# IB00168 网络空间安全基础


## 授课教师：姜婧妍

jiangjingyan@sztu.edu.cn


2023年

# 第三章 数字签名与身份认证


**授课教师：姜婧妍**

jiangjingyan@sztu.edu.cn


**2023年**

# 2.4 散列函数和消息认证码

## 2.4.1散列函数

- **散列函数又叫做散列算法，是一种将任意长度的消息映射到某一固定长度消息摘要（散列值或哈希值）的函数。**消息摘要相当于是消息的“指纹”，用来防止对消息的非法篡改。散列函数也常常被称为**Hash函数(哈希函数)**。
- 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

$$H=H(M)$$

- 且H(M)长度固定。假设H安全，发送方将散列值H附于消息M后发送；接收方通过重新计算散列值H'，并比较H=H'是否成立，可以**验证该消息的完整性**。

# 散列函数的安全性

- 密码学中的散列函数必须满足一定的安全特征，主要包括三个方面：单向性、强抗碰撞性和弱抗碰撞性。
- **A. 单向性**是指对任意给定的散列码H，找到满足H(x)=H的x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，但是由散列值H(M)计算M是不可行的。
- **B. 强抗碰撞性**是指散列函数满足下列四个条件。
  (1) 散列函数H的输入是任意长度的消息M;
  (2) 散列函数H的输出是固定长度的数值;
  (3) 给定H和M，计算H(M)是容易的;

# 散列函数的安全性

(4) 给定散列函数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。（如果有两个消息$M_1$和$M_2$，$M_1 \neq M_2$但是$H(M_1)=H(M_2)$，则称$M_1$和$M_2$是碰撞的。）

- **C．弱抗碰撞性**的散列函数满足强抗碰撞散列函数的前三个条件，但具有一个不同的条件：给定H和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。

# MD5和SHA

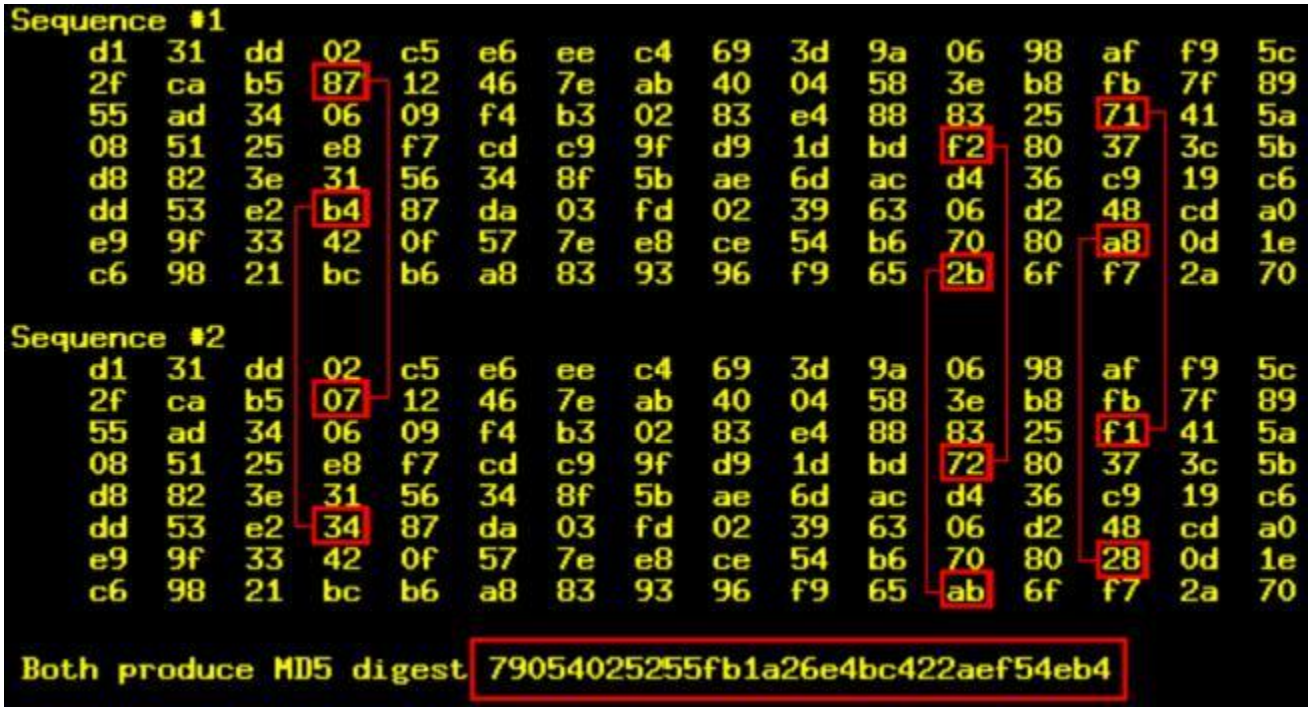
- 目前使用最多的两种散列函数是MD5和SHA序列函数。

- MD5的散列码长度为128比特。Ubuntu Linux发行版本内置了MD5算法: md5sum filename

- SHA序列函数是美国联邦政府的标准，如SHA-1散列码长度为160比特，SHA-2散列码长度为256、384和512位。
- SHA-1算法见教材第30页的内容。

# MD5的碰撞问题

- MD5的散列码长度为128比特， 已经被证明是不安全的。2004年，山东大学的王小云（现为科学院院士，清华大学和山东大学的教授）第一次发现MD5算法存在碰撞的可能，并给出了实例。



# SHA-1的碰撞问题

- On February 23, 2017, CWI Amsterdam and Google announced they had performed a collision attack against SHA-1. They had given 2 different PDF files with the same SHA-1 outputs.
- https://shattered.io/


**目前（2020年10月）：**
**① 对于安全要求较高的应用，不能使用MD5**
**② 找到SHA-1的碰撞是较困难的； SHA-2是安全的**

# 2.4.2 消息鉴别码

- 完整性是安全的基本要求之一，是信息安全的三要素之一。恶意篡改、信道的偶发干扰和故障都将破坏消息的完整性。

- 另外，攻击者还可能实施假冒攻击，破坏信息的真实性。

- **保障消息完整性和真实性的重要手段是消息鉴别技术。**

# 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完整性和真实性验证的过程。
- **过程：用鉴别函数产生一个鉴别符，根据收发端的鉴别符是否一致，对消息进行鉴别。**

- 检验内容应包括：

**(1)证实报文的源**

**(2)报文内容是否曾受到偶然的或有意的篡改**

**(3)报文的序号和时间栏**

# 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完整性和真实性验证的过程。
- **过程：用鉴别函数产生一个鉴别符，根据收发端的鉴别符是否一致，对消息进行鉴别。**
- 根据鉴别符的生成方式，鉴别函数可以分为如下三类：

① 基于消息加密：以整个消息的密文作为鉴别符。

② 基于**消息鉴别码(MAC, Message identification code)**：利用公开函数和密钥产生一个较短的定长值作为鉴别符，并与消息一同发送给接收方，实现对消息的验证。

③ 基于散列函数：利用公开的散列函数将任意长的消息映射为定长的散列值，并以该散列值作为鉴别符。

# 常用的鉴别方法：由Hash函数导出MAC

- 加密整个消息的鉴别方法是昂贵的，对于大量消息的鉴别是不合适的，特别是对于不需要保密的图像、声音、视频等媒体信息，对存储和传输都带来巨大的资源需求，如果加密整个消息，所需的计算资源是巨大的，在某些情况下甚至是不可行的。

- 实用的鉴别方法所需的计算和存储资源不应该太大。因此，**基于Hash函数导出MAC的方法成为了主流**。

# 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**密码校验和(cryptographic check-sum)**，其实现鉴别的原理是：利用公开函数和密钥生成一个固定大小的小数据块，即MAC，并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，MAC可以在不安全的信道中传输，因为MAC的生成需要密钥。

# 2．基于MAC的鉴别

消息认证码是一种确认完整性并进行认证的技术，简称MAC。消息认证码的输入包括任意长度的消息和一个发送者与接收者之间共享的密钥，它可以输出固定长度的数据，这个数据称为MAC值。要计算MAC值必须持有共享密钥，没有共享密钥的人就无法计算MAC值，消息验证码正是利用这一性质来完成认证的。我们可以暂且将“消息认证码”理解为是“一种特殊的与密钥相关的单项散列函数”。

**MAC = 散列 + 对称密钥**

# MAC鉴别原理

假设以下情景中有发送者Alice和接收者Bob。

（1）发送者Alice和接收者Bob事先共享密钥；

（2）发送者Alice使用共享密钥根据消息计算MAC值；

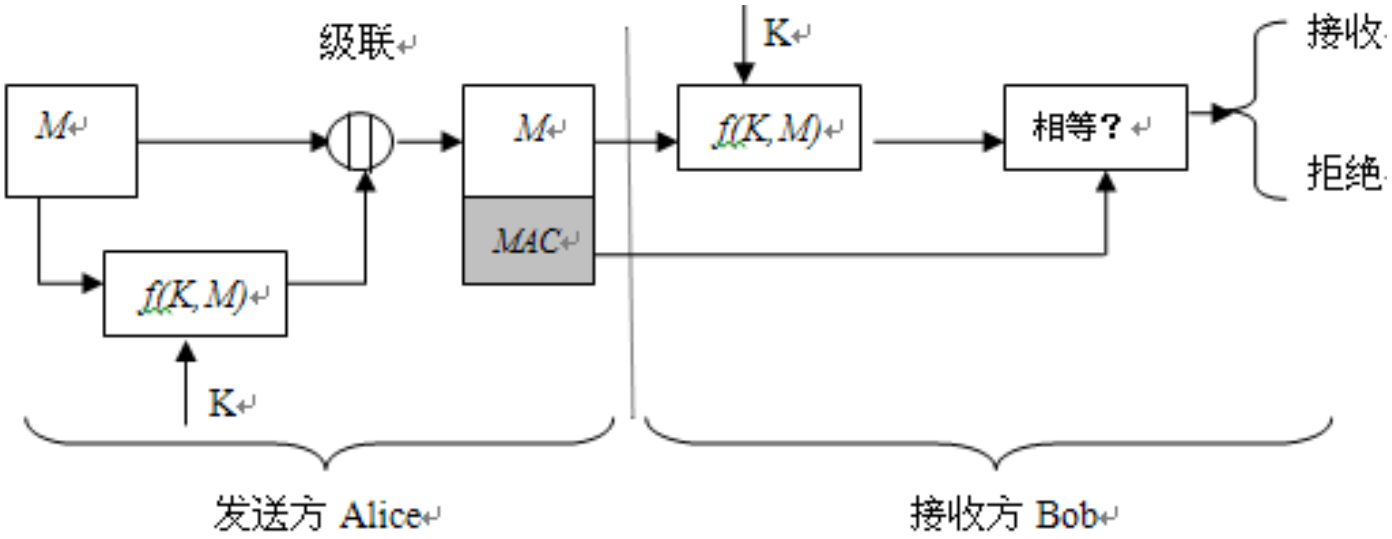
（3）发送者Alice将消息和MAC值发送给接收者Bob；

（4）接收者Bob根据自己收到的消息计算MAC值，并将其与从Alice收到的MAC值进行对比；

（5）如果两个MAC值一致，Bob就可以确认刚才收到的消息来自Alice，认证成功，如果不一致，则认证失败，可判断消息不是来自于Alice。

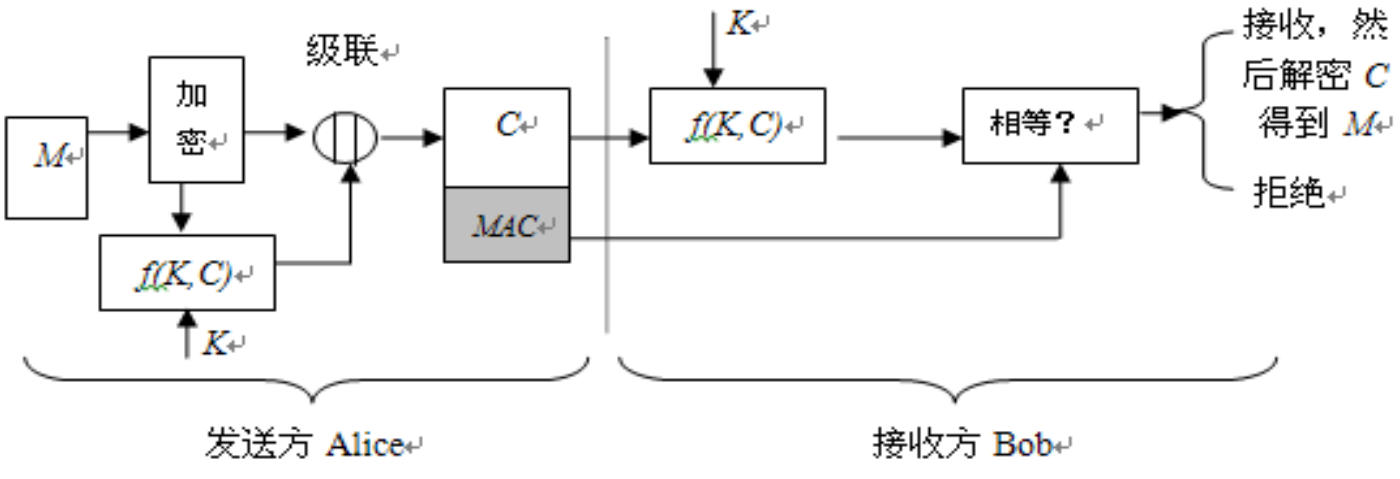
# MAC的使用步骤

**认证码被附加到消息后以M ‖ MAC方式一并发送，接收方通过重新计算MAC以实现对M的认证，如图所示。**

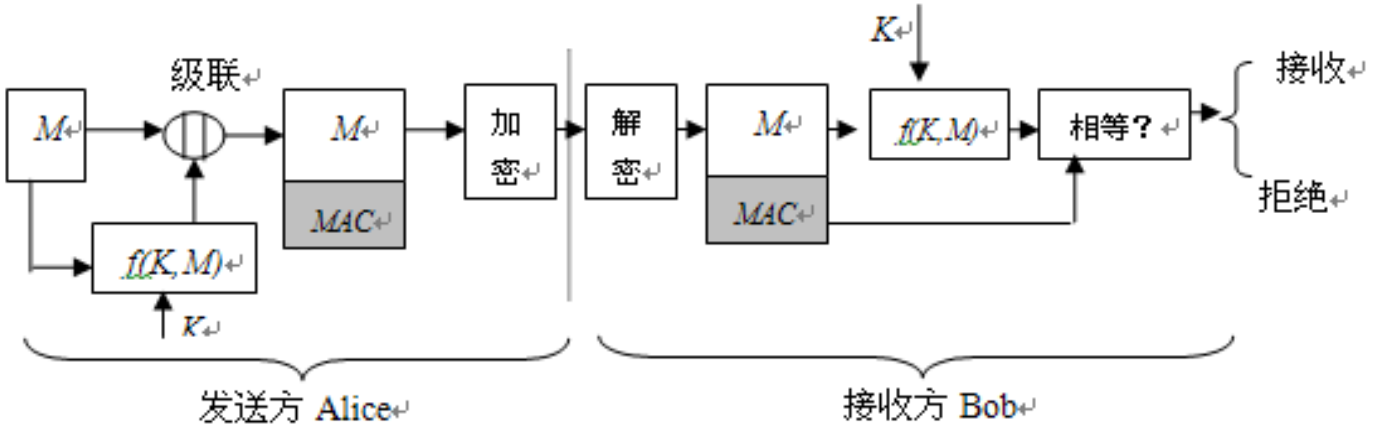



MAC 认证方式示意图

**提问：把f(K,M)改成Hash（M），安全吗？**




MAC 认证方式示意图——只加密 M




MAC 认证方式示意图——加密整个消息

## 采用HASH的消息认证--把上述三幅图f(K,M)变为HASH（M）

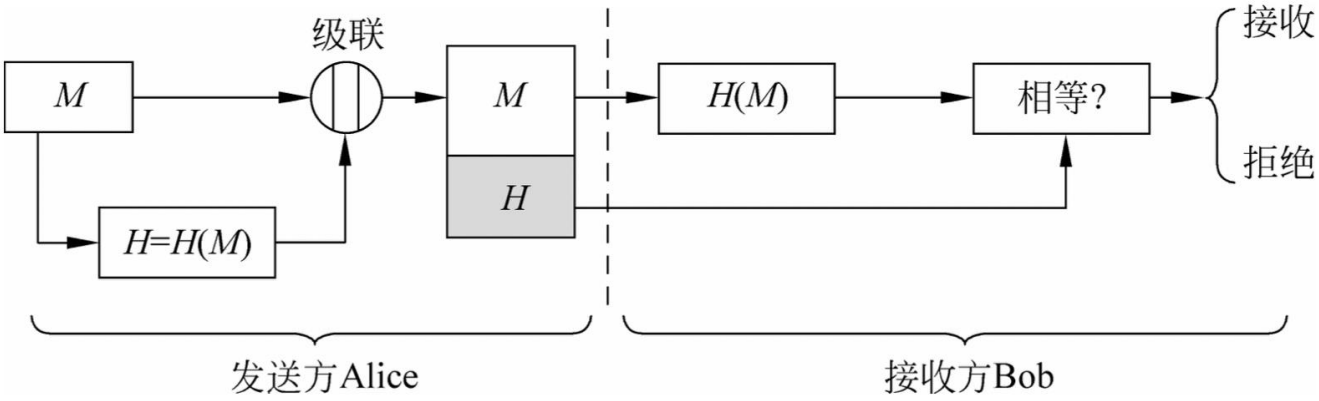



图 3-7　Hash 函数使用方式示意

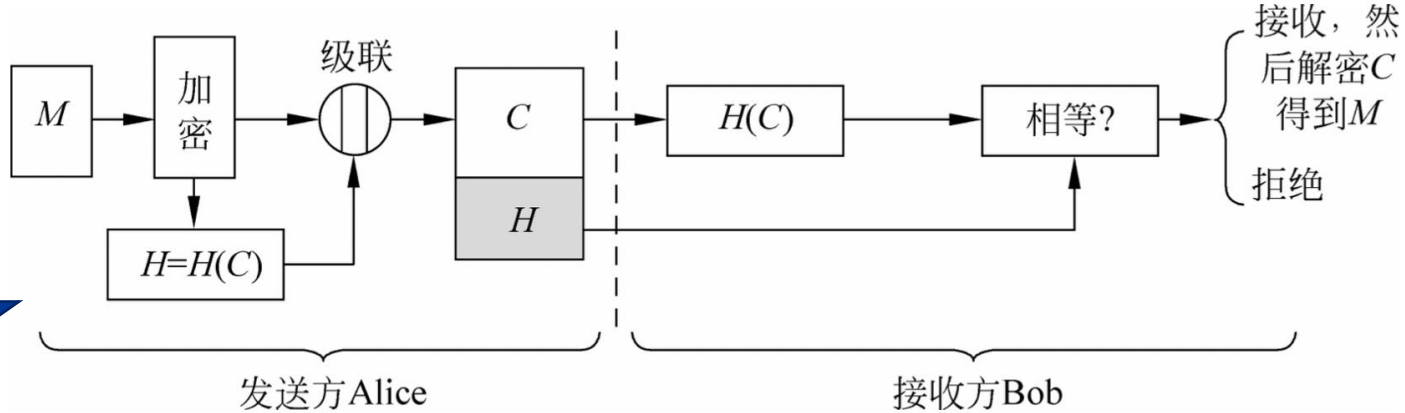



图 3-8　Hash 函数使用方式示意——只加密 $M$

# MAC鉴别原理

- 图2-7所示的过程仅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被攻击的弱点要少。

- **与加密相比，MAC算法更不易被攻破。**

# MAC鉴别原理

- 图2-7所示的过程仅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被攻击的弱点要少。

- **与加密相比，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(密文分组链接消息鉴别码)**，建立在DES之上，是使用最广泛的MAC算法之一，也是ANSI的一个标准。

- 数据鉴别算法采用DES运算的密文块链接(CBC)方式，参见图2-8。

# 采用DES运算的密文块链接(CBC)（略）

- 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$D_1$，$D_2$，$d$，$D_N$，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)的过程为：

$$
\begin{aligned}
O_0 &= \boldsymbol{IV} \\
O_1 &= E_K(D_1 \oplus O_0) \\
O_2 &= E_K(D_2 \oplus O_1) \\
O_3 &= E_K(D_3 \oplus O_2) \\
&\vdots \\
O_N &= E_K(D_N \oplus O_{N-1})
\end{aligned}
$$

# 基于DES的MAC （略）

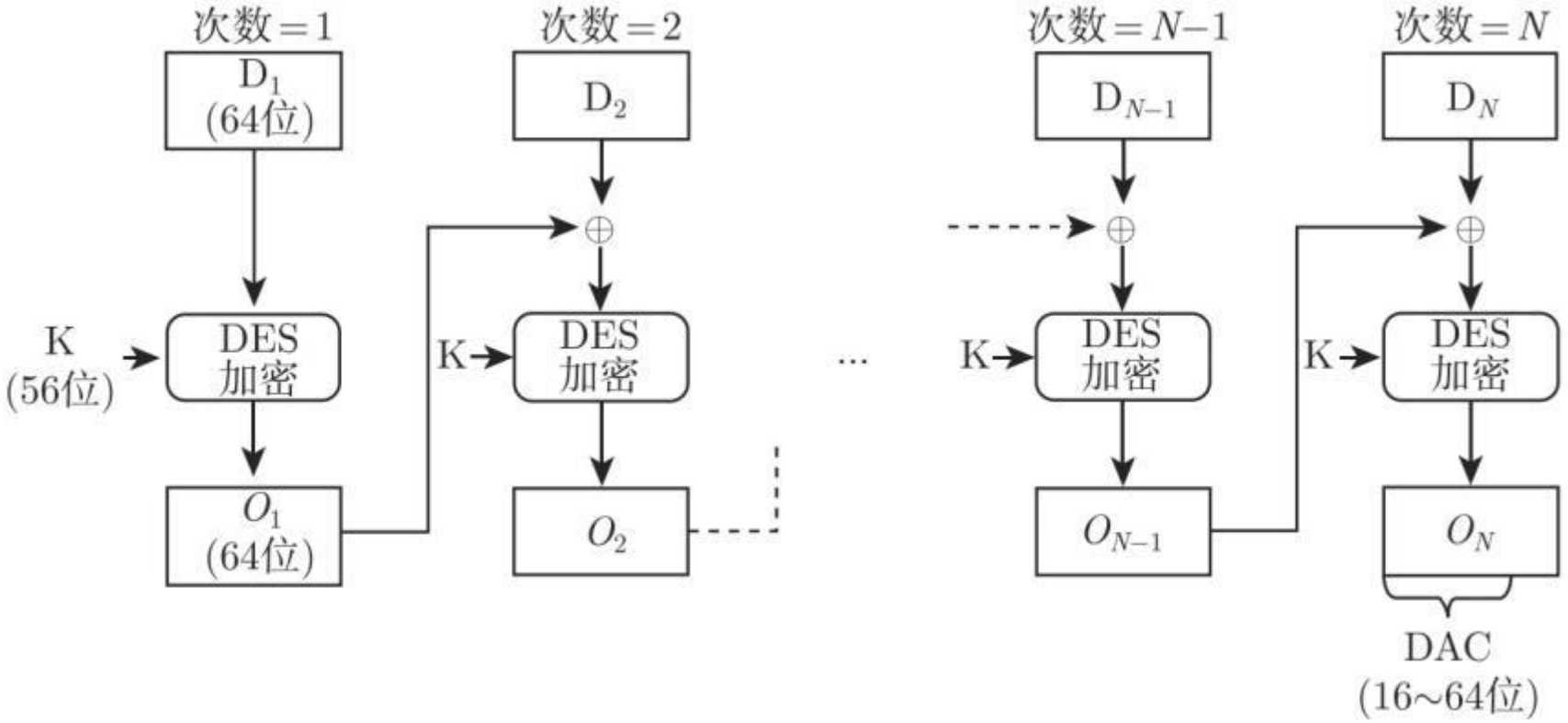



图2-8 基于DES的数据鉴别算法

- 其中，DAC可以取整个块$O_N$，也可以取其最左边的M位，其中16≤M≤64。

# 3. 基于散列函数的鉴别

- 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是最受支持的方案，它是一种依赖于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安全里必须实现的MAC方案，并且其他Internet协议中(如SSL)也使用了HMAC。

- HMAC可描述如下：

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

（略）

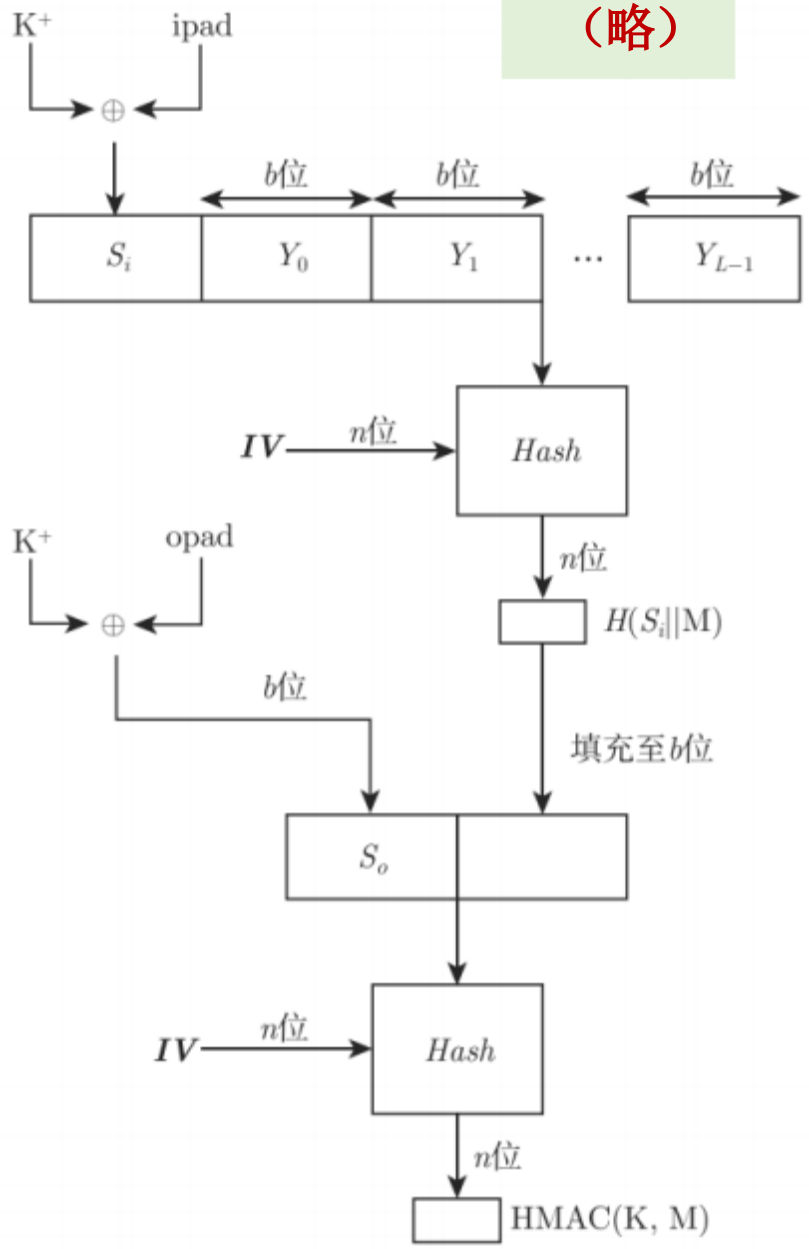



图29 HMAC的总体结构

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（包括由嵌入散列函数定义的填充位）；

L——M中的分组数；

Yi——M的第i个分组，0≤i≤L-1；

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≥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位长度而在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的过程（略）

(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0字节）；

(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；

(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（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。

# 2.5 数字签名

- 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，并产生某种承诺或法律上的效力。
- 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最重要的概念之一，也是现代密码学的一个最重要的组成部分之一。
- 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，得到了广泛的应用。各国对数字签名的使用颁布了相应的法案。
- 美国2000年通过的《电子签名全球与国内贸易法案》就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。

# 2.5.1 数字签名简介

- 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，但是它不能处理通信双方内部的相互的攻击，这些攻击可以有多种形式。
- 数字签名是解决通信双方内部相互攻击的最好方法，它的作用相当于手写签名。用户A发送消息给B，B只要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于A。同时，因为消息上有A的签名，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。

# 数字签名的概念及其特征

- 数字签名在ISO7498-2标准中定义为：
- “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（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=Sig(m)，而对签名s的验证可记为Ver(s)∈{0，1}。

# 数字签名体制的形式化定义

- 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$a \in A$，$Ver_k(m, a)=1$，当且仅当$a=Sig_k(m)$时。

# 数字签名必须具有的4个特征

① **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。

② **不可伪造性**。除了签名人之外，任何人不能伪造签名人的合法签名。

③ **不可否认性**。发送方在发送签名的消息后，无法抵赖发送的行为；接收方在收到消息后，也无法否认接收的行为。

④ **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。即数字签名具有消息鉴别的功能。

# 数字签名应满足的6个条件

① 签名必须是与消息相关的二进制位串。

② 签名必须使用发送方某些独有的信息，以防伪造和否认。

③ 产生数字签名比较容易。

④ 识别和验证签名比较容易。

⑤ 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。

⑥ 保存数字签名的拷贝是可行的。

**※基于公钥密码算法和对称密码算法都可以获得数字签名，目前主要是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。签名的有效性能被任何人验证，因为签名人的公钥是公开可访问的。

# 2.5.2 基于公钥密码的数字签名原理

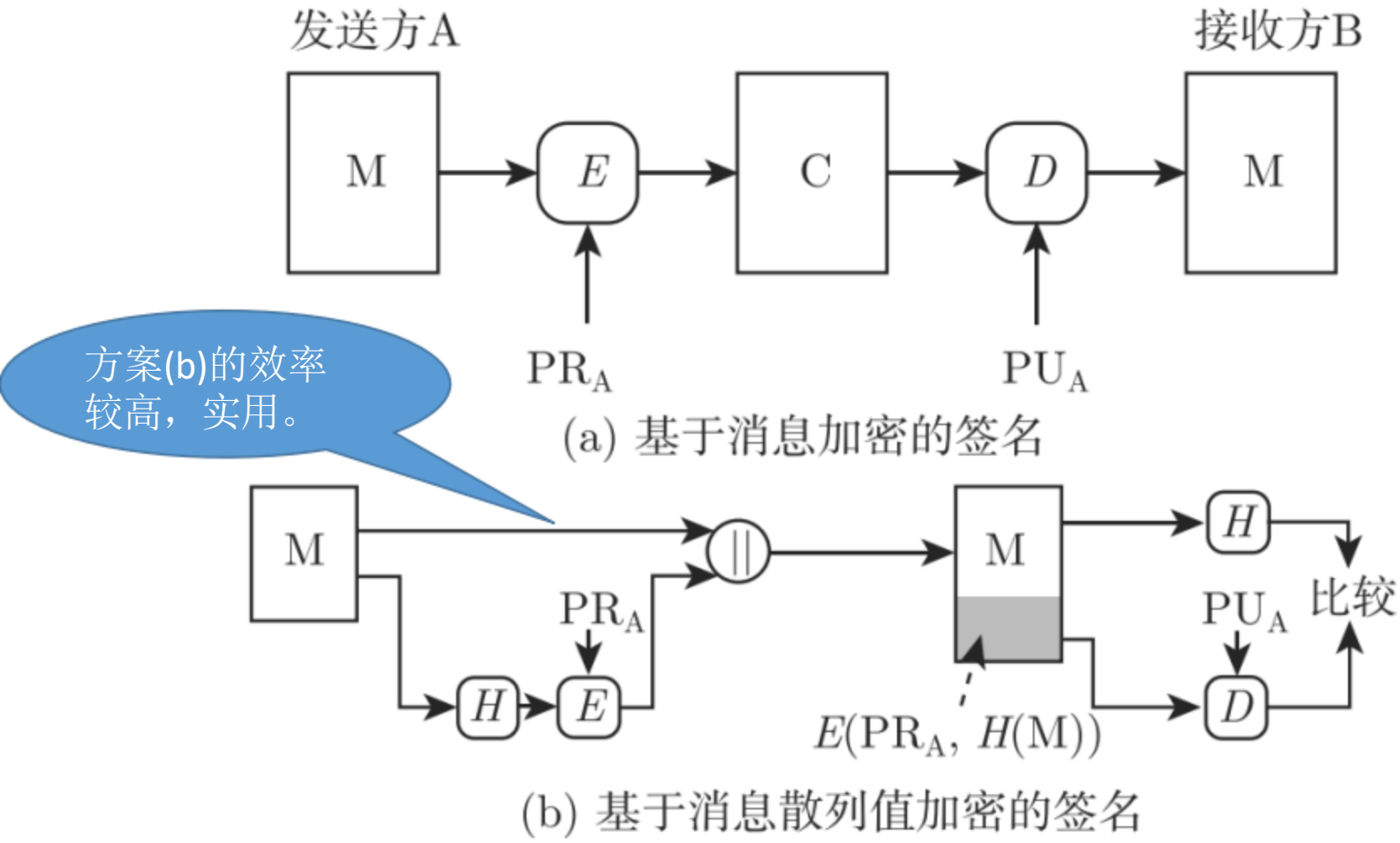



**图2-10 基于公钥密码的数字签名原理**

# 2.5.3 数字签名算法

## 1．基于RSA的数字签名

- RSA数字签名方法要使用一个散列函数H，散列函数的输入是要签名的消息，输出是定长的散列码。

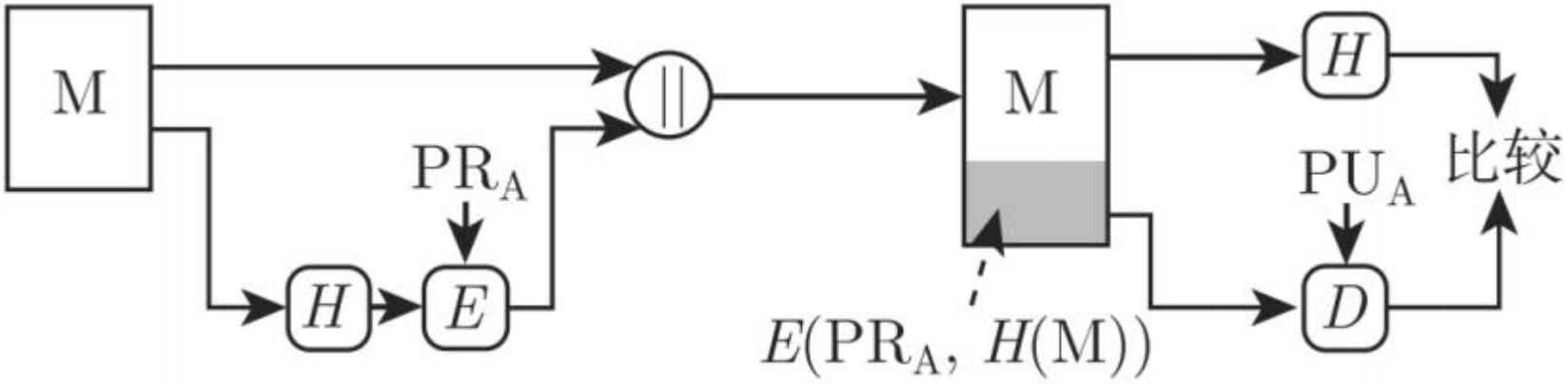



**图2-11 RSA数字签名**

- 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，因此只有发送方能够产生有效的签名。

# 2．数字签名标准DSS

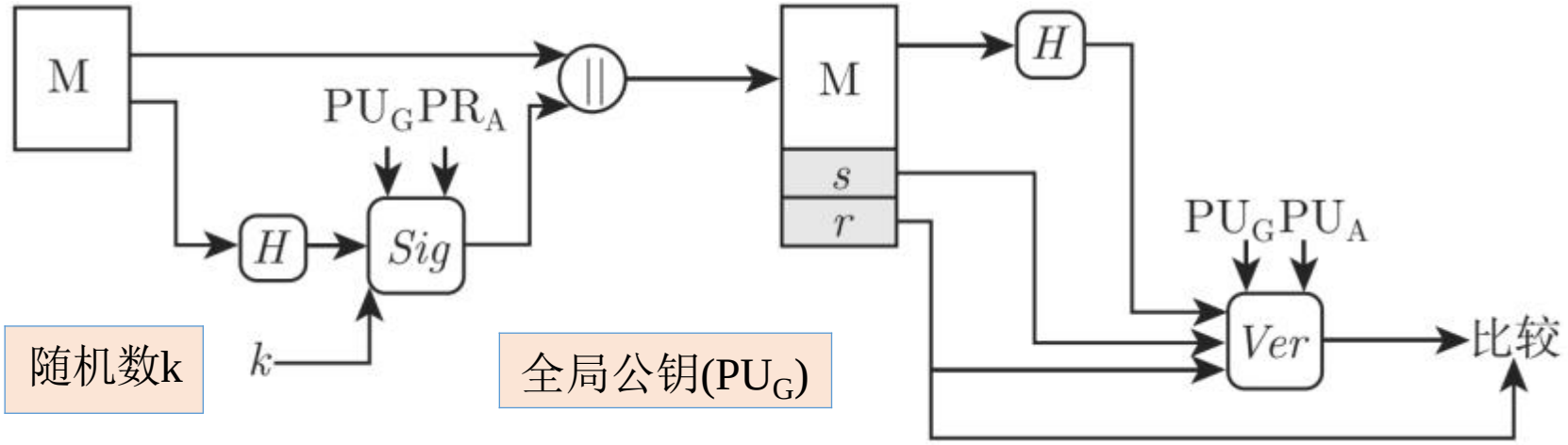



**图2-12 DSS数字签名方法**

- 美国国家标准与技术研究所(NIST)于1991年提出了一个联邦数字签名标准，称之为数字签名标准(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2-12所示。

# DSS签名过程（略）

- 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，签名函数依赖于发送方的私钥($PR_A$)和一组参数，这些参数为一组通信伙伴所共有，我们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，验证函数依赖于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 ， 并起源于EIGamal和Schnorr提出的数字签名方法。

# DSA算法（略）

**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私密值**

$k$为随机或伪随机整数且$0<k<\mathrm{q}$

**签名**

$r=(g^{r} \bmod p) \bmod q$
$s=[k^{-1}(H(\mathrm{M})+xr)] \bmod q$
签名$=(r,s)$

**验证**

$w=(s')^{-1} \bmod q$
$u_1=[(H(\mathrm{M}')w)] \bmod q$
$u_2=(r')w \bmod q$
$v=[(g^{u_1}y^{u_2}) \bmod p] \bmod q$
检验：$v=r'$

M —— 要签名的消息
$H(\mathrm{M})$——使用SHA－1求得M的三列码
$\mathrm{M}',r',s'$——接收到的$\mathrm{M},r,s$

**图2-13 数字签名算法DSA**

# DSA算法的说明（略）

- 公钥由三个参数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为1～p-1的整数且g大于1。

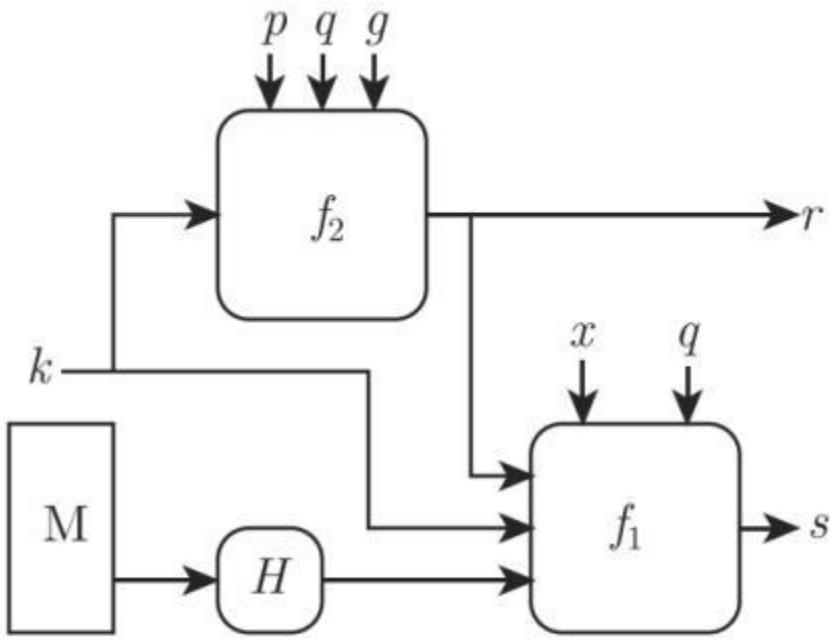
- 选定这些参数后，每个用户选择私钥并产生公钥。私钥x必须是随机或伪随机选择的素数，取值区间是[1, q-1]。公钥则根据公式$y = g^x \bmod p$计算得到。由给定的x计算y比较简单，而由给定的y确定x则在计算上是不可行的，因为这就是求y的以g为底的模p的离散对数，而求离散对数是困难的。

# DSA算法的说明（续）（略）

- 假设要对消息M进行签名。发送方需计算两个参数r和s，它是公钥(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k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2-14描述了上述签名和验证函数。

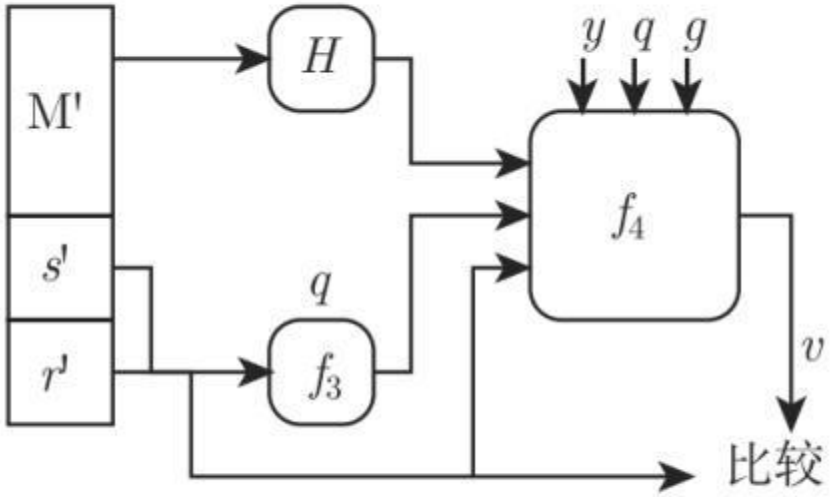
# DSA算法的说明（续） （略）




$$s = f_1(H(\mathrm{M}), k, x, r, q) = (k^{-1}(H(\mathrm{M}) + xr)) \bmod q$$

$$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$$

(a) 签名




$$w = f_3(s', q) = (s')^{-1} \bmod q$$

$$v = f_4(y, q, g, H(\mathrm{M}'), w, r')$$
$$= ((g^{(H(\mathrm{M}')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$$

(b) 验证

**图2-14 DSS签名和验证函数**

# DSA算法的说明（续）（略）

- DSA算法有这样一个特点，接收端的验证依赖于r，但是r却根本不依赖于消息，它是k和全局公钥的函数。k(mod q) 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、它的公钥以及全局公钥来恢复r。

- 由于求离散对数的困难性，攻击者从r恢复出k或从s恢复出x都是不可行的。