第五章

消息认证与数字署名

问题的提出

通信威胁

1. 泄露：把消息内容公布给任何人或没有正当密钥的进程。
2. 伪造：从一种假冒信息源向网络中插入消息。
3. 内容修改：消息内容被插入删除变换修改。
4. 顺序修改：插入删除或重组消息序列。
5. 时间修改：消息延迟或重放。
6. 否定：接受者否定收到消息，发送者否定发送过消息。

回忆与总结

- 回忆
  - 对称密码算法
    - 运算速度快、密钥短、多种用途(随机数产生、Hash函数)、历史悠久
    - 密钥管理困难(分发、更换)
  - 非对称密码算法
    - 只需保管私钥、能够相当长的时间保持不变、需要的数目较小
    - 运算速度慢、密钥尺寸大、历史短

信息安全的需求

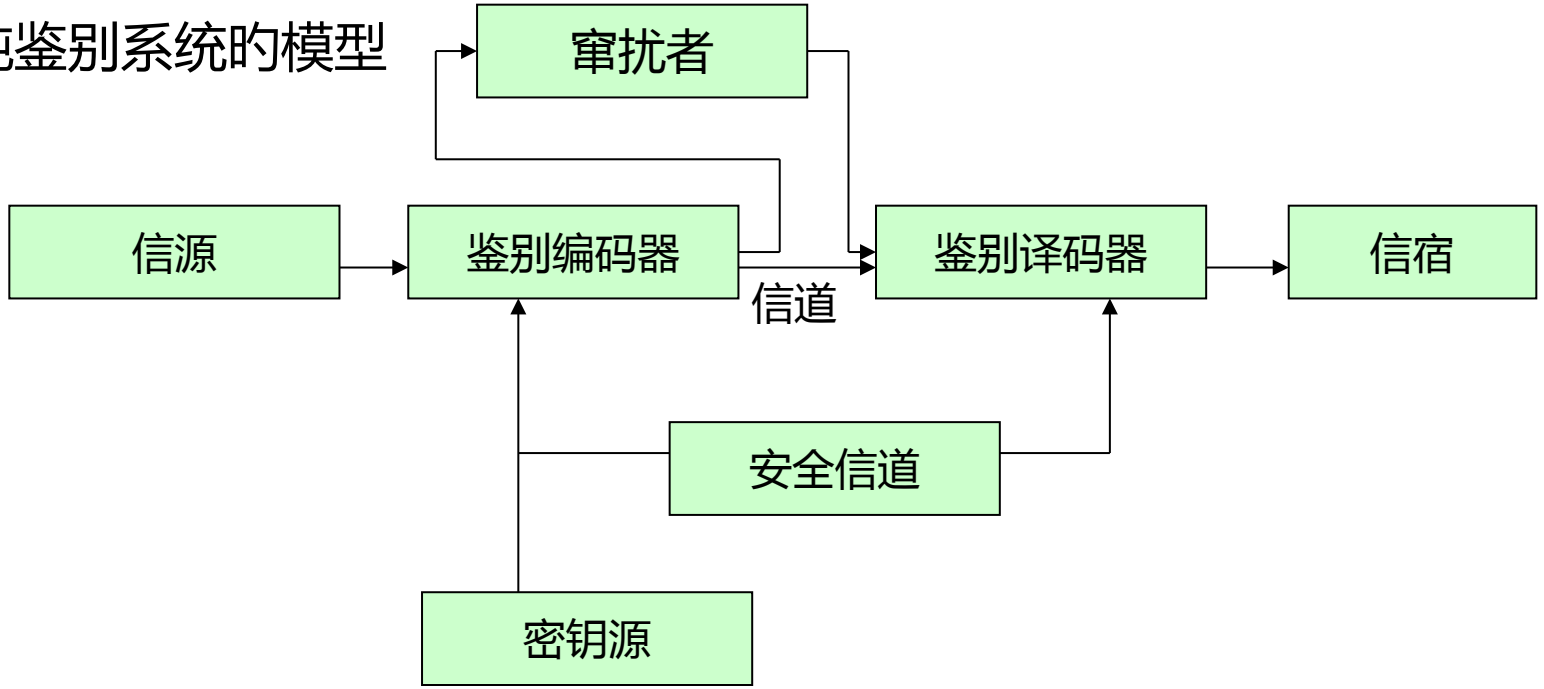
- 保密性 (Confidentiality)
- 完整性 (Integrity)
  - 数据完整性，未被未授权篡改或者损坏
  - 系统完整性，系统未被非授权操纵，按既定的功能运营
- 可用性 (Availability)
- 鉴别 (Authenticity)：实体身份的鉴别，合用于顾客、进程、系统、信息等
- 不可否定性 (Non-repudiation)：预防源点或终点的抵赖
- ……

讨论议题

- 定义
  - 消息鉴别 (Message Authentication)：是一种证明收到的消息来自可信的源点且未被篡改的过程。
    - 散列函数 (Hash Functions)：一种散列函数以一种变长的报文作为输入，并产生一种定长的散列码，有时也称报文摘要，作为输出。
  - 数字署名(Digital Signature)
    - 是一种预防源点或终点抵赖的鉴别技术。

5.1 消息认证

- 鉴别的目的
  - 鉴别的主要目的有二：
    - 第一，验证信息的发送者是真正的，而不是冒充的，此为信源辨认；
    - 第二，验证信息的完整性，在传送或存储过程中未被篡改，重放或延迟等。
- 鉴别模型
  - 一种单纯鉴别系统的模型



5.1 消息认证

❖ 鉴别系统的构成

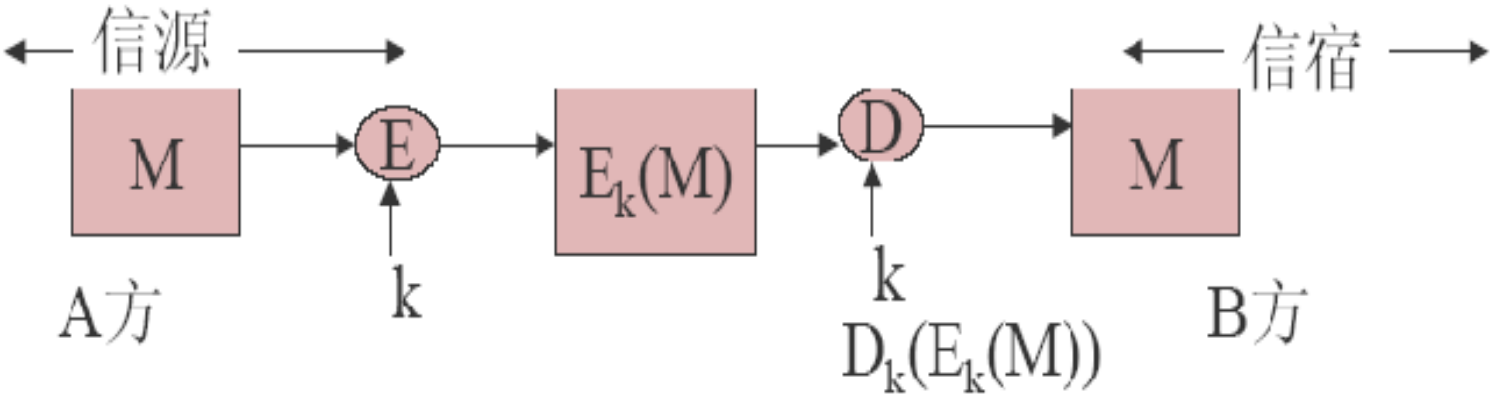
- 鉴别编码器和鉴别译码器可抽象为鉴别函数。一种安全的鉴别系统，需满足
  - 1) 接受者能够检验和证明消息的正当性、真实性和完整性
  - 2) 消息的发送者和接受者不能抵赖
  - 3) 除了正当的消息发送者，其他人不能伪造正当的消息
- 鉴别系统首先要选好恰当的鉴别函数，该函数产生一种鉴别标识，然后在此基础上，给出合理的鉴别协议(Authentication Protocol)，使接受者完毕消息的鉴别。

5.1 消息认证

- 鉴别函数
  - 可用来做鉴别的函数分为三类：
    - 1) 消息加密函数(Message encryption)：用完整信息的密文作为对信息的鉴别。
    - 2) 消息鉴别码MAC(Message Authentication Code)：用一种密钥控制的公开函数作用后，产生固定长度的数值作为认证符，也称密码校验和。
    - 3) 散列函数(Hash Function)：是一种公开的函数，它将任意长的信息映射成一种固定长度的信息。常见的散列函数有：MD4、 MD5、SHA和 SHA-1

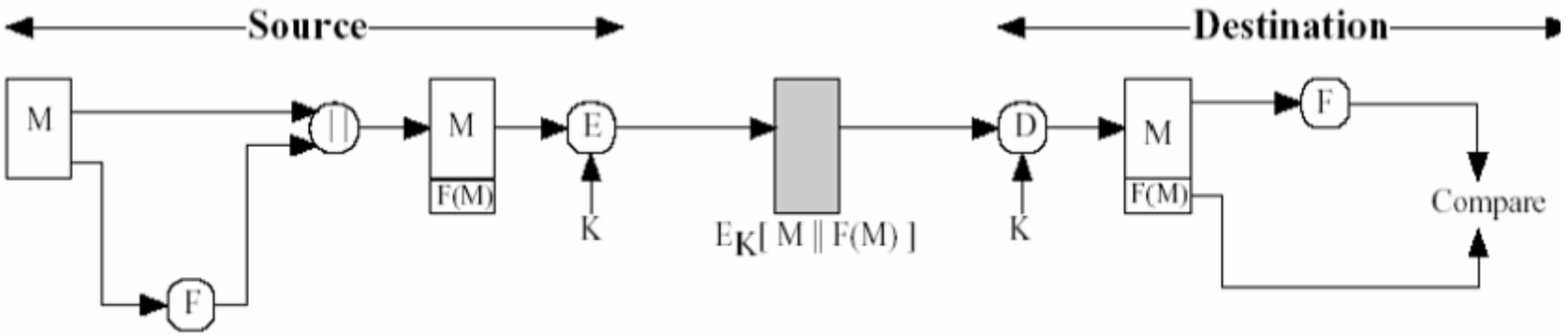
5.1.1 加密认证

- 对称密码体制加密认证
  - 发送者A用只有他与接受者B懂得的密钥K加密信息发送给接受者。
    - 因为A是除B以外惟一拥有密钥K的一方，而敌手不懂得怎样变化密文来产生明文中所期望的变化，所以B 懂得解密得到的明文是否被变化。这么对称加密就提供了消息认证功能。
    - 对称加密：具有机密性，可认证，不提供署名

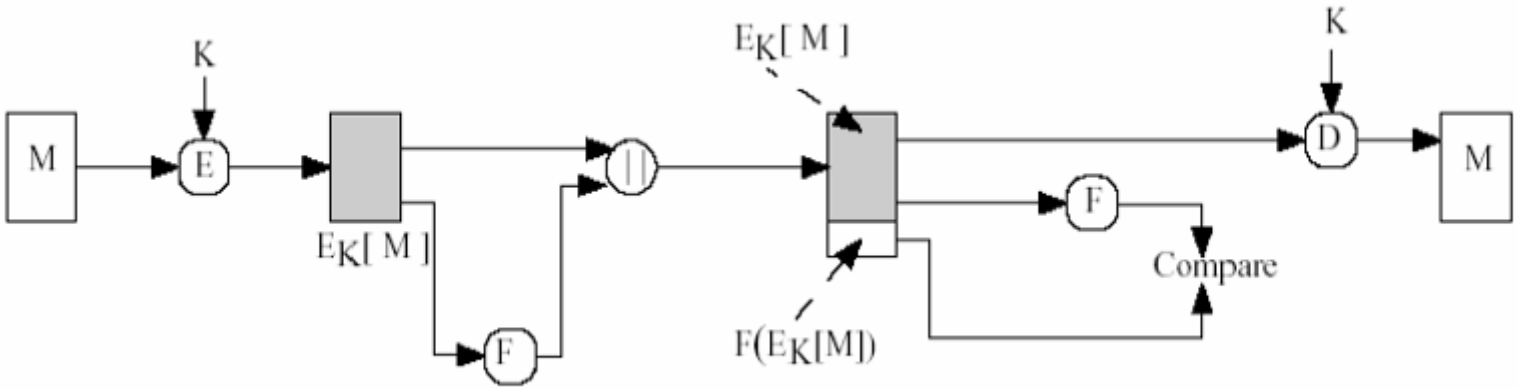


- 怎样自动拟定是否收到的明文可解密为可懂的明文?
- 一种处理方法是强制明文有某种构造.

差错控制：Error Control



(a) Internal error control



(b) External error control

5.1.1 加密认证

- 公钥密码体制加密认证
  - 使用公开密钥加密信息的明文只能提供保密而不能提供认证。为了提供认证，发送者A用私钥对信息的明文进行加密，任意接受者都能够用A的公钥解密。
  - 采用这么的构造既可提供了认证，也可提供数字署名。因为只有A能够产生该密文，其他任何一方都不能产生该密文。
    - 从效果上看A 已经用私钥对信息的明文进行了署名。
    - 应该注意只用私钥加密不能提供保密性。因为，任何人只要有A的公开密钥就能够对该密文进行解密。

5.1.1 加密认证

- 问题：经过加密得到信息真实性？
  - 保密性与真实性是两个不同的概念。根本上，信息加密提供的是保密性而非真实性。
  - 加密代价大(公钥算法代价更大)。
  - 鉴别函数与保密函数的分离能提供功能上的灵活性。
    - 广播的信息难以使用加密(信息量大)
    - 某些信息只需要真实性,不需要保密性

5.1.2 消息认证码

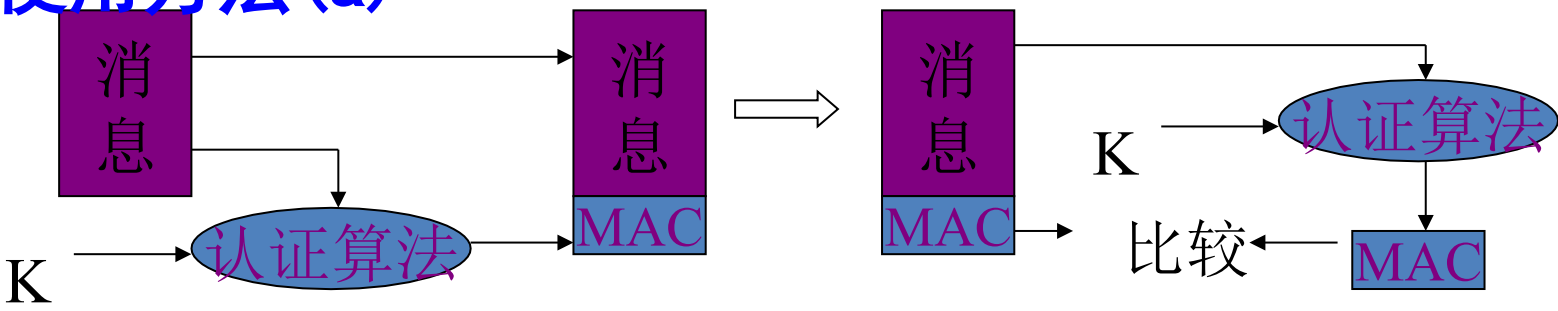
- 消息认证码
  - 消息认证码MAC 或称密码检验和是在一种密钥的控制下将任意长的消息映射到一种简短的定长数据分组，并将它附加在消息后。
  - 设M是变长的消息，K是仅由收发双方共享的密钥，则M的MAC由如下的函数C生成：$MAC = C_k(M)$
    - 这里的$C_k(M)$是定长的。发送者每次将MAC附加到消息中，接受者经过重新计算MAC来对消息进行认证。

5.1.2 消息认证码

- 假如收到的MAC与计算得出的MAC相同，则接受者能够以为：
  - 消息未被更改正
  - 消息来自与他共享密钥的发送者
- MAC函数类似于加密函数，主要区别在于MAC 函数不需要可逆性，而加密函数必须是可逆的，所以认证函数比加密函数更不易破解。
- 因为收发双方共享相同的密钥，上述过程只提供认证而不提供保密，也不能提供数字署名

5.1.2 消息认证码（MAC）

MAC的基本使用方法(a)

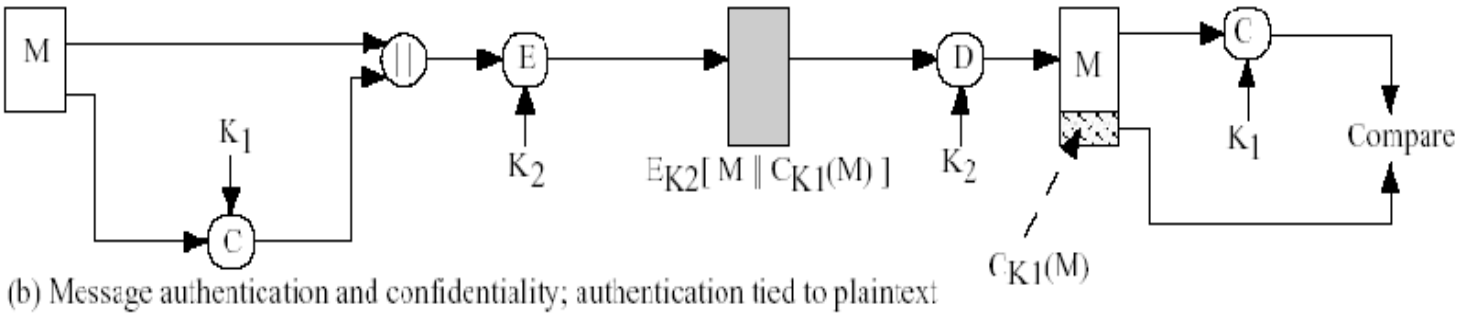


认证码的使用

- A B: $M||C_K(M)$
- 即A使用双方共享的密钥K对明文进行计算，产生一种短小的数据块，即消息验证码 $MAC=C_K(M)$ 。发送给接受方B时，将它附加在报文中。
- 接受方收到报文使用相同的密钥K执行相同的计算，得到新的MAC。接受方将收到的MAC与计算得到MAC进行比较，假如相匹配，那么能够确保报文在传播过程中维持了完整性：
- （1）报文未被更改正
- （2）接受者确信报文来自真实的发送者。（无人知晓密钥）
- 注意：上述认证过程只提供认证、不提供保密性。

5.1.3 MAC的基本使用方法

- MAC的基本使用方法(b)



(b) Message authentication and confidentiality; authentication tied to plaintext

$$A \longrightarrow B: \quad E_{K2}(M||C_{K1}(M))$$

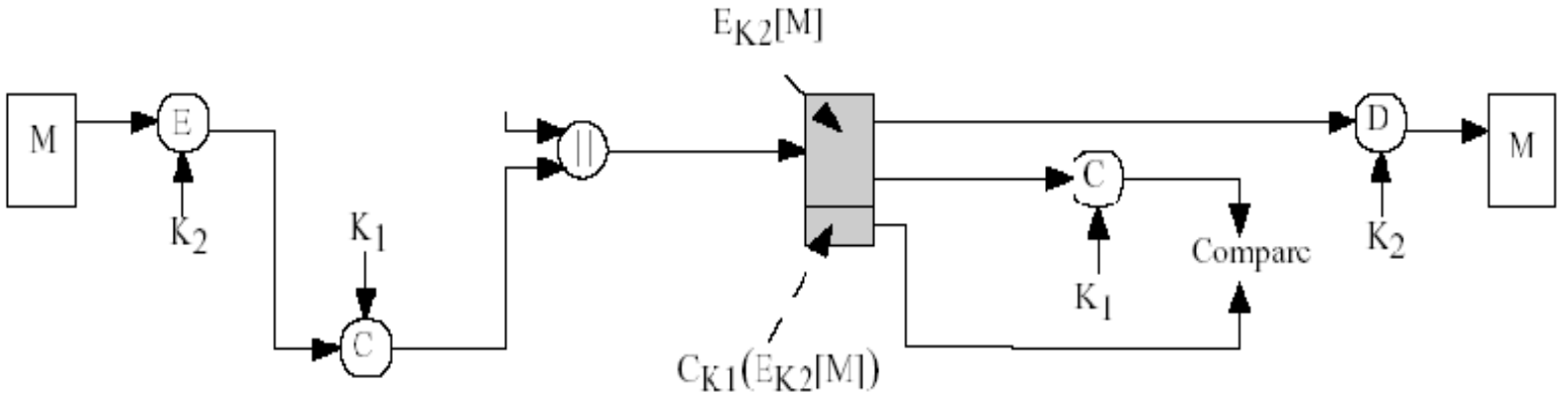
提供消息认证与保密，认证码与明文连接

Provides authentication -- only A and B share K1

Provides confidentiality -- only A and B share K2

5.1.3 MAC的基本使用方法

- MAC的基本使用方法(c)



(c) Message authentication and confidentiality; authentication tied to ciphertext

$$A \longrightarrow B: \quad E_{K2}(E_{K2}(M) \| C_{K1}(E_{K2}(M)))$$

提供消息认证与保密，认证码与密文连接

Provides authentication -- Using K1

Provides confidentiality -- Using K2

消息认证 VS 常规加密

- MAC函数类似于加密函数，主要区别在于MAC函数不需要可逆而加密函数必须是可逆的，所以，认证函数比加密函数更不易破解。
- 保密性与真实性是两个不同的概念
- 根本上,信息加密提供的是保密性而非真实性
- 加密代价大(公钥算法代价更大)
- 认证函数与保密函数的分离能提供功能上的灵活性
- 某些信息只需要真实性,不需要保密性
  - 广播的信息难以使用加密(信息量大)
  - 网络管理信息等只需要真实性
  - 政府/权威部门的公告

5.2 散列(Hash)函数

- 散列(Hash)函数
  - 散列函数(又称杂凑函数)是对不定长的输入产生定长输出的一种特殊函数：h = H(M)
    - 其中M是变长的消息
    - h=H(M)是定长的散列值或称为消息摘要。
  - 散列函数H是公开的，散列值在信源处被附加在消息上，接受方经过重新计算散列值来确保消息未被篡改。
  - 因为函数本身公开，传送过程中对散列值需要另外的加密保护(假如没有对散列值的保护，篡改者能够在修改消息的同步修改散列值，从而使散列值的认证功能失效)。

5.2 散列(Hash)函数

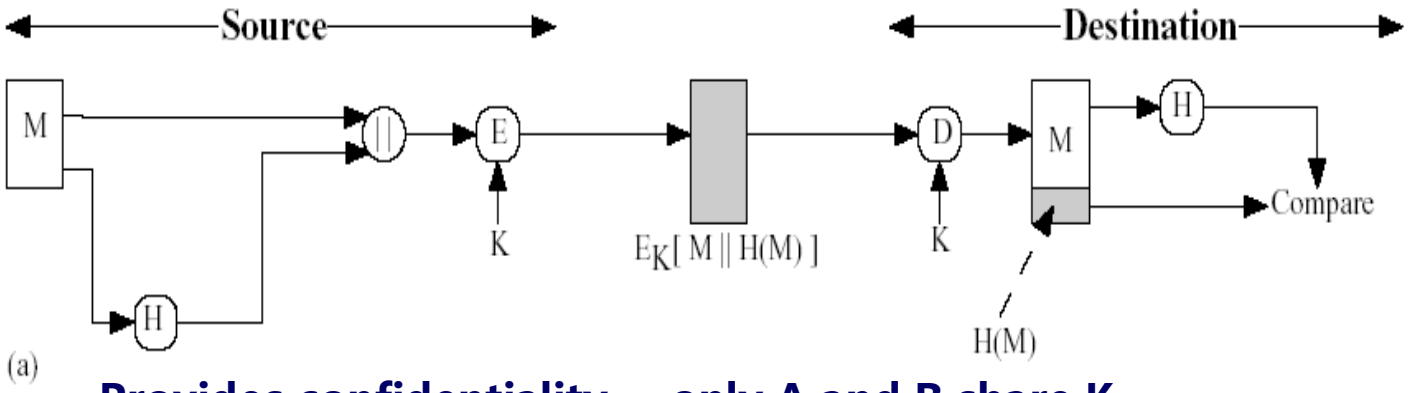
- 散列(Hash)函数的安全性
  - 以一种x开始。首先计算Z=h(x)，并企图找到一种x′满足h(x′)=h(x)。若他做到这一点， x′也将为有效。为预防这一点，要求函数h具有无碰撞特征。
    - 定义1(弱无碰撞)：散列函数h称为是弱无碰撞的，指对给定消息$x\in X$，在计算上几乎找不到不等于x的$x'\in X$，使h(x)=h(x′)。
    - 定义2(强无碰撞)，散列函数h被称为是强无碰撞的，是指在计算上几乎不可能找到任意的相异的x，x′，使得h(x)=h(x′)。
    - 注：强无碰撞自然含弱无碰撞！

5.2.1 散列函数的性质

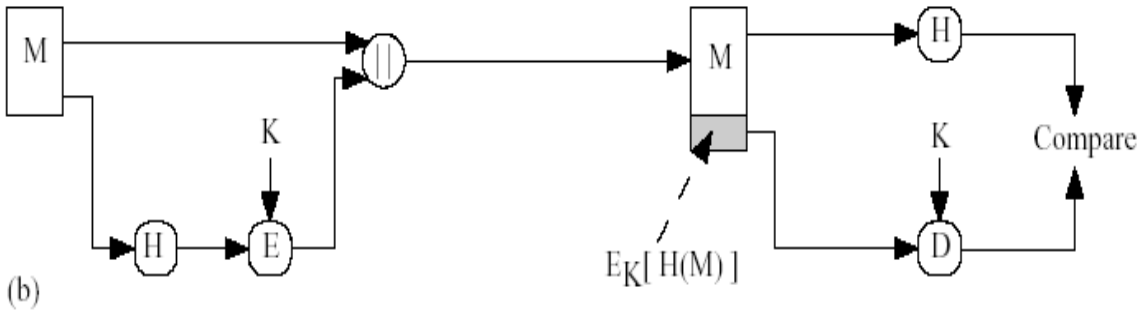
- 散列函数的性质
  - 散列函数的目的是为文件、消息或其他的分组数据产生“指纹”。用于消息认证的散列函数H必须具有如下性质：
    - 1. H能用于任何大小的数据分组，都能产生定长的输出
    - 2. 对于任何给定的x，H(x)要相对易于计算
    - 3. 对任何给定的散列码h，寻找x使得H(x)=h在计算上不可行
    - 4. 对任何给定的分组x，寻找不等于x的y，使得H(x)=H(y)在计算上不可行(弱抗冲突)
    - 5. 寻找任何的(x，y)，使得H(x)=H(y)在计算上不可行(强抗冲突)

5.2.2 散列函数的使用方式

- 散列函数的基本使用方法(a、b)



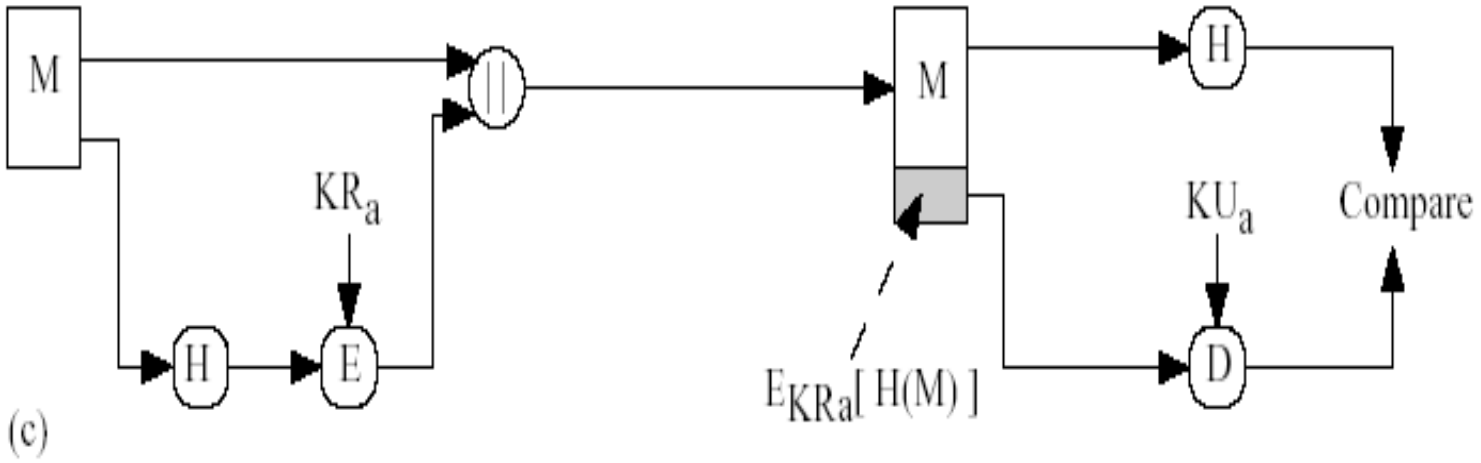
Provides confidentiality -- only A and B share K
Provides authentication -- H(M) is cryptographically protected



Provides authentication -- H(M) is cryptographically protected

5.2.2 散列函数的使用方式

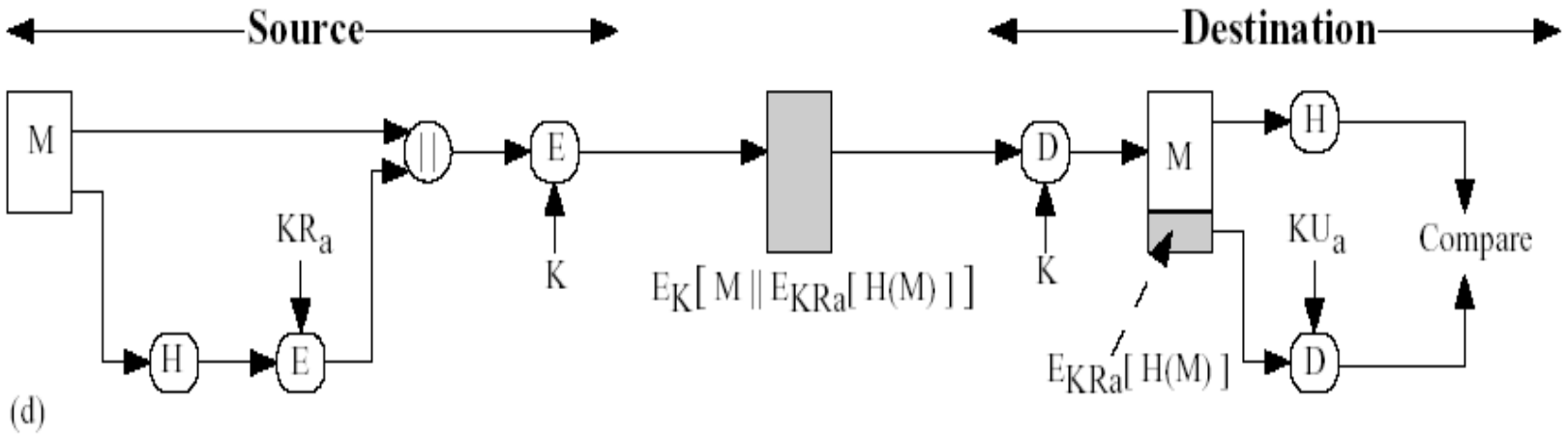
- 散列函数的基本使用方法(C)



使用公开密钥加密措施及发方的私有密钥仅对散列码加密。和b一样，这种措施也提供鉴别。它还提供数字署名，因为只有发方能够生成加密的散列码。实际上这是数字署名的本质。

5.2.3 散列函数的使用方式

- 散列函数的基本使用方法(d)

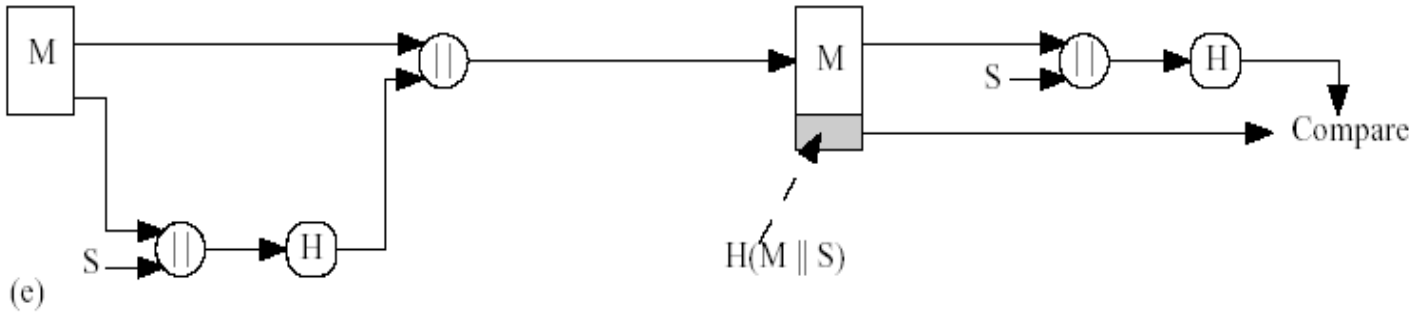


(d) $A \rightarrow B: E_K[M||E_{KRa}[H(M)]]$

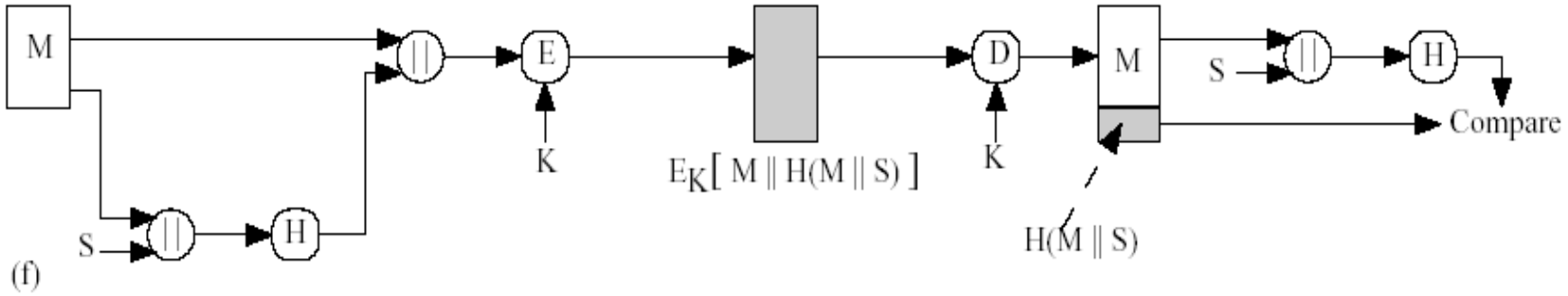
Provides authentication and digital signature
Provides confidentiality

5.2.2 散列函数的使用方式

- 散列函数的基本使用方法(e、f)



Provides authentication -- only A and B share S



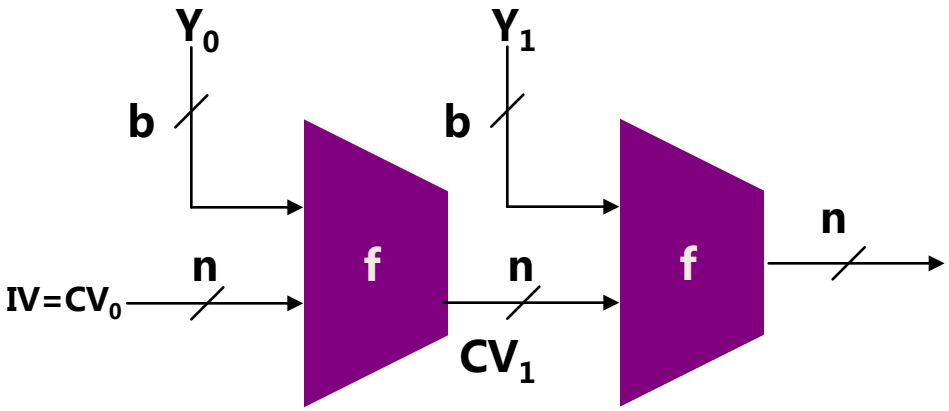
Provides authentication -- only A and B share S
Provides confidentiality -- only A and B share K

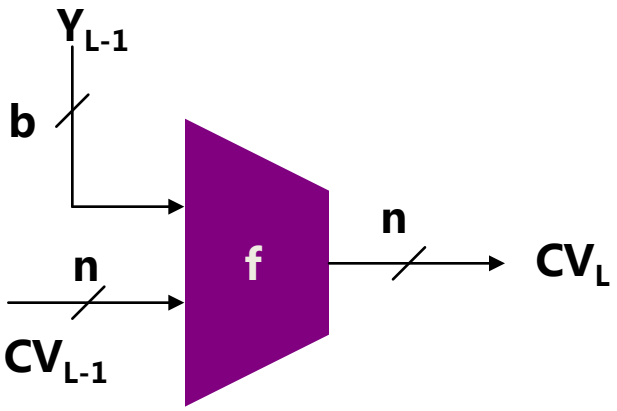
5.2.3 散列函数的构造

- 散列函数的构造
  - 由Merkle于1989年提出
  - Ron Rivest于1990年提出MD4
  - 几乎被全部hash函数使用
  - 详细做法：
    - 把原始消息M提成某些固定长度的块Y_i
    - 最终一块padding并使其包括消息M长度
    - 设定初始值CV_0
    - 反复使用压缩函数f，$CV_i = f(CV_{i-1}, Y_{i-1})$
    - 最终一种CV_i为hash值

5.2.3 散列函数的构造

- 散列函数的构造





IV = initial value 初始值

CV = chaining value 链接值

Y_i = ith input block (第i 个输入数据块)

f = compression algorithm (压缩算法)

n = length of hash code (散列码的长度)

b = length of input block(输入块的长度)

CV_0 = IV = initial n-bit value

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

$H(M) = CV_L$

5.2.4 三种常用的HASH算法

- 三种常用的HASH算法
  - MD5
  - SHA-1
  - HMAC

5.2.4.1 MD5

- MD5简介
  - Merkle于1989年提出hash function模型
  - Ron Rivest于1990年提出MD4
  - 1992年, MD5 (RFC 1321) developed by Ron Rivest at MIT
  - MD5把数据提成512-bit块
  - MD5的hash值是128-bit
  - 在近来数年之前，MD5是最主要的hash算法
  - 现行美国原则SHA-1以MD5的前身MD4为基础

5.2.4.1 MD5

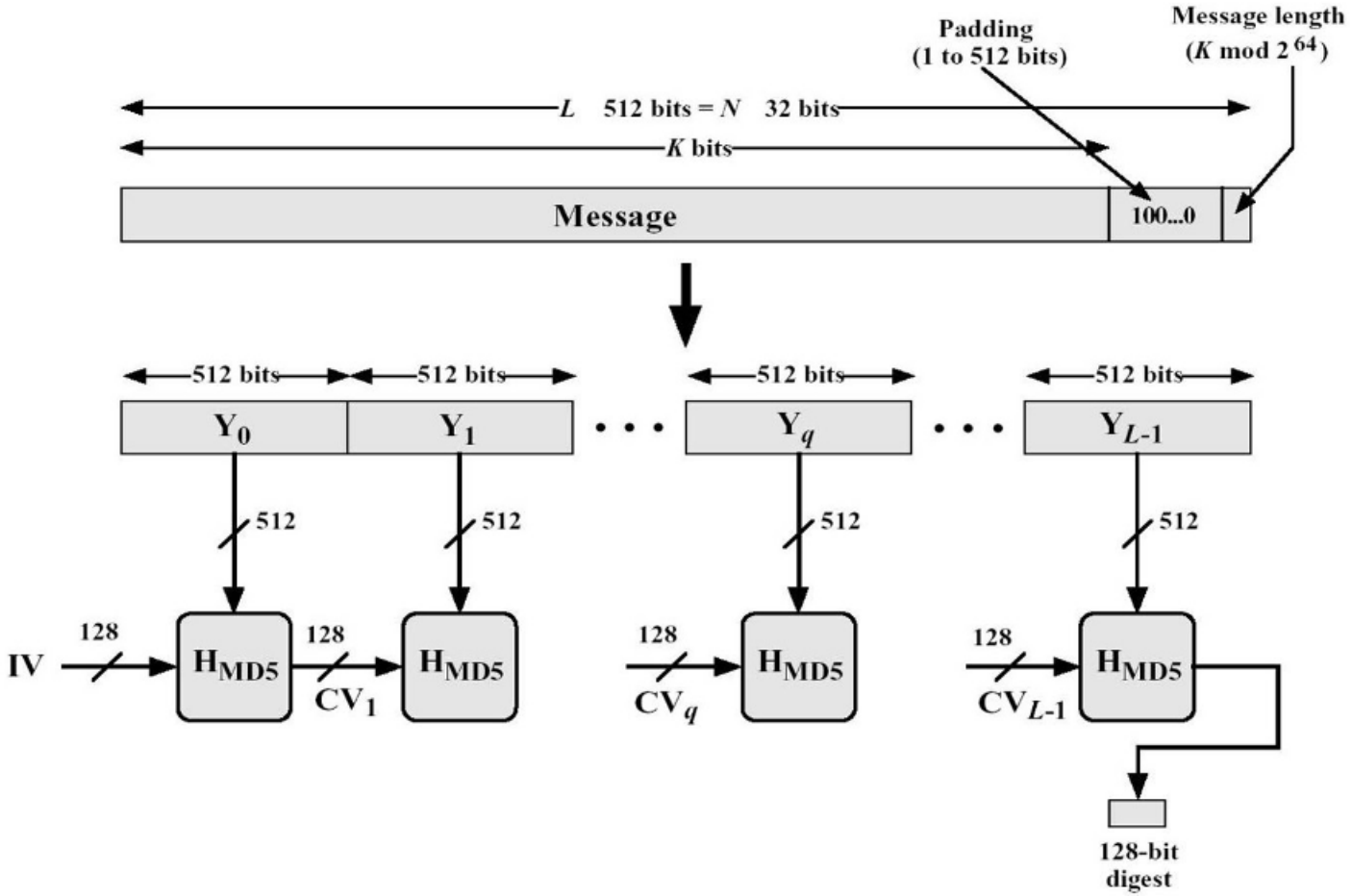
- MD5算法
  - 该算法以一种任意长度的报文作为输入，产生一种128bit的报文摘要作为输出。输入是按512bit的分组进行处理的。
  - Step 1：
    - 附加填充比特 M ➔ M1，|M1| ≡ 448 mod 512 (即填充长度为512的整数倍减去64)。
    - 例如，假如|M|是448bit，那么填充512bit，形成960bit的报文
    - Padding内容：100……0
- 首位为1，其他补0至满足要求，即填充后的比特数为512的整数倍减去64，或使得填充后的数据长度与448模512同余。

5.2.4.1 MD5

- MD5算法
  - Step 2：附加长度值
    - Append 64-bit length M1 ➔ M2(初始报文的位长度)
    - 若|M| > 2^{64},则仅取低64位
    - 低字节在前 (little-endian)
    - |M2|为512的倍数: $Y_0, Y_1, \ldots, Y_{L-1}$

5.2.4.1 MD5

- MD5算法
  - MD5示意图

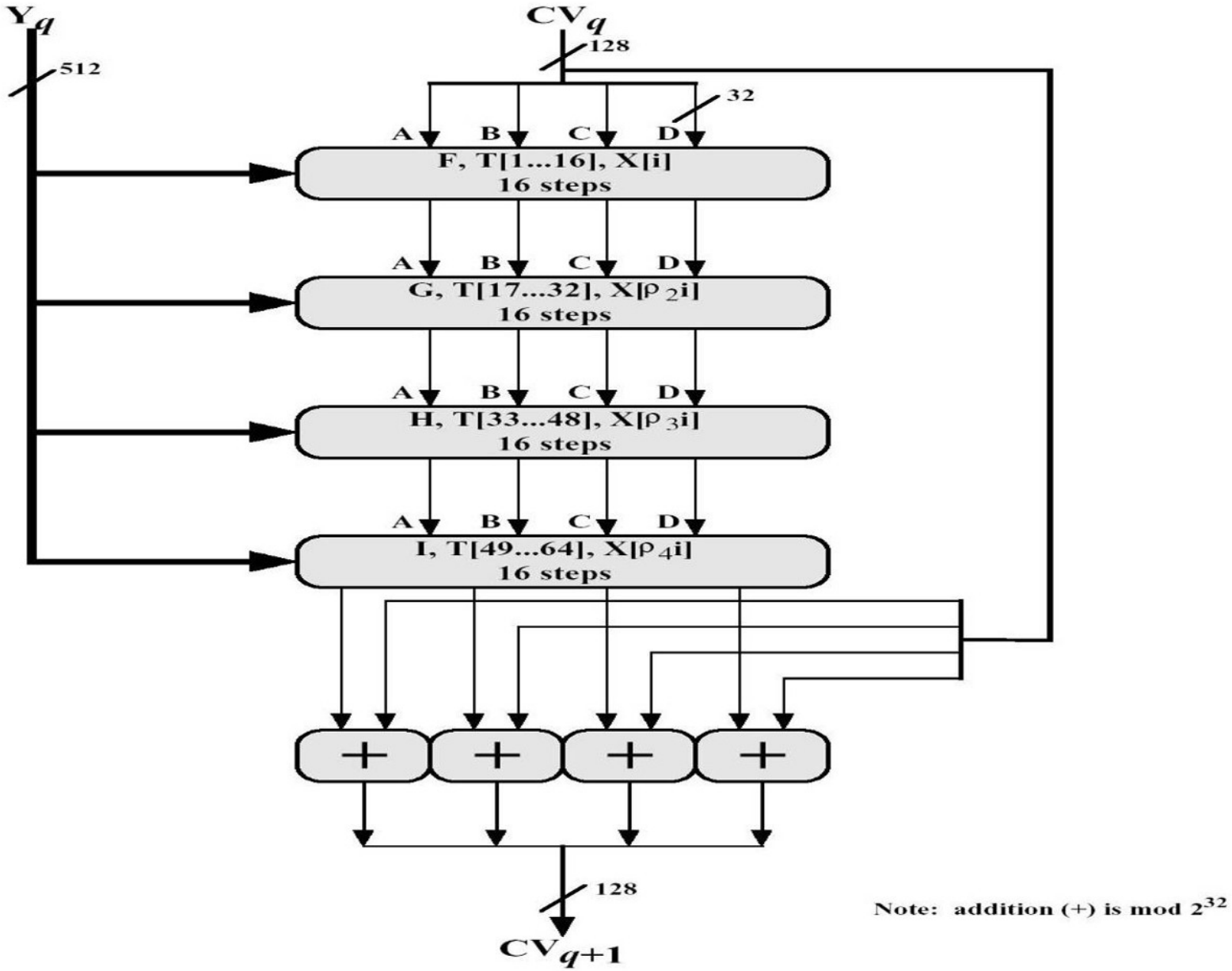


5.2.4.1 MD5

- MD5算法
  - Step 3 : Initialize MD buffer (little-endian)
    - A = 01 23 45 67 (0x67452301)
    - B = 89 AB CD EF (0xEFCDAB89)
    - C = FE DC BA 98 (0x98BADCFE)
    - D = 76 54 32 10 (0x10325476)
  - Step 4 : Compression
    - CV_0=IV
    - CV_i=HMD5(CV_{i-1} , Y_i)
  - Step 5: Output
    - MD = CV_L

5.2.4.1 MD5

MD5 Step 4：示意图



5.2.4.1 MD5

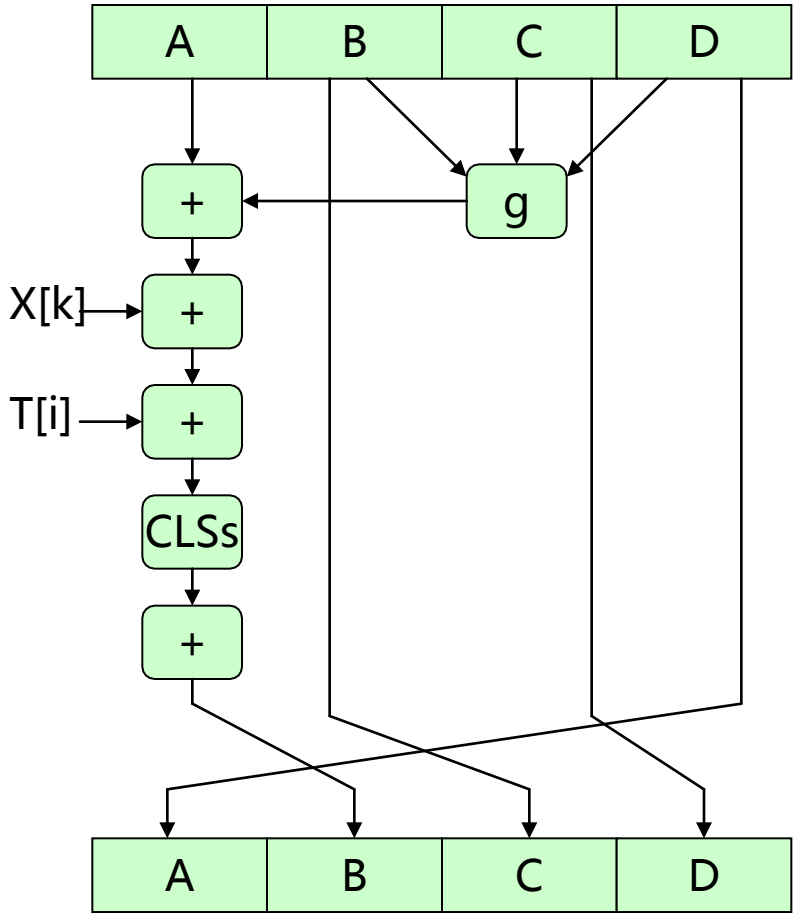
- MD5 Step 4：$CV_0=IV$，$CV_i=H_{MD5}(CV_{i-1},Y_i)$
  - (A0,B0,C0,D0)←(A,B,C,D)
    - RoundOne(A,B,C,D,T[1…16],X[0…15])
    - RoundTwo(A,B,C,D,T[17…32],X[0…15])
    - RoundThree(A,B,C,D,T[33…48],X[0…15])
    - RoundFour(A,B,C,D,T[49…64],X[0…15])
  - (A,B,C,D)←(A+A0,B+B0,C+C0,D+D0)
  - 512-bit块(X[…]为32-bit表达)在四个Round使用，每个Round包括16次循环，每次处理一种32-bit，
  - T[j]= [sin(j)*2^{32}]的整数部分, $1 \le j \le 64$

5.2.4.1 MD5

- MD5 Step 4：Compression
  - 每一轮包括对缓冲区ABCD的16步操作所构成的一种序列。
    - a ← b + ((a + g(b,c,d) + X[k] +T[i])<<<s)
  - 其中：
    - a,b,c,d = 缓冲区的四个字，以一种给定的顺序排列；
    - g = 基本逻辑函数F,G,H,I之一；
    - <<<s = 对32位字循环左移s位
    - X[k] = M[q×16 + k] = 在第q个512位数据块中的第k个32位字
    - T[i] = 表T中的第i个32位字；
    - + = 模 2^{32}的加；

5.2.4.1 MD5

- MD5 Step 4：Compression



Function g	g(b,c,d)
1 F(b,c,d)	(b∧c)∨(b∧d)
2 G(b,c,d)	(b∧d)∨(c∧d)
3 H(b,c,d)	b⊕c⊕d
4 I(b,c,d)	c⊕(b∨d)

5.2.4.1 MD5

- MD5 Step 4：
  - $CV_0=IV$，$CV_i=H_{MD5}(CV_{i-1},Y_i)$
    - (A0,B0,C0,D0)←(A,B,C,D)
      - RoundOne(A,B,C,D,T[1…16],X[0…15])
      - RoundTwo(A,B,C,D,T[17…32],X[0…15])
      - RoundThree(A,B,C,D,T[33…48],X[0…15])
      - RoundFour(A,B,C,D,T[49…64],X[0…15])
    - (A,B,C,D)←(A+A0,B+B0,C+C0,D+D0)
    - MD = CV_L

5.2.4.1 MD5

- MD5 小结：
  - MD5使用小数在前
  - Dobbertin在1996年找到了两个不同的512-bit块，它们在MD5计算下产生相同的hash
  - 至今还没有真正找到两个不同的消息，它们的MD5的hash相等
  - MD5不是足够安全的

5.2.4.1 MD5

- MD5 小结：
  - MD5在线查询破解
  - MD5的32位和16位加密算法
    - MD5一般是32位的编码，而在不少地方会用到16位的编码。16位就是从32位MD5散列中把中间16位提取出来。
    - admin 的摘要：
      - 16位：7a57a5a743894a0e
      - 32位：21232f297a57a5a743894a0e4a801fc3