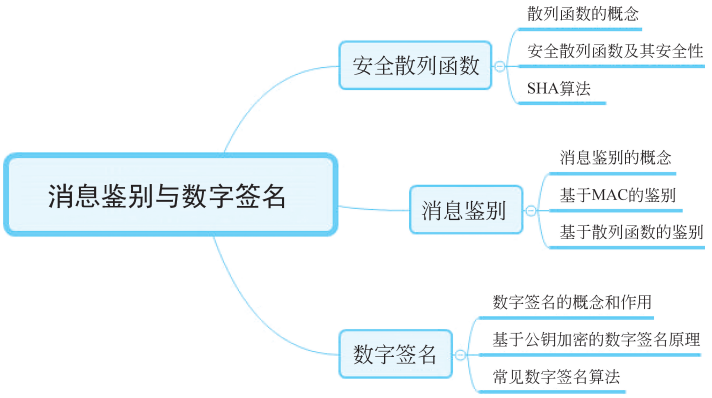
第 3 章

消息鉴别与数字签名

课程思政

导　　读



经典的密码学是关于加密和解密的理论，主要用于保密。目前，密码学已得到更加深入、广泛的发展和应用，不再局限于单一的加解密技术，而是被有效、系统地用于保证电子数据的机密性、完整性和真实性。这是因为，在公开的计算机网络环境中，传输中的数据可能遭到的威胁不只局限于泄密，而是多种形式的攻击。

（1）泄密。消息的内容被泄露给没有合法权限的任何人或过程。

（2）通信业务量分析。指分析通信双方的通信模式。在面向连接的应用中，确定连接的频率和持续时间；在面向连接或无连接的环境中，确定双方的消息数量和长度。

（3）伪造消息。攻击者假冒真实发送方的身份，向网络中插入一条消息，或者假冒接收方发送一个消息确认。

（4）篡改消息。这分成 3 种情形。

- 内容篡改。对消息内容的改动，包括插入、删除、调换和修改。
- 序号篡改。在依赖序号的通信协议（如 TCP）中，对通信双方消息序号进行修改，包括插入、删除和重新排序。
- 时间篡改。对消息进行延时和重放。在面向连接的应用中，整个消息序列可能是前面某合法消息序列的重放，也可能是消息序列中的一条消息被延时或重放；在面向无连接的应用中，可能是一条消息（如数据报）被延时或重放。

（5）行为抵赖。发送方否认发送过某消息或者接收方否认接收到某消息。

对抗前两种攻击的方法属于消息保密性范畴，前面讲过的对称密码学和公钥密码学都是围绕这个主题展开的；对付第 3 种和第 4 种攻击的方法一般称为消息鉴别；对付第 5

种攻击的方法属于数字签名。一般而言，数字签名方法也能够对抗第 3 种和第 4 种中的某些或全部攻击。本章将介绍消息鉴别和数字签名。


教学课件


教学视频

3.1　安全散列函数

散列函数又叫作散列算法、哈希函数，是一种将任意长度的消息映射到某一固定长度散列值（又称为哈希值、消息摘要）的函数。散列值相当于消息的“指纹”，用来防止对消息的非法篡改。对消息任何一个或几个比特的改变都将极大地改变消息的散列值，即消息的“指纹”变得不正确。因此，散列函数的首要目标是保证消息的完整性。

令 H 代表一个散列函数，M 代表一个任意长度的消息，则 M 的散列值 h 表示为 $h=H(M)$，h 长度固定。我们称 h 是 M 的像，而 M 是 h 的原像。因为 M 长度任意，而 h 通常长度较短，所以散列函数是多对一映射。对于给定的散列值 h，对应多个原像。如果有两个消息 M_1 和 M_2，$M_1 \neq M_2$ 但 $H(M_1)=H(M_2)$，则称 M_1 和 M_2 是碰撞的。因为使用散列函数的目的是保证消息完整性，所以出现碰撞是我们不希望的。

3.1.1　安全散列函数及其安全性

在安全应用中使用的散列函数称为安全散列函数。安全散列函数应该是用途最多的密码算法，它被广泛运用于各种不同的安全应用和网络协议中。

安全散列函数必须满足一定的安全特征，主要包括 3 个被广泛认同的特性：单向性、弱抗碰撞性和强抗碰撞性。

- 单向性，又称为抗原像攻击，是指对任意给定的散列值 h，找到满足 $H(M)=h$ 的消息 M 在计算上是不可行的；即给定散列函数 h，由消息 M 计算散列值 $H(M)$ 是容易的，但由散列值 $H(M)$ 计算 M 是不可行的。
- 弱抗碰撞性，又称为抗第二原像攻击，指给定一个随机选择的消息 M，寻找消息 M'，使得 $H(M)=H(M')$ 在计算上是不可行的，即不能找到与给定消息具有相同散列值的另一消息。
- 强抗碰撞性，又称为抗碰撞攻击，是指对于给定的散列函数 H，寻找两个不同的消息 M_1 和 M_2，使得 $H(M_1)=H(M_2)$ 在计算上是不可行的。

与加密算法一样，对安全散列函数的攻击也分成两类：穷举攻击和密码分析攻击。

穷举攻击不依赖算法细节，仅和散列值的长度有关。假设散列值长度为 m 比特。对于原像攻击和第二原像攻击，穷举攻击的方式是随机选择消息 M，计算其散列值直到碰撞出现；平均情况下，攻击者要尝试 2^{m-1} 次才能找到满足要求的 M。对于碰撞攻击，攻击者试图找到两个不同的消息 M_1 和 M_2，使得 $H(M_1)=H(M_2)$；平均情况下，攻击者要尝试 $2^{m/2}$ 次才能成功。

对安全散列函数进行密码分析攻击则是利用散列函数的某种性质而不是通过穷举方法来开展。因此，评价安全散列函数抗密码分析攻击的方法就是，将其与穷举攻击的代价进行比较。对理想的安全散列函数进行密码分析攻击的代价大于或等于穷举攻击所需的代价。最近一二十年，人们对安全散列函数的密码分析攻击方面做了大量工作，其中有些

攻击是成功的。例如，2004年，时任山东大学信息安全研究所所长的王小云教授给出了曾经广泛使用的安全散列函数MD5的多个碰撞对，证明了MD5已不再安全。

3.1.2 SHA

人们设计出了大量的散列算法，其中SHA（Secure Hash Algorithm，安全散列算法）是近年来使用最广泛的。事实上，由于其他曾经广泛使用的安全散列函数被证实存在安全缺陷，SHA是2005年以来仅存的安全散列函数标准。

NIST于1993年发布了第1版的SHA，后来人们给它取了一个非正式的名称SHA-0，以避免与它的后继者混淆。SHA-0很快被发现存在缺陷，其修订版于1995年发布，即SHA-1。SHA-1算法产生160比特的散列值。2005年，王小云教授和姚期智教授给出了一个攻击SHA-1的方法，用2^{69}次操作就可以找到一对碰撞，远远少于此前人们认为的2^{80}次。同年，NIST宣布逐步废除SHA-1的意图，计划到2010年过渡到SHA-2及后续版本。

2002年，NIST给出了3种新的SHA版本，散列值长度依次为256位、384位和512位，分别称为SHA-256、SHA-384和SHA-512；2008年增加了224位版本。这些算法被统称为SHA-2。

值得注意的是，SHA-2与其前任具有类似的结构和基本数学运算。尽管当前还未被“攻破”，但鉴于MD5和SHA-0都被证实是不安全的，人们担心或许若干年后SHA-2的缺陷也会被发现。因此，NIST决定开展新的安全散列函数标准的制定工作，并且于2012选择Keccak算法作为SHA-3的标准算法。SHA-3采用了与前任算法不同的结构。

本节以512位版本为例介绍SHA-2。

1. SHA-512算法概要

SHA-512算法的输入是最大长度小于2^{128}位的消息，输出是长度为512位的消息散列值。

图3-1显示了SHA-512消息处理的总体过程，包含以下几个步骤。

步骤1：附加填充位。

填充消息使其长度模1024与896同余，即长度在对1024取模以后的余数是896。即使消息已经满足上述长度要求，仍然需要进行填充。填充是这样进行的：先补一个1，然后再补0，直到长度满足对1024取模后余数是896。因此，至少填充1位，最多填充1024位。

步骤2：附加长度。

将原始数据的长度补到已经进行了填充操作的消息后面。这个长度是一个128位的无符号整数。

前两步的结果是产生一个长度为1024整数倍的扩展消息，可看成一串长度为1024比特的分组$M_1, M_2, \cdots, M_N$。

步骤3：初始化散列缓冲区。

算法运算的中间结果和最终结果保存于一个长度为512位的缓冲区中，该缓冲区可以看成8个长度为64位的寄存器：a、b、c、d、e、f、g、h。这些寄存器按照如下方式初始化。

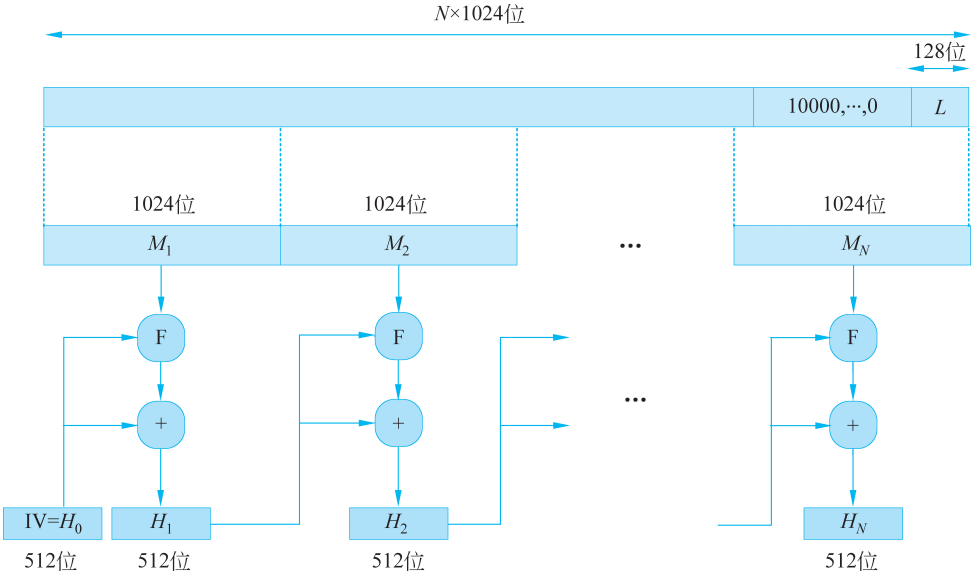


图 3-1　SHA-512 算法总体过程

a = 6A09E667F3BCC908　　e = S1OES27FADE682D1

b = BB67AE8S84CAA73B　　f = 9BOS688C2B3E6CIF

c = 3C6EF372FE94F82B　　g = IF83D9ABFB41BD6B

d = AS4FFS3ASFID36F1　　h = SBEOCD19137E2179

步骤 4：计算消息摘要。

算法的核心是图中标记为 F 的运算模块，该模块对每个长度为 1024 位的分组都进行 80 轮的运算处理。因此，如果共有 N 个 1024 位的分组，消息摘要的计算过程就包括 N 个阶段，每个阶段进行 80 轮的运算。

F 模块的输入有两个：当前阶段处理的分组 M_i，以及前一阶段对前一分组 M_{i-1} 处理的结果 H_{i-1}。

图 3-2 给出了 F 模块的逻辑原理。F 模块把 512 位的缓冲区 abcdefgh 作为输入并更新缓冲区的值。第 0 轮时，缓冲区的值是中间值 H_{i-1}。每一轮，如第 t 轮（$0 \leqslant t \leqslant 79$），使用一个 64 位的数值 W_t，该数值从当前处理的分组 M_t 导出。每一轮还使用一个常数 K_t，作用是使得每轮的运算不同。用如下方式获得 K_t：对前 80 个素数分别求立方根，取小数部分的前 64 位。最后一轮的输出和第 0 轮的输入进行模 2^{64} 的加法运算得到 H_i。

步骤 5：输出。

所有的 N 个 1024 位的分组都处理完以后，第 N 阶段的输出就是长度为 512 比特的消息摘要。

因此，SHA-512 算法的步骤可描述如下。

$$H_0 = \mathrm{IV}$$

$$H_i = \mathrm{SUM}_{64}(H_{i-1}, \mathrm{abcdefgh}_i)$$

$$\mathrm{MD} = H_N$$

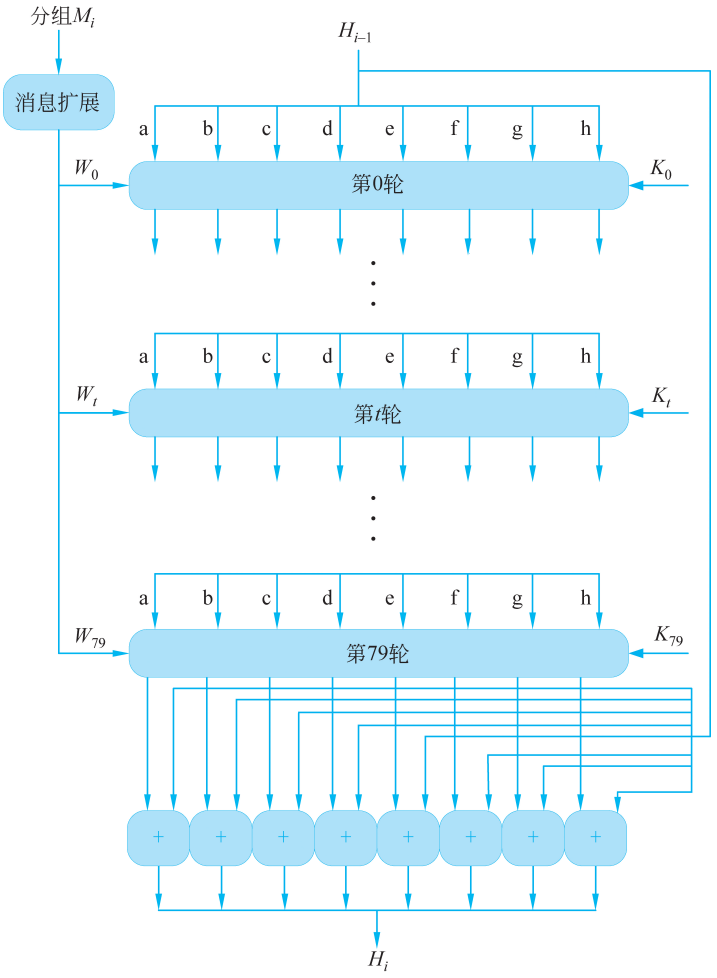


图 3-2　F 模块的逻辑原理

其中，IV 是缓冲区 abcdefgh 的初始值，abcdefgh$_i$ 表示第 i 个分组处理后的最终结果，N 是分组数，SUM$_{64}$ 表示模 2^{64} 加，MD 为最终输出的散列值。

2. SHA-512 轮函数

F 模块每一轮的运算可用以下公式描述。

$$T_1 = h + \mathrm{Ch}(e,f,g) + \left(\sum_{1}^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\sum_{1}^{512} a\right) + \mathrm{Maj}(a,b,c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

其中：

- t 为轮数，W_t 是从当前处理的分组导出的 64 位数值，K_t 为附加常数。
- $\mathrm{Ch}(e,f,g)=(e\ \mathrm{AND}\ f)\oplus(\mathrm{NOT}\ e\ \mathrm{AND}\ g)$，这是一个条件函数：如果 e，则 f；否则 g。
- $\mathrm{Maj}(a,b,c)=(a\ \mathrm{AND}\ b)\oplus(a\ \mathrm{AND}\ c)\oplus(b\ \mathrm{AND}\ c)$，当 a、b、c 中的多数(2 个或 3 个)为真时，结果为真。
- $\left(\sum_{0}^{512} a\right)=\mathrm{ROTR}^{28}(a)\oplus\mathrm{ROTR}^{34}(a)\oplus\mathrm{ROTR}^{39}(a)$。
- $\mathrm{ROTR}^{n}(x)$ 表示对 64 位变量 x 循环右移 n 位。
- W_t 是从当前处理的分组导出的 64 位数值。每个分组共 1024 位，可以看成 16 个 64 位的字，序号为 0～15；每一阶段包括 80 轮，前 16 轮的 W_t $(0\leqslant t\leqslant 15)$ 直接取自分组中第 t 个 64 位的字；后 64 轮的 W_t $(16\leqslant t\leqslant 79)$ 按照如下方式产生。

$$W_t=\sigma_1^{512}(W_{t-2})+W_{t-7}+\sigma_0^{512}(W_{t-15})+W_{t-16}$$
$$\sigma_1^{512}(x)=\mathrm{ROTR}^{19}(x)\oplus\mathrm{ROTR}^{8}(x)\oplus\mathrm{SHR}^{7}(x)$$
$$\sigma_0^{512}(x)=\mathrm{ROTR}^{19}(x)\oplus\mathrm{ROTR}^{61}(x)\oplus\mathrm{SHR}^{6}(x)$$

其中，$\mathrm{ROTR}^{n}(x)$ 表示对 64 位变量 x 循环右移 n 位，$\mathrm{SHR}^{n}(x)$ 则表示左移 n 位(右边补 0)。

SHA-512 算法产生的消息散列值的每一位都是全部输入位的函数。除非 SHA-512 算法存在目前未知的隐藏缺陷，否则找到一对碰撞需要 2^{256} 次操作，针对给定散列值寻找对应的消息则需要 2^{512} 次操作。基于目前对 SHA-512 的认知，这是一个安全的散列算法。

3.2　消 息 鉴 别


教学课件


教学视频

完整性是安全的基本要求之一。篡改消息是对通信系统进行主动攻击的常见形式，被篡改的消息是不完整的；信道的偶发干扰和故障也破坏了消息的完整性。接收者应该能够检查所收到的消息是否完整。另外，攻击者还可以将一条声称来自合法授权用户的虚假消息插入网络，或者冒充消息的合法接收者发回假确认。因此，消息接收者还应该能够识别收到的消息是否确实来源于该消息所声称的主体，即验证消息来源的真实性。

保障消息完整性和真实性的重要手段是消息鉴别技术。

3.2.1　消息鉴别的概念

消息鉴别也称为“报文鉴别”或“消息认证”，是一个对收到的消息进行验证的过程。验证的内容包括两方面。

- 真实性。消息的发送者是真正的而不是冒充的。

- 完整性。消息在传送和存储过程中未被篡改过。

从功能上看，一个消息鉴别系统可分成两个层次，如图 3-3 所示。

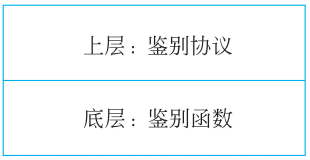


图 3-3　消息鉴别系统的功能分层结构

底层是一个鉴别函数，其功能是产生一个鉴别符。鉴别符是一个用来鉴别消息的值，即鉴别的依据。在此基础上，上层的鉴别协议调用该鉴别函数，实现对消息真实性和完整性的验证。鉴别函数是决定鉴别系统特性的主要因素。

根据鉴别符的生成方式，鉴别函数可分为如下 3 类。

- 基于消息加密。以整个消息的密文作为鉴别符。
- 基于消息鉴别码(MAC)。利用公开函数和密钥产生一个较短的定长值作为鉴别符，并且与消息一同发送给接收方，实现对消息的验证。
- 基于散列函数。利用公开函数将任意长的消息映射为定长的散列值，并且以该散列值作为鉴别符。

目前，像对称加密、公钥加密等常规加密技术已经发展得非常成熟。但是，由于多种原因，常规加密技术并没有被简单地应用到消息鉴别符的生成中，实际应用中一般采用独立的消息鉴别符。用避免加密的方法提供消息鉴别符受到广泛的重视，而最近几年消息鉴别的热点转向由散列函数导出 MAC。

3.2.2　基于 MAC 的鉴别

1. 消息鉴别码原理

消息鉴别码(Message Authentication Code，MAC)又称为密码校验和，其实现鉴别的原理是：用公开函数和密钥生成一个固定大小的小数据块，即 MAC，并且将其附加在消息之后传输；接收方利用与发送方共享的密钥进行鉴别。基于 MAC 提供消息完整性保护时，MAC 可以在不安全的信道中传输，因为 MAC 的生成需要密钥。

基于 MAC 的鉴别原理见图 3-4。假定通信双方(如 A 和 B)共享密钥 K。A 向 B 发送消息时，A 计算 MAC，它是消息和密钥 K 的函数，即 $\mathrm{MAC}=C(K, M)$。其中，M 为输入消息，C 为 MAC 函数，K 为共享的密钥，MAC 为消息鉴别码。

消息和 MAC 一起被发送给接收方 B。接收方 B 对收到的消息用相同的密钥 K 进行相同的计算得出新的 MAC，并且与接收到的 MAC 进行比较。假定只有收发双方知道该密钥，若接收到的 MAC 与计算得出的 MAC 相等，则说明以下 3 点。

(1) 接收方 B 可以相信消息在传送途中未被非法篡改。因为假定攻击者不知道密钥 K，所以攻击者可能修改消息，但他不知道应如何改变 MAC 才能使其与修改后的消息相一致。这样，接收方计算出的 MAC 将不等于接收到的 MAC。

(2) 接收方 B 可以相信消息来自真正的发送方 A。因为其他各方均不知道密钥，因此他们不能产生具有正确 MAC 的消息。

(3) 如果消息中含有序列号(如 TCP 序列号)，那么接收方可以相信消息顺序是正确的，因为攻击者无法成功地修改序列号并保持 MAC 与消息一致。

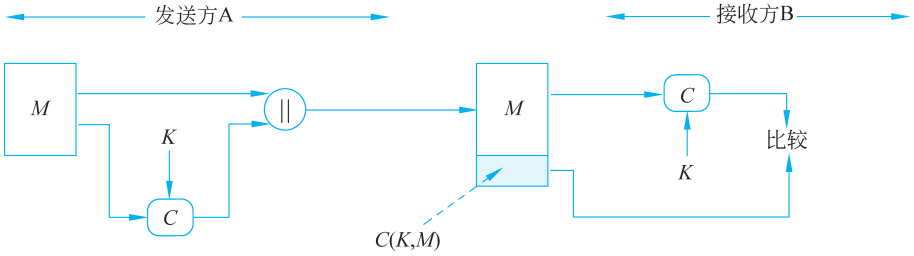


图 3-4　基于 MAC 的鉴别原理

图 3-4 所示的过程仅提供鉴别而不能提供保密性，因为消息是以明文形式传送的。若将 MAC 附加在明文消息后对整个信息块加密，则可以同时提供保密和鉴别。这需要两个独立的密钥，并且收发双方共享这两个密钥。

MAC 函数与加密类似，但加密算法必须是可逆的，而 MAC 算法则不要求可逆性，在数学上比加密算法易受攻击的弱点要少。与加密算法相比，MAC 算法更不易被攻破。

2. 基于 DES 的消息鉴别码

构造 MAC 的常用方法之一就是基于分组密码并按 CBC（密文块链接）模式操作。在 CBC 模式中，每个明文分组在用密钥加密之前，要先与前一个密文分组进行异或运算。用一个初始向量 IV 作为密文分组初始值。

数据鉴别算法也称为 CBC-MAC（密文块链接-消息鉴别码），它建立在 DES 之上，是使用极广泛的 MAC 算法之一，也是 ANSI 的一个标准。

数据鉴别算法采用 DES 运算的 CBC 方式，参见图 3-5。其初始向量 IV 为 **0**，需要鉴别的数据分成连续的 64 位的分组 $\boldsymbol{D}_1,\boldsymbol{D}_2,\cdots,\boldsymbol{D}_N$，若最后分组不足 64 位，则在其后填 0 直至成为 64 位的分组。利用 DES 加密算法 E 和密钥 K 计算数据鉴别码（DAC）的过程如下。

$$\begin{aligned}
&\boldsymbol{O}_0 = \mathrm{IV}\\
&\boldsymbol{O}_1 = E_K(\boldsymbol{D}_1 \oplus \boldsymbol{O}_0)\\
&\boldsymbol{O}_2 = E_K(\boldsymbol{D}_2 \oplus \boldsymbol{O}_1)\\
&\boldsymbol{O}_3 = E_K(\boldsymbol{D}_3 \oplus \boldsymbol{O}_2)\\
&\vdots\\
&\boldsymbol{O}_N = E_K(\boldsymbol{D}_N \oplus \boldsymbol{O}_{N-1})
\end{aligned}$$

DAC 可以取整个块 $\boldsymbol{O}_N$，也可以取其最左边的 M 位，其中 $16 \leqslant M \leqslant 64$。

3.2.3　基于散列函数的鉴别

散列函数是消息鉴别码的一种变形。与消息鉴别码一样，散列函数的输入是可变大小的消息 M，输出是固定大小的散列码 $H(M)$，也称为消息摘要或散列值。与 MAC 不同的是，散列函数并不使用密钥，它仅是输入消息的函数。使用没有密钥的散列值作为消息鉴别码的机制是不安全的，因此实践中常将散列函数和加密结合起来使用。

图 3-6 给出了将散列码用于消息鉴别的两种常用方法。

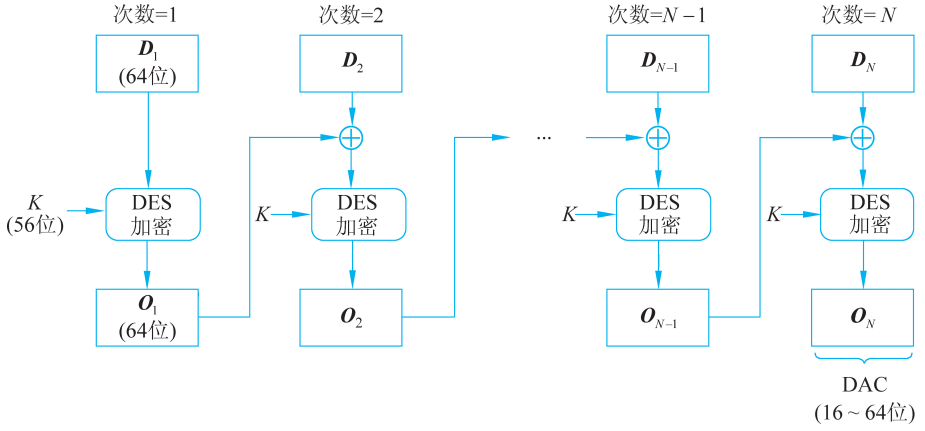


图 3-5　数据鉴别算法

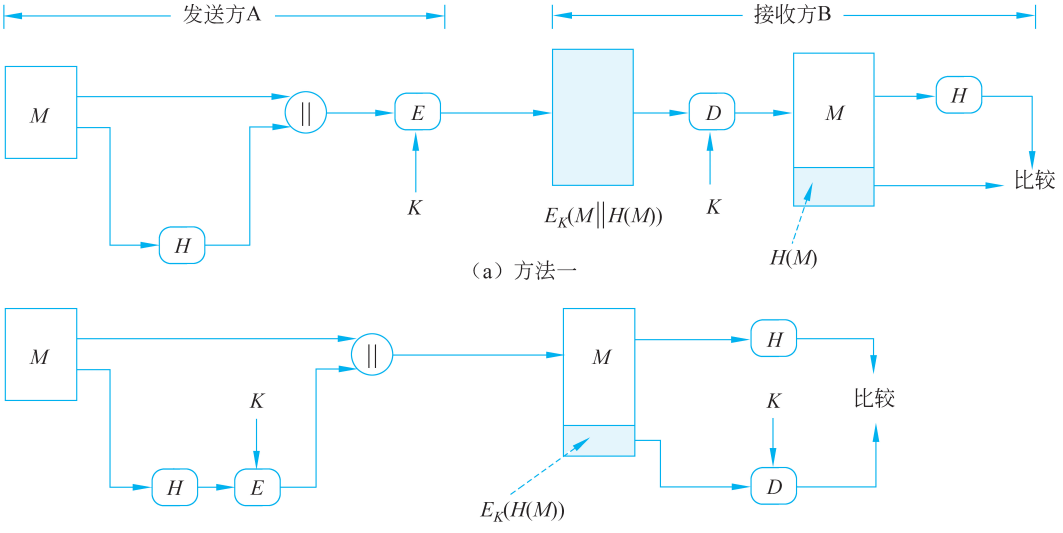


图 3-6　基于散列函数的消息鉴别

在图 3-6(a)中，消息发送方 A 首先计算明文消息 M 的散列值 $H(M)$并将 $H(M)$串接在 M 后，然后用对称加密算法对消息及附加在其后的散列值加密，将密文发送给对方。接收方 B 首先解密密文得到散列值 $H(M)$和明文消息 M，然后根据同样的散列算法计算散列值 $H'(M)$并验证 $H(M)=H'(M)$是否成立。如果成立，因为只有 A 和 B 共享密钥并且散列函数是一个单向函数，所以 B 可以确认消息一定是来自 A 且未被修改过。散列值提供了鉴别所需的结构或冗余，并且由于该方法是对整个消息和散列值加密，因此也提供了保密性。

图 3-6(b)用对称加密算法仅对散列码加密。$E_K(H(M))$是变长消息 M 和密钥 K 的函数，它产生定长的输出值，若攻击者不知道密钥，则他无法得出这个值。这个方案只

能提供鉴别，而无法提供保密。

近年来，人们对于利用散列函数设计 MAC 越来越感兴趣。这是因为利用对称加密算法产生 MAC 要对全部消息进行加密，运算速度较慢，而散列函数执行速度比对称分组加密要快。

散列函数并不是专为 MAC 而设计的，不依赖于密钥，因此它不能直接用于 MAC。目前，已经提出了许多方案将密钥加到现有的散列函数中。HMAC（Hash-based Message Authentication Code）是最受支持的方案，它是一种依赖密钥的单向散列函数，同时提供对数据的完整性和真实性的验证。HMAC 是 IP 安全必须实现的 MAC 方案，并且其他 Internet 协议（如 SSL）中也使用它。

RFC 2104 给出了 HMAC 的设计目标。

- 不必修改而直接使用现有的散列函数，即将散列函数看成“黑盒”，从而可以使用多种散列函数。
- 如果找到或需要更快或更安全的散列函数，应能很容易地替代原来嵌入的散列函数。
- 应保持散列函数的原有性能，不能过分降低其性能。
- 对密钥的使用和处理应较简单。
- 如果已知嵌入的散列函数的强度，则完全可以知道认证机制抗密码分析的强度。

图 3-7 给出了 HMAC 的总体结构。

其中：

- H——嵌入的散列函数（如 MD5、SHA-1、RIPEMD-160）。
- IV——作为散列函数输入的初始值。
- M——HMAC 的消息输入（包括由嵌入的散列函数定义的填充位）。
- Y_i——M 的第 i 个分组，$0 \leqslant i \leqslant L-1$。
- L——M 中的分组数。
- b——每一分组所含的位数。
- n——嵌入的散列函数所产生的散列码长。
- K——密钥（建议密钥长度 $\geqslant n$）。若密钥长度大于 b，则将密钥作为散列函数的输入来产生一个 n 位的密钥。
- K^+——为使 K 为 b 位长而在 K 左边填充 0 后所得的结果。
- ipad——内层填充，00110110（十六进制数 36）重复 $b/8$ 次的结果。
- opad——外层填充，01011100（十六进制数 5C）重复 $b/8$ 次的结果。

HMAC 可描述如下。

$$\mathrm{HMAC}(K,M)=H[(K^+ \oplus \mathrm{opad}) \| H[(K^+ \oplus \mathrm{ipad}) \| M]]$$

说明：

（1）在 K 左边填充 0，得到 b 位的 K^+（例如，若 K 是 160 位，$b=512$，则在 K 中加入 44 个全 0 字节 0x00）。

（2）K^+ 与 ipad 执行异或运算（逐位异或）产生 b 位的分组 S_i。

（3）将 M 附于 S_i 后。

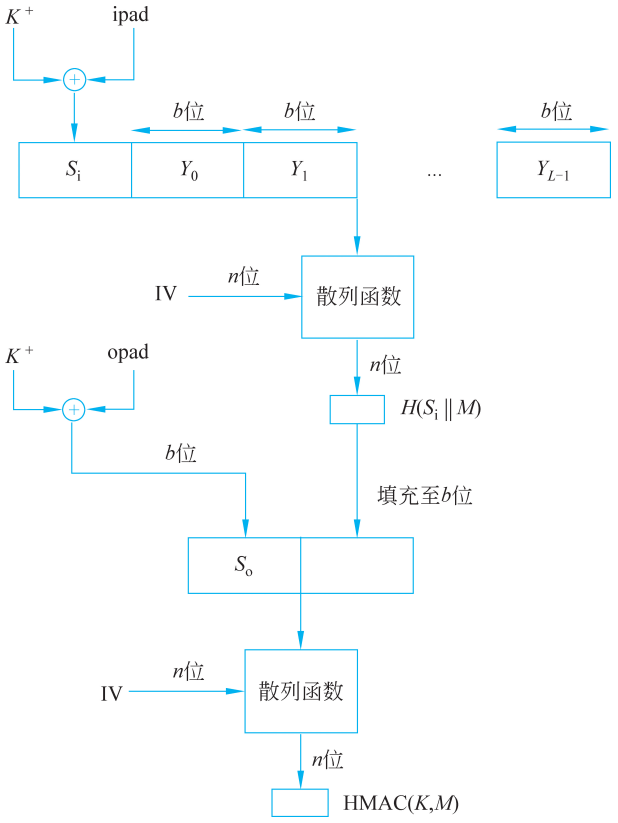


图 3-7　HMAC 的总体结构

(4) 将 H 作用于步骤(3)所得出的结果。

(5) K^+ 与 opad 执行异或运算(逐位异或)产生 b 位的分组 S_o。

(6) 将步骤(4)中的散列码附于 S_o 后。

(7) 将 H 作用于步骤(6)所得出的结果并输出该函数值。

注意，K 与 ipad 异或后，其信息位有一半发生了变化；同样，K 与 opad 异或后，其信息位的另一半也发生了变化。这样，通过将 S_i 与 S_o 传给散列算法中的压缩函数，可以从 K 伪随机地产生出两个密钥。

HMAC 多执行了 3 次散列压缩函数，但是对于长消息，HMAC 和嵌入的散列函数的执行时间大致相同。

3.3　数 字 签 名

教学课件

教学视频

在实际生活中，许多事情的处理需要人们手写签名。签名起到了鉴别、核准、负责等作用，表明签名者对文档内容的认可，并且产生某种承诺或法律上的效应。数字签名是手写签名的数字化形式，是公钥密码学发展过程中非常重要的概念之一，也是现代密码学非常重要的组成部分之一。数字签名的概念自 1976 年被提出时就受到了特别的关注。数

字签名已成为计算机网络不可缺少的一项安全技术，在商业、金融、军事等领域得到了广泛应用。很多国家对数字签名的使用颁布了相应的法案。美国于 2000 年通过的《电子签名全球与国内贸易法案》规定数字签名与手写签名具有同等法律效力，我国的《电子签名法》也规定可靠的数字签名与手写签名或印章有同等法律效力。

3.3.1　数字签名简介

1. 数字签名的必要性

消息鉴别通过验证消息完整性和真实性可以保护信息交换双方不受第三方的攻击，但它不能处理通信双方内部的相互攻击，这些攻击可以有多种形式。

例如，B 可以伪造一条消息并称该消息发自 A。此时，B 只需要产生一条消息，用 A 和 B 共享的密钥产生消息鉴别码，并且将消息鉴别码附于消息之后。因为 A 和 B 共享密钥，所以 A 无法证明自己没有发送过该消息。

又如，A 可以否认曾发送过某条消息。同样道理，因为 A 和 B 共享密钥，B 可以伪造消息，所以无法证明 A 确实发送过该消息。

在通信双方彼此不能完全信任对方的情况下，就需要用除消息鉴别外的其他方法来解决这些问题。数字签名是解决这个问题的最好方法，它的作用相当于手写签名。用户 A 发送消息给 B，B 通过验证附在消息上的 A 的签名，就可以确认消息是否确实来自 A。同时，因为消息上有 A 的签名，所以 A 在事后也无法抵赖所发送过的消息。因此，数字签名的基本目的是认证、核准和负责，防止相互欺骗和抵赖。数字签名在身份认证、数据完整性、不可否认性和匿名性等方面有着广泛的应用。

2. 数字签名的概念及其特征

数字签名在 ISO 7498-2 标准中被定义为“附加在数据单元上的一些数据，或是对数据单元所作的密码变换，这种数据和变换允许数据单元的接收者用于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（例如接收者）进行伪造”。

数字签名体制也叫数字签名方案，一般包含两个主要组成部分，即签名算法和验证算法。对消息 M 签名记为 $s=\mathrm{Sig}(m)$，而对签名 s 的验证可记为 $\mathrm{Ver}(s)\in\{0,1\}$。数字签名体制的形式化定义如下。

定义 3-1　一个数字签名体制是一个五元组 (M, A, K, S, V)，其中：

- M 是所有可能的消息的集合，即消息空间。
- A 是所有可能的签名组成的一个有限集，称为签名空间。
- K 是所有密钥组成的集合，称为密钥空间。
- S 是签名算法的集合，V 是验证算法的集合，满足：对任意 $k\in K$，有一个签名算法 Sig_k 和一个验证算法 Ver_k，使得对任意消息 $m\in M$，每一签名 $a\in A$，$\mathrm{Ver}_k(m, a)=1$，当且仅当 $a=\mathrm{Sig}_k(m)$。

在数字签名体制中，$a=\mathrm{Sig}_k(m)$ 表示使用密钥 k 对消息 m 签名，(m, a) 被称为一个消息-签名对。发送消息时，通常将签名附在消息后。

数字签名必须具有下列特征。

- 可验证性。信息接收方必须能够验证发送方的签名是否真实有效。

- 不可伪造性。除签名人外，任何人都不能伪造签名人的合法签名。
- 不可否认性。发送方在发送签名的消息后，无法否认发送的行为。
- 数据完整性。数字签名使得发送方能够对消息的完整性进行校验。换句话说，数字签名具有消息鉴别的功能。

根据这些特征，数字签名应满足下列条件。

- 签名必须是与消息相关的二进制位串。
- 签名必须使用发送方某些独有的信息，以防伪造和否认。
- 产生数字签名比较容易。
- 识别和验证签名比较容易。
- 伪造数字签名在计算上是不可行的。无论是从给定的数字签名伪造消息，还是从给定的消息伪造数字签名，在计算上都是不可行的。
- 保存数字签名的副本是可行的。

基于公钥密码算法和对称密码算法都可以获得数字签名，目前主要采用基于公钥密码算法的数字签名。在基于公钥密码的签名体制中，签名算法必须使用签名人的私钥，而验证算法则只使用签名人的公钥。因此，只有签名人才可能产生真实有效的签名，只要他的私钥是安全的。签名的有效性可以被任何人验证，因为签名人的公钥是公开可访问的。

3.3.2 基于公钥密码的数字签名原理

假定接收方已知发送方的公钥，则发送方可以用自己的私钥对整个消息或消息的散列码加密来产生数字签名，接收方用发送方的公钥对签名进行验证从而确认签名和消息的真实性，如图 3-8 所示。

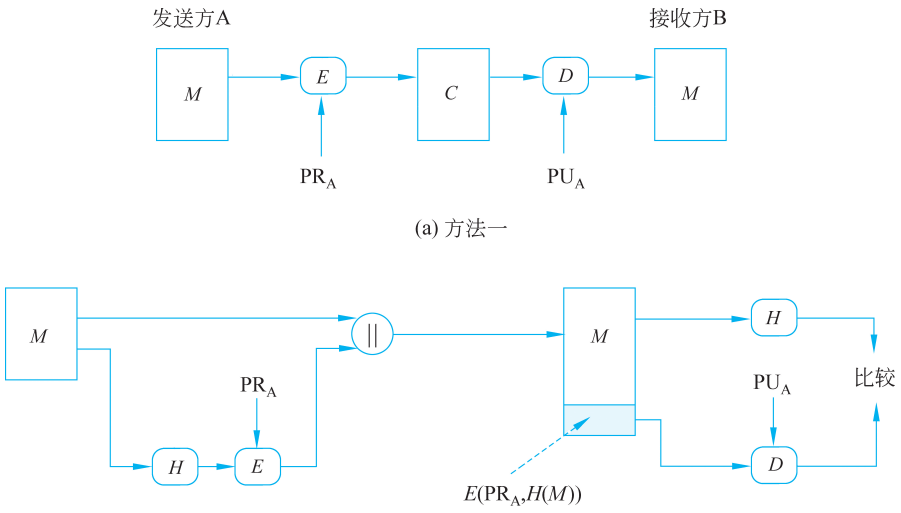
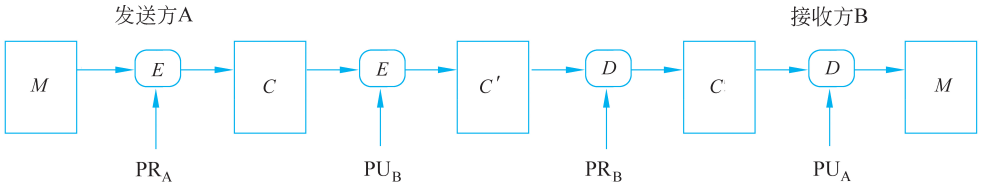


图 3-8 基于公钥密码的数字签名原理

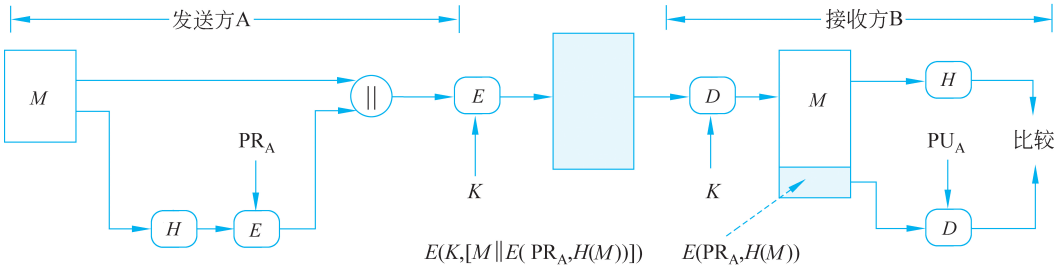
在实际应用中，考虑到效率，一般采用第二种方法，即发送方用自己的私钥对消息的

散列值加密来产生数字签名。假设发送方否认发送过消息 M，则接收方只需要提供 M 和签名 $E(PR_A, H(M))$。第三方可以用 A 的公钥解密签名得到 $H(M)$并与自己计算得到的散列值进行比较。如果相等，由于签名是 A 用自己的私钥加密的，则 M 肯定是 A 发送的，A 无法否认自己的发送行为。

如果发送方用接收方的公钥（公钥密码）和共享的密钥（对称密码）再对整个消息和签名加密，则可以获得保密性，如图 3-9 所示。



（a）方法一



（b）方法二

图 3-9　签名和保密

注意，这里是先进行签名，然后才执行外层的加密，这样在发生争执时，第三方可以查看消息及其签名。若先对消息加密，然后才对消息的密文签名，那么第三方必须知道解密密钥才能读取原始消息。但是签名若是在内层进行，那么接收方可以存储明文形式的消息及其签名，以备将来解决争执时使用。

签名的有效性依赖发送方私钥的安全性。如果发送方想否认以前曾发送过某条消息，那么他可以称其私钥已丢失或被盗用，其他人伪造了他的签名。可以通过在私钥的安全性方面进行控制来阻止或至少减少这种情况的发生。比较典型的做法是要求每条要签名的消息都包含一个时间戳（日期和时间），以及在密钥被泄露后应立即向管理中心报告。

3.3.3　数字签名算法

自数字签名的概念被提出后，人们设计了多种数字签名的算法。比较知名的有 RSA、ElGamal、Schnorr、DSS 等。

1. 基于 RSA 的数字签名

RSA 密码体制既可以用于加密，又可以用于签名。RSA 数字签名是最容易理解和实现的数字签名方案，其安全性基于大整数因子分解的困难性。

图 3-10 描述了基于 RSA 的数字签名方法。它要使用一个散列函数，散列函数的输入是要签名的消息，输出是定长的散列码。发送方用其私钥和 RSA 算法对该散列码加密形成签名，然后发送消息及其签名。接收方收到消息后计算散列码并用发送方的公钥对签名解密得到发送方计算的散列码，如果两个散列码相同，则认为签名是有效的。因为只有发送方拥有私钥，所以只有发送方能够产生有效的签名。

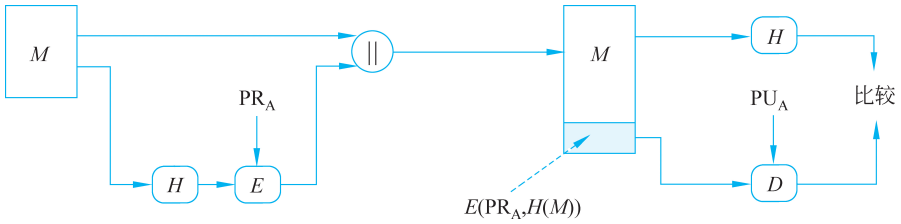


图 3-10 RSA 数字签名方法

2. 数字签名标准

NIST 于 1991 年提出了一个联邦数字签名标准，称为数字签名标准（DSS）。DSS 使用安全散列算法（SHA）给出了一种新的数字签名方法，即数字签名算法（DSA）。与 RSA 不同，DSS 是一种公钥方法，但只提供数字签名功能，不能用于加密或密钥分配。

DSS 数字签名方法如图 3-11 所示。

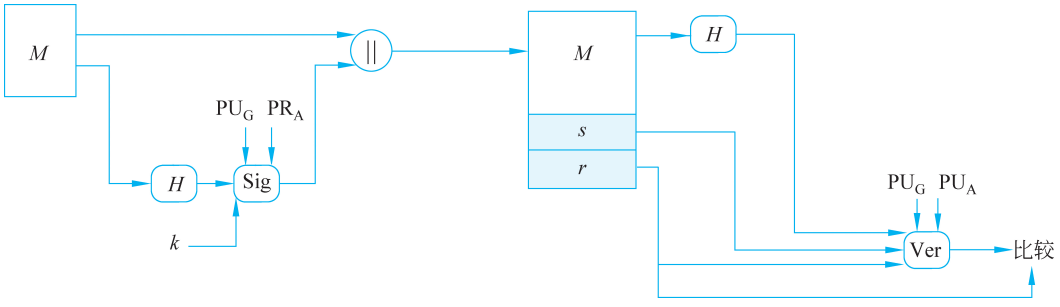


图 3-11 DSS 数字签名方法

DSS 方法也使用散列函数，它产生的散列码和为此次签名而产生的随机数 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）。

接收方对接收到的消息产生散列码，这个散列码和签名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 PU_A，若验证函数的输出等于签名中的 r 成分，则签名是有效的。签名函数保证只有拥有私钥的发送方才能产生有效签名。

DSA 安全性基于计算离散对数的困难性，并且起源于 ElGamal 和 Schnorr 提出的数字签名方法。

图 3-12 归纳总结了 DSA 算法。公钥由 3 个参数 p、q、g 组成并为一组用户所共有。首先选择一个 160 位的素数 q，然后选择一个长度为 512～1024 的素数 p，并且使得 q 是 $p-1$ 的素因子；最后选择形为 $h^{(p-1)/q} \bmod p$ 的 g，其中 h 是 $1 \sim p-1$ 的整数且 g 大于 1。

全局公钥组成

p为素数，其中$2^{L-1}<p<2^{L}$，$512\leqslant L\leqslant 1024$且$L$是64的倍数，即$L$的位长为512～1024并且其增量为64位。$q$为$p-1$的素因子，其中$2^{159}<q<2^{160}$，即位长为160位。$g=h^{(p-1)/q} \bmod p$，其中$h$是满足$1<h<p-1$并且$h^{(p-1)/q} \bmod p>1$的任何整数

用户的私钥

x为随机或伪随机整数且$0<x<q$

用户的公钥

$y=g^{x} \bmod p$

与用户每条消息相关的秘密值

k等于随机或伪随机整数且$0<k<q$

签名

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

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

签名$=(r,s)$

验证

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

$u_1=[H(M')w] \bmod q$

$u_2=(r')w \bmod q$

$v=[(g^{u_1}y^{u_2}) \bmod p] \bmod q$

检验：$v=r'$

M——要签名的消息
$H(M)$——使用SHA-1求得的M的散列码
M'、r'、s'——接收到的M、r、s

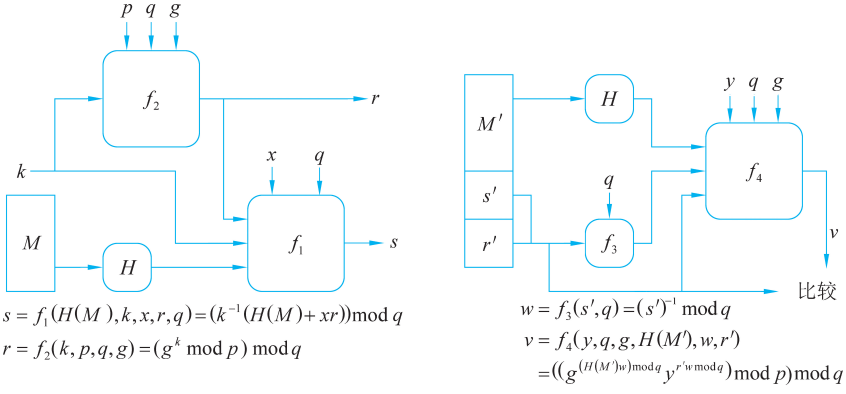
图 3-12　DSA 算法

选定这些参数后，每个用户选择私钥并产生公钥。私钥 x 必须是随机或伪随机选择的素数，取值区间是[1，$q-1$]；公钥则根据公式 $y=g^{x} \bmod p$ 计算得到。由给定的 x 计算 y 比较简单，而由给定的 y 确定 x 则在计算上是不可行的，因为这就是求 y 的以 g 为底的模 p 的离散对数，而求离散对数是困难的。

假设要对消息 M 进行签名。发送方需要计算两个参数 r 和 s，它们是公钥（p、q、g）、用户私钥 x、消息的散列码 $H(M)$和附加整数 k 的函数，其中 k 是随机或伪随机产生的（$0<k<q$），且对每次签名是唯一的。

为了对签名进行验证，接收方计算值 v，它是公钥（p、q、g）、发送方公钥、接收到的消息的散列码的函数。若 v 与签名中的 r' 相同，则签名是有效的。

图 3-13 描述了上述签名和验证函数。



（a）签名　　（b）验证

图 3-13　DSS 签名和验证函数

DSA算法有这样一个特点：接收端的验证依赖r，但r却根本不依赖消息，它是k和全局公钥的函数。$k \bmod p$的乘法逆元传给函数的输入还包含消息的散列码和用户私钥，函数的这种结构使接收方可利用其收到的消息、签名、自己的公钥以及全局公钥来恢复r。

由于求离散对数的困难性，因此攻击者从r恢复出k或从s恢复出x都是不可行的。

思 考 题

1. 消息鉴别主要用于对抗哪些类型的攻击？
2. 根据鉴别符的生成方式，鉴别函数可分为哪几类？各自具有什么特点？
3. 什么是消息鉴别码（MAC）？其实现的基本原理是什么？
4. 散列函数应该具有哪些安全特性？
5. 简述SHA-512算法处理消息和输出摘要的过程。
6. 什么是数字签名？数字签名具有哪些特征？
7. 为什么说消息鉴别无法处理内部矛盾（如发送方否认发送过某消息或者接收方否认接收到某消息），而数字签名可以？
8. 简述基于公钥密码的数字签名原理。