第5章

消息认证

主要内容

- 消息认证基本概念
- 消息加密认证
- 消息认证码
- hash函数

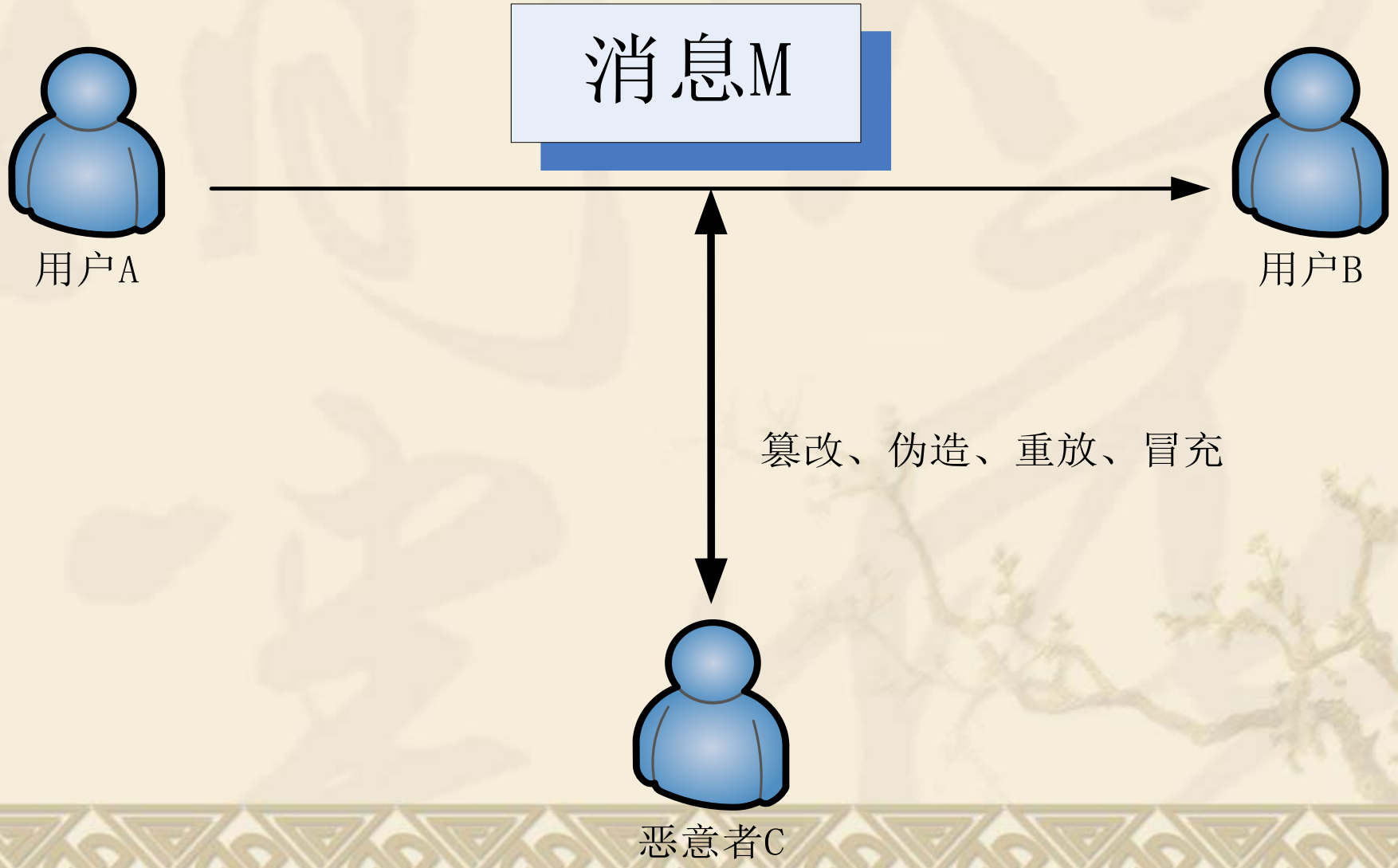
5.1 消息认证基本概念

- 认证(Authentication)：即鉴别、确认，它是证实某事是否名副其实，或是否有效的一个过程。
- 认证与加密的区别：
  - 加密用以确保数据的保密性，阻止对手的被动攻击，如截取、窃听。
  - 认证用以确保报文发送者和接受者的真实性以及报文的完整性，阻止对手的主动攻击，如冒充、篡改、重播等。
- 认证往往是应用系统中安全保护的第一道防线，极为重要。

基本思想

- 通过验证称谓者(人或事)的一个或多个参数的真实性和有效性，来达到验证称谓者是否名副其实的目的。
- 常用的参数有：口令、标识符、密钥、信物、智能卡、指纹、视网纹等。
- 利用人的生理特征参数进行认证的安全性高，但技术要求也高，至今尚未普及。目前广泛应用的还是基于密码的认证技术。

没有消息认证的通信系统是极为危险的



消息认证(Message Authentication)

- 消息认证用于抗击主动攻击
- 验证接收消息的完整性
- 完整性
  - 未被篡改、插入和删除
- 验证消息的顺序性和时间性（未重排、重放和延迟）

需求

1. 泄密：将消息透露给没有合法秘密钥的任何人或程序。
2. 传输分析：分析通信双方的通信模式，如连接频率，时间等
3. 伪装：攻击者产生一条消息并声称来自某合法实体
4. 内容修改：对消息进行插入、删除、转化、修改
5. 顺序修改：对消息顺序进行插入、删除、重新排序
6. 计时修改：对消息的延时和重放
7. 发送方否认
8. 接收方否认

- 对付1、2可用加密；
- 对付3、4、5、6可用消息认证；
- 对付7、8可用数字签名

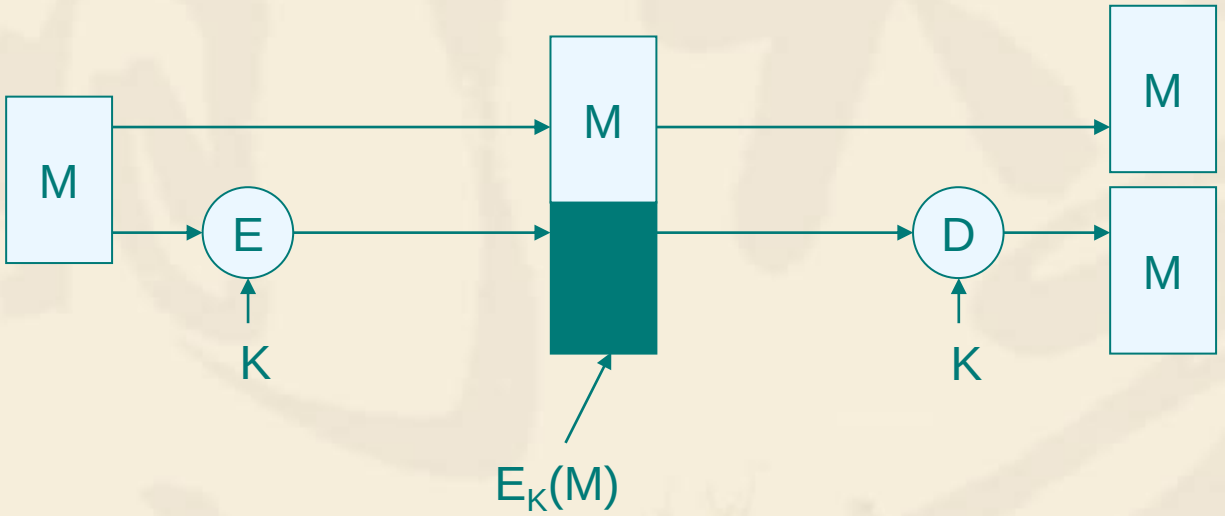
消息认证的基本概念

- 消息认证：验证所收到的消息和该消息被发送时是否相同，即是否被修改过。
- 认证符(authenticator)：一个用来认证消息的值。由消息的发送方产生认证符，并传递给接收方。
- 认证函数：产生认证符的函数，认证函数实际上代表了一种产生认证符的方法。
  - 消息加密(Message encryption)
  - 消息认证码(**Message authentication code ,MAC**)
  - Hash函数(**Hash function**)

5.2 消息加密认证

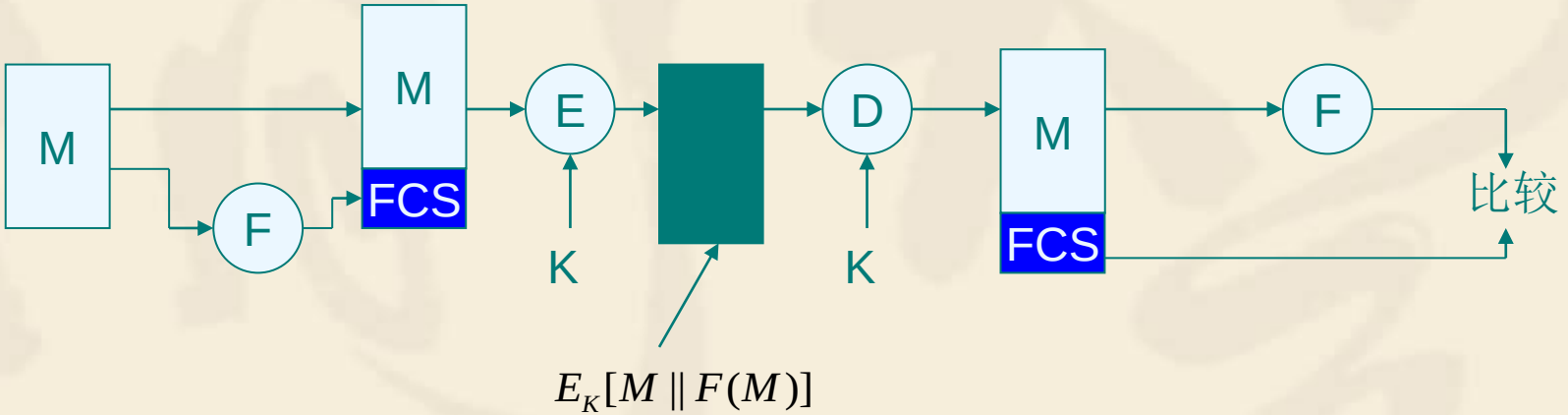
- 对称密码实现消息加密认证
- 公钥密码实现消息加密认证

消息加密认证---在对称加密体制下



- 由于攻击者不知道密钥K，他也就不知道如何改变密文中的信息位才能在明文中产生预期的改变。

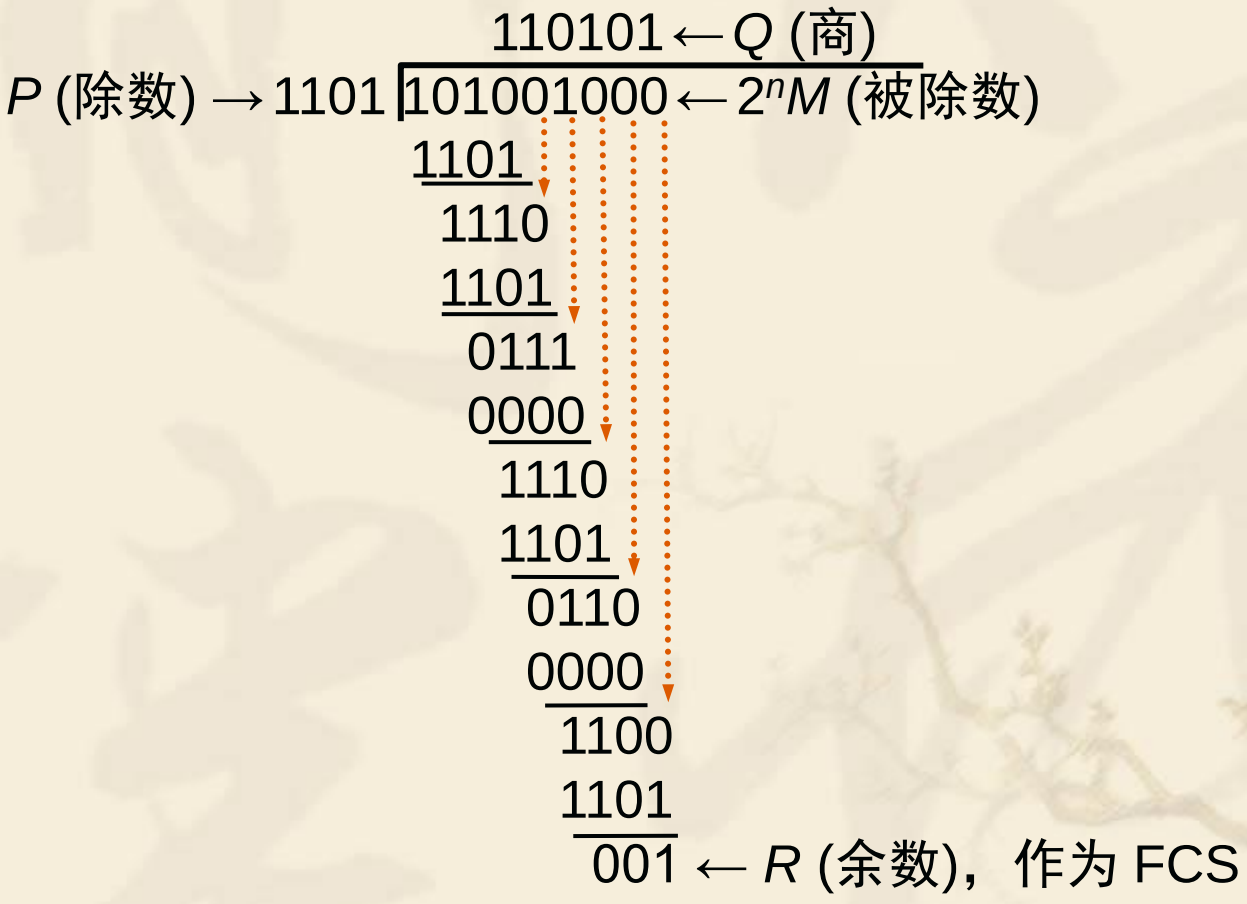
内部错误控制



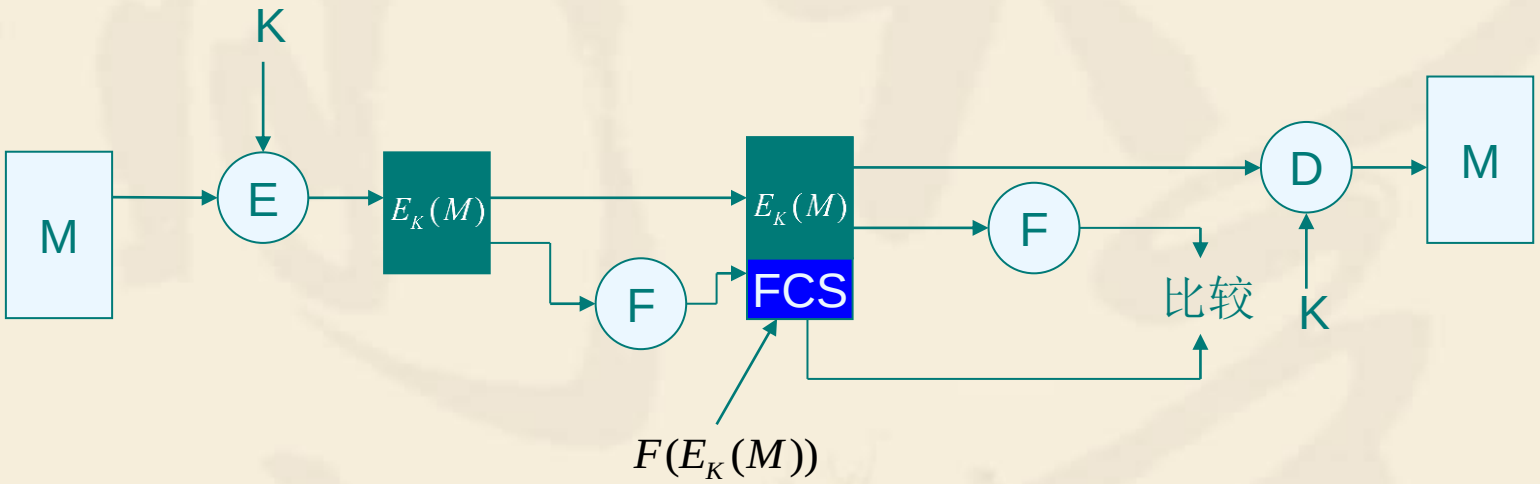
- 根据明文M和公开的函数F产生FCS，即错误检测码，或帧校验序列，校验和。
- 把M和FCS合在一起加密，并传输。
- 接收端把密文解密，得到M。
- 根据得到的M，按照F计算FCS，并与接收到的FCS比较是否相等。

FCS的原理说明

M = 101001

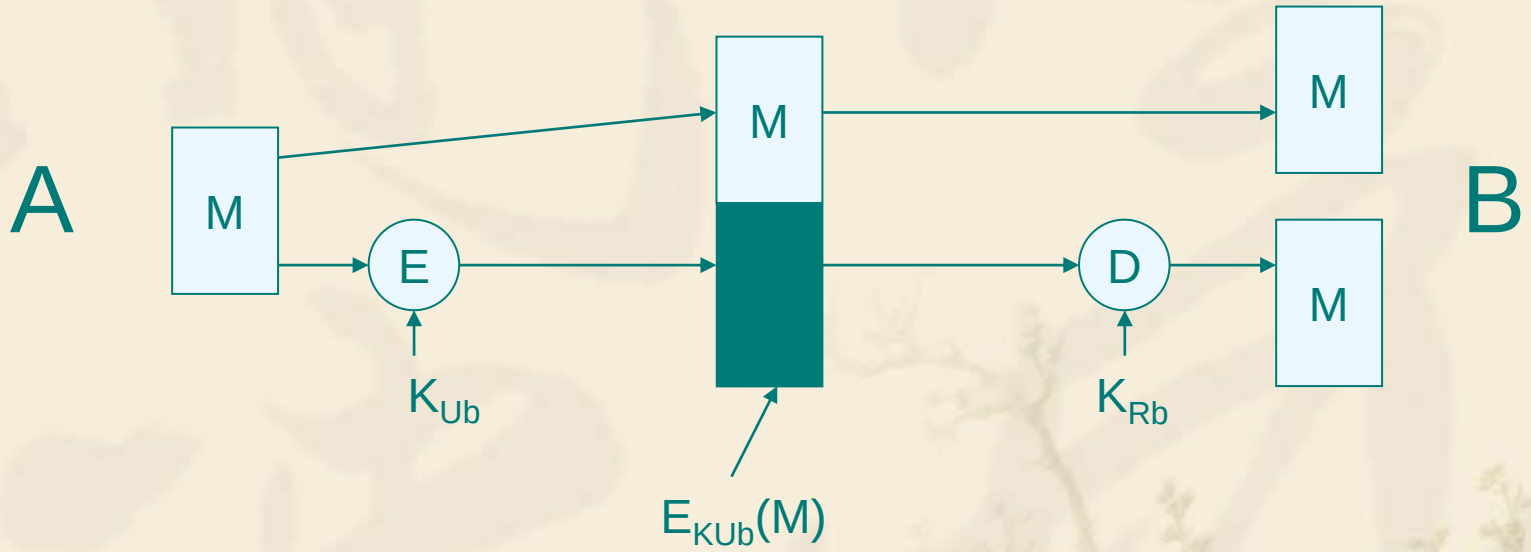


外部错误控制



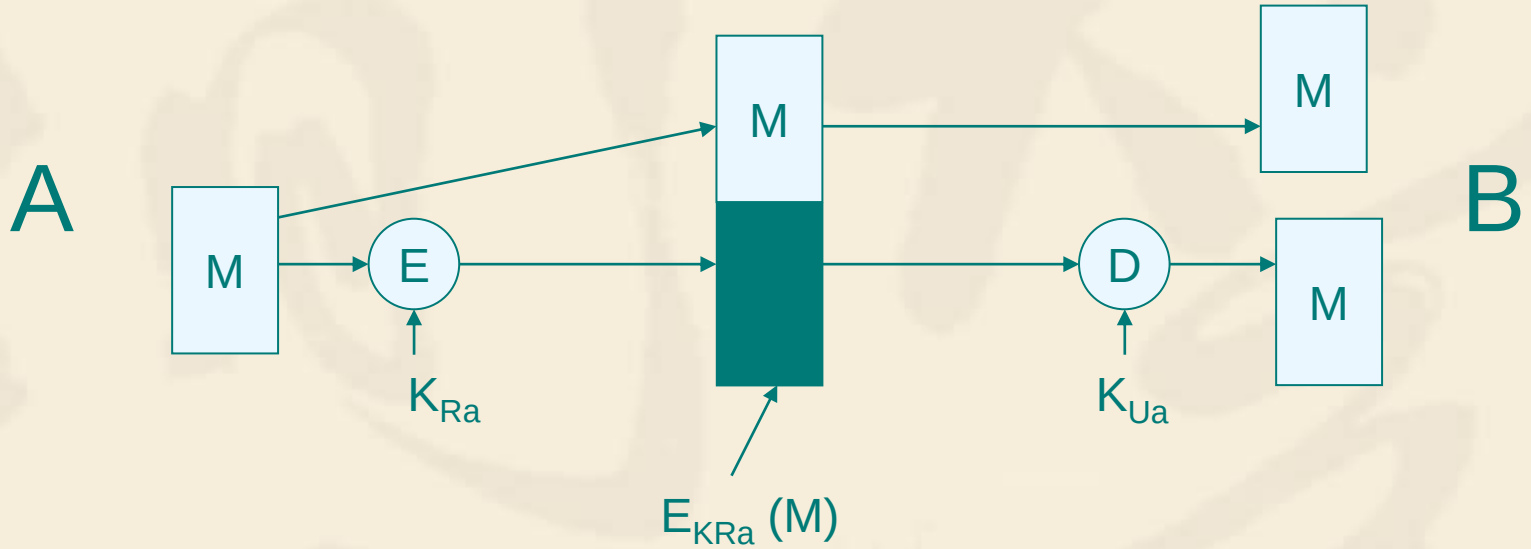
$E_K(M)$

消息加密认证---在公钥加密体制下



I. 普通加密

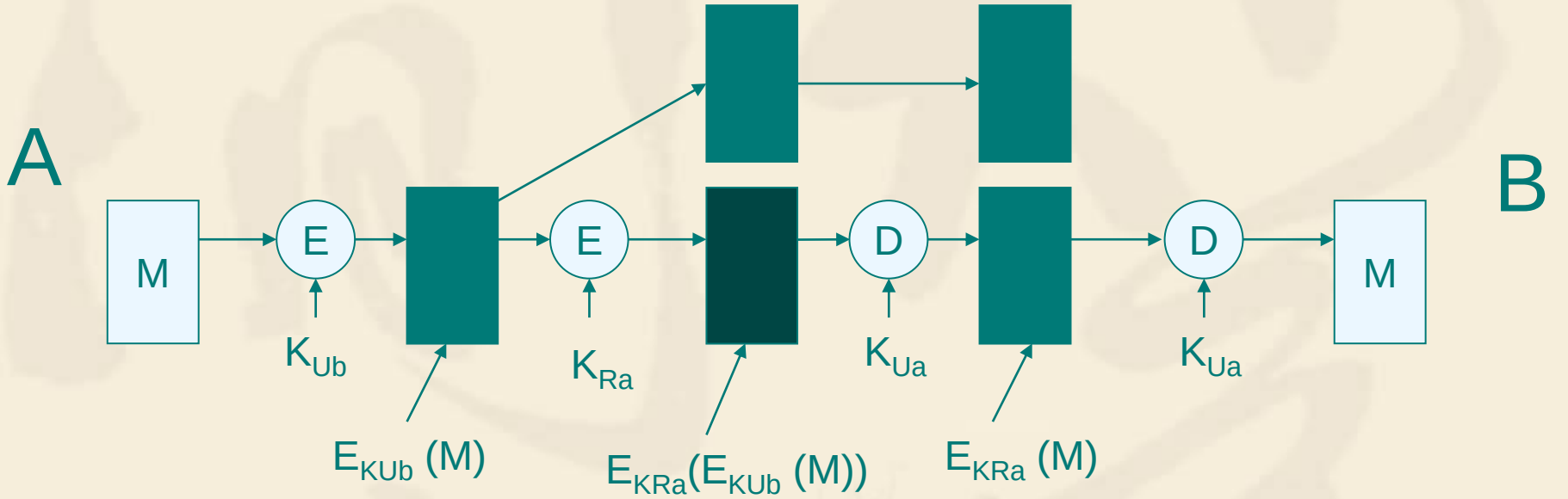
消息加密认证---在公钥加密体制下



II. 认证和签名

- 由于只有A有用于产生E_{KRa} (M)的密钥，所以此方法提供认证。

消息加密认证---在公钥加密体制下



III. 加密认证和签名

- 提供认证和加密。
- 一次通信中要执行四次复杂的公钥算法。

把加密与认证分开

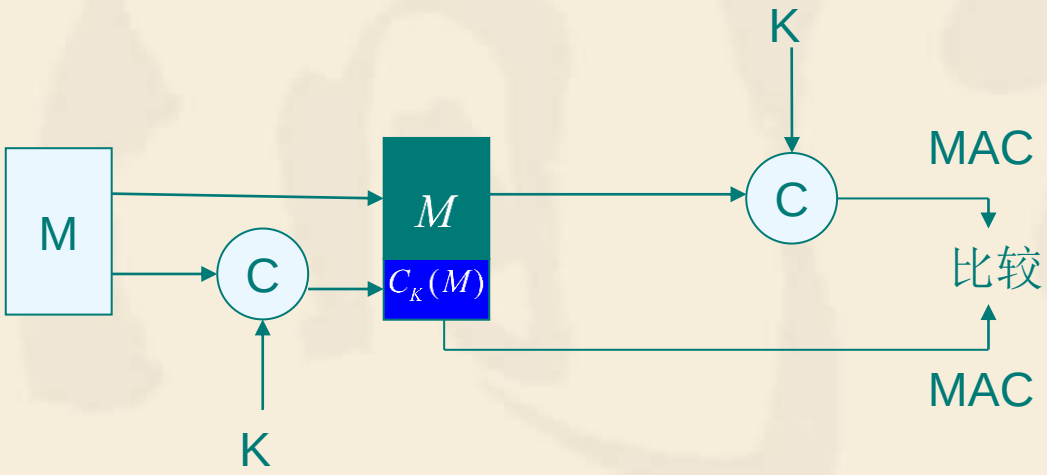
- There are a number of applications in which the same message is broadcast to a number of destinations.
- The receiver has a heavy load and cannot afford the time to decrypt all incoming messages.
- For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages.
- Separation of authentication and confidentiality functions affords architectural flexibility.

5.3 消息认证码(MAC)

- Message Authenticaion Code
- 消息认证码是消息和密钥的公开函数，它产生定长的值，以该值作为认证符。
- 利用密钥和消息生成一个固定长度的短数据块，并将其附加在消息之后。
- 通信双方共享密钥K

- A MAC, also known as a cryptographic checksum(密码校验和) , is generated by a function C of the form： MAC = C(K, M)
- Where：
  - M is a variable-length message
  - K is a secret key shared only by sender and receiver
  - MAC is the fixed-length authenticator.
- The MAC is appended to the message at the source at a time when the message is assumed or known to be correct.
- The receiver authenticates that message by recomputing the MAC.

消息认证码用于认证



- A和B共享密钥K
- A计算MAC=C_k(M),
- M和MAC一起发送到B
- B对收到的M，计算MAC，比较两个MAC是否相同。

如果两个MAC相等，则：

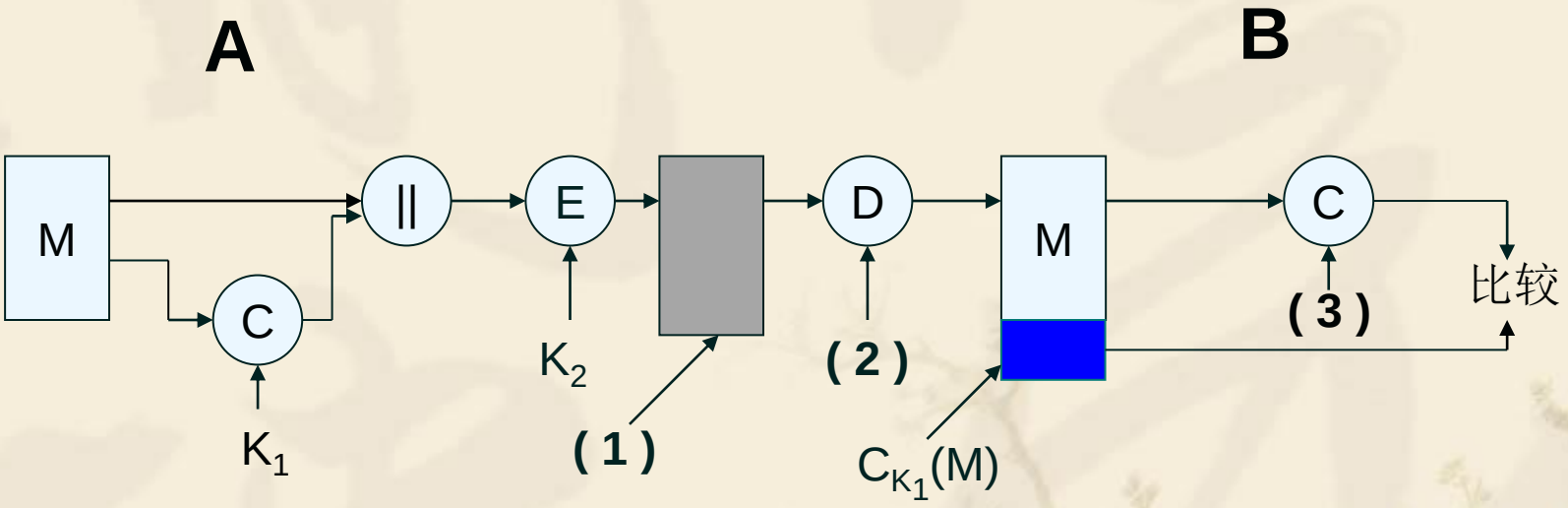
1. 接收方可以相信消息未被修改，因为如果攻击者改变了消息，由于不知道k，无法生成正确的MAC。
2. 接收方可以相信消息的确来自确定的发送方。因为其他人不能生成和原始消息相应的MAC。

MAC函数与加密函数的区别

- MAC函数与加密函数类似，都需要明文、密钥和算法的参与。
- 但MAC算法不要求可逆性，而加密算法必须是可逆的。
- 例如：使用100比特的消息和10比特的MAC，那么总共有2^{100}个不同的消息，但仅有2^{10}个不同的MAC。也就是说，平均每2^{90}个消息使用的MAC是相同的。
- 因此，认证函数比加密函数更不易被攻破，因为即便攻破也无法验证其正确性。关键就在于加密函数是一对一的，而认证函数是多对一的。

消息认证码的基本用途

- 只提供消息认证，不提供保密性。（见前）
- 提供消息认证和保密性：

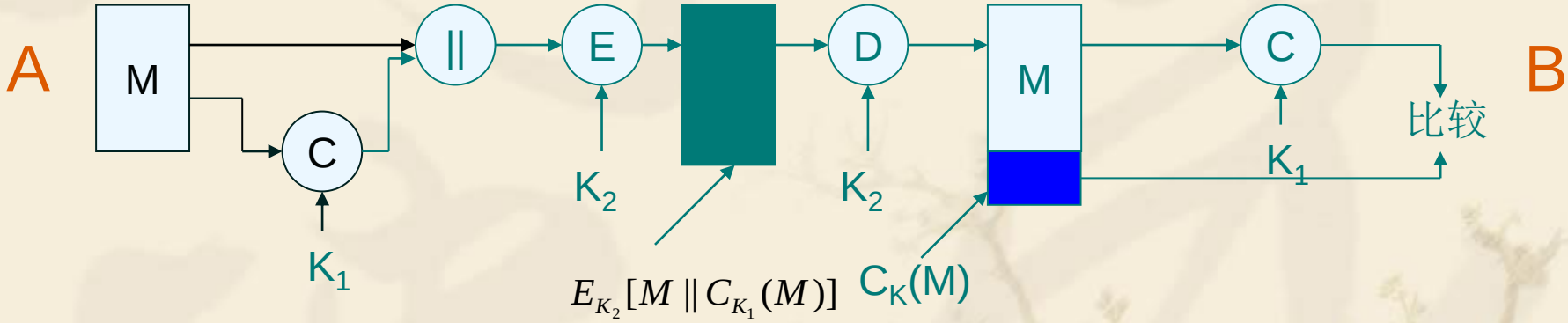


A和B共享K1和K2
K1：用于生成MAC
K2：用于加密

与明文有关的认证

消息认证码的基本用途

- 只提供消息认证，不提供保密性。（见前）
- 提供消息认证和保密性：

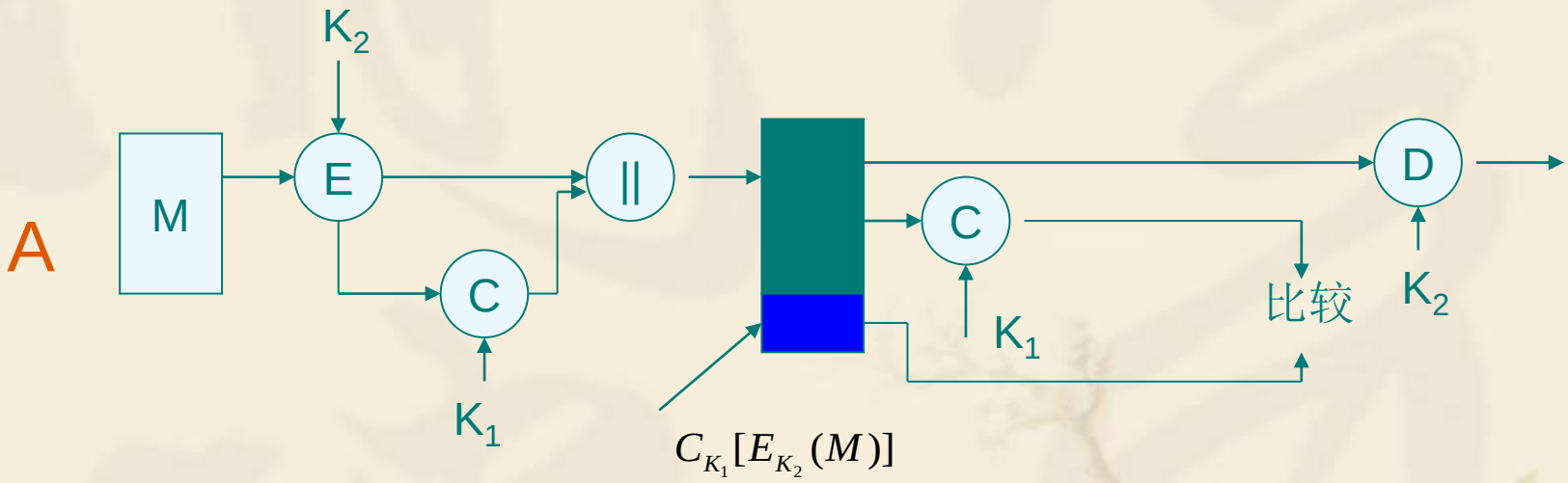


A和B共享K1和K2
K1：用于生成MAC
K2：用于加密

与明文有关的认证

消息认证码的基本用途

- 提供消息认证和保密性：



A和B共享K1和K2
K1：用于生成MAC
K2：用于加密

与密文有关的认证

5.3.2 消息认证码的安全性

- 攻击密钥
- 攻击算法

对MAC的攻击—攻击密钥

- 已知消息M_1和MAC算法C，以及$MAC_1 = C_{k1}(M_1)$，现要破解k_1。密钥为k个bit，MAC为n个bit。
- 当**k>n**：

  可能的密钥个数为2^k。可能的MAC个数为2^n个。

  所以许多不同的密钥(约2^{k-n}个)，计算出来的MAC都等于MAC_1。这些密钥中哪一个是正确的密钥不得而知。这时需要新的M-MAC对来测试这2^{k-n}个密钥，于是有如下的重复攻击：

重复攻击

- Step 1:
  - 给定M_1和$MAC_1 = C_{k1}(M_1)$
  - 对所有2^k个密钥，判断$MAC_i = C_{ki}(M_1)$
  - 匹配数约为： 2^{k-n}
- Step 2:
  - 给定M_2和$MAC_2 = C_{k2}(M_1)$
  - 对所有2^k个密钥，判断$MAC_i = C_{ki}(M_2)$
  - 匹配数约为： 2^{k-2n}
- 平均来讲，若k=x*n，则需x次循环才能找到正确的密钥。
- 所以，用穷举法攻破MAC比攻破加密算法要困难得多。

对MAC的攻击—攻击算法

- 考虑下面的算法：

  消息M=($X_1 \| X_2 \| \ldots \| X_m$)是由64比特长的分组$X_i$(i=1,…,m)链接而成

  MAC算法是：

$$\Delta(M) = X_1 \oplus X_2 \oplus \cdots \oplus X_m$$

$$C_K(M) = E_K\left[\Delta(M)\right]$$

  加密算法是DES。因此，密钥长为56比特。

  如果敌手得到M‖C_K(M)，那么敌手使用穷搜索攻击寻找K将需做2^{56}次加密。很困难!

- 但攻击者可以改变M的内容，却使MAC正确。

  方法如下：

- 用Y_1替换X_1，Y_2替换X_2，…，Y_m替换X_m，其中Y_1，Y_2，…，Y_m是攻击者编造的假消息。且

  $Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$，

- 当接收者收到这个消息：$M'=(Y_1 \| Y_2 \| \ldots \| Y_m)$

  则$\Delta(M')= Y_1 \oplus Y_2 \oplus \ldots \oplus Y_m$

  $= \Delta(M)$

所以：$C_K(M)=C_K(M')$ 通过了验证，攻击得逞。

MAC函数应具有的性质

- 若攻击者已知M和$C_K(M)$，则他构造满足：

  $C_K(M)=C_K(M')$的消息M'在计算上不可行
- $C_K(M)$应是均匀分布的，即对于随机消息M和M'，

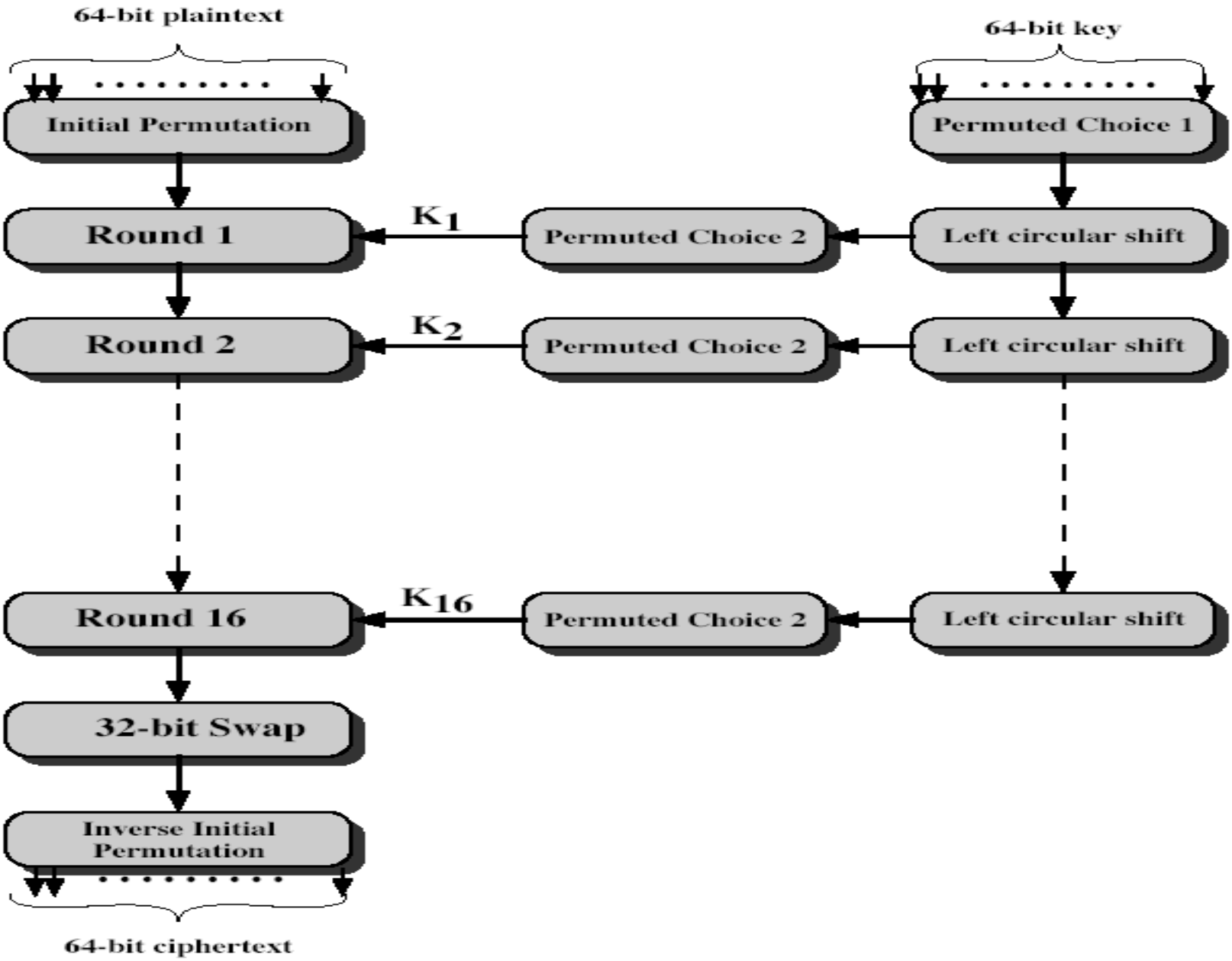
  $C_K(M)=C_K(M')$的概率是2^{-n}，n是MAC的位数
- 若M'是M的某个变换，即M'=f(M)，那么

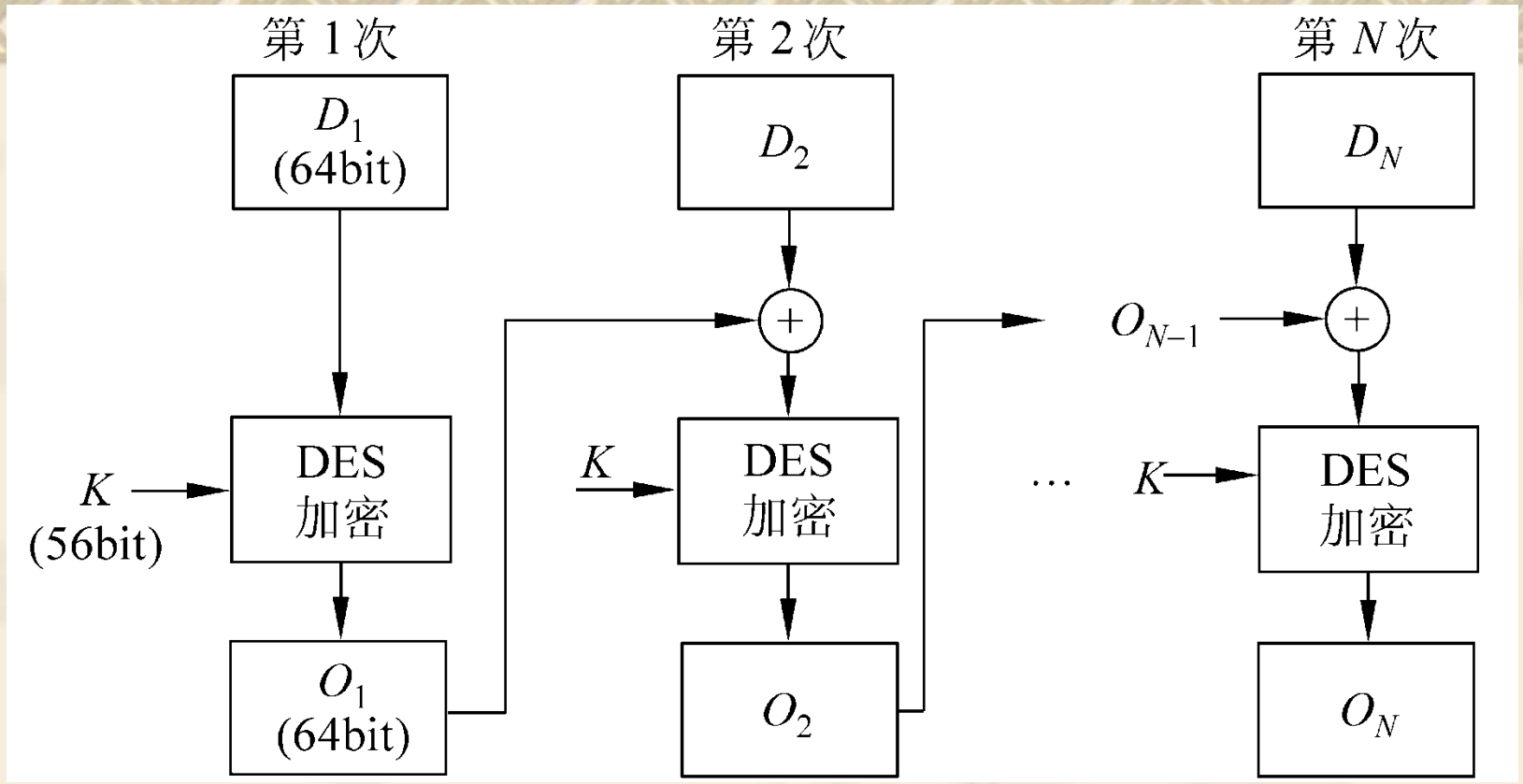
  $Pr[C_K(M)=C_K(M')]= 2^{-n}$。

5.3.3 基于DES的消息认证码

Message Authentication Code Based on DES

- 使用最广泛的MAC算法之一：数据认证算法
- 过程：
  - 把需要认证的数据分成连续的64位的分组。
  - 若最后一个分组不是64位，则填0
  - 利用DES加密算法E和密钥K，计算认证码。


64-bit plaintext
64-bit key
Initial Permutation
Permuted Choice 1
Round 1
K1
Permuted Choice 2
Left circular shift
Round 2
K2
Permuted Choice 2
Left circular shift
Round 16
K16
Permuted Choice 2
Left circular shift
32-bit Swap
Inverse Initial Permutation
64-bit ciphertext



$$O_1 = E_K(D_1)$$

$$O_2 = E_K(D_2 \oplus O_1)$$

$$O_3 = E_K(D_3 \oplus O_2)$$

$$\vdots$$

$$O_N = E_K(D_N \oplus O_{N-1})$$

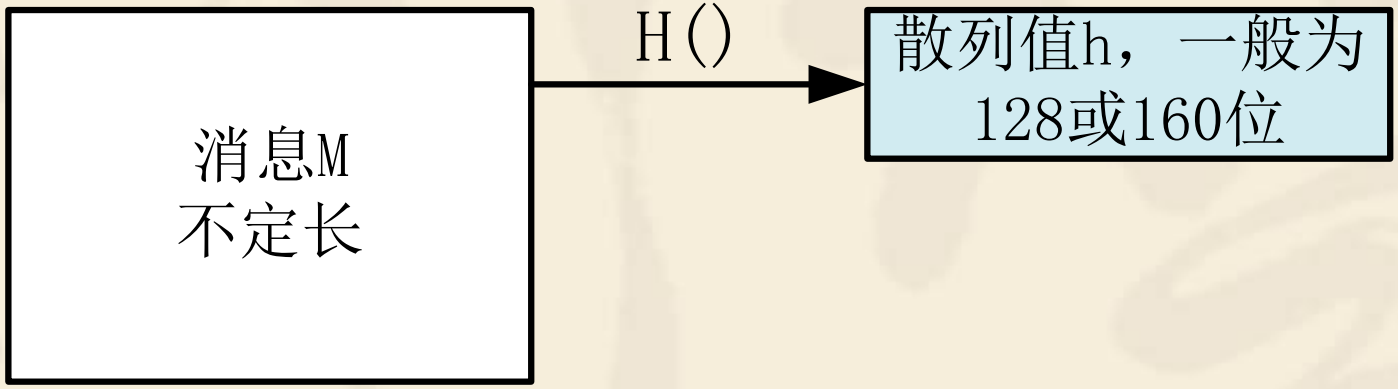
DAC M-bits
(16 to 64 bits)

数据认证算法似乎可以满足前面提出的要求。

5.4 Hash函数(杂凑函数、散列函数)

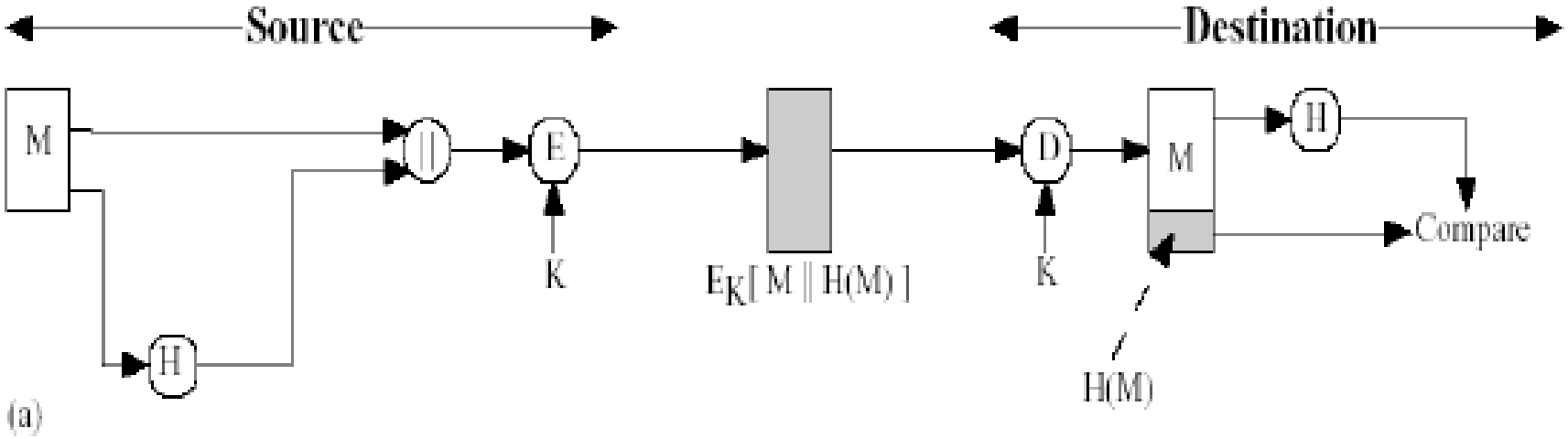
- Hash的特点：
  - 与消息认证码一样，hash函数的输入是可变的消息M，输出是固定大小的hash码H(M)，或称消息摘要**(Message Digest)**、hash值。
  - 与消息认证码不同的是， hash码的产生过程中并不使用密钥。
  - Hash码是所有消息的函数，改变消息的任何一位或多位，都会导致hash码的改变。
  - Hash算法通常是公开的。
  - 又称为：哈希函数、数字指纹（**Digital finger print)**、压缩（**Compression)**函数、紧缩（**Contraction** ）函数、数据鉴别码**DAC**（**Data authentication code**）、篡改检验码**MDC(Manipulation detection code)**

h=H(M)



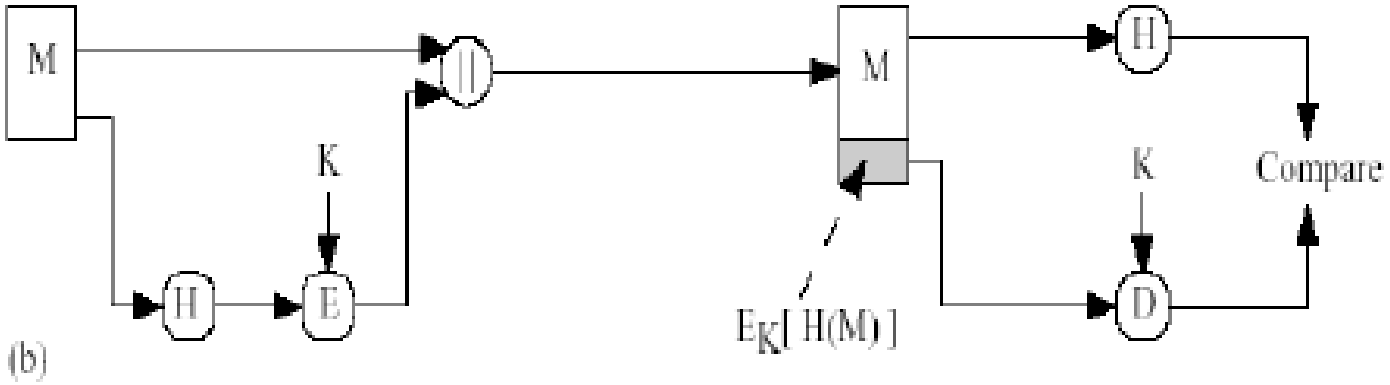
- 假定两次输入同样的数据，那么散列函数应该能够生成相同的散列值。输入数据中的一位发生了变化，会导致生成的散列值完全不一样。
- 散列函数有个非常重要的特性为单向性，也就是从M计算h容易，而从h计算M不可能。

散列函数的基本用法（a、b）



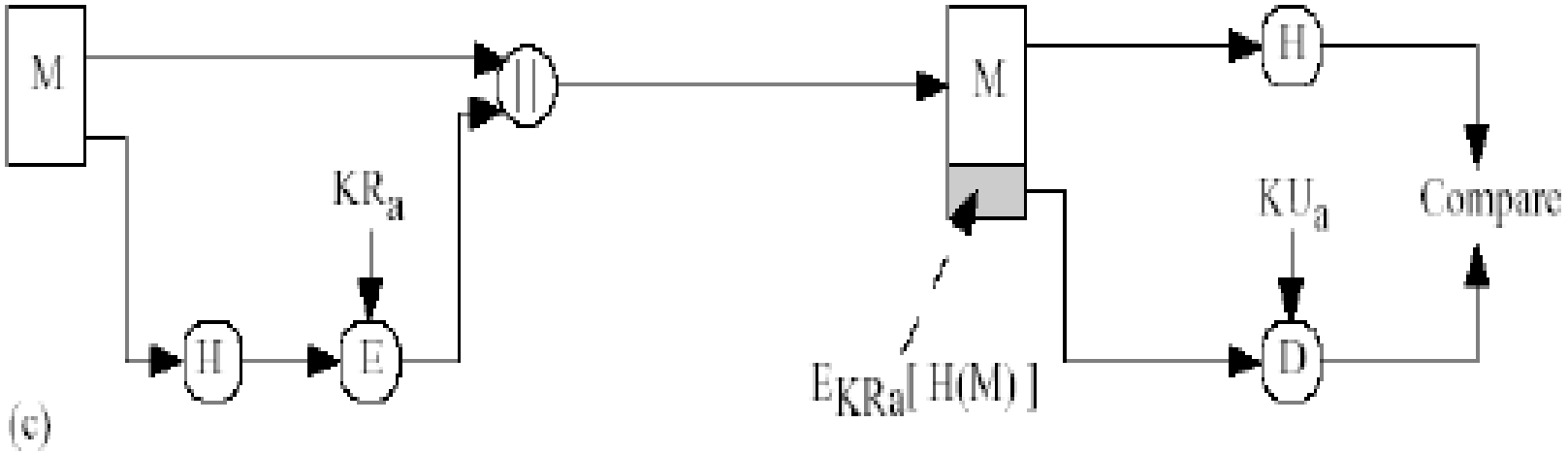
Provides confidentiality -- only A and B share K

Provides authentication -- H(M) is cryptographically protected



Provides authentication -- H(M) is cryptographically protected

散列函数的基本用法（c）

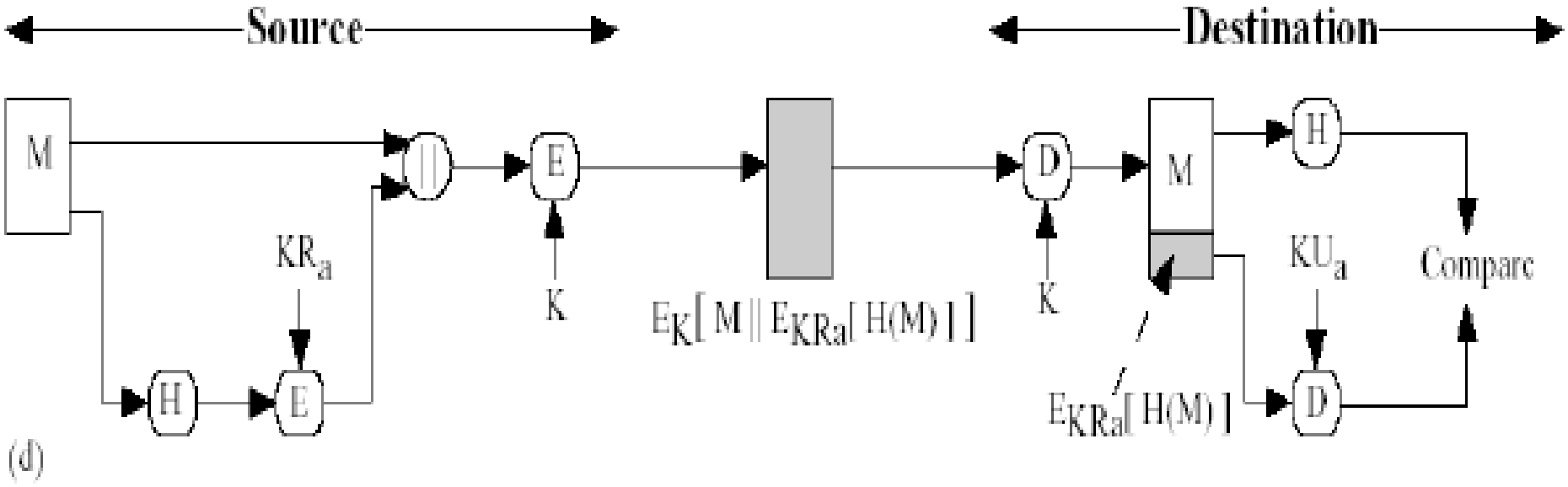


Provides authentication and digital signature

-- H(M) is cryptographically protected

-- only A could create $E_{KRa}[H(M)]$

散列函数的基本用法(d)

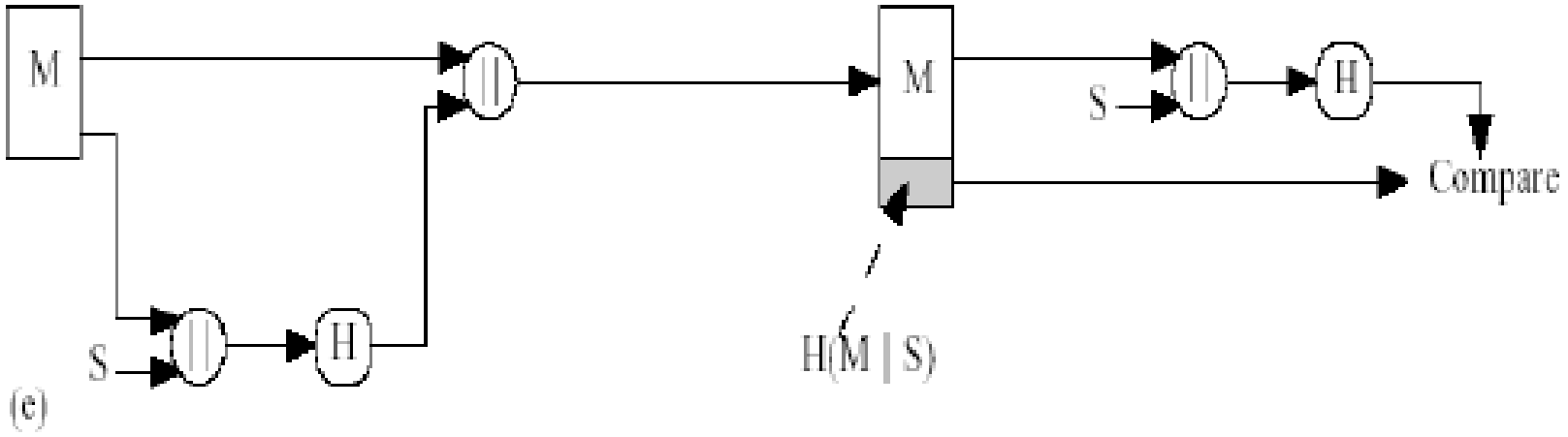


(d) A→B: $E_K[M||E_{KRa}[H(M)]]$

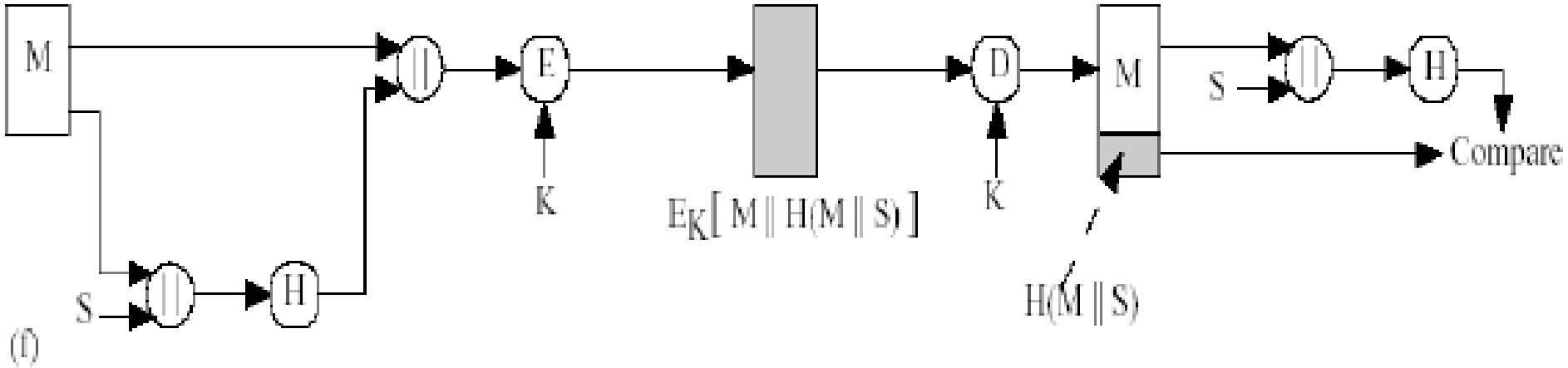
Provides authentication and digital signature

Provides confidentiality

散列函数的基本用法(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

方法e的优点

- 不含加密处理
  - 加密软件很慢
  - 加密硬件的开销很大
  - 加密硬件是对大长度数据进行优化的
  - 加密算法可能受专利保护
  - 加密算法可能受出口的限制.

对HASH函数h = H(M)的要求

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property（单向性）.

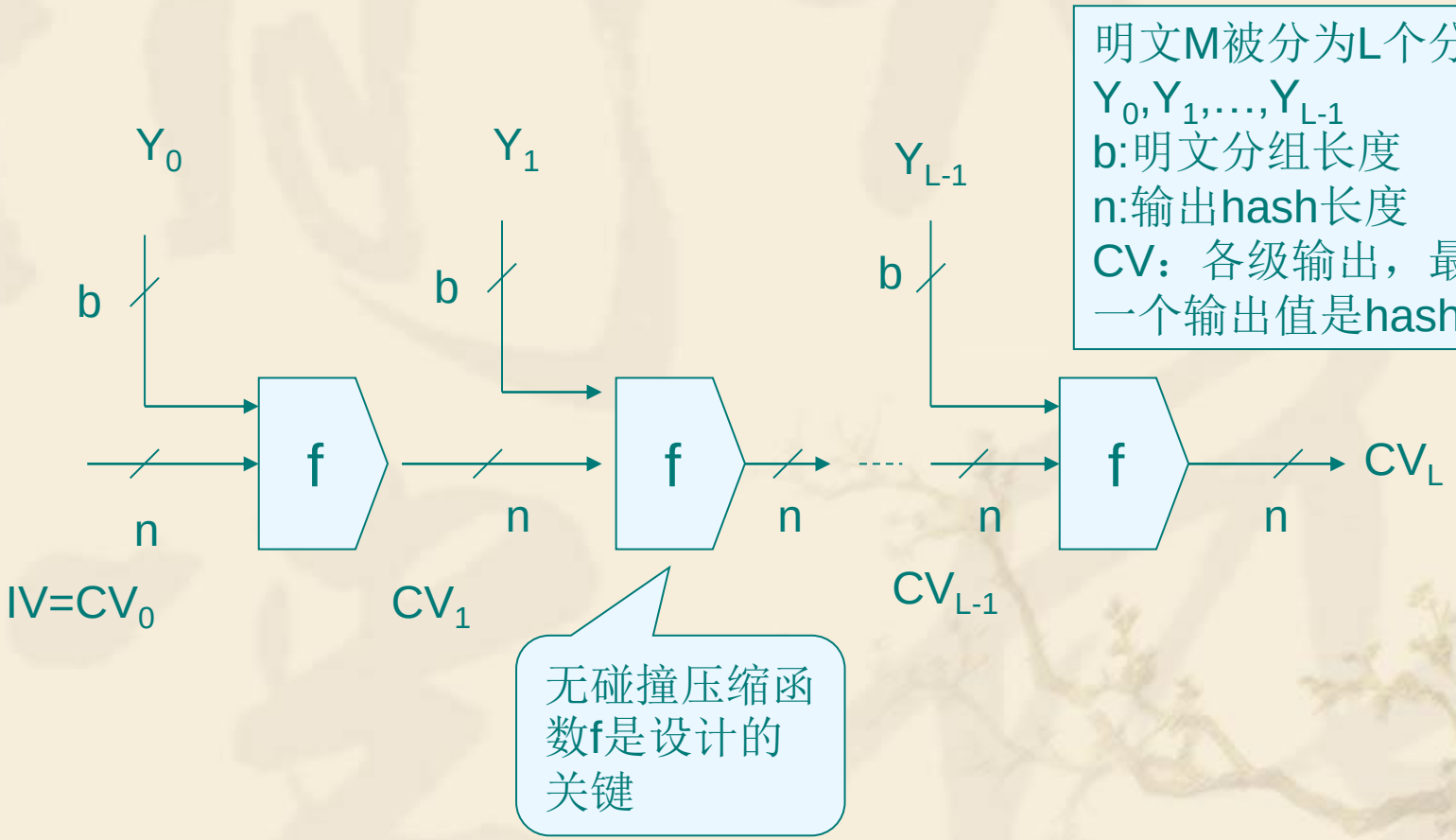
对HASH函数h = H(M)的要求

5. For any given block x, it is computationally infeasible to find y≠x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**（抗弱碰撞性）.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**（抗强碰撞性）.

- 前三条要求具有实用性
- 第4条是单向性质，即给定消息可以产生一个散列码，而给定散列码不可能产生对应的消息
- 第5条性质是保证一个给定的消息的散列码不能找到与之相同的另外的消息。即防止伪造。
- 第6条是对已知的生日攻击方法的防御能力,强无碰撞。

5.4.3 常用Hash算法

迭代型hash函数的一般结构



迭代型hash函数

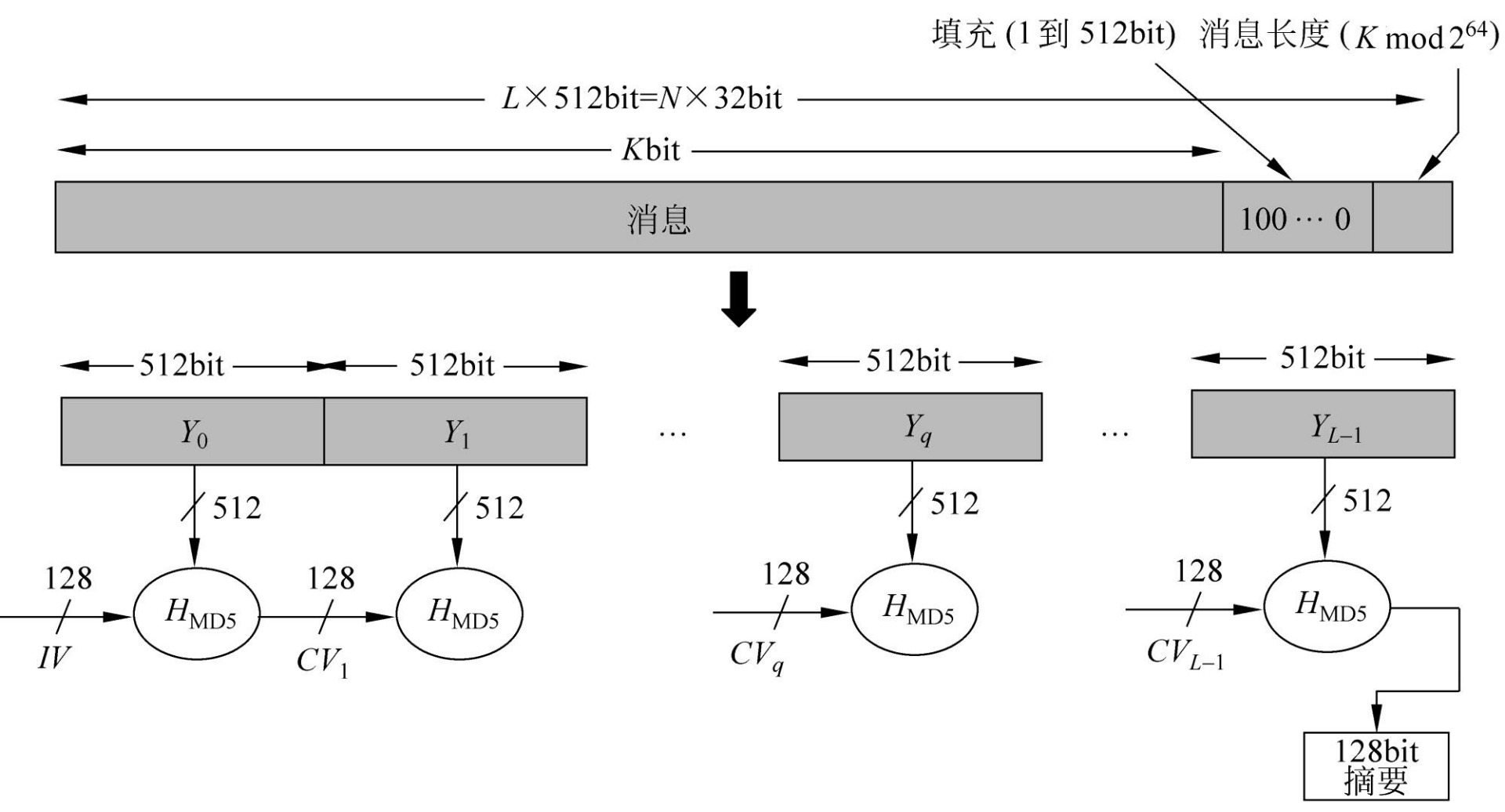
- 这种结构的hash函数已被证明是合理的，如果采用其他结构，不一定安全。
- 设计新的hash函数只是改进这种结构，或者增加hash码长。
- 算法的核心技术是设计无碰撞的压缩函数f，而敌手对算法的攻击重点是f 的内部结构，由于f 和分组密码一样是由若干轮处理过程组成，所以对f 的攻击需通过对各轮之间的位模式的分析来进行，分析过程常常需要先找出f 的碰撞。由于f 是压缩函数，其碰撞是不可避免的，因此在设计f 时就应保证找出其碰撞在计算上是不可行的。

MD5 hash算法
MD5 Hash Algorithm

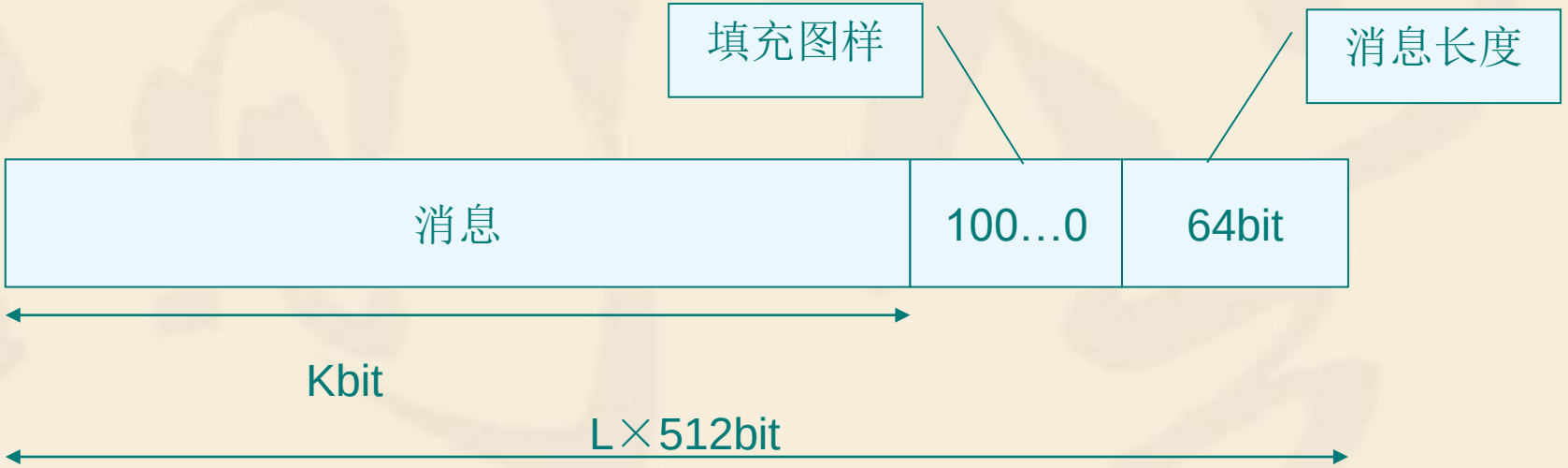
MD4是MD5杂凑算法的前身，由Ron Rivest于1990年10月作为RFC提出，1992年4月公布的MD4的改进（RFC 1320，1321）称为MD5。

MD5的算法框图

- 输入消息可任意长，压缩后输出为128bits。



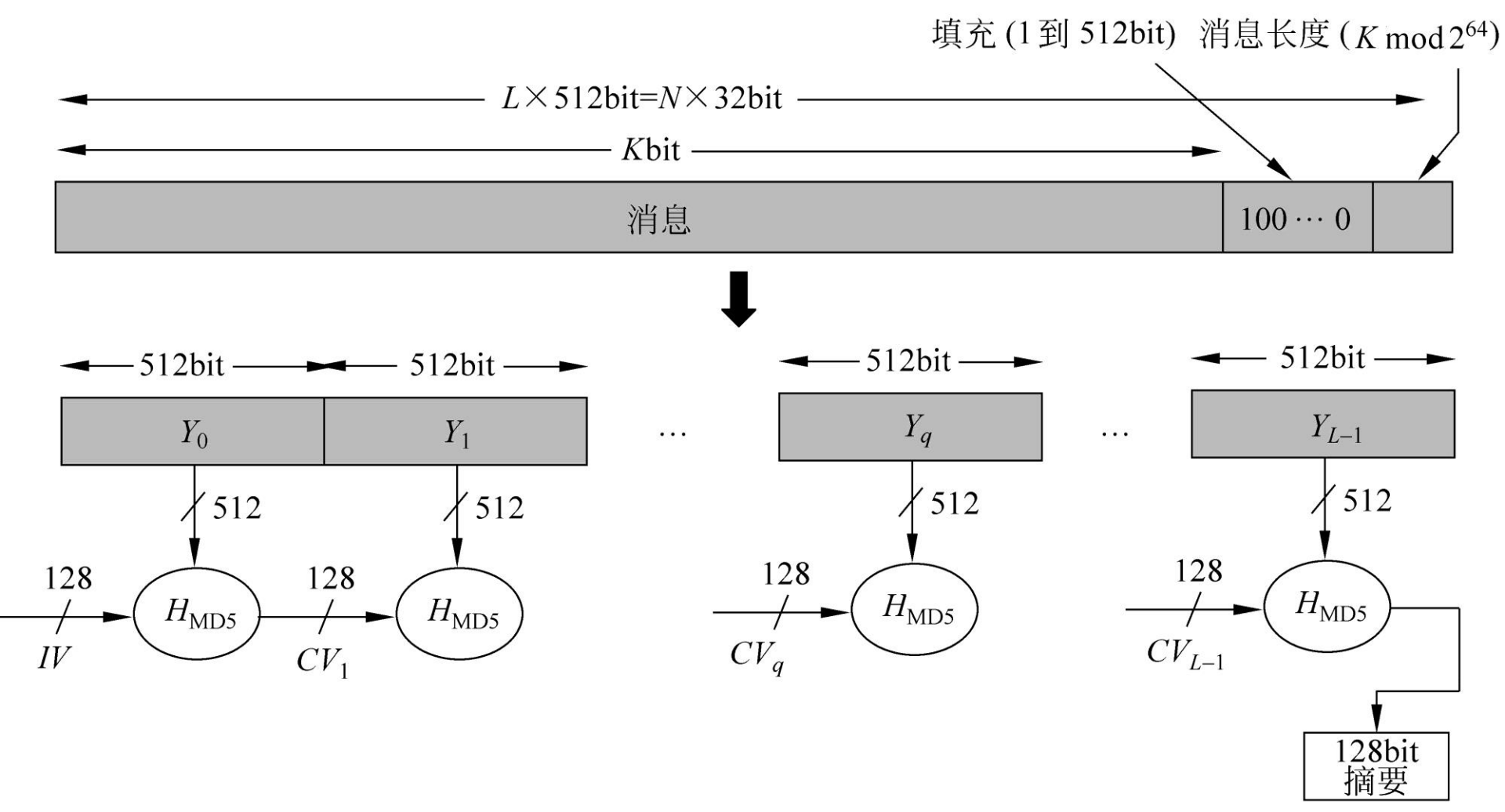
算法步骤(1)—分组填充



- 如果消息长度大于2^{64}，则取其对2^{64}的模。
- 执行完后，消息的长度为512的倍数（设为L倍），则可将消息表示为分组长为512的一系列分组Y_0，Y_1，…，Y_{L-1}，而每一分组又可表示为16个32比特长的字，这样消息中的总字数为N=L×16，因此消息又可按字表示为M[0,…,N-1]。

MD5的算法框图

- 输入消息可任意长，压缩后输出为128bits。



算法步骤(2)—缓冲区初始化

hash函数的中间结果和最终结果保存于128位的缓冲区中，缓冲区用32位的寄存器表示。可用4个32bits字表示：*A,B,C,D*。初始存数以十六进制表示为

A=01234567

B=89*ABCDEF*

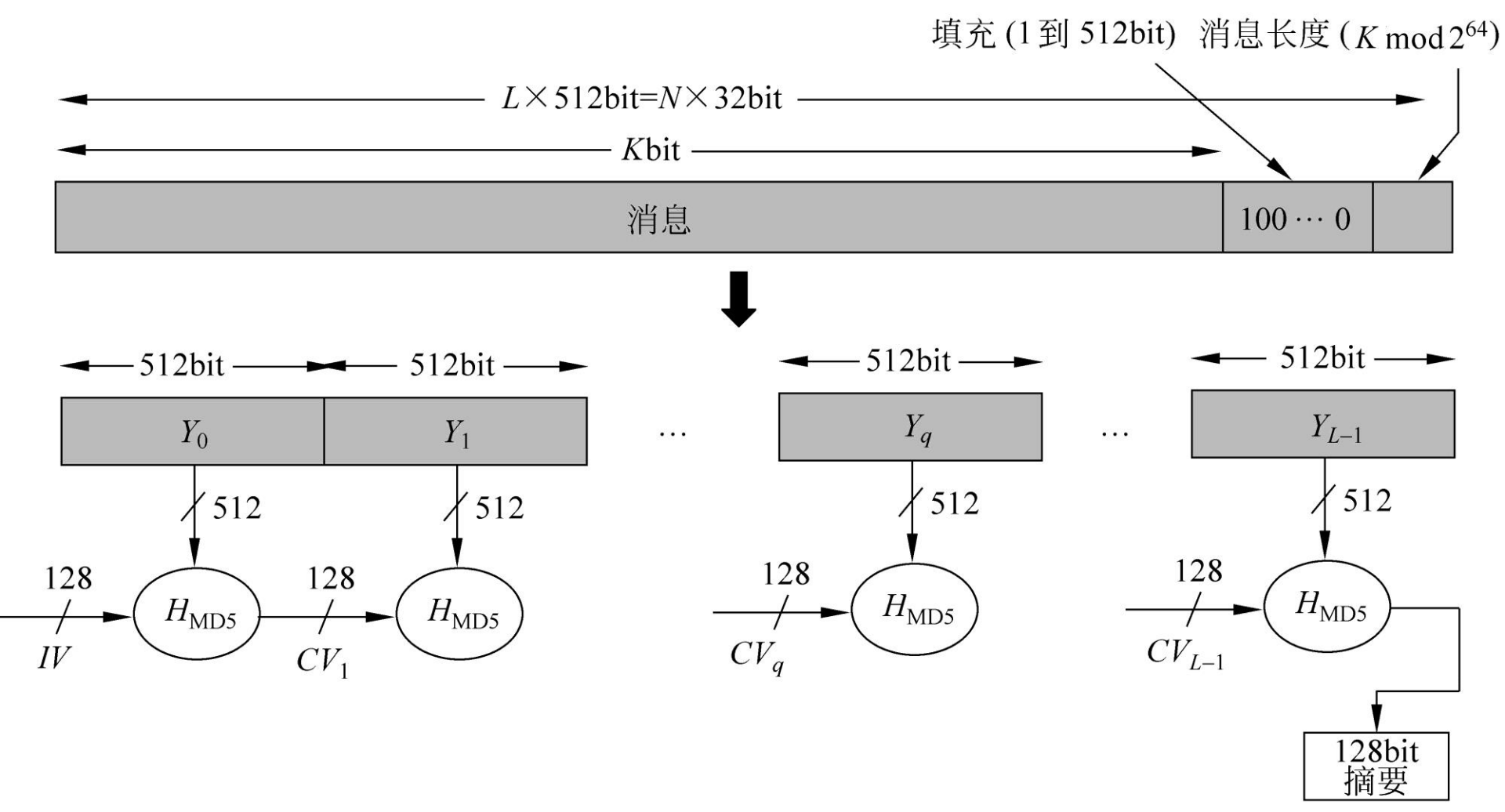
C=*FEDCBA*98

D=76543210

```
A = 01 23 45 67 (0x67452301)
B = 89 AB CD EF (0xEFCDAB89)
C = FE DC BA 98 (0x98BADCFE)
D = 76 54 32 10 (0x10325476)
```

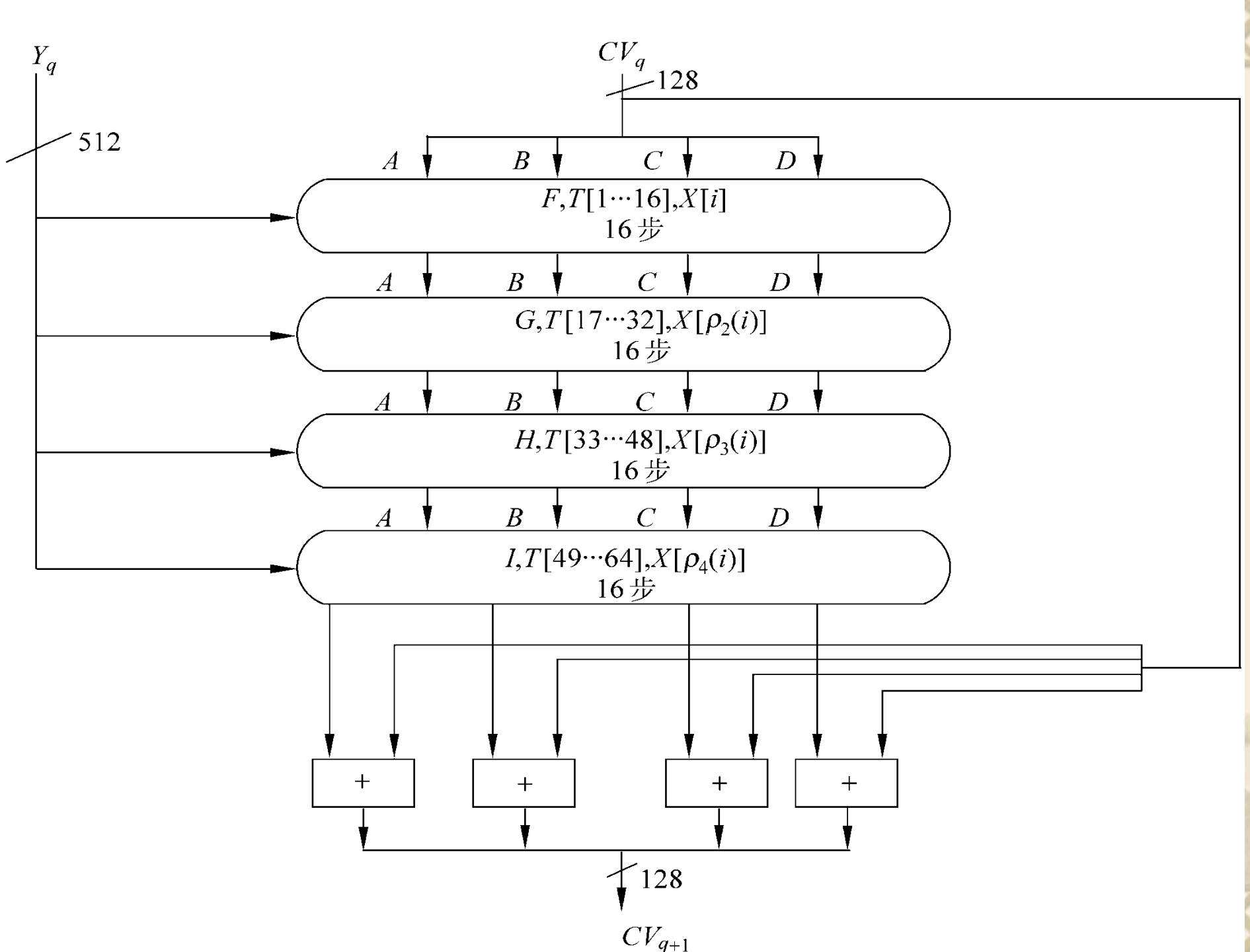
MD5的算法框图

- 输入消息可任意长，压缩后输出为128bits。

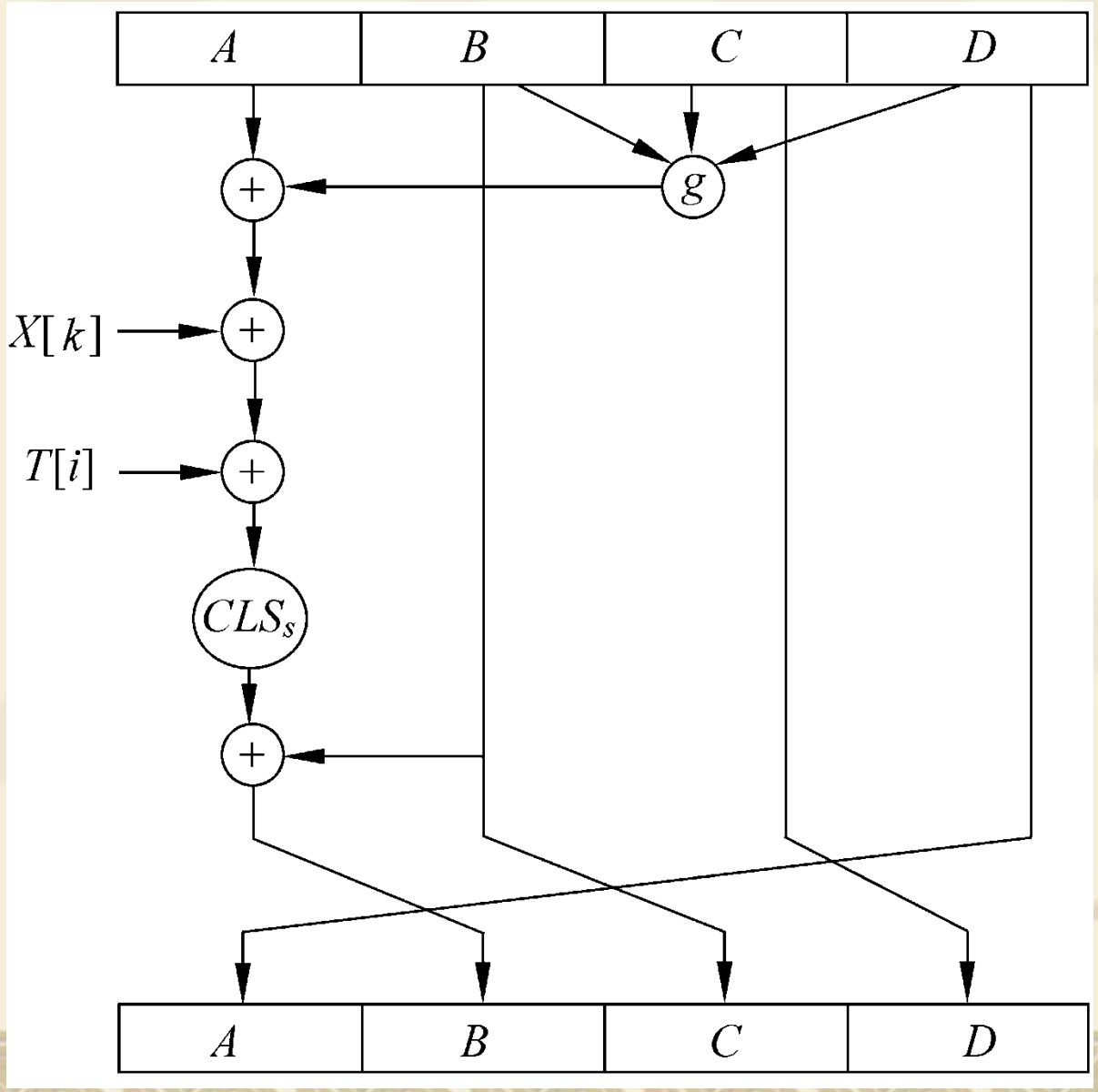


算法步骤(3) -H_{MD5}运算

- 以分组为单位对消息进行处理每一分组Y_q(q=0,…,L-1)都经一压缩函数H_{MD5}处理。H_{MD5}是算法的核心，其中又有4轮处理过程。
- H_{MD5}的4轮处理过程结构一样，但所用的逻辑函数不同，分别表示为F、G、H、I。每轮的输入为当前处理的消息分组Y_q和缓冲区的当前值A、B、C、D，输出仍放在缓冲区中以产生新的A、B、C、D。
- 每轮又要进行16步迭代运算，4轮共需64步完成。
- 第四轮的输出与第一轮的输入相加得到最后的输出。


Y_q
512
CV_q
128
A
B
C
D
F,T[1…16],X[i]
16步
A
B
C
D
G,T[17…32],X[ρ2(i)]
16步
A
B
C
D
H,T[33…48],X[ρ3(i)]
16步
A
B
C
D
I,T[49…64],X[ρ4(i)]
16步
+
+
+
+
128
CV_{q+1}

压缩函数中的一步迭代



基本逻辑函数定义

轮	基本函数 g	g(b, c, d)
f_F	F(b,c,d)	$(b \wedge c) \vee (\bar{b} \wedge d)$
f_G	G(b,c,d)	$(b \wedge d) \vee (c \wedge \bar{d})$
f_H	H(b,c,d)	$b \oplus c \oplus d$
f_I	I(b,c,d)	$c \oplus (b \vee \bar{d})$

X[k]

- 当前分组的第k个32位的字。

第1轮	x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]	x[9]	x[10]	x[11]	x[12]	x[13]	x[14]	x[15]
第2轮	x[1]	x[6]	x[11]	x[0]	x[5]	x[10]	x[15]	x[4]	x[9]	x[14]	x[3]	x[8]	x[13]	x[2]	x[7]	x[12]
第3轮	x[5]	x[8]	x[11]	x[14]	x[1]	x[4]	x[7]	x[10]	x[13]	x[0]	x[3]	x[6]	x[9]	x[12]	x[15]	x[2]
第4轮	x[0]	x[7]	x[14]	x[5]	x[12]	x[3]	x[10]	x[1]	x[8]	x[15]	x[6]	x[13]	x[4]	x[11]	x[2]	x[9]

T[i]

- T[1,...,64]为64个元素表，分四组参与不同轮的计算。T[i]为$2^{32}\times abs(Sin(i))$的整数部分，i是弧度。T[i]可用32 bit二元数表示，T是32 bit随机数源。T[i]参与运算的作用是消除输入数据的规律性。

T[1]= d76aa478	T[17]= f61e2562	T[33]= fffa3942	T[49]= f4292244
T[2]= e8c7b756	T[18]= c040b340	T[34]= 8771f681	T[50]= 432aff97
T[3]= 242070db	T[19]= 265e5a51	T[35]= 6d9d6122	T[51]= ab9423a7
T[4]= c1bdceee	T[20]= e9b6c7aa	T[36]= fde5380c	T[52]= fc93a039
T[5]= f57c0faf	T[21]= d62f105d	T[37]= a4beea44	T[53]= 655b59c3
T[6]= 4787c62a	T[22]= 02441453	T[38]= 4bdecfa9	T[54]= 8f0ccc92
T[7]= a8304613	T[23]= d8a1e681	T[39]= f6bb4b60	T[55]= ffeff47d
T[8]= fd469501	T[24]= e7d3fbc8	T[40]= bebfbc70	T[56]= 85845dd1
T[9]= 698098d8	T[25]= 21e1cde6	T[41]= 289b7ec6	T[57]= 6fa87e4f
T[10]= 8b44f7af	T[26]= c33707d6	T[42]= eaa127fa	T[58]= fe2ce6e0
T[11]= ffff5bb1	T[27]= f4d50d87	T[43]= d4ef3085	T[59]= a3014314
T[12]= 895cd7be	T[28]= 455a14ed	T[44]= 04881d05	T[60]= 4e0811a1
T[13]= 6b901122	T[29]= a9e3e905	T[45]= d9d4d039	T[61]= f7537e82
T[14]= fd987193	T[30]= fcefa3f8	T[46]= e6db99e5	T[62]= bd3af235
T[15]= a679438e	T[31]= 676f02d9	T[47]= 1fa27cf8	T[63]= 2ad7d2bb
T[16]= 49b40821	T[32]= 8d2a4c8a	T[48]= c4ac5665	T[64]= eb86d391

CLS_s：循环左移s位

- 第一轮：7、12、17、22
- 第二轮：5、9、14、20
- 第三轮：4、11、16、23
- 第四轮：6、10、15、21

MD-5的安全性

- MD-5的输出为128-bit，若采用纯强力攻击寻找一个消息具有给定Hash值的计算困难性为2^{128}，用每秒可试验1 000 000 000个消息的计算机需时1.07×10^{22}年。
- 采用生日攻击法，找出具有相同hash值的两个消息需执行2^{64}次运算。

碰撞

- 如果两个输入串的hash函数的值一样，则称这两个串是一个**碰撞(Collision)**。既然是把任意长度的字符串变成固定长度的字符串，所以，必有一个输出串对应无穷多个输入串，碰撞是必然存在的。
- 2004年8月17日，美国加州圣巴巴拉正在召开国际密码学会议，山东大学王小云教授公布了快速寻求MD5算法碰撞的算法。

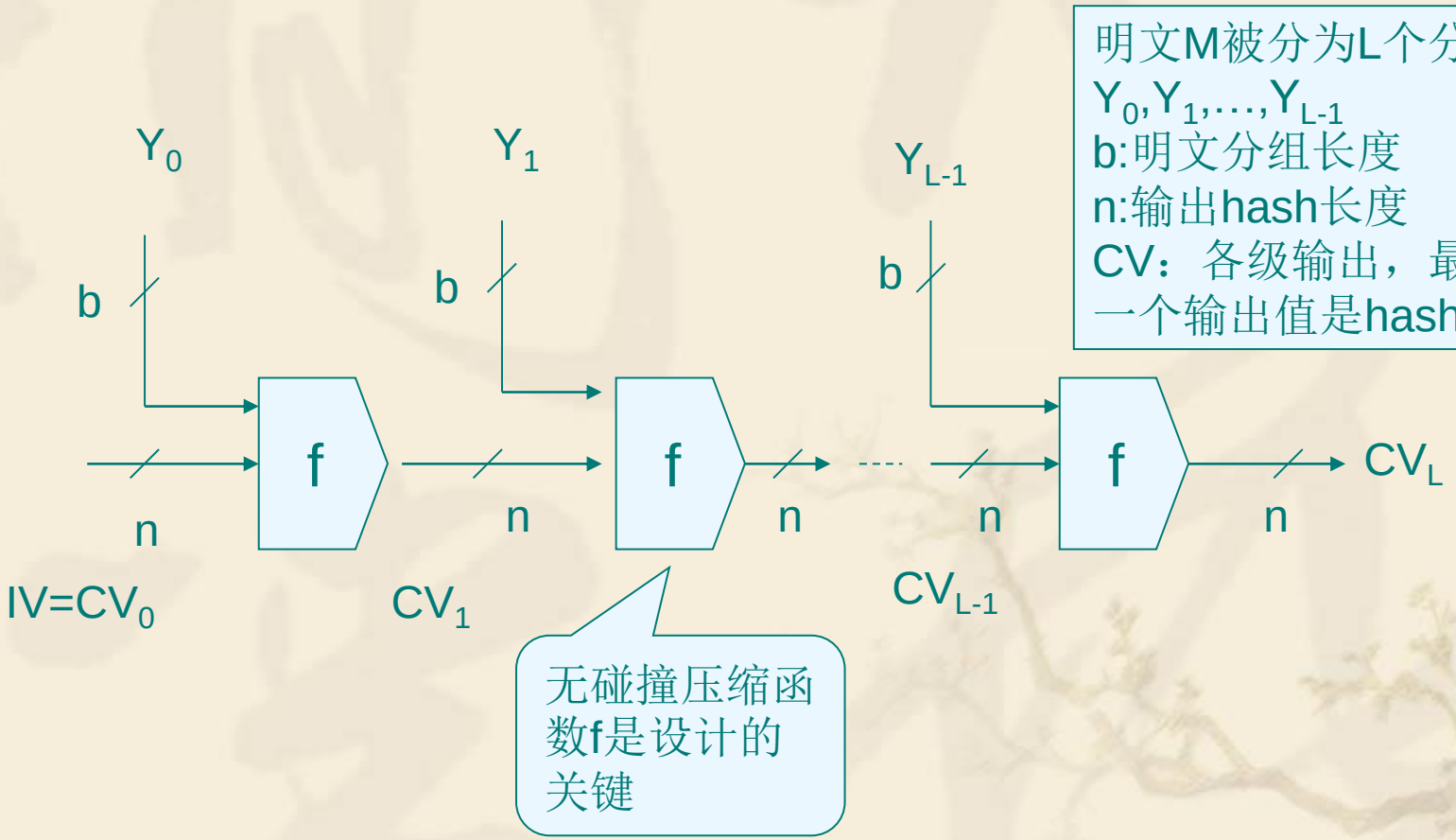
SHA 算法

Secure Hash Algorithm

算法简介

- 美国标准与技术研究所NIST设计
- 1993年成为联邦信息处理标准(FIPS PUB 180)
- 基于MD4算法，与之非常类似。
- 输入为小于2^{64}比特长的任意消息
- 分组512bit长
- 输出160bit

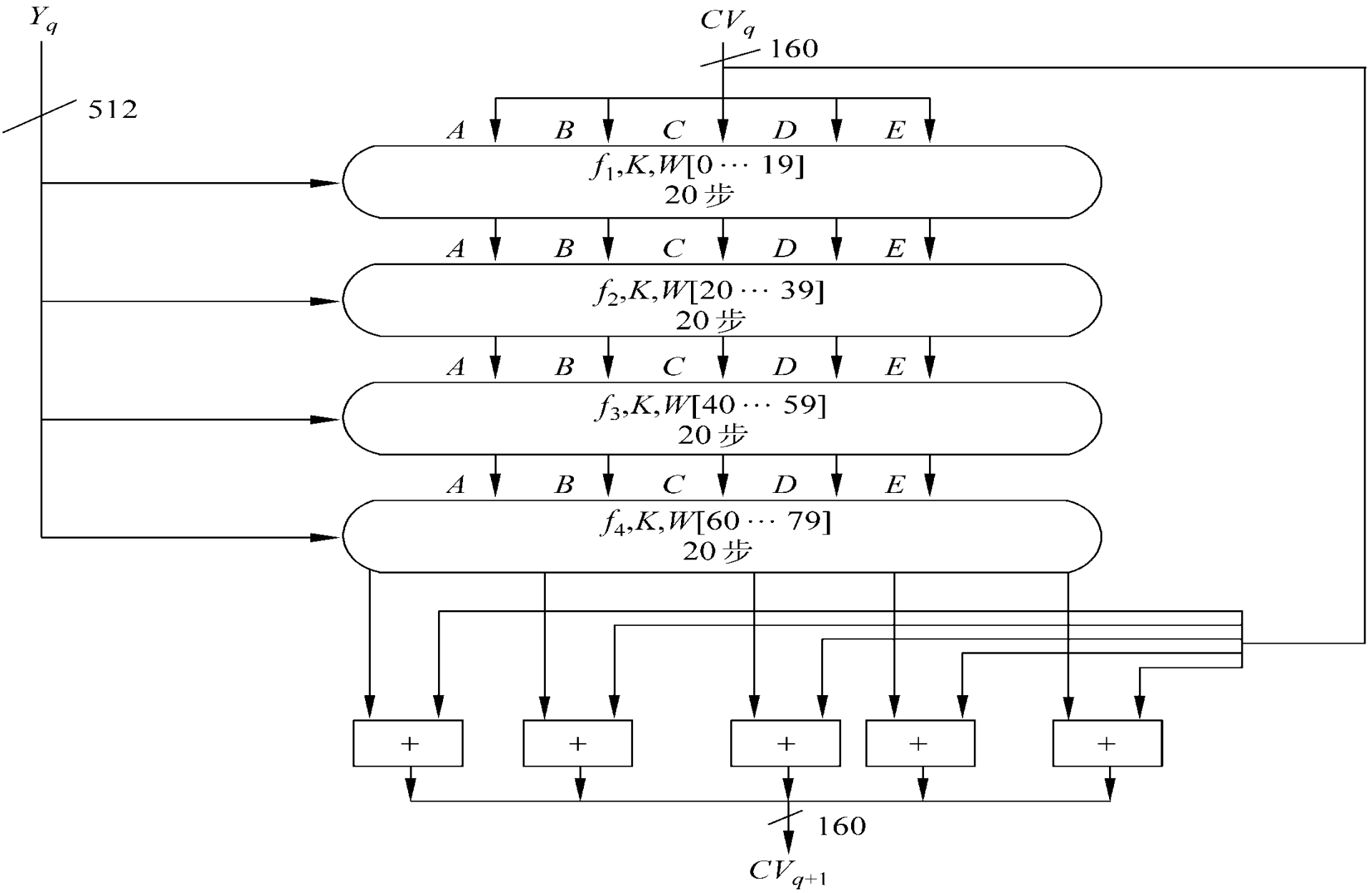
迭代型hash函数的一般结构



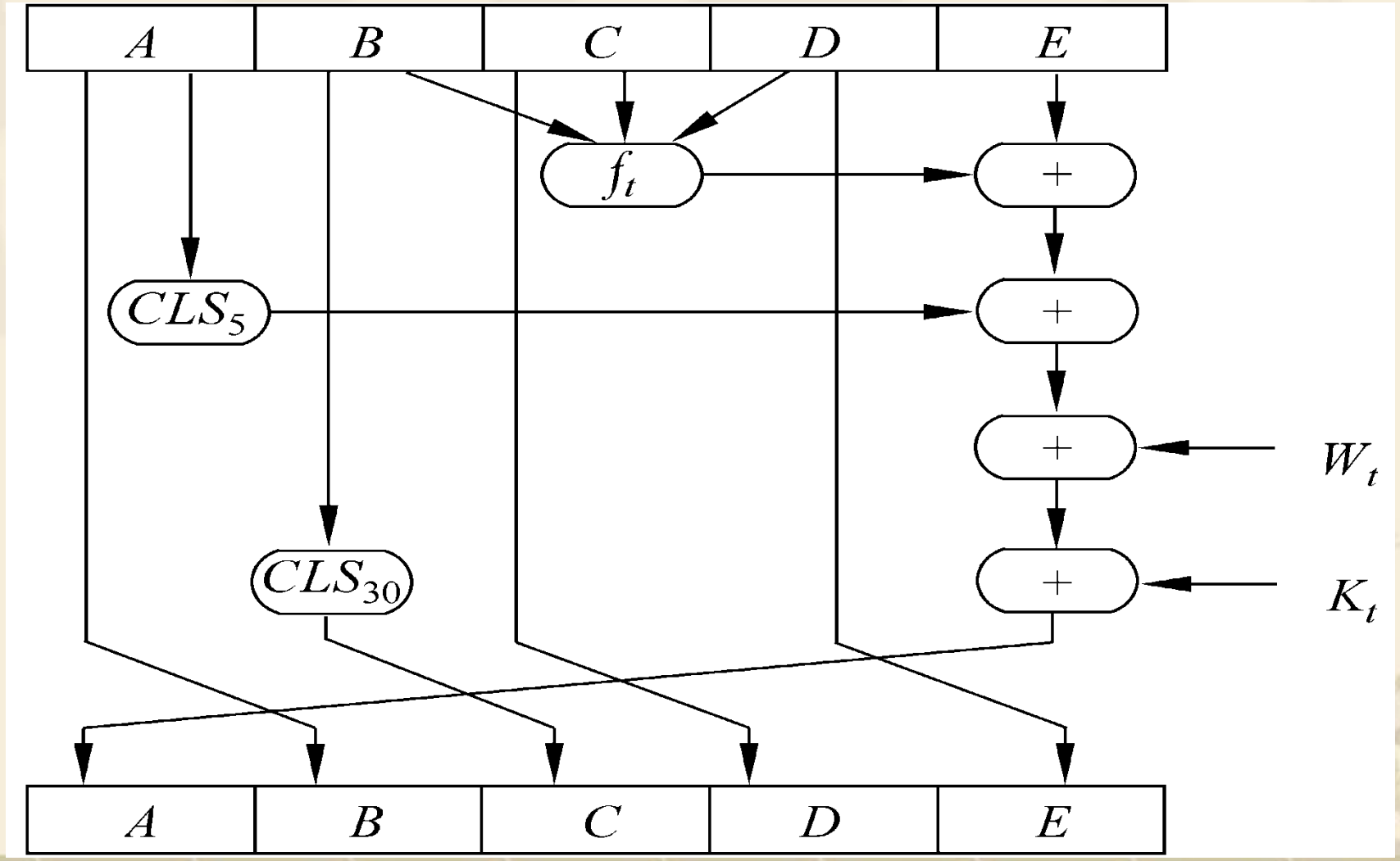
算法描述

- 消息填充：与MD5完全相同
- 缓冲区初始化
  - A = 67452301
  - B = EFCDAB89
  - C = 98BADCFB
  - D = 10325476
  - E = C3D2E1F0

分组处理


Y_q
512
CV_q
160
A B C D E
$f_1, K, W[0 \cdots 19]$
20 步
A B C D E
$f_2, K, W[20 \cdots 39]$
20 步
A B C D E
$f_3, K, W[40 \cdots 59]$
20 步
A B C D E
$f_4, K, W[60 \cdots 79]$
20 步
+
+
+
+
+
160
CV_{q+1}

SHA-1压缩函数（单步）

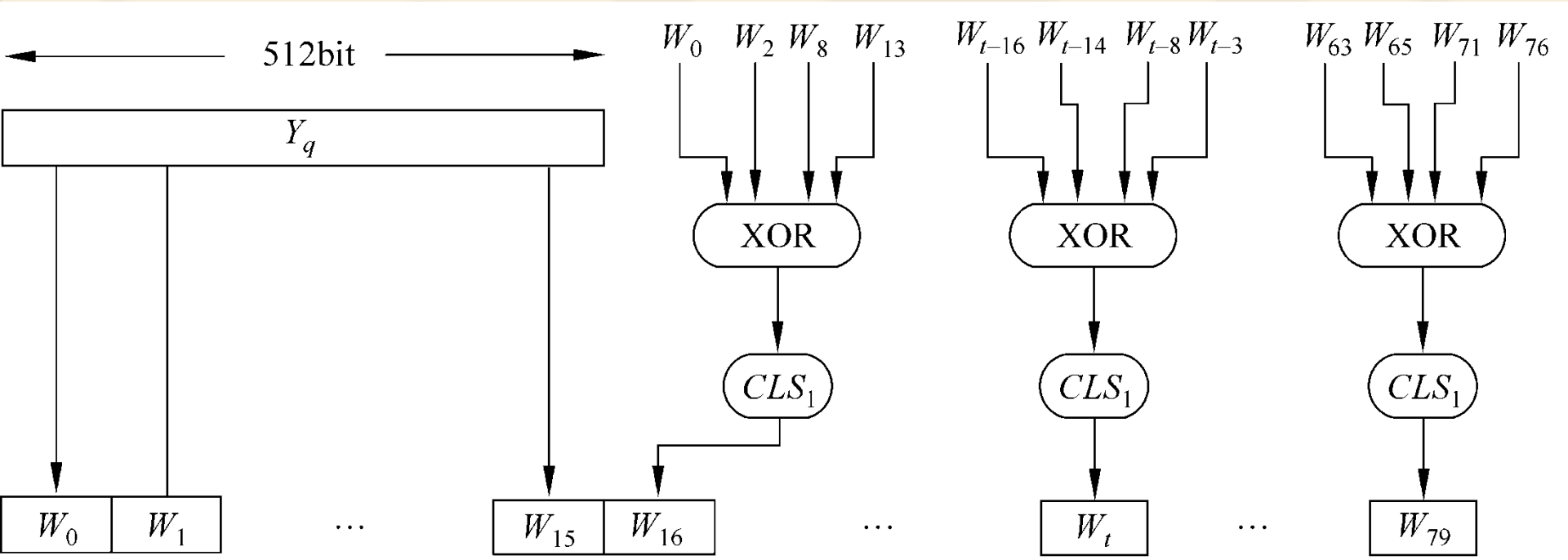


f_t----基本逻辑函数

轮	基本函数	函数值
1	$f_1(B,C,D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
2	$f_2(B,C,D)$	$B \oplus C \oplus D$
3	$f_3(B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	$f_4(B,C,D)$	$B \oplus C \oplus D$

- CLS_5：32位的变量循环左移5位。
- CLS_{30}：32位的变量循环左移30位。

W_t ---从当前512位输入分组导出的32位字



- 前16个值（即$W_0,W_1,\ldots,W_{15}$）直接取为输入分组的16个相应的字，其余值（即$W_{16},W_{17},\ldots,W_{79}$）取为

$$W_t = CLS_1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$

K_t ---加法常量

步骤	十六进制
$0 \leqslant t \leqslant 19$	K_t=5A827999
$20 \leqslant t \leqslant 39$	K_t=6ED9EBA1
$40 \leqslant t \leqslant 59$	K_t=8F1BBCDC
$60 \leqslant t \leqslant 79$	K_t=CA62C1D6

SHA与MD5的比较

- 抗穷举搜索能力
  - 寻找指定hash值， SHA：$O(2^{160})$，MD5：$O(2^{128})$
  - 生日攻击： SHA：$O(2^{80})$，MD5：$O(2^{64})$
- 抗密码分析攻击的强度
  - SHA似乎高于MD5
- 速度
  - SHA较MD5慢
- 简捷与紧致性
  - 描述都比较简单，都不需要大的程序和代换表

其它hash算法

- MD4
  - MD4使用三轮运算，每轮16步； MD5使用四轮运算，每轮16步。
  - MD4的第一轮没有使用加法常量，第二轮运算中每步迭代使用的加法常量相同，第三轮运算中每步迭代使用的加法常量相同，但不同于第二轮使用的加法常量；MD5的64部使用的加法常量T[i]均不同。
  - MD4使用三个基本逻辑函数，MD5使用四个。
  - MD5中每步迭代的结果都与前一步的结果相加，MD4则没有。
  - MD5比MD4更复杂，所以其执行速度也更慢，Rivest认为增加复杂性可以增加安全性。

RIPEMD-160

- 欧共体RIPE项目组研制。
- 输入可以是任意长的报文，输出160位摘要。
- 对输入按512位分组。以分组为单位处理。
- 算法的核心是具有十轮运算的模块，十轮运算分成两组，每组五轮，每轮16步迭代。

5.4.4 对Hash函数的攻击

- 对一个hash算法的攻击可分三个级别：
  - 预映射攻击（Preimage Attack）：给定Hash值h，找到其所对应的明文M，使得Hash(M)=h，这种攻击是最彻底的，如果一个hash算法被人找出预映射，那这种算法是不能使用的。
  - 次预映射攻击（Second Preimage Attack）：给定明文M1，找到另一明文M2（M1≠M2），使得hash(M1)=hash(M2)，这种攻击其实就是要寻找一个弱碰撞；
  - 碰撞攻击 (Collision Attack)：找到M1和M2，使得hash(M1)=hash(M2) ，这种攻击其实就是要寻找一个强碰撞。

- 给定一个散列函数**H和某hash值H(x)，假定H有n个可能的输出。**如果**H有k个**随机输入**, k必须为多大才能使至少存在**一个输入**y,使得H(y)=H(x)**的概率大于**0.5?**

结论

- 如果hash码为m位，则有2^m个可能的hash码。
- 如果给定h=H(X)，要想找到一个y，使H(y) =h的概率为0.5，则要进行多次的尝试，尝试的次数 $k=2^m/2=2^{m-1}$
- 所以，对于一个使用64位的hash码，攻击者要想找到满足H(M')=H(M)的M'来替代M，平均来讲，他要找到这样的消息大约要进行2^{63}次尝试。
- 但是，存在一种攻击，称为“生日攻击”，却可以大大减小尝试的次数，对于64位的hash码，所需的代价仅为2^{32}次。

生日悖论

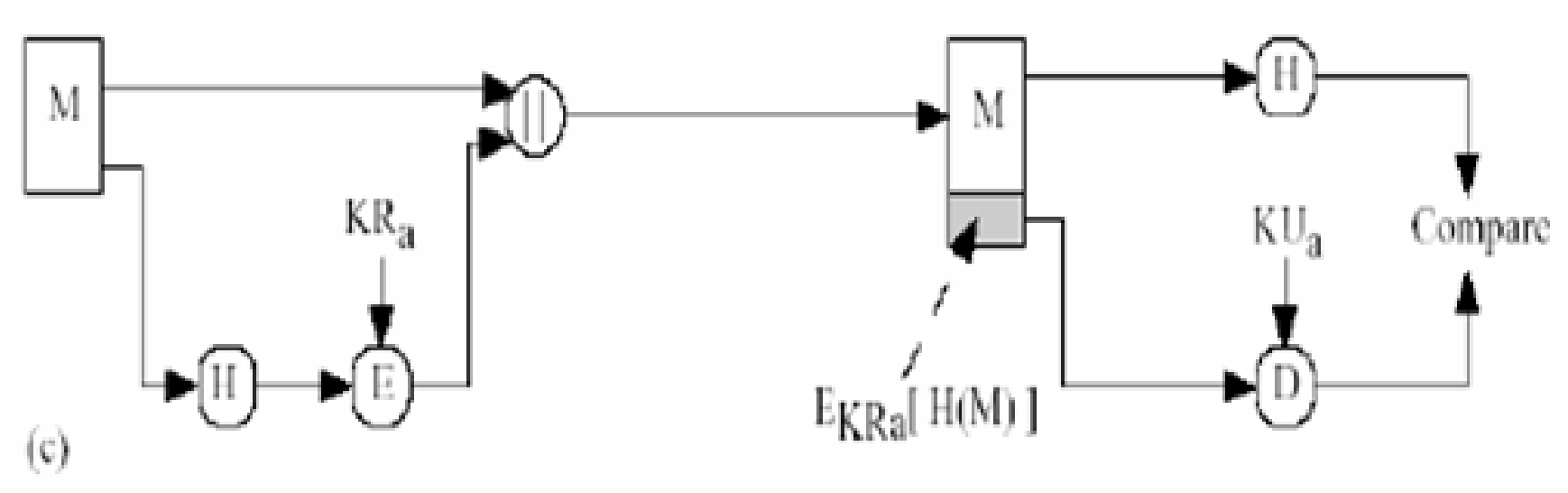
- 一个教室中，最少应有多少学生，才使至少有两人具有相同生日的概率不小于**1/2**？

生日悖论

- 概率结果与人的直觉是相违背的.
- 实际上只需**23**人,即任找**23**人，从中总能选出两人具有相同生日的概率至少为**1/2**。
- 即P(365,23)=0.5073.
  P(365,100)=0.9999997

实施生日攻击

- 前面提到过，对于一个使用64位的hash码，攻击者要想找到满足H(M')=H(M)的M'来替代M，平均来讲，他要找到这样的消息大约要进行2^{63}次尝试。这太困难了！



- 给定一个散列函数，有n个可能的输出，输出值为H(x)，如果H有k个随机输入，k必须为多大才能使至少存在一个输入y，使得H(y)=H(x)的概率大于0.5.
- 对单个y，H(y)=H(x)的概率为1/n，反过来H(y)≠H(x)的概率为1-(1/n).
- 如果产生k个随机值y，他们之间两两不等的概率等于每个个体不匹配概率的乘积，即$[1-(1/n)]^k$，这样，至少有一个匹配的概率为$1-[1-(1/n)]^k \approx 1-[1-(k/n)]=k/n$.要概率等于0.5，只需k=n/2.
- 对长度为m位的散列码，共有2^m个可能的散列码，若要使任意的x、y 有H(x)=H(y)的概率为0.5，只需$k= 2^{m/2}$.

- 设M和hash算法生成64位的hash值。
- 攻击者可以根据M，产生2^{32}个表达相同含义的变式(例如在词与词之间多加一个空格)。
- 同时准备好伪造的消息M’，产生2^{32}个表达相同含义的变式。
- 在这两个集合中，找出产生相同hash码的一对消息M_1和M_1’。根据生日悖论，找到这样一对消息的概率大于0.5。
- 最后，攻击者将拿M_1给发送者签名，但发送时，把M_1’和经加密的hash码一起发送。

Birthday Attacks: example

- A准备两份合同M和M′，一份B会同意，一份会取走他的财产而被拒绝
- A对M和M′各做32处微小变化(保持原意),分别产生232个64位hash值
- 根据前面的结论,超过0.5的概率能找到一个M和一个M′,它们的hash值相同
- A提交M,经B审阅后产生64位hash值并对该值签名,返回给A
- A用M′替换M