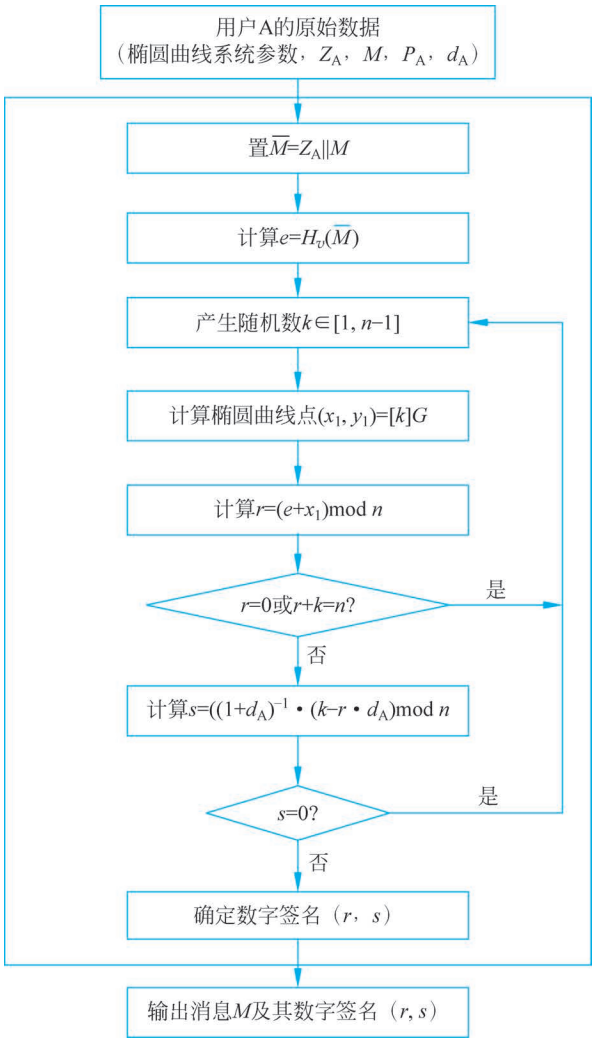

图 5-18　SM2 数字签名算法

数字签名的验证算法如图 5-19 所示。

5.3.4　数字签名标准

1991 年，NIST 发布了 FIPS PUB186《数字签名标准》(DSS)。DSS 采用了 SHA 散列算法，给出了一种新的数字签名方法即数字签名算法(DSA)。DSS 被提出后，1996 年又被稍做修改，2000 年发布了该标准的扩充版，即 FIPS 186-2。DSA 的安全性是建立在求解离散对数难题之上的，算法基于 ElGamal 和 Schnorr 签名算法，其后面发布的版本还包括基于 RSA 和椭圆曲线密码的数字签名算法。这里给出的算法是最初的 DSA 算法。

DSA 只提供数字签名功能的算法，虽然它是一种公钥密码机制，但是不能像 RSA 和 ECC 算法那样用于加密或密钥分配。

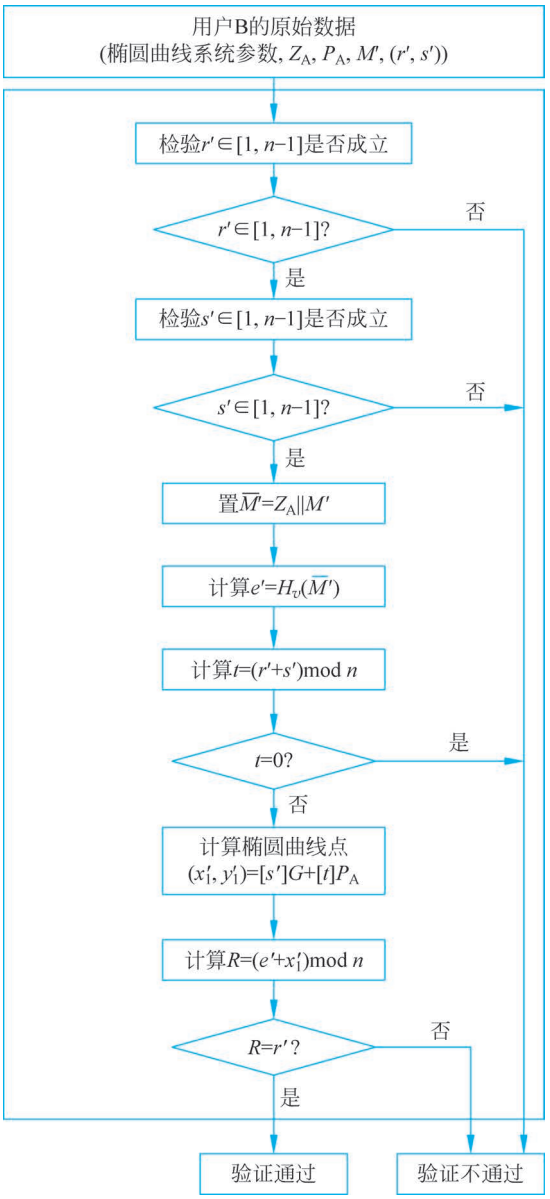

图 5-19　数字签名的验证算法

DSS 方法使用 Hash 函数产生消息的散列值和随机生成的 k 作为签名函数的输入。签名函数依赖发送方的私钥(PR_A)和一组参数，这些参数为一组通信伙伴所共有，可以认为这组参数构成全局公钥 PU_G。签名由两部分组成，标记为 r 和 s。

接收方对收到的消息计算散列值和收到的签名(r，s)一起作为验证函数的输入。验证函数依赖全局公钥和发送方公钥，若验证函数的输出等于签名中的 r，则签名合法。DSA 算法(图 5-20)如下。

(1) DSA 的系统参数选择。p 为 512 的素数，其中 $2^{L-1}<p<2^L$，$512 \leqslant L \leqslant 1024$，且

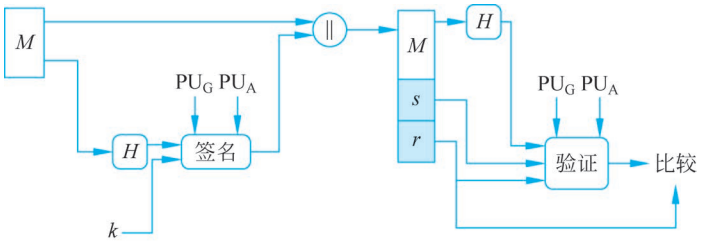

图 5-20 DSA 算法

L 是 64 的倍数，即 L 的位长在 512～1024 位之间并且其增量为 64 位。q 为 160 位的素数且 $q \mid p-1$。

g 满足 $g=h^{(p-1)/q} \bmod p$。H 为 Hash 函数。x 为用户的私钥，$0<x<q$。y 为用户的公钥，$y=g^x \bmod p$。p、q、g 为系统发布的公共参数，与公钥 y 公开；私钥 x 保密。

(2) 签名。设要签名的消息为 $M(0<M<p)$，签名者随机选择一整数 $k(0<k<q)$，并计算

$$r=(g^k \bmod p) \bmod q$$

$$s=[k^{-1}(H(M)+xr)] \bmod q$$

(r,s)即为 M 的签名。签名者将 M 和(r,s)一起存放或发送给验证者。

(3)验证。验证者获得 M 和(r,s)，需要验证(r,s)是否是 M 的签名。首先检查 r 和 s 是否属于$[0,q]$，若不属于，则(r,s)不是签名值。否则，计算

$$w=s^{-1} \bmod q$$

$$u_1=(H(M)w) \bmod q$$

$$u_2=rw \bmod q$$

$$v=((g^{u1}y^{u2}) \bmod p) \bmod q$$

若 $v=r$，则所获得的(r,s)是 M 的合法签名。

在 DSA 中，签名者和验证者都需要进行一次模 q 的求逆运算，这个运算是比较耗时的。为免去签名者或验证者的求逆运算，Yen 和 Laih 提出了以下两种改进方法。

(1) DSA 改进方法一：

签名：$r=(g^k \bmod p) \bmod q s=(rk-H(M))x^{-1} \bmod q$

验证：$t=r^{-1} \bmod q v=(g^{h(M)t}y^{st} \bmod p) \bmod q$

判断 v 和 r 是否相等。

(2) DSA 改进方法二：

签名：$r=(g^k \bmod p) \bmod q$

$$s=(k(H(M)+xr)^{-1}) \bmod q$$

验证：$t=sH(M) \bmod q$

$$v=(g^t y^{sr} \bmod p) \bmod q$$

判断 v 和 r 是否相等。

在上述方法中有些可以预先计算。在改进方法一中，签名时会用到的 x^{-1}，若 x 不

是经常更换，则 x^{-1} 可以预先计算并保存以便多次使用，这样就可以省掉一次求逆运算。在改进方法二中，验证者无须计算逆元。即便对于初始 DSA，也可以采用预计算的方法提高效率：签名时所计算的 $g^k \bmod p$ 并不依赖消息，因此可以预先计算。用户还可以根据需要预先计算多个可用于签名的 r，以及相应的 r^{-1}，这样可以大大提高效率。

以上给出的签名方案是直接数字签名（或称普通数字签名），包括 RSA、Schnorr、DSA、ECC、Fiat-Shamir、Guillou-Quisquarter、Schnorr、Ong-Schnorr-Shamir 等。这类数字签名只涉及通信双方，即签名方使用自己的私钥对整个消息或者对于消息的散列值进行签名，验证者使用签名者的公钥进行验证。即便发生纠纷，也是根据密钥及签名值进行仲裁。该方案的有效性完全依赖签名方的私钥。若签名者的私钥丢失或者被攻击者获取，则有可能被他人伪造签名，这时产生纠纷后，仲裁者无法给出实时的判断。因此，在实际应用中除了普通数字签名外，还有些特殊的签名方案，更多的可以说是一种安全协议，如仲裁数字签名、盲签名、代理签名、多重签名、不可否认签名、公平盲签名、门限签名、具有消息恢复功能的签名等，它们与具体应用环境密切相关。

5.4 身份认证

身份认证技术在信息安全中处于非常重要的地位，是其他安全机制的基础，只有实现了有效的身份认证，才能保证访问控制、安全审计、入侵防范等安全机制的有效实施。怎样才可以确保这个以数字身份进行操作的操作者就是这个数字身份合法拥有者，也就是说保证操作者的物理身份与数字身份相对应，就成为一个很重要的问题。身份认证就是为了解决这个问题。总的说来，身份认证的任务可以概括为以下四方面。

（1）会话参与方身份的认证，保证参与者不是经过伪装的潜在威胁者。

（2）会话内容的完整性，保证会话内容在传输过程中不被篡改。

（3）会话的机密性，保证会话内容（明文）不会被潜在威胁者所窃听。

（4）会话抗抵赖性，保证在会话后双方无法抵赖自己所发出过的信息。

由上可以看出，建立可信网络的核心问题是参与实体的身份认证问题，认证是建立信任的前提。在现实生活中每个人都有一个真实的物理身份，如居民身份证、户口本等。在计算机网络中如何保证以数字代码来标识用户身份时的真实性，如何通过技术手段保证用户的实体身份与数字身份一致，这便是身份认证要解决的问题。在实际应用中，验证用户的身份主要通过以下三种方式。

（1）根据用户所知道的信息来证明用户的身份。假设某些信息只有某个用户知道，如暗号、知识、密码等，通过询问这个信息就可以确认这一用户的身份。

（2）根据用户所拥有的东西来证明用户的身份。假设某一样东西只有某个用户拥有，如印章、身份证、护照、信用卡等，通过出示这些东西也可以确认用户的身份。

（3）直接根据用户独一无二的体态特征来证明用户的身份，如人的指纹、笔迹、DNA、视网膜及身体的特殊标志等。

认证（Authentication）是解决确定某个用户或其他实体是否被允许访问特定的系统或资源的问题。在网络中，任何用户或实体在进行任何操作之前必须要有相应的方法来

识别用户或实体的真实身份。为此,认证又称为鉴别或确认。身份认证主要鉴别或确认访问者的身份是否属实,以防止攻击,保障网络安全。

授权(Authorization)是指当用户或实体的身份被确定为合法后,赋予该用户的系统访问或资源使用权限。只有通过认证的用户才允许访问系统资源,然而在许多情况下当一个用户通过认证后通常不可能赋予访问所有系统资源的权限。例如,在 Windows 操作系统中,通过认证的系统管理员账户(Administrator)可以对系统配置进行设置,而通过认证的临时账户(Guest)只能查看系统的一些基本信息。为此,必须根据用户身份的不同,给不同的用户授予不同的权限,限制通过认证的用户的行为。

审计(Accounting)也称为记账(Accounting)或审核,出于安全考虑,所有用户的行为都要留下记录,以便进行核查。采集的数据包括登录和注销的用户名、主机名及时间。安全要求较高的网络,审计数据应该包括任何人所有的试图通过身份认证和获得授权的尝试。另外,以匿名(Anonymous)或临时账户(Guest)身份对公共资源的访问情况也应该进行采集,以便进行安全性评估时使用。审计信息一般存放在日志文件中,是对系统安全性进行评估的基础。目前使用的网络入侵检测、网络入侵防御、网络安全动态感知等系统,工作依据主要来源于各系统和设备产生的日志信息。

5.4.1 基于密码的身份认证

密码认证也称为“口令认证”,它是计算机系统和网络系统中应用最早,也最为广泛的一种身份认证方式。密码是用户与计算机之间以及计算机与计算机之间共享的一个秘密,在通信过程中一方向另一方提交密码,表示自己知道该秘密,从而通过另一方的认证。密码通常由一组字符串组成,为便于用户记忆,用户使用的密码一般有长度的限制。

因为密码认证易于管理、操作简单,而且不需要额外的成本,用户只要记得账号与密码就可以进行系统资源的访问。所以,为了防止非法用户进入计算机系统,常用的方法是密码认证,以保护计算机或网络系统不被入侵者破坏。合法用户利用正确的密码可以登录计算机和网络系统,拒绝非法用户登录。

传统的密码认证的方式是先建立用户账户,再为每个用户账户分配一个密码。用户登录首先发送一个包含用户账户与密码的请求登录信息,主机系统根据储存在用户数据库中的用户账户与密码验证该账户及所对应的密码是否正确。如果正确,认证过程结束,允许用户登录;否则,拒绝用户登录。

随着计算机网络的迅速发展和应用系统的不断增加,大多数的用户不得不通过网络远程登录主机系统,用户和主机系统之间就必须在网络进行密码认证。而互联网上泛滥的网络监听工具(如 Sniffer 等)可以非常容易地监听到网络上传递的各类明文信息,包括 FTP、Telnet、POP3 等网络服务的账户及密码。如果在网络上传送的账户和密码没有经过加密处理,就很容易被窃取,这是一种非常不安全的密码认证方式。

为了保证口令的私密性,使用了一种容易加密但很难解密的“单向散列”方法(Hash 算法)来处理口令。也就是说,操作系统本身并不知道用户输入的口令,只知道口令经过加密的形式,这使得口令不再以明文方式进行传输,攻击者获取“明文”口令只有采用暴

力方法在口令可能的区间内进行穷举。

1. 一次性口令认证

一次性口令(One Time Password,OTP)认证技术是一种比传统口令认证技术更加安全的身份认证技术,其基本思想是在登录过程中加入不确定因素,使每次登录时计算的密码都不相同,系统也做同样的运算来验证登录,以提高登录过程的安全性。

一次性口令认证是一种相对简单的身份认证机制,它可以简单快速地加载到需要认证的系统,而无须添加额外的硬件,也不需要存储密钥或口令等敏感信息,避免了遭受重放攻击的可能。不管是基于静态口令还是基于动态口令,一次性口令认证的原理仍然是基于"用户知道什么"来实现的。例如,静态密码是使用者和认证系统之间所共同知道的信息,但其他人或系统不知道,认证系统通过验证使用者提供的口令就能够判断是否是合法用户。动态口令也是这样,用户和认证系统之间必须遵循相同的通行短语的"通信暗语",对于外界来说这条通行短语是保密的。动态口令和静态口令不同的是,这个通行短语是不在网络上进行传输的,所以攻击者无法通过网络窃听方式获得通行短语。每条动态口令一般由三个参数按照一定单向散列算法进行计算所得,具体如下。

(1) 种子:认证服务器为用户分配的一个非密文的字符串,一个用户对应于一个种子,种子在认证系统中具有其唯一性。

(2) 序号:单向 Hash 函数的迭代次数,是系统通知用户将生成的本次口令在一次性序列中的顺序号。在动态口令认证中,迭代次数一般是不断变化的,即本次计算动态口令时单向 Hash 函数的迭代次数一般与上一次不同。迭代次数的作用就是让动态口令不断地发生变化。

(3) 通行短语:只有用户知道的值,这个值是保密的。

2. 动态口令认证

在动态口令认证中,通行短语和种子一般是不变的,序号是动态变化的;种子和序号是可以公开的,通行短语是保密的。

动态口令认证的实现方式较多,根据不确定因子选择方式可将动态口令的实现分为以下三种类型。

(1) 时间同步方式:利用用户的登录时间作为随机数,连同用户的通行短语一起生成一个口令。这种方式对客户端和认证服务器时间准确度的要求较高。该方式的优点是方便使用,管理容易。缺点是在分布式环境下对不同设备的时间同步难度较大,因为时间的改变可能造成密码输入的错误码。

(2) 事件同步方式:事件同步方式的基本原理是通过特定事件次序及相同的种子值作为输入,通过特定算法运算出相同的口令。事件动态口令是让用户的密码按照使用的次数不断动态地发生变化。每次用户登录时(当作一个事件),用户按下事件同步令牌上的按键产生一个口令(如银行的密码器),与此同时系统也根据登录事件产生一个口令,两者一致则通过验证。与时间同步的动态口令不同的是,事件同步不需要精准的时间同步,而是依靠登录事件保持与服务器的同步。因此,相比时间同步,事件同步适用于恶劣

的环境中。

(3) 挑战/应答方式：当客户端发出登录请求时，认证系统会生成一个挑战信息发送给客户端。客户端再使用某种单向 Hash 函数把这条消息连同自己的通行短语连起来生成一个口令，并将这个口令发送给认证系统。认证系统用同样的方法生成一个口令，然后通过比较验证用户身份。该方式的优点是不要考虑同步的问题，安全性较高，是目前身份认证中常采用的一种认证方式；缺点是使用者输入信息较多，操作比较复杂。

挑战/应答方式的工作原理是当客户端试图访问一个服务器主机时，在认证服务器收到客户端的登录请求后，将给客户端返回它生成的一个信息，用户在客户端输入只有自己知道的通行短语，并将其发送给认证服务器，并由动态口令计算器生成一个动态口令。这个动态口令再通过网络传送到认证服务器。认证服务器检测这个口令。如果这个动态口令和认证服务器上生成的动态口令相同，则认证成功，使用者被认证服务器授权访问，同时这个动态口令将不能再次使用。由于客户端用户所输入通行短语生成的动态口令在网上传输，而使用者的通行短语本身不在网上传输，也不保存客户端和服务器端中的任何地方，只有使用者本人知道，所以这个通行短语不会被窃取，即使此动态口令在网络传输过程中被窃取，也无法再次被使用，避免了重放攻击的发生。

挑战/应答机制的实现过程(图 5-21)如下。

① 客户向认证服务器发出请求，要求进行身份认证。

② 认证服务器从用户数据库中查询用户是否是合法的用户，如果不是，则不做进一步处理。

③ 认证服务器内部产生一个随机数，作为“提问”(挑战)，发送给客户。

④ 客户端将自己的密钥(通行短语)和随机数合并，使用单向 Hash 函数(如 MD5 算法)运算得到一个结果作为认证证据传给服务器(应答)。

⑤ 认证服务器使用该随机数与存储在服务器数据库中的该客户密钥(通行短语)进行相同的单向 Hash 运算，若运算结果与客户端传回的响应结果相同，则认为客户端是合法用户。

⑥ 认证服务器通知客户认证成功或失败。

图 5-21 挑战/应答机制的实现过程

5.4.2 基于生物特征的身份认证

生物特征认证又称“生物特征识别”，是指通过计算机利用人体固有的物理特征或行

为特征鉴别个人身份。在信息安全领域，推动基于生物特征认证的主要动力来自基于密码认证的不安全性，即利用生物特征认证来替代密码认证。

人的生理特征与生俱来，一般是先天性的。行为特征则是习惯养成，多为后天形成。生理和行为特征统称为生物特征。常用的生物特征包括脸像、虹膜、指纹、声音、笔迹等。同时，随着现代生物技术的发展，尤其对人类基因研究的重大突破，研究人员认为 DNA 识别技术将是未来生物识别技术的又一个发展方向。满足以下条件的生物特征才可以用来作为进行身份认证的依据。

（1）普遍性：每人都应该具有这一特征。

（2）唯一性：每人在这一特征上有不同的表现。

（3）稳定性：这一特征不会随着年龄和生活环境而改变。

（4）易采集性：这一特征应该便于采集和保存。

（5）可接受性：人们是否能够接受这种生物识别方式。

1. 指纹识别

指纹识别技术也称“指纹认证技术”。1858 年，印度的 William Hershel 爵士就使用指纹和掌纹作为合同签名的一种形式。目前，在全球范围内都建立了指纹鉴定机构及罪犯指纹数据库，我国早在 20 世纪 80 年代的重点人口管理中就开始采集具有犯罪前科的重点人口的指纹，并相继建立了全国范围内联网的指纹比对数据库。早期的指纹认证主要用于司法鉴定，现在已广泛应用于门禁系统、考勤、部分笔记本电脑和移动存储设备，认证技术也在不断成熟，应用范围也在不断拓宽。

指纹识别的特点如下。

（1）独特性：指纹具有高度的不可重复性。

（2）稳定性：指纹脊的样式终端不变。指纹不会随着人的年龄、健康程序的变化而发生变化。

（3）方便性：目前已建有标准化的指纹样本库，以方便指纹认证系统的开发；同时，在指纹识别系统中用于指纹采集的硬件设备也较容易实现。

指纹识别包括指纹注册过程和指纹比对过程（图 5-22），具体如下。

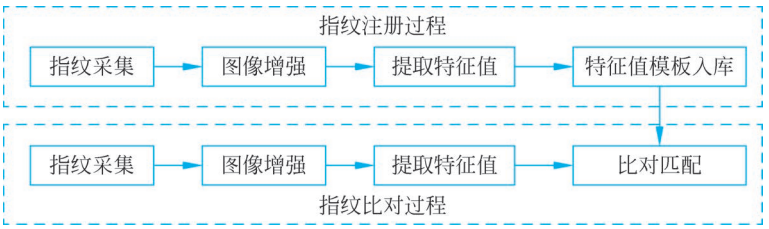

图 5-22　指纹识别系统

（1）指纹采集。通过指纹传感器获取人的指纹图像数据，其本质是指纹成像。指纹采集大都通过各种采集仪，可分为光学和 COMS 两类，光学采集仪采集图像失真小但成本较高，而 COMS 采集仪成本低但图像质量较差。

（2）图像增强。根据某种算法，对采集到的指纹图案进行效果增强，以利于后续对指

纹特征值的提取。

(3) 提取特征值。提取特征值对指纹图案上的特征信息进行选择、编码和形成二进制数据的过程。

(4) 特征值模板入库。特征值模板入库是根据指纹算法的数据结构,即特征值模板,对提取的指纹特征值进行结构化并保存。

(5) 比对匹配。比对匹配是指把当前取得的指纹特征值集合与已存储的指纹特征值模板进行匹配的过程。

2. 虹膜识别

作为生物特征认证的依据,指纹的应用已经比较广泛,然而指纹识别易受脱皮、出汗、干燥等外界条件的影响,并且这种接触式的识别方法要求用户直接接触公用的传感器,给使用者带来了不便。为此,非接触式的生物特征认证将成为身份认证发展的必然趋势。与脸像、声音等其他非接触式的身份鉴别方法相比,虹膜以更高的准确性、可采集性和不可伪造性,成为目前身份认证研究和应用的热点。基于虹膜的身份认证要求对被认证者的虹膜特征进行现场实时采集,用户在使用虹膜进行身份认证时无须输入 ID 号等标志信息。

虹膜认证是基于生物特征的认证方式中最好的一种。虹膜(眼睛中的彩色部分)是眼球中包围瞳孔的部分,上面布满极其复杂的锯齿网络状花纹,而每个人虹膜的花纹都是不同的。虹膜识别技术就是应用计算机对虹膜花纹特征进行量化数据分析,用以确认被识别者的真实身份。

每人的虹膜具有随机的细节特征和纹理图像,这些特征在人的一生中保持相对的稳定性,不易改变。据统计,到目前为止,虹膜认证的错误率在所有的生物特征识别中是最低的(相同纹理的虹膜出现的概率是 10^{-46})。所以虹膜识别技术在国际上得到广泛关注,有很好的应用前景。一个虹膜识别系统一般由以下四部分组成(图 5-23)。

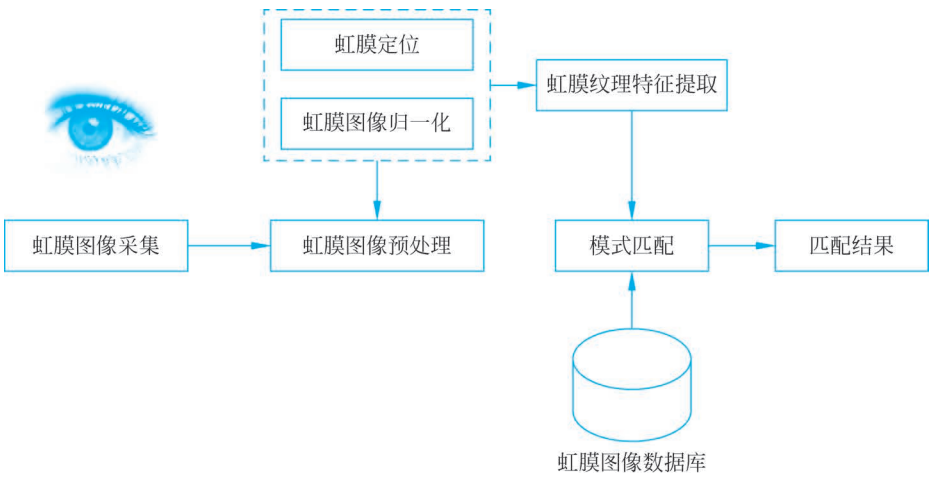

图 5-23 虹膜识别系统

（1）虹膜图像采集。虹膜图像采集是虹膜识别系统一个重要且困难的步骤。因为虹膜尺寸比较小且颜色较暗，所以用普通的照相机来获取质量好的虹膜图像是比较困难的，必须使用专门的采集设备。

（2）虹膜图像预处理。这一操作分为虹膜定位和虹膜图像归一化两个步骤。虹膜定位就是要找出瞳孔与虹膜之间（内边界）、虹膜与巩膜之间（外边界）的两个边界，再通过相关的算法对获得的虹膜图像进行边缘检测。虹膜图像归一化是由于光照强度及虹膜震颤的变化，瞳孔的大小会发生变化，而且在虹膜纹理中发生的弹性变形也会影响虹膜模式匹配。因此，为了实现精确的匹配，必须对定位后的虹膜图像进行归一化，补偿大小和瞳孔缩放引起的变异。

（3）虹膜纹理特征提取。采用转换算法将虹膜的可视特征转换成为固定字节长度的虹膜代码。

（4）模式匹配。识别系统将生成的代码与代码数据库中的虹膜代码进行逐一比较，当相似率超过某一个阀值时，系统判定检测者的身份与某一个样本相符；否则，系统将认为检测者的身份与该样本不相符，进入下一轮的比较。

3. 人脸识别

人脸识别技术是通过计算机提取人脸的特征，并根据这些特征进行身份验证的一种技术。人脸与人体的其他生物特征（如指纹和虹膜等）一样与生俱来，它们所具有的唯一性和不易被复制的良好特性为身份鉴别提供了必要的前提。同其他生物特征识别技术相比，人脸识别技术具有操作简单、结果直观、隐蔽性好的优越性。

人脸识别技术是基于人的脸部特征，对输入的人脸图像或者视频流首先判断其是否存在人脸，如果存在，则进一步给出每个脸的位置、大小和各个主要面部器官的位置信息，并依据这些信息进一步提取每个人脸中所蕴含的身份特征，并将其与存放在数据库中的已知的人脸信息进行对比，从而识别每个人脸的身份。

人脸识别技术从最初对背景单一的正面灰度图像的识别，经过对多姿态（正面、侧面等）人脸的识别研究，发展到能够动态实现人脸识别，目前正在朝三维人脸识别的方向发展。在此过程中，人脸识别技术涉及的图像逐渐复杂，识别效果不断地得到提高。人脸识别技术融合了数字图像处理、计算机图形学、模式识别、计算机视觉、人工神经网络和生物特征技术等多个学科的理论和方法。另外，人脸自身及所处环境的复杂性，如表情、姿态、图像的环境光照强度等条件的变化以及人脸上的遮挡物（眼镜、胡须）等，都会使人脸识别方法的正确性受到很大的影响。

从人脸识别的过程来看，可以将人脸识别过程分为以下四个部分。

（1）人脸图像采集及检测。人图像采集及检测包括人脸图像采集和人脸检测两个过程。人脸图像采集是指通过摄像镜头采集人脸的图像，包含静态图像、动态图像、不同的位置以及不同表情等。被采集者进入采集设备的拍摄范围内时，采集设备会自动搜索并拍摄被采集者的人脸图像。人脸检测主要用于人脸识别的预处理，即在图像中准确标定出人脸的位置和大小。人脸图像中包含的模式特征十分丰富，如直方图特征、颜色特征、模板特征、结构特征等，人脸检测就是把其中有用的信息挑选出来，并利用这些特征实现

人脸检测。

(2) 人脸图像预处理。对于人脸的图像预处理是基于人脸检测结果,对图像进行处理并最终服务于特征提取的过程。系统获取的原始图像由于受到各种条件的限制和随机干扰,往往不能直接使用,必须在图像处理的早期阶段对它进行灰度校正、噪声过滤等图像预处理。对于人脸图像而言,其预处理过程主要包括人脸图像的光线补偿、灰度变换、直方图均衡化、归一化、几何校正、滤波以及锐化等。

(3) 人脸图像特征提取。人脸识别系统可使用的特征通常分为视觉特征、像素统计特征、人脸图像变换系数特征、人脸图像代数特征等。人脸特征提取就是针对人脸的某些特征进行的。人脸特征提取也称人脸表征,它是对人脸进行特征建模的过程。人脸特征提取的方法归纳起来分为两大类:一是基于知识的表征方法;二是基于代数特征或统计学习的表征方法。基于知识的表征方法主要是根据人脸器官的形状描述以及他们之间的距离特性来获得有助于人脸分类的特征数据,其特征分量通常包括特征点间的欧几里得距离、曲率和角度等。人脸由眼睛、鼻子、嘴、下巴等局部构成,对这些局部和它们之间结构关系的几何描述可作为识别人脸的重要特征,这些特征称为几何特征。基于知识的人脸表征主要包括基于几何特征的方法和模板匹配法。

(4) 人脸图像匹配与识别。提取的人脸图像的特征数据与数据库中存储的特征模板进行搜索匹配,通过设定一个阈值,当相似度超过这一阈值时,把匹配得到的结果输出。人脸识别就是将待识别的人脸特征与已得到的人脸特征模板比较,根据相似程度对人脸的身份信息进行判断。这一过程又分为两类:一类是确认,一对一进行图像比较的过程;另一类是辨认,一对多进行图像匹配对比的过程。

5.4.3 基于零知识证明的身份认证

一般身份认证过程中,验证者在收到证明者提供的认证账户和密码(或生物特征信息)后,在数据库中进行核对,如果在验证者的数据库中找到了证明者提供的账户和密码(或完全匹配的生物特征信息),那么该认证通过;否则,认证失败。在这一认证过程中,验证者必须事先知道证明者的账户和密码(或生物特征信息),这显然会带来不安全因素。那么,能否实现在验证者不需要知道证明者任何信息(包括用户账户和密码)的情况下就能够完成对证明者的身份认证?零知识证明身份认证就可以实现这一功能。

零知识证明是在 20 世纪 80 年代初出现的一种身份认证技术。零知识证明是指证明者能够在不向验证者提供任何有用信息的情况下,使验证者相信某个论断是正确的。

零知识证明实质上是一种涉及两方或多方的协议,即两方或多方完成一项任务所需采取的一系列步骤。证明者向验证者证明并使验证者相信自己知道某一消息或拥有某一物品,但证明过程不需要(也不能够)向验证者泄露。零知识证明分为交互式零知识证明和非交互式零知识证明两种类型。下面给出一个零知识证明的例子。

用户 A 要向用户 B 证明自己是某一间房子的主人,即 A 要向 B 证明自己能够正常进入该房间。假设该房间只能用钥匙打开锁后进入,而其他任何方法都打不开。这时有两种办法:一种方式是 A 把钥匙交给 B,B 拿着这把钥匙打开该房间的锁,如果 B 能够打

开则证明 A 是该房间的主人；另一种方式是 B 确定该房间内有一个物品，只要 A 用自己的钥匙打开该房间后，将该物品拿出来出示给 B，从而证明 A 是该房间的主人。后一种方式属于零知识证明。虽然两种方式都证明了用户 A 是该房子的主人，但是在后一种方式中用户 B 始终没有得到用户 A 的任何信息（包括没有拿到用户 A 的钥匙），从而可以避免用户 B 的信息（钥匙）被泄露。

用户 A 拥有用户 B 的公钥，用户 B 需要向 A 证明自己的身份是真实的，同样有两种证明的方法：一种方式是用户 B 把自己的私钥交给 A，A 用这个私钥对某个数据进行加密操作，然后将加密后的密文用 B 的公钥来解密，如果能够成功解密，则证明用户 B 的身份是真实的；另一种方式是用户 A 给出一个随机值，B 用自己的私钥对该随机值进行加密操作，然后把加密后的数据交给 A，A 用 B 的公钥进行解密操作，如果能够得到原来的随机值，则证明用户 B 的身份是真实的。后一种方式属于零知识证明，在整个过程中 B 没有向 A 提供自己的私钥。

零知识证明验证者 B 选择随机数，证明者 A 根据随机数出示不同的证明。一方面 A 不能欺骗 B，因为 A 的随机数要求使用其私钥运算；另一方面 B 不能伪装 A 来欺骗 C，因为随机数将由 C 选择，B 不能重发 A 与 A 之间的验证信息。证明者欺骗验证者的概率随着随机数选择的位数和验证次数的增多而减少。

零知识证明协议可定义为证明者和验证者之间进行交互时使用的一组规则。交互式零知识证明是由这样一组协议确定的：在零知识证明过程结束后，证明者只告诉验证者关于某一个断言成立的信息，而验证者不能从交互式证明协议中获得其他任何信息。即使在协议中使用欺骗手段，验证者也不可能揭露其信息。这一概念其实就是零知识证明的定义。

如果一个交互式证明协议满足以下三点，就称此协议为一个零知识交互式证明协议。

（1）完备性。如果证明者的声称是真的，则验证者以绝对优势的概率接受证明者的结论。

（2）有效性。如果证明者的声称是假的，则验证者也以绝对优势的概率拒绝证明者的结论。

（3）零知识性。无论验证者采取任何手段，当证明者的声称是真的，且证明者不违背协议时，验证者除了接受证明者的结论以外，得不到其他额外的信息。

在交互式零知识证明过程中，证明者和验证者之间必须进行交互。20 世纪 80 年代末，出现了“非交互式零知识证明”的概念。在非交互式零知识证明过程中，通信双方不需要进行任何交互，从而任何人都可以对证明者公开的消息进行验证。

在非交互式零知识证明中，证明者公布一些不包括他本人任何信息的秘密消息，却能够让任何人相信这个秘密消息。在这一过程（其实是一组协议）中，起关键作用的因素是一个单向 Hash 函数。如果证明者要进行欺骗，他必须能够知道这个 Hash 函数的输出值。事实上，由于他不知道这个单向 Hash 函数的具体算法，所以无法实施欺骗。也就是说，这个单向 Hash 函数在协议中是验证者的代替者。

5.4.4 基于量子密码的身份认证

量子密码技术是密码学与量子力学相结合的产物，采用量子态作为信息载体，经由量子通道在合法用户之间传递密钥。

量子密码的安全性是由量子力学原理保证。“海森堡测不准原理”是量子力学的基本原理，是指在同一时刻以相同精度测定量子的位置与动量是不可能的，只能精确测定两者之一。“单量子不可复制定理”是“海森堡测不准原理”的推论，它是指在不知道量子状态的情况下复制单个量子是不可能的，因为要复制单个量子就只能先做测量，而测量必然改变量子的状态。量子密码技术可达到经典密码学所无法达到的两个最终目的：一是合法的通信双方可察觉潜在的窃听者并采取相应的措施；二是使窃听者无法破解量子密码，无论企图破解者有多么强大的计算能力。将量子密码技术的不可窃听性和不可复制性用于认证技术可以用来认证通信双方的身份，原则上提供了不可破译、不可窃听和大容量的保密通信体系，真正做到了通信的绝对安全。主要有三类量子身份认证的实现方案，分别为基于量子密钥的经典身份认证系统、基于经典密钥的量子身份认证系统和基于纯量子身份认证系统。

思维认证是一种全新的身份认证方式，它是以脑-机接口(Brain-Computer Interface，BCI)技术为基础，有望替代传统的身份认证方式。脑-机接口技术通过实时记录人脑的脑电波，在一定程度上解读人的思维信号。其原理是当受试主体的大脑产生某种动作意识或者受到外界刺激后，其神经系统的活动会发生相应改变，这种变化可以通过一定的手段检测出来，并作为意识发生的特征信号。试验表明，即使对于同一个外部刺激或者主体在思考同一个事件时，不同人的大脑所产生的认知脑电信号是不同的。也就是说，这些思维的信号携带有主体的独一无二的特性，因此人们可以通过探测被试者的脑部的响应变化来进行身份认证。

典型的基于 BCI 的思维认证系统的结构如图 5-24 所示。

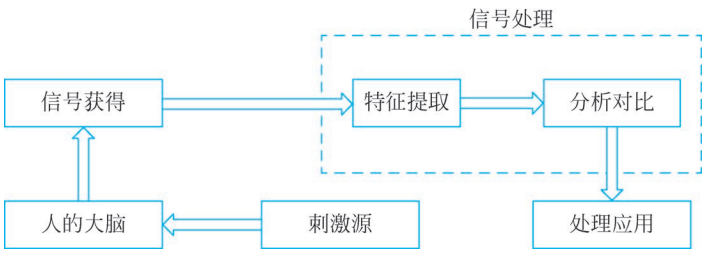

图 5-24 基于 BCI 的思维认证系统的结构

社会工程因素是整个安全链条中最薄弱的一环，思维认证方法使得对社会工程的攻击变得无效，即使非法者通过偷窥，骗取信任的方式获得了系统的口令，也不能够模拟合法用户的特征思维信号，这种独一无二的特性能够有效地抵御各种攻击和入侵，因此无法通过验证系统。

行为认证技术有别于传统的认证技术，它是以用户的行为为依据的认证技术。行为认证技术的基本思想：对于一个固定的用户，其行为总是遵循一定的习惯，表现为在行动操作

中存在规律性，行为认证技术正是基于对用户的行为习惯来判断用户的身份是否假冒。

行为认证技术要求跟踪记录每个用户的历史行为习惯，并按照一定的算法从中抽取出规律，建立用户行为模型，当用户的行为习惯突然改变时，与行为模型库中不匹配，从而这种异常就会被检查出来。典型的行为认证系统的结构如图 5-25 所示。

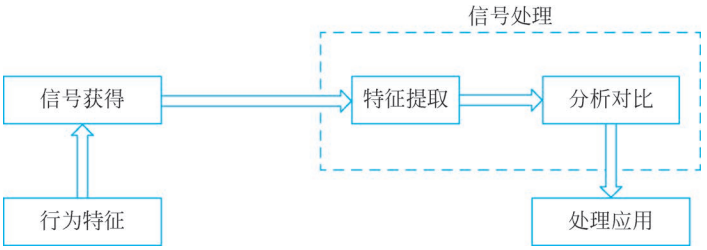

图 5-25　行为认证系统的结构

不同人的行为习惯及表现具有独一无二的特性，因此行为认证技术能够有效抵御各种假冒攻击和入侵，保证身份识别的准确。

自动认证技术是认证技术的演进方向，可以融合多种认证技术（标识认证技术、基于量子密码的认证技术、思维认证技术、行为认证技术等），可以接受多种认证手段（口令、KEY、证书、生物特征信息等），提供接入多元化、核心架构统一化、应用服务综合化的智能认证技术。自动认证技术其原理：综合利用各个认证因子作为整个认证系统的输入，利用专家知识系统对其进行判断，对其没有通过认证的和假冒成功的综合因子的特征信息通过免疫技术学习进化使得专家知识库不断更新，同时通过数据挖掘技术来识别专家知识库中不曾进化的潜在威胁。典型的自动认证技术的结构如图 5-26 所示。

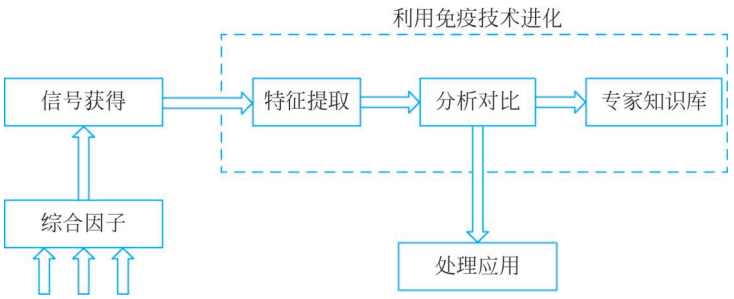

图 5-26　自动认证技术的结构

单一因子的认证技术很容易存在被假冒的风险，利用综合因子正好可以提高系统的安全性，同时可以确保多种认证手段的无缝接入，因此自动认证技术有着其他认证技术不可比拟的优势。

5.5 零信任

云计算和大数据时代，网络安全边界逐渐瓦解，内外部威胁愈演愈烈，传统的边界安全架构难以应对，零信任（ZT）安全架构应运而生。

零信任安全架构基于“以身份为基石、业务安全访问、持续信任评估、动态访问控制”四大关键能力，构筑以身份为基石的动态虚拟边界产品与解决方案，助力组织实现全面身份化、授权动态化、风险度量化、管理自动化的新一代网络安全架构。

5.5.1 零信任概述

网络基础设施日益复杂，安全边界逐渐模糊。数字化转型的时代浪潮推动着信息技术的快速演进，云计算、大数据、物联网、移动互联等新兴技术为各行各业带来了新的生产力，同时也给组织网络基础设施带来了极大的复杂性。安全边界正在逐渐瓦解，传统的基于边界的网络安全架构和解决方案难以适应现代网络基础设施。

网络安全形势不容乐观，有组织的、武器化的、以数据及业务为攻击目标的高级持续攻击仍然能轻易找到各种漏洞突破组织的边界；内部业务的非授权访问、雇员犯错、有意的数据窃取等内部威胁愈演愈烈。

安全事件层出不穷，传统安全架构失效背后的根源是信任。传统的基于边界的网络安全架构某种程度上假设，或默认了内网的人和设备是值得信任的，认为网络安全就是构筑组织的数字“护城河”，通过防火墙、WAF、IPS 等边界安全产品/方案对组织网络边界进行重重防护就足够了。

事实证明，正确的思维应该是假设系统一定有未被发现的漏洞，假设一定有已发现但仍未修补的漏洞，假设系统已经被渗透，假设内部人员不可靠。“四个假设”彻底推翻了传统网络安全通过隔离、修边界的技术方法，推翻了边界安全架构下对“信任”的假设和滥用，基于边界的网络安全架构和解决方案已经难以应对如今的网络威胁。

需要全新的网络安全架构应对现代复杂的组织网络基础设施，应对日益严峻的网络威胁形势，零信任架构(ZTA)正是在这种背景下应运而生，是安全思维和安全架构进化的必然。

零信任架构一直在快速发展和成熟，不同版本的定义基于不同的维度进行描述。EvanGilman 和 DougBarth 在《零信任网络》一书中将零信任的定义建立在如下五个基本假定之上。

(1) 网络无时无刻不处于危险的环境中。

(2) 网络中自始至终存在外部或内部威胁。

(3) 网络的位置不足以决定网络的可信程度。

(4) 所有的设备、用户和网络流量都应当经过认证和授权。

(5) 安全策略必须是动态的，并基于尽可能多的数据源计算而来。

简而言之，默认情况下不应该信任网络内部和外部的任何人、设备和系统，需要基于认证和授权重构访问控制的信任基础。零信任对访问控制进行了范式上的颠覆，其本质是以身份为基石的动态可信访问控制。

NIST 在其发布的《零信任架构》(草案)中指出，零信任架构是一种网络/数据安全的端到端方法，关注身份、凭证、访问管理、运营、终端、主机环境和互联的基础设施，认为零信任是一种关注数据保护的架构方法，传统安全方案只关注边界防护，对授权用户开放

了过多的访问权限。零信任的首要目标是基于身份进行细粒度的访问控制，以应对越来越严峻的越权横向移动风险。

零信任提供了一系列概念和思想，旨在面对失陷网络时，减少在信息系统和服务中执行准确的、按请求访问决策时的不确定性。零信任架构是一种组织网络安全规划，它利用零信任概念，并囊括其组件关系、工作流规划与访问策略。

从零信任的发展历史进行分析，也不难发现零信任的各种不同维度的观点也在持续发展、融合，并最终表现出较强的一致性。

零信任的最早雏形源于 2004 年成立的耶利哥论坛(Jericho Forum)，其成立的使命正是为了定义无边界趋势下的网络安全问题并寻求解决方案。2010 年，零信任术语正式出现，并指出所有的网络流量都是不可信的，需要对访问任何资源的任何请求进行安全控制，零信任提出之初，其解决方案专注于通过微隔离对网络进行细粒度的访问控制以便限制攻击者的横向移动。

随着零信任的持续演进，以身份为基石的架构体系逐渐得到业界主流的认可，这种架构体系的转变与移动计算、云计算的大幅采用密不可分。从 2014 年开始，谷歌公司基于其内部项目 Beyond Corp 的研究成果，陆续发表了多篇论文，阐述了在谷歌公司内部如何为其员工构建零信任架构。Beyond Corp 的出发点在于仅仅针对组织边界构建安全控制已经不够了，需要把访问控制从边界迁移到每个用户和设备。通过构建零信任，谷歌公司成功地摒弃了对传统 VPN 的采用，通过全新架构体系确保所有来自不安全网络的用户能安全地访问组织业务。

通过业界对零信任理论和实践的不断完善，零信任已经超越了最初的网络层微分段的范畴，演变为以身份为基石的，能覆盖云环境、大数据中心、微服务等众多场景的新一代安全解决方案。

综合分析各种零信任的定义和框架，不难看出零信任架构的本质是以身份为基石的动态可信访问控制，聚焦身份、信任、业务访问和动态访问控制等维度的安全能力，基于业务场景的人、流程、环境、访问上下文等多维的因素，对信任进行持续评估，并通过信任等级对权限进行动态调整，形成具备较强风险应对能力的动态自适应的安全闭环体系。

5.5.2 零信任参考架构

零信任安全的关键能力可以概括为以身份为基石、业务安全访问、持续信任评估和动态访问控制，这些关键能力映射到一组相互交互的核心架构组件，对各业务场景具备较高的适应性，如图 5-27 所示。

零信任的本质是在访问主体和客体之间构建以身份为基石的动态可信访问控制体系，通过以身份为基石、业务安全访问、持续信任评估和动态访问控制的关键能力，基于对网络所有参与实体的数字身份，对默认不可信的所有访问请求进行加密、认证和强制授权，汇聚关联各种数据源进行持续信任评估，并根据信任的程度动态对权限进行调整，最终在访问主体和访问客体之间建立一种动态的信任关系。

零信任架构下，访问客体是核心保护的资源，针对被保护资源构建保护面，资源包括

图 5-27　零信任架构的关键能力模型

但不限于组织的业务应用、服务接口、操作功能和资产数据。访问主体包括人员、设备、应用和系统等身份化之后的数字实体，在一定的访问上下文中这些实体还可以进行组合绑定，进一步对主体进行明确和限定。

1. 以身份为基石

基于身份而非网络位置来构建访问控制体系，首先需要为网络中的人和设备赋予数字身份，将身份化的人和设备进行运行时组合构建访问主体，并为访问主体设定所需的最小权限。

在零信任安全架构中，根据一定的访问上下文，访问主体可以是人、设备和应用等实体数字身份的动态组合，在《零信任网络》一书中将这种组合称为网络代理。网络代理是指在网络请求中用于描述请求发起者的信息集合，一般包括用户、应用程序和设备三类实体信息，用户、应用程序和设备信息是访问请求密不可分的上下文。网络代理具有短时性特征，在进行授权决策时按需临时生成。访问代理的构成要素（用户或设备）信息一般存放在数据库中，在授权时实时查询并进行组合，因此，网络代理代表的是用户和设备各个维度的属性在授权时刻的实时状态。

最小权限原则是任何安全架构必须遵循的关键实践之一，然而零信任架构将最小权限原则又推进了一大步，遵循了动态的最小权限原则。如果用户需要更高的访问权限，那么用户只能在需要的时候获得这些特权。而反观传统的身份与访问控制相关实现方案，一般对人、设备进行单独授权，零信任这种以网络代理作为授权主体的范式，在授权决策时按需临时生成主体，具有较强的动态性和风险感知能力，可以极大地缓解凭证窃取、越权访问等安全威胁。

2. 业务安全访问

零信任架构关注业务保护面的构建，通过业务保护面实现对资源的保护，在零信任架构中，应用、服务、接口、数据都可以视作业务资源。通过构建保护面实现对暴露面的收缩，要求所有业务默认隐藏，根据授权结果进行最小限度的开放，所有的业务访问请求都应该进行全流量加密和强制授权，业务安全访问相关机制需要尽可能工作在应用协议层。

构建零信任安全架构，需要关注待保护的核心资产，梳理核心资产的各种暴露面，并通过技术手段将暴露面进行隐藏。这样，核心资产的各种访问路径就隐藏在零信任架构组件之后，默认情况对访问主体不可见，只有经过认证、具有权限、信任等级符合安全策

略要求的访问请求才予以放行。通过业务隐藏，除了满足最小权限原则，还能很好地缓解针对核心资产的扫描探测、拒绝服务、漏洞利用、非法爬取等安全威胁。

3. 持续信任评估

持续信任评估是零信任架构从零开始构建信任的关键手段，通过信任评估模型和算法，实现基于身份的信任评估能力，同时需要对访问的上下文环境进行风险判定，对访问请求进行异常行为识别并对信任评估结果进行调整。

在零信任架构中，访问主体是人、设备和应用程序三位一体构成的网络代理，因此在基础的身份信任的基础上还需要评估主体信任。主体信任是对身份信任在当前访问上下文中的动态调整，与认证强度、风险状态和环境因素等相关，身份信任相对稳定。主体信任和网络代理一样，具有短时性特征，是一种动态信任，基于主体的信任等级进行动态访问控制也是零信任的本质所在。

信任和风险如影随形，在某些特定场景下甚至是一体两面。在零信任架构中，除了信任评估，还需要考虑环境风险的影响因素，需要对各类环境风险进行判定和响应。但需要特别注意，并非所有的风险都会影响身份或主体的信任度。

基于行为的异常发现和信任评估能力必不可少，主体（所对应的数字身份）与个体行为的基线偏差、主体与群体的基线偏差、主体环境的攻击行为、主体环境的风险行为等都需要建立模型进行量化评估，是影响信任的关键要素。当然，行为分析需要结合身份态势进行综合度量，以减少误判，降低对使用者操作体验的负面影响。

4. 动态访问控制

动态访问控制是零信任架构的安全闭环能力的重要体现。建议通过 RBAC（基于角色的访问控制）和 ABAC（基于属性的访问控制）的组合授权实现灵活的访问控制基线，基于信任等级实现分级的业务访问；同时，当访问上下文和环境存在风险时，需要对访问权限进行实时干预并评估是否对访问主体的信任进行降级。

任何访问控制体系的建立离不开访问控制模型，需要基于一定的访问控制模型制定权限基线。零信任强调灰度哲学，从实践经验来看，大可不必纠结 RBAC 好还是 ABAC 好，而是考虑如何兼顾融合，建议基于 RBAC 模型实现粗粒度授权，建立权限基线满足组织基本的最小权限原则，并基于主体、客体和环境属性实现角色的动态映射和过滤机制，充分发挥 ABAC 的动态性和灵活性。权限基线决定了一个访问主体允许访问的权限的全集，而在不同的访问时刻，主体被赋予的访问权限和访问上下文、信任等级、风险状态息息相关。

需要注意，并非所有的风险都对信任有影响，特别是环境风险，风险一旦发生，就需要对应的处置策略，常见手段是撤销访问会话。因此，零信任架构的控制平面需要能接收外部风险平台的风险通报，并对当前访问会话进行按需处理，从而实现风险处置的联动，真正将零信任架构体系与组织现存的其他安全体系融合。

5.6 安全协议

安全协议也称密码协议，是以密码学为基础的消息交换协议，此定义包含两层含义：

安全协议以密码学为基础，体现了安全协议与普通协议之间的差异；安全协议也是通信协议，其目的是在网络环境中提供各种安全服务。

密码学是网络安全的基础，但网络安全不能单纯依靠安全的密码算法。安全协议是网络安全的一个重要组成部分，需要通过安全协议进行实体之间的认证，在实体之间安全地分配密钥或其他各种秘密，确认发送和接收的消息的不可否认性等。

总之，安全协议是建立在密码体制基础上的一种交互通信协议，它运用密码算法和协议逻辑来实现认证和密钥分配等目标。

全面掌握安全协议定义，还需了解安全协议在运行环境中的角色。协议参与者，即协议执行过程中的双方或多方，也就是发送方与接收方，可能是完全信任的人，也可能是攻击者。攻击者即协议过程中企图破坏协议安全性的人。如被动攻击者主要是试图获取信息，主动攻击者则是为了达到欺骗、获取敏感信息、破坏协议完整性的目的。攻击者可能是合法的参与者，或是外部实体，也可能是协议的参与者。可信第三方，即在完成协议的过程中，能帮助可信任的双方完成协议的值得信任的第三方，如仲裁者（用于解决协议过程中出现的纠纷）和密钥分发中心等。

5.6.1 安全协议的分类

按照协议完成的功能进行划分，常用的安全协议主要有以下四类。

1. 密钥生成协议

密钥生成协议是在通信的实体中建立共享的会话密钥，会话密钥通常使用对称密码算法对每一次单独的会话加密。密钥生成协议可采用对称密码体制或非对称密码体制建立会话密钥，可借助于一个可信的服务器为用户分发密钥，即密钥分发协议，可通过两个用户协商建立会话密钥。

2. 认证协议

认证是对数据和实体标识的认证。数据完整性可由数据来源认证保证。实体认证是确认某个实体的真实性的过程。认证协议主要用于防止假冒攻击。

3. 电子商务协议

电子商务是利用电子信息技术进行各种商务活动。电子商务协议中主体往往代表交易的双方目标利益不太一致，因此电子商务协议最关注公平性，即协议应保证交易双方都不能通过损害对方利益得到不应该得到的利益。常见的电子商务协议有拍卖协议、SET 协议等。

4. 安全多方计算协议

安全多方计算协议是保证分布式环境中各参与者以安全的方式来共同执行分布式的计算任务。安全多方计算协议的两个最基本的安全要求是保证协议的正确性和参与方私有输入的秘密性，即协议执行完后每个参与方都应该得到正确的输出，并且除此之外不能获知其他任何信息。根据参与者以及密码算法的使用情况，可以分为无可信第三

方的对称密钥协议、应用密码校验函数的认证协议、具有可信第三方的对称密钥协议、使用对称密钥的签名协议、使用对称密钥的重复认证协议、无可信第三方的公钥协议和有可信第三方的公钥协议。

5.6.2 双向身份认证协议

双向身份认证协议需要消息的发送方和接收方同时在线，而单向认证没有这样的要求。在重要的商务活动中，通信双方在通信之前要相互确认对方的真实身份，并且希望之间的通信不会被第三者阅读。无论是单向认证还是双向认证，都可以分为基于对称密钥加密和非对称密钥加密两种情况。

(1) 时间戳：在电子商务中，时间是十分重要的信息。与普通文件一样，在网络上传输的电子商务信息的日期是十分重要的，它是防止文件被伪造、篡改、防抵赖的关键性内容。时间戳是经加密后形成的凭证文档。它包括需加时间戳的文件的摘要，数字时间服务(Digital Timestamp Service，DTS)，收到文件的日期和时间，DTS 的数字签名。使用时间戳的各方的系统时钟应该是同步的。

(2) 随机数：计算机利用一定的算法产生的数值。严格地说，计算机产生的随机数应该称为“伪随机数”，因为这个数不是真正随机的。为了叙述方便，一般都将“伪”字去掉。

(3) 序列号：一个随机数值，通信双方协商各自的序列号初始值。一个新消息当且仅当有正确的序列号时才被接收。但序列号的方法不适合在身份认证协议中使用，因为它要求每个用户要单独记录与其他每一用户交换的消息的序列号，这样增加了用户的负担。

(4) 挑战—应答：假设 A 向 B 发送了一个一次性随机数询问，B 的回答中应该包含正确的随机数，该机制称为挑战—应答机制。

5.7 本章小结

认证分为消息认证和身份认证。消息认证是用来防止主动攻击的重要技术，用以保证消息的完整性。身份认证是证实一个实体与其所声称身份是否相符的过程。本章首先着重介绍了 Hash 函数、MD5 算法、SHA 算法、SM3 算法、消息认证、数字签名、身份认证、零信任、安全协议等；其次详细描述了基于对称密码体制、基于非对称密码体制、利用 Hash 函数和基于 MAC 的消息认证方法；再次介绍了利用对称加密和非对称加密方式实现数字签名的原理；最后重点介绍了基于密码的身份认证、基于生物特征的身份认证、基于零知识证明的身份认证和基于量子密码的身份证技术。

习　　题

5-1　什么是身份认证？为什么需要身份认证？

5-2 什么是生物特征认证？与传统的密码认证等方式相比，生物特征认证有哪些优势？并比较分析指纹识别、虹膜认证和人脸识别的实现技术和应用特点。

5-3 什么是零知识证明认证？有什么特点？

5-4 为什么需要消息认证？

5-5 数字签名需要满足哪些条件？身份认证与数字签名的区别是什么？

5-6 数字签名技术有哪几种？分析其优缺点。