



密码学基础

汪 定

南开大学 网络空间安全学院

2021年11月6日



提 纲

CONTENTS



1. 加密算法

2. 消息认证

3. 数字签名

密码算法与密钥

□ 密码系统： 密码算法 + 密钥

□ 密码算法

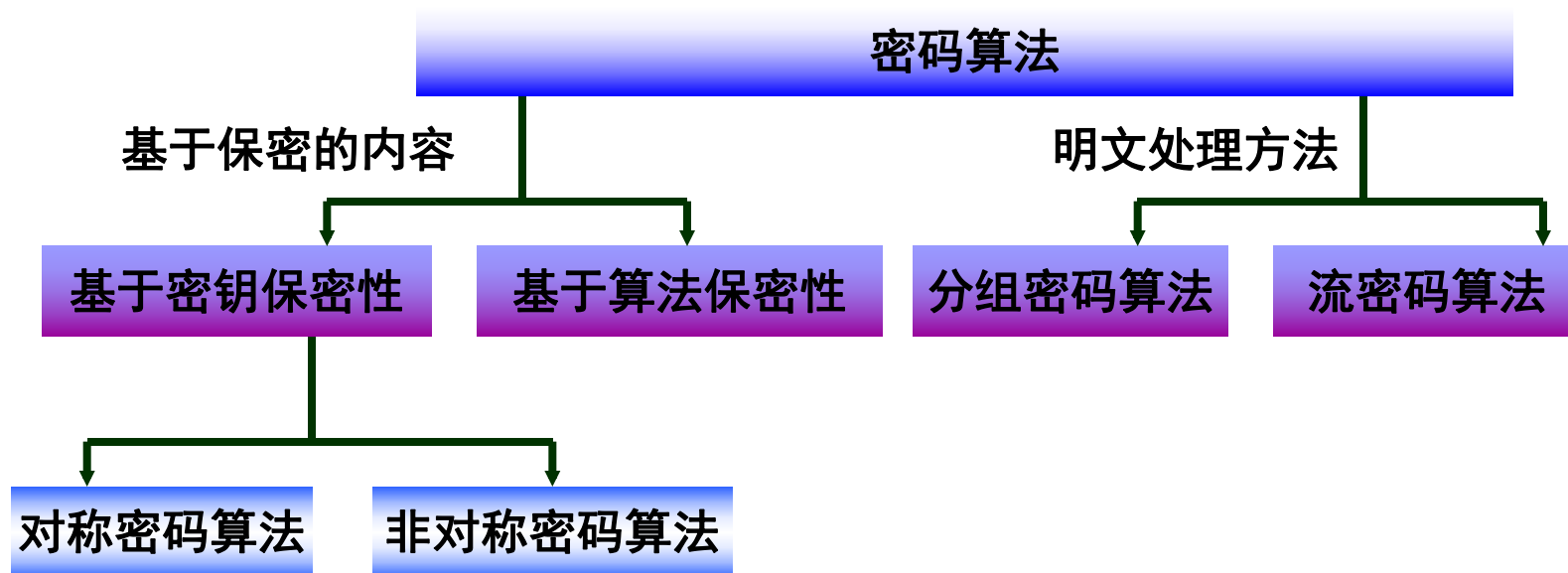
- 加解密算法， Hash函数， 数字签名等

□ 密钥

- 长密钥， 如军事中的流密钥； $Len > 8192$ bits； 密钥枪， 密钥盘，“密码本”
- 中长密钥， 如商用中的公钥加密算法的密钥； $len \in [512, 8192]$ ； **USB Key**
- 中等密钥， 如商用中的对称加密算法的密钥； $len \in (128, 512)$ ； **USB Key**
- 短密钥， 如Password

40亿网民可感知、直接
接触最多的密码学。脑

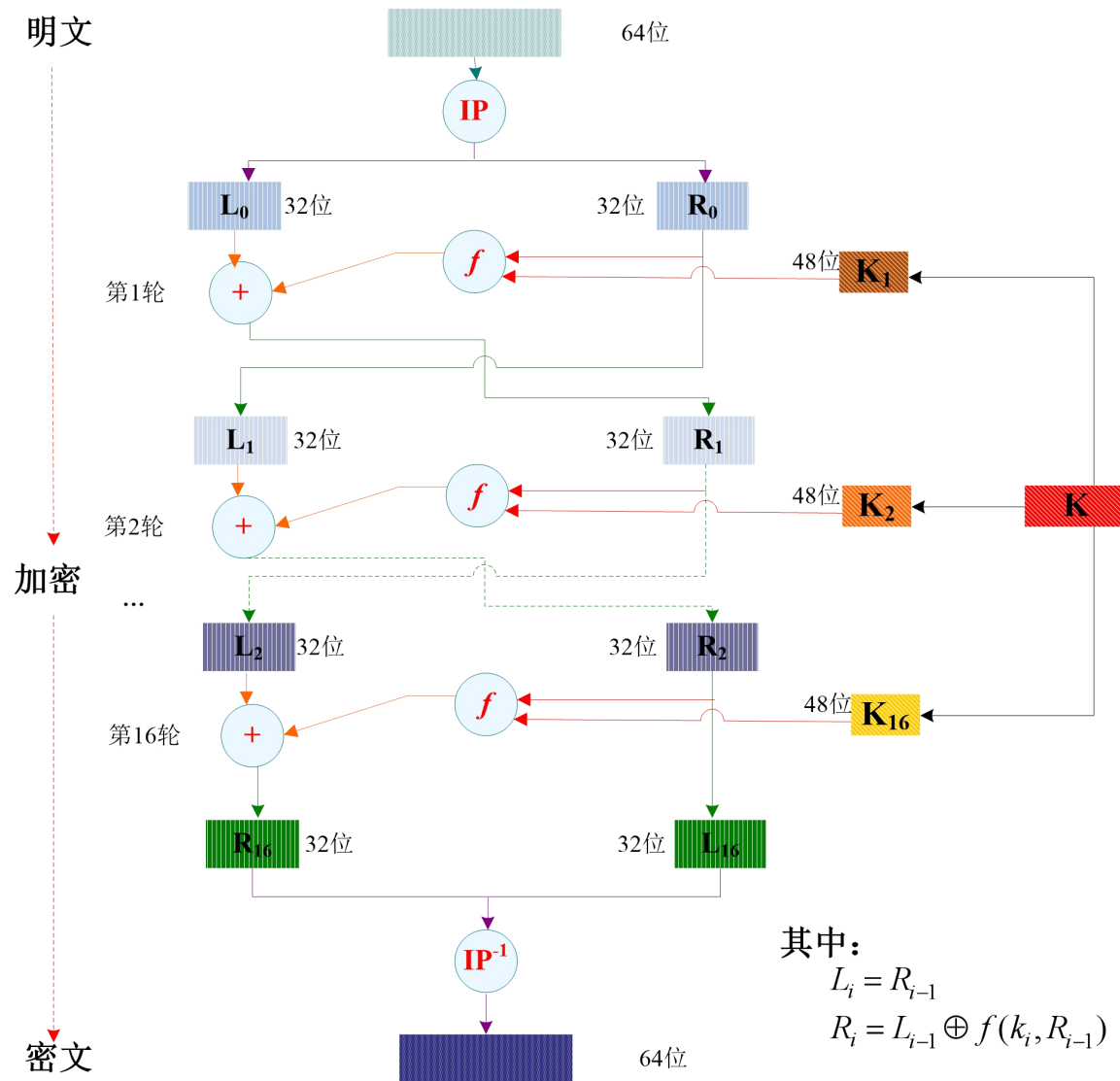
密码算法分类



对称密码学： **DES→AES**

非对称密码学： **RSA→ECC →抗量子(如，格Lattice)**

DES



RSA

□ 算法概要：

- Bob选择保密的素数 p 和 q ，并计算 $n=pq$ ；
- Bob通过 $\gcd(e, (p-1)(q-1))=1$ 来选择 e ；
- Bob通过 $de \equiv 1 \pmod{(p-1)(q-1)}$ 来计算 d ；
- Bob将 n 和 e 设为公开的， p 、 q 、 d 设为秘密的；
- Alice将 m 加密为 $c \equiv m^e \pmod{n}$ ，并将 c 发送给Bob；
- Bob通过计算 $m \equiv c^d \pmod{n}$ 解密。

RSA: 例1

- 选择两个素数: $p=17$ & $q=11$
- 计算 $n = pq = 17 \times 11 = 187$
- 计算 $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
- 选择 e : $\gcd(e, 160) = 1$; 其中 $e=7$
- 计算 d : $de = 1 \bmod 160$ 且 $d < 160$, 则 $d=23$ (因为 $23 \times 7 = 161 = 10 \times 160 + 1$)
- 公布公钥 $K_U = \{7, 187\}$
- 保存私钥 $K_R = \{23, 17, 11\}$

RSA: 例1 (续)

- 如果待加密的消息 $M = 88$ (注意: $88 < 187$)
- 加密: $C = 88^7 \bmod 187 = 11$
- 解密: $M = 11^{23} \bmod 187 = 88$

RSA: 例2

- Bob选择 $p=885320693$, $q=238855417$, 则可以计算

$$n=p*q=211463707796206571,$$

设加密系数为 $e=9007$, 将 n 和 e 发送给Alice。

- 假设Alice传递的信息是cat。

令 $a=01$ $c=03$ $t=20$, 则 $cat=030120=30120$ 。

Alice计算: $c \equiv m^e \equiv 30120^{9007}$

$$\equiv 113535859035722866 \pmod{n}$$

她将 c 传给Bob。

RSA：例2（续）

- Bob已知p和q的值，他用扩展欧几里德算法计算d:

$$d^e \equiv 1 \pmod{(p-1)(q-1)}, \quad \text{得到}$$

$$d=116402471153538991$$

然后Bob计算：

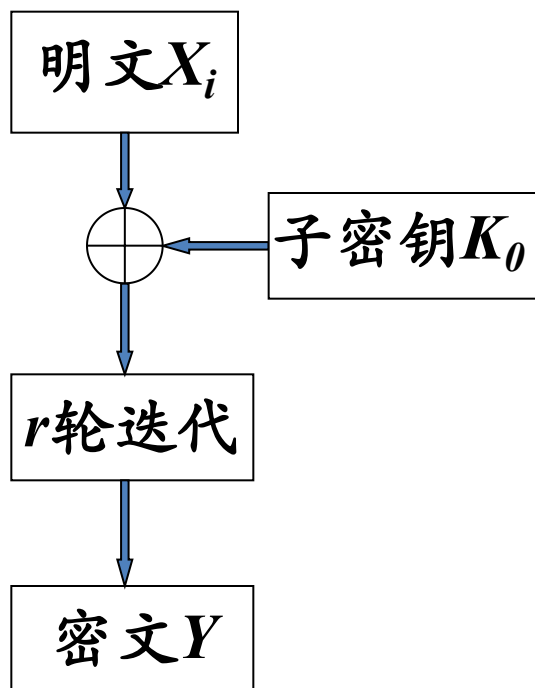
$$c^d \equiv 113535859035722866116402471153538991$$

$$\equiv 30120 \pmod{n}$$

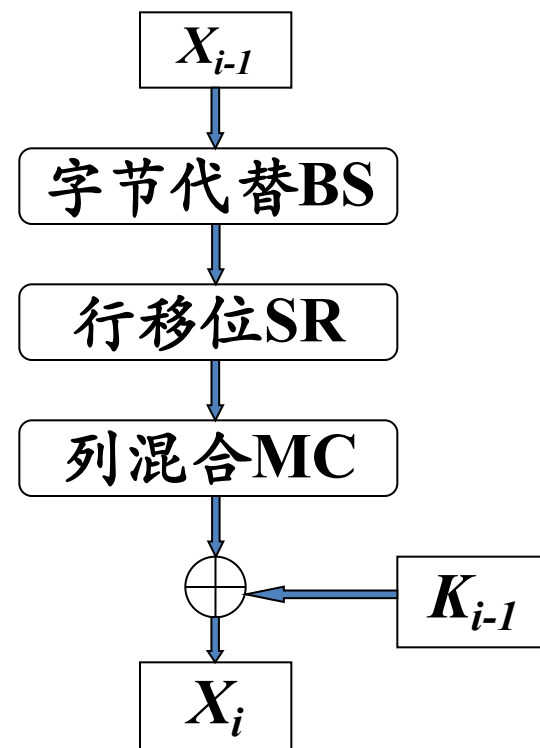
由此他可以得到最初的信息。

AES

□ 算法概要：



(a) AES 算法框图



(b) 一轮 AES 结构

AES的设计原则：能够抵御已知攻击、硬件实现容易且速度快、设计简单

ECC

- ECC: Elliptic Curve Cryptography
- 1985年，N.Koblitz及V.S.Miller分别提出了椭圆曲线密码体制（ECC）。
- 已经开发出的椭圆曲线标准的文档有：IEEE P1363 P1363a、ANSI X9.62 X9.63、ISO/IEC14888等。
- 近年来，ECC已走向工程实现和实际应用阶段。
- 目前，德国、日本、法国、美国、加拿大等许多西方国家的密码学研究小组和公司已经实现了椭圆曲线密码体制。

ECC（续）

□ 为什么要提出ECC？

- RSA 主要问题之一：
为了保证必要的安全强度，其密钥必须很长
- ECC的优势：
在同等安全强度下，ECC所需密钥比RSA短

Table 4.1: NIST's recommendation for practical applications revision 4 [130]

Security level in bits	Block cipher	\mathbb{F}_p	\mathbb{F}_2^m	RSA
80	SKIPJACK	192	163	1024
112	Triple-DES	224	233	2048
128	AES Small	256	283	3072
192	AES Medium	384	409	7680
256	AES Large	521	571	15360

ECC（续）

- 椭圆曲线指的是由韦尔斯特拉斯（Weierstrass）方程所确定的平面曲线

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- 其中，系数 a_i ($i=1, 2, \dots, 6$) 定义在基域 K 上（ K 可以是有理数域、实数域、复数域，还可以是有限域，椭圆曲线密码体制中用到的椭圆曲线都定义在有限域上）。
- 椭圆曲线并非椭圆

ECC（续）

□ 群：

对于非空集合 G ，其上的一个二元运算 $(.)$ 满足：封闭性、结合率、单位元和可逆性

□ 环：对于 R 上的两个二元运算 $(+, \times)$ 满足：

- 关于 $+$ 是一个交换群（群的条件 $+$ 交换率）
- 对于乘法 \times 满足：封闭性 $+$ 结合率 $+$ 分配率

□ 域：对于 F 上的两个运算 $(+, \times)$ 满足：

- F 是一个整环：交换环 $+$ 乘法逆元 $+$ 无零因子
- 乘法逆元存在

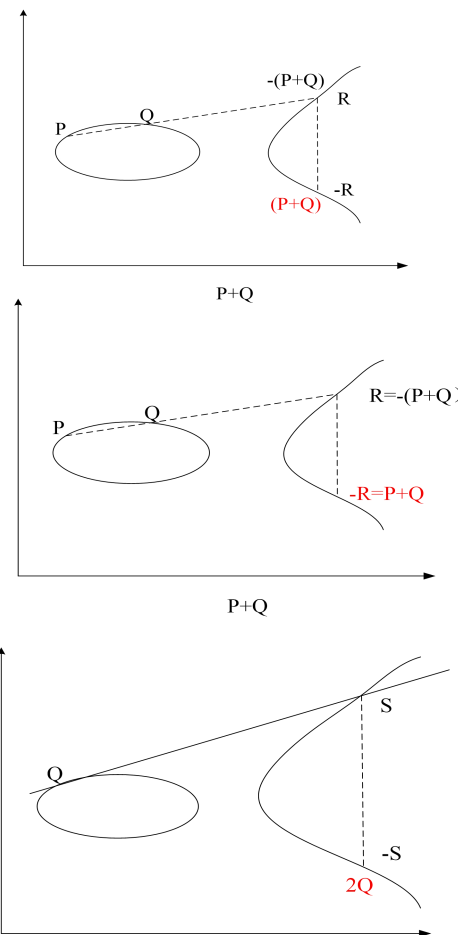
ECC (续)

域 { 整环 { 可交换环 { 环 { 可交换群 { 群

ECC (续)

□ 椭圆曲线加法运算规则:

- O 是加法的单位元, $O = -O$; 对于椭圆曲线上的任一点 P , 有 $P + O = P$
- 点 P 的负元是与 P 具有现同 x 坐标和相反 y 坐标的点, 即若 $P = (x, y)$, 则 $-P = (x, -y)$; $P + (-P) = O$
- 若 $P = (x_1, y_1)$, $Q = (x_2, y_2)$, 则 $P + Q = -R$ 。其中 R 是直线 PQ 与椭圆曲线的第三个交点。
- 若 P 和 Q 的 x 坐标相同, 则为**无穷远点** O
- 若 $Q = (x, y)$, 则 $Q + Q = 2Q = -S$, 其中 S 为椭圆曲线在 Q 点的切线与椭圆曲线的另一交点。



ECC（续）

□ 有限域上椭圆曲线

- $y^2 \equiv x^3 + ax + b \pmod{p}$

p 是奇素数,且 $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$

（构成Abel群的条件，证明过程略）

- $y^2 + xy \equiv x^3 + ax^2 + b \pmod{2^m}$ (Galois 域的椭圆曲线)

ECC (续)

有限域上椭圆曲线

- $y^2 \equiv x^3 + ax + b \pmod{p}$

(1)
 $P + O = P$

(2)
 $P = (x, y)$
 $P + (x, y) = O$
其中 $(x, -y)$
是 P 的负元
 $-P$

(3) 加法公式:
 $P = (x_P, y_P), Q = (x_Q, y_Q)$
若 $x_P = x_Q$ 且 $y_P = -y_Q$, 则 $P + Q = O$;
否则 $P + Q = (x_R, y_R)$
 $x_R = \lambda^2 - x_P - x_Q$
 $y_R = \lambda(x_P - x_R) - y_P$
其中
 $\lambda = (y_Q - y_P) / (x_Q - x_P)$, 如果 $P \neq Q$
 $\lambda = (3x_P^2 + a) / (2y_P)$, 如果 $P = Q$

(4) 重复相加:
 $nP = P + \dots + P$

按照上述定义构成了一个椭圆曲线上的Abel群

ECC (续)

□ 示例：有限域上椭圆曲线

- $y^2 \equiv x^3 + ax + b \pmod{p}$

□ 条件： $a=1$, $b=1$, $x=9$, $y=7$, $p=23$

- $y^2 = 7^2 \pmod{23} = 3$

- $x^3 + ax + b = (9^3 + 9 + 1) \pmod{23} = 3$

- $y^2 \equiv x^3 + ax + b \pmod{p}$

ECC示例

□ 示例：有限域上椭圆曲线

- $y^2 \equiv x^3 + ax + b \pmod{p}$

□ 条件： $a=1$, $b=1$, $x=9$, $y=7$, $p=23$

□ 问题：

- 求满足上述方程的所有整数对 (x, y) 以及无穷远点 O 组成的集合 $E_p(a, b) = E_{23}(1, 1)$?

ECC示例（续）

$E_{23}(1, 1)$

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

ECC示例（续）

$E_{23}(1, 1)$

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

1) $P=(0, 1)$,
 $P+O=(0, 1)$

2) $P=(13, 7)$
 $-P=(13, -7)$
 $= (13, 16)$

3) $P=(3, 10)$,
 $Q=(9, 7)$,
 $P+Q=(17, 20)$

4) $P=(3, 10)$,
 $2P=(7, 12)$

ECC示例（续）

□ P+Q计算过程:

$$P = (3, 10), Q = (9, 7)$$

$$\therefore \lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P} \right) \bmod p & (P \neq Q) \\ \left(\frac{3x_P^2 + a}{2y_P} \right) \bmod p & (P = Q) \end{cases}$$

$$\therefore \lambda = \left(\frac{7 - 10}{9 - 3} \right) \bmod 23 = 11$$

$$\therefore x_R = (\lambda^2 - x_P - x_Q) \bmod p = 17$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p = 20$$

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

其中

$$\lambda = (y_2 - y_1) / (x_2 - x_1), \text{ 如果 } P \neq Q$$

$$\lambda = (3x_1^2 + a) / 2y_1, \text{ 如果 } P = Q$$

ECC (续)

有限域上椭圆曲线

● $y^2 + xy \equiv x^3 + ax^2 + b \pmod{2^m}$ (Galois 域的椭圆曲线)

(1)
 $P + O = P$

(2)
 $P = (x, y)$
 $P + (x, y) = O$
其中 $(x, -y)$
是 P 的负元
 $-P$

(3) 加法公式:
 $P = (x_P, y_P)$,
 $Q = (x_Q, y_Q)$, 且 $P \neq -Q$,
 $P \neq Q$
则 $P + Q = (x_R, y_R)$
 $x_R = \lambda^2 + \lambda + x_P + x_Q + a$
 $y_R = \lambda(x_P + x_R) + x_R + y_P$
其中
 $\lambda = (y_Q + y_P) / (x_Q + x_P)$

(4) 若 $P = (x_P, y_P)$, 则
 $R = 2P = (x_R, y_R)$
其中:
 $x_R = \lambda^2 + \lambda + a$
 $y_R = x_P^2 + (\lambda + 1)x_R$
 $\lambda = x_P + y_P / x_P$

按照上述定义构成了一个椭圆曲线上的Abel群

ECC（续）

□ 椭圆曲线上的离散对数“难题”

- 对于方程 $Q=kP$,其中 P, Q 属于 $E_p(a,b)$ 。对于给定的 k 和 P , 计算 Q 比较容易, 而对于给定的 P 和 Q , 计算 k 比较困难

方程 $y^2=(x^3+9x+17) \bmod 23$ 所定义的群 $E_{23}(9,17)$ 。

求: $P=(16, 5)$ 和 $Q=(4, 5)$ 的离散对数 k ?

穷举计算:

$P=(16, 5)$, $2P=(20, 20)$, $3P=(14, 14)$, $4P=(19, 20)$, $5P=(13, 10)$,
 $6P=(7, 3)$, $7P=(8, 7)$, $8P=(12, 17)$, $9P=(4, 5)$

因此 $k=9$

ECC加密/解密实现

□ Alice → Bob

- **Step 1:** Bob选择 $E_p(a, b)$ 的元素 G , 使得 G 的阶 n 是一个大素数, 秘密选择整数 k . 计算 $P=kG$, 公开 (p, a, b, G, P) , 保密 k 。其中 $K_b=kG$ 为Bob公钥, $K_b'=k$ 为Bob私钥
- **Step 2:** 将消息 m 编码为 x - y 形式的点 P_m
- **Step 3:** Alice随机选择一个正整数 r , 对 P_m 产生密文 $C_m = \{rG, P_m + rK_b\}$
- **Step 4:** Bob解密

$$\begin{aligned} & C_m - K_b'(rG) \\ &= P_m + rK_b - krG \\ &= P_m + r(kG) - rkG \\ &= P_m \end{aligned}$$

ECC加密/解密实现（续）

□ Alice→Bob (示例)

- **Step 1:** Bob选择 $E_{88331}(3, 45)$, $G=(4,11)$, Bob私钥 $K_b'=K=3$, Bob公布公钥 $K_b=(413,1808)$
- **Step 2:** $P_m=(5,1734)$
- **Step 3:** Alice随机选择一个正整数 $r=8$, 对 P_m 产生密文:
 $C_m = \{rG, P_m + rK_b\} = \{(5415,6321), (6626,3576)\}$
- **Step 4:** Bob解密

$$K_b'(rG) = 3(5415,6321) = (673,146)$$

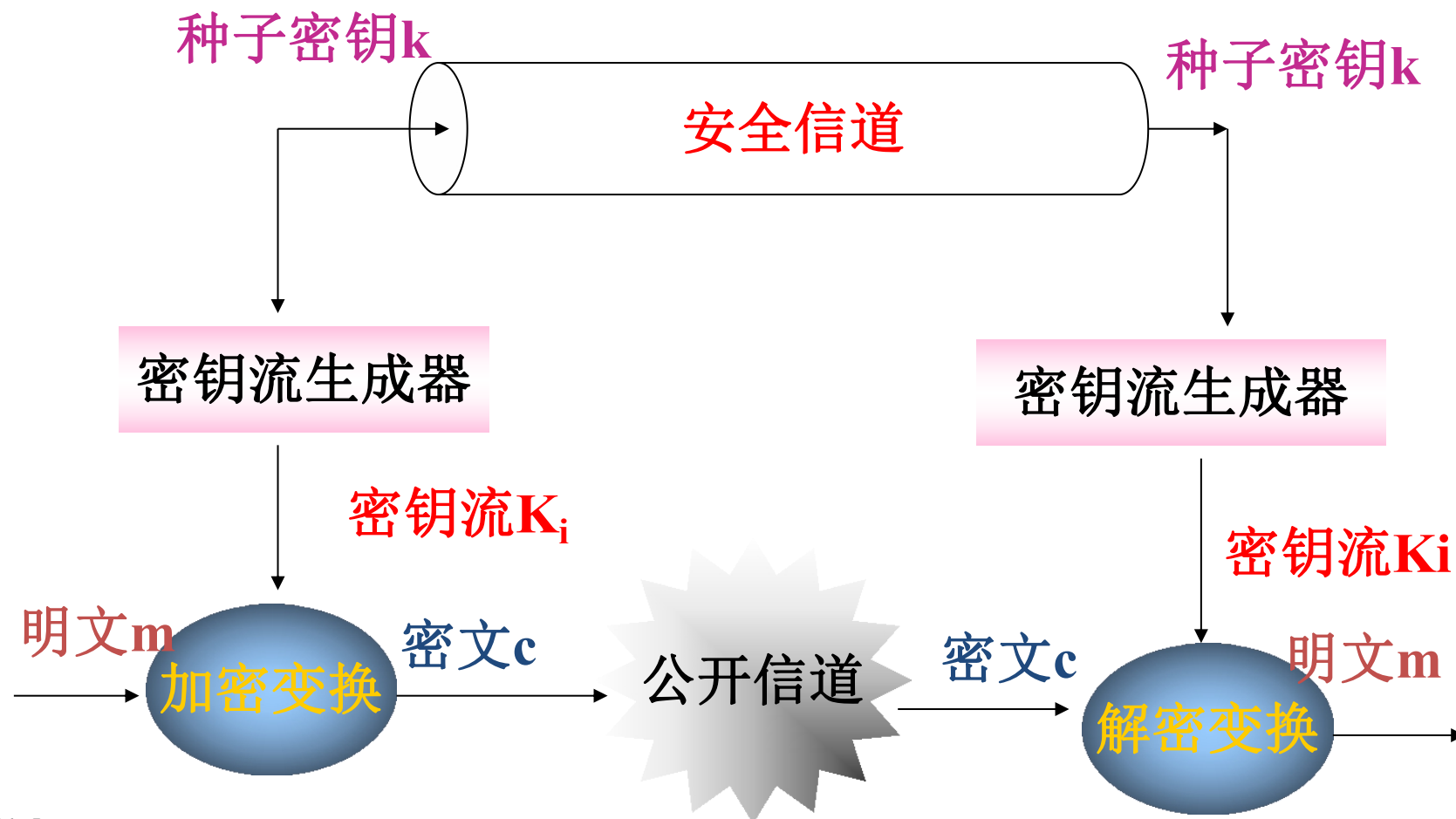
$$C_m - K_b'(rG) = (6626,3576) - (673,146) = (5,1734)$$

序列密码

- 按位处理消息
- 一般具有一个伪随机密钥流发生器
- 对明文按位进行异或运算 (XOR)
- 以随机的**密钥流**来破坏消息的统计特征
 - $C_i = M_i \text{ XOR } \text{StreamKey}_i$

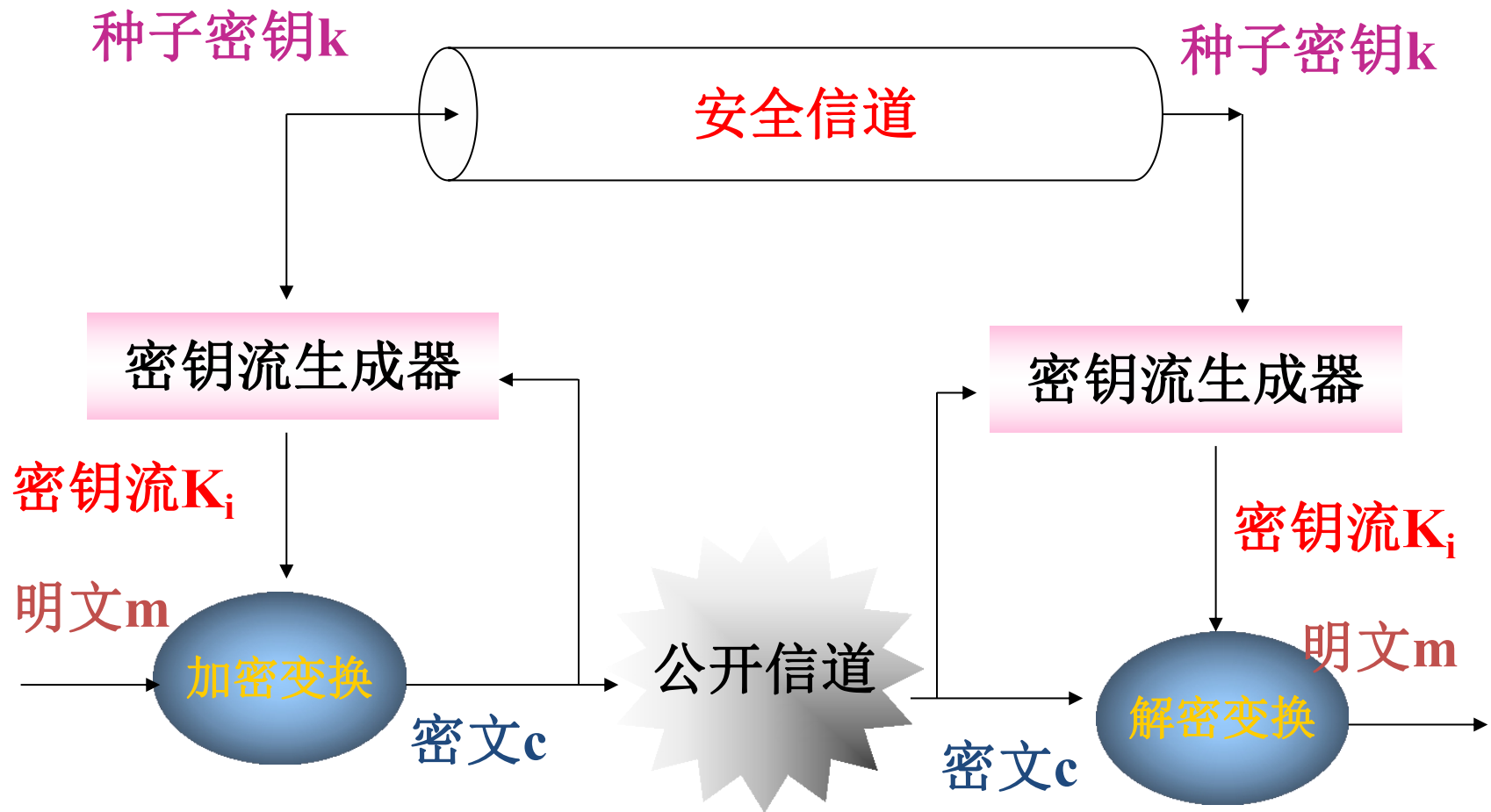
序列密码（续）

□ 同步流密码原理



序列密码（续）

□ 自同步流密码原理



序列密码：RC4

- RC: “RC” is Ron’s Code or Rivest Cipher
- 1987年Ron Rivest 为RSA公司所设计的流密码算法
- 可变密钥长度的、面向字节操作的流密码：8~2048位可变
- 1994年算法才公开
- 在SSL/TLS和IEEE 802.11无线网络中有广泛应用：**WEP协议**

序列密码：RC4（续）

□ RC4初始化

- 用从1到256个字节的可变长度**密钥初始化矢量**
S: 0..255 , S[0],S[1],...,S[255]
- S 形成了算法的内部状态
- **密钥流字节K由S中255个元素按照一定方式选出一个元素来生成**
- 每生成一个K, S中的元素就被重新置换一次

序列密码：RC4（续）

□ 密钥生成

● 初始化S

- 初始条件：密钥种子 $Key[]$, 密钥初始化向量S
- 初始化S：S中元素被置为按升序从0到255： $S[0]=0, S[1]=1, \dots, S[255]=255$
- 建立一个临时矢量K

如果密钥种子Key的长度为256字节，则将Key赋值给K；

否则将K的值赋给K的前N个元素（N为密钥长度），

循环重复用Key的值赋给K剩下的元素,直到K的所有元素都被赋值

- 然后用K产生S的初始置换

从S[0]到S[255]，对每个S[i]，根据由K[i]确定的方案，将S[i]置换为S中的另外一个字节

由于对S的操作仅是交换（即置换），因此S仍然包含所有值为0到255的元素

序列密码：RC4（续）

Input : Key[], N=len(key)

Output: S[] /*密钥流初始值*/

/*Array “key” contains N bytes of key*/

/*Array “S” always has a permutation of 0,1,...,255*/

for i = 0 to 255

S[i] = i

K[i] = key[i (mod N)]

next i

j = 0

for i = 0 to 255

j = (j + S[i] + K[i]) mod 256

swap(S[i], S[j])

next i

序列密码：RC4（续）

□ 加密与解密

- 密钥流KeyStreamByte的生成是从S[0]到S[255]，对每个S[i]，根据当前S的值，将S[i]与S中另外一个字节置换
当S[255]完成置换后，操作继续重复
- 加密：
将KeyStreamByte值与下一个明文字节异或
- 解密：
将KeyStreamByte值与下一个密文字节异或

序列密码：RC4（续）

Input : $M, S[]$

Output: C

$i=j=0$

for each message byte M_i

$i = (i + 1) \bmod 256$

$j = (j + S[i]) \bmod 256$

swap($S[i], S[j]$)

$t = (S[i] + S[j]) \bmod 256$

KeyStreamByte = $S[t]$

$C_i = M_i \text{ XOR KeyStreamByte}$

Next

$C=C_1|C_2|\dots|C_n$

序列密码：RC4（续）

□ RC4初始化

- 对已知的密码分析很安全
 - 有很多攻击分析，但不是很实际
- 结果非线性
- 绝对不能重复使用密钥
- 在无线保密协议（WEP）中使用，但是有潜在的安全问题



提 纲

CONTENTS



1. 加密算法

2. 消息认证

3. 数字签名

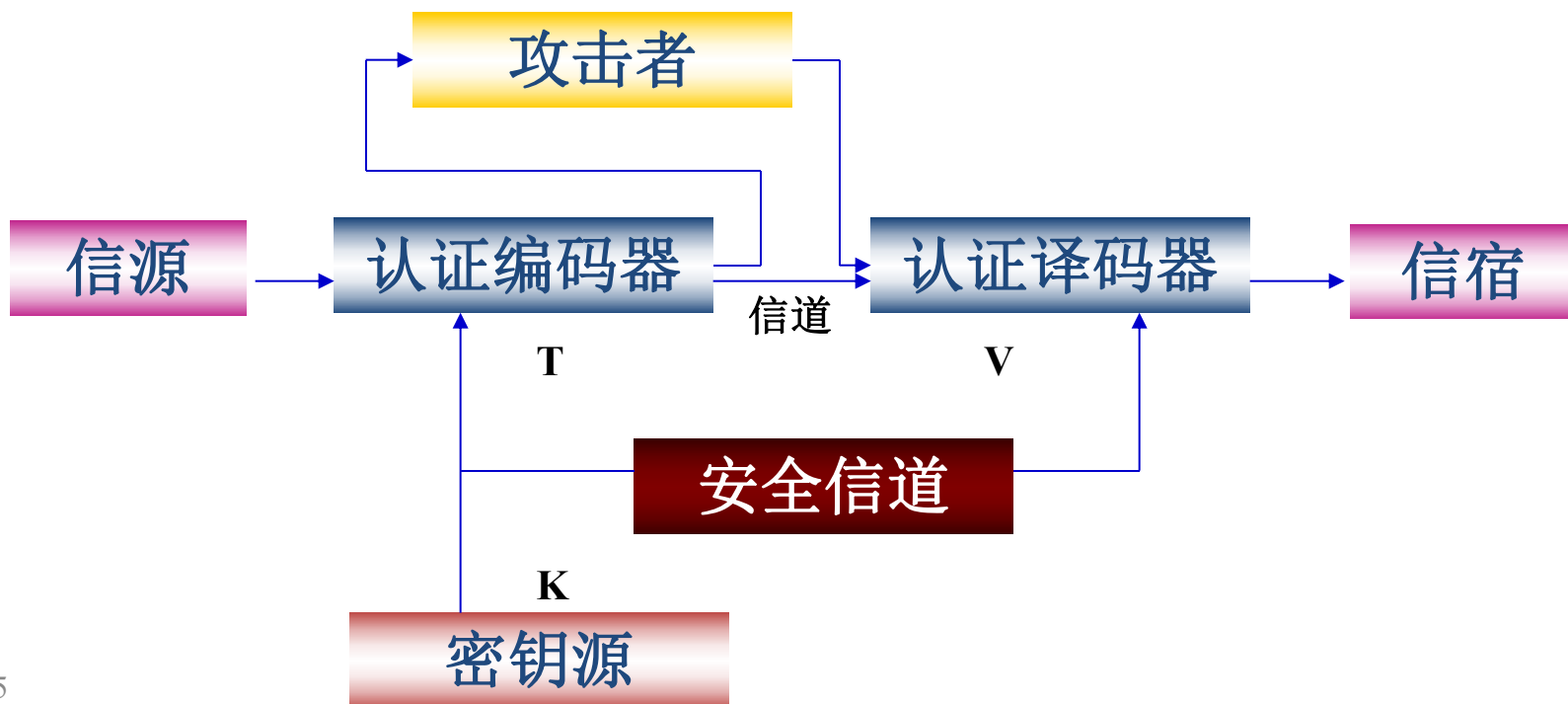
Message Authentication

- **Message Authentication: 报文鉴别（消息认证，消息鉴别）**
 - **Message:** 消息、报文。
 - **Authentication:** 鉴别、认证。
- **认证：消息的接收者对消息进行的验证**
 - 真实性：消息确实来自于其真正的发送者，而非假冒；
 - 完整性：消息的内容没有被篡改。
- **是一个证实收到的消息来自可信的源点且未被篡改的过程。**
它也可以验证消息的顺序和及时性

消息认证概念

□ 三元组 (K, T, V)

- 密钥生成算法K
- 标签算法T
- 验证算法V



认证函数

- 鉴别编码器和鉴别译码器可抽象为**认证函数**
- **认证函数**
 - 产生一个**鉴别标识**（**Authentication Identification**）
 - 给出合理的**认证协议**(**Authentication Protocol**)
 - 接收者完成消息的**鉴别**（**Authentication**）

认证函数分类

□ 认证函数分为三类：

- 消息加密函数(Message encryption)

用完整信息的密文作为对信息的认证。

- 消息认证码MAC (Message Authentication Code)

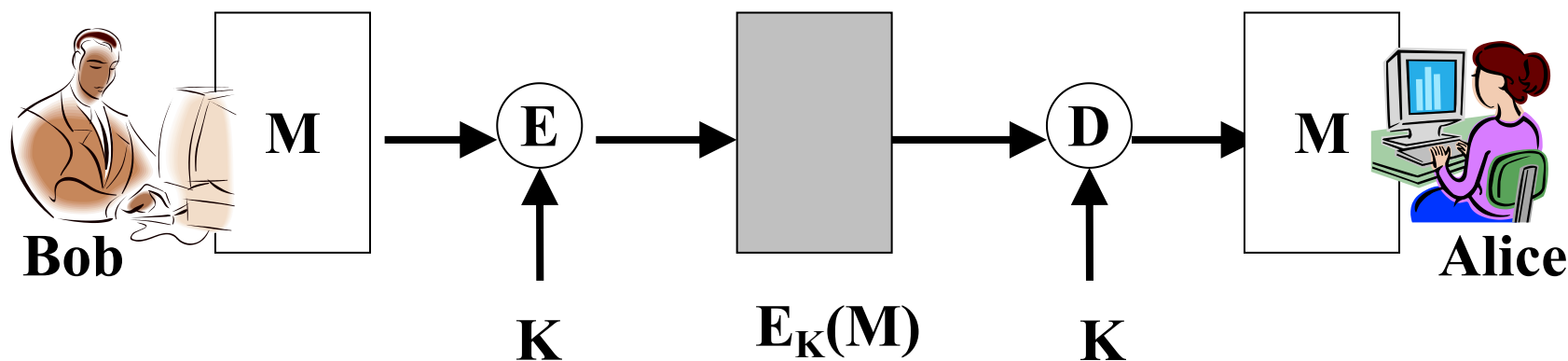
是对信源消息的一个编码函数。

- 散列函数 (Hash Function)

是一个公开的函数，它将任意长的信息映射成一个固定长度的信息。

认证函数：加密函数

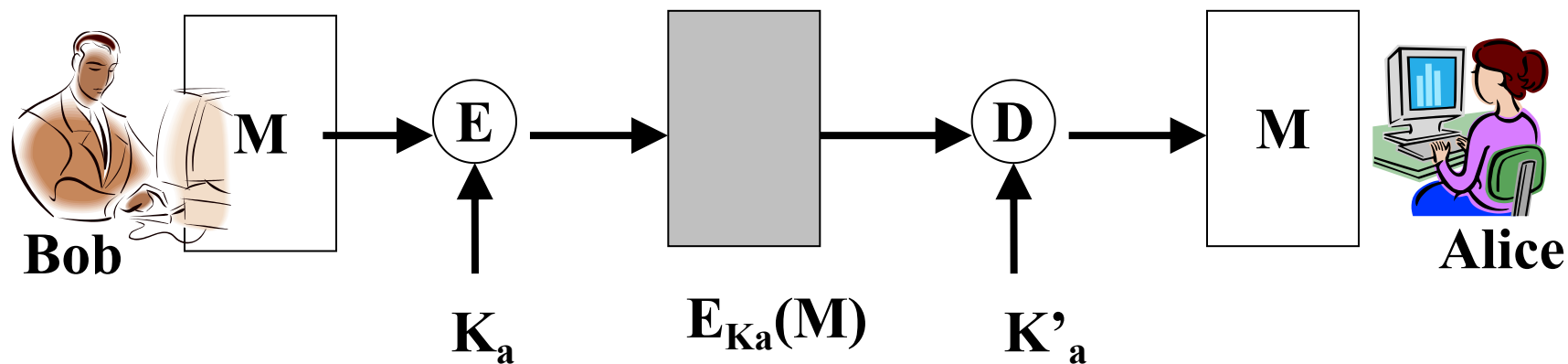
□ 对称加密：保密性与认证



提供保密
提供认证
不提供签名

认证函数：加密函数（续）

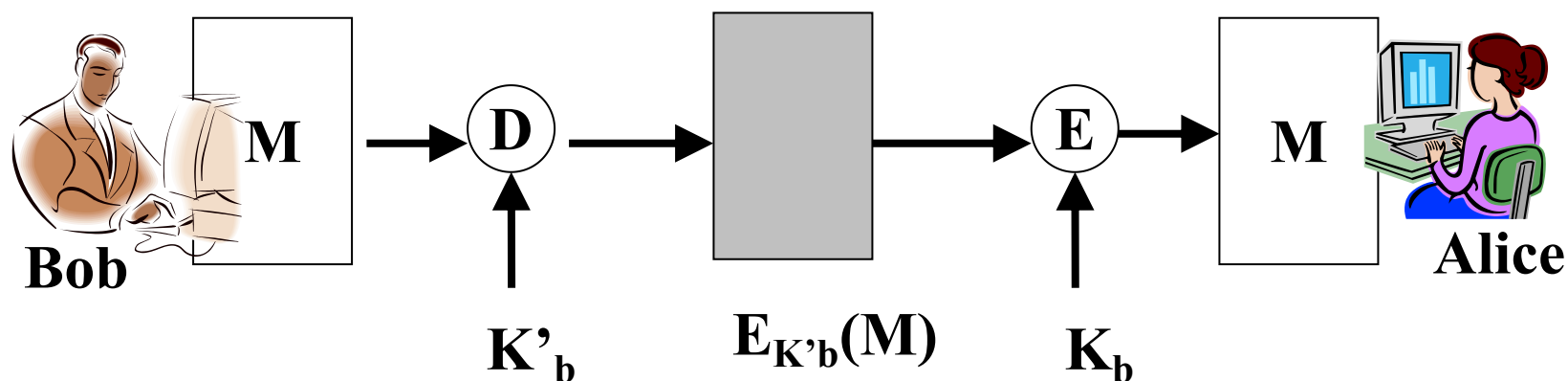
□ 公钥加密：保密性



提供保密
不提供认证

认证函数：加密函数（续）

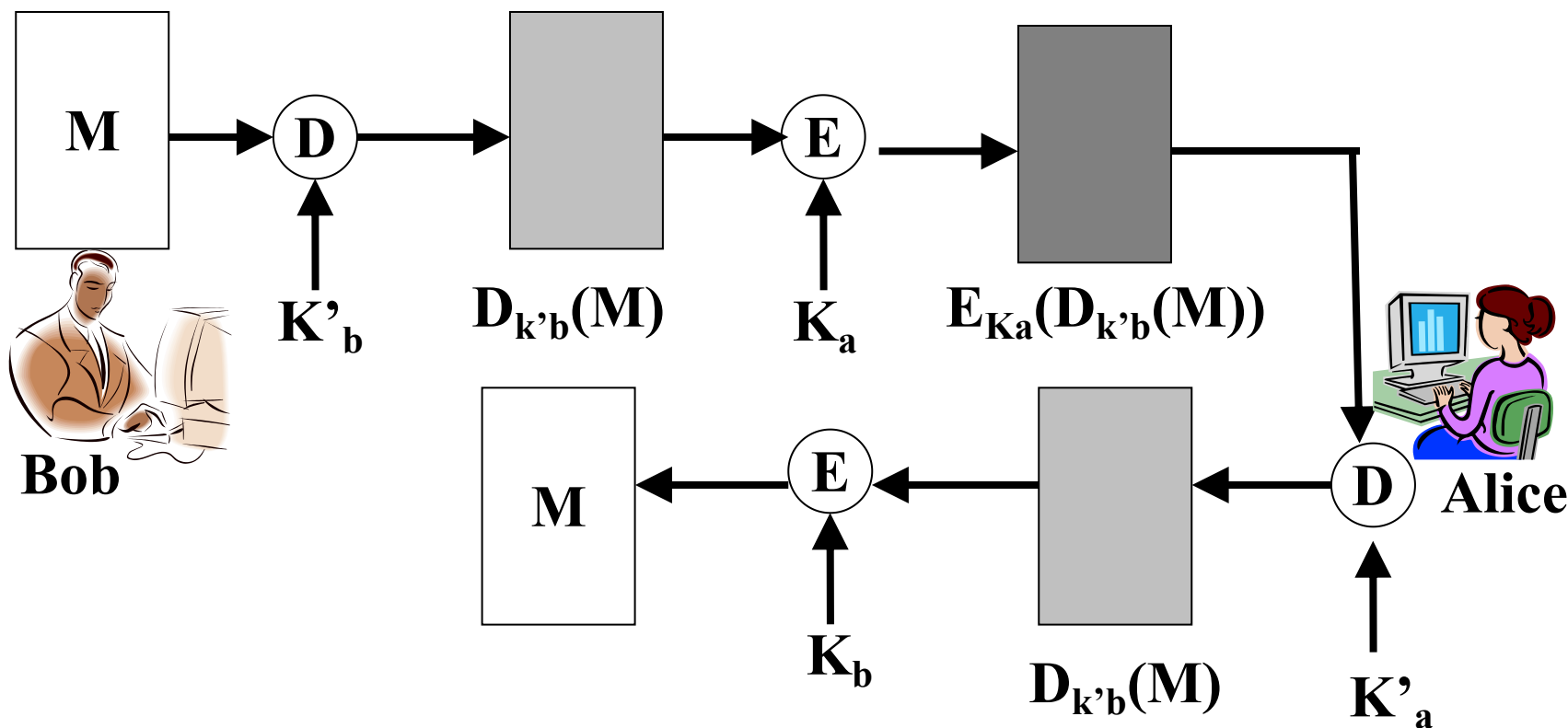
□ 公钥加密：认证与签名



提供认证

认证函数：加密函数（续）

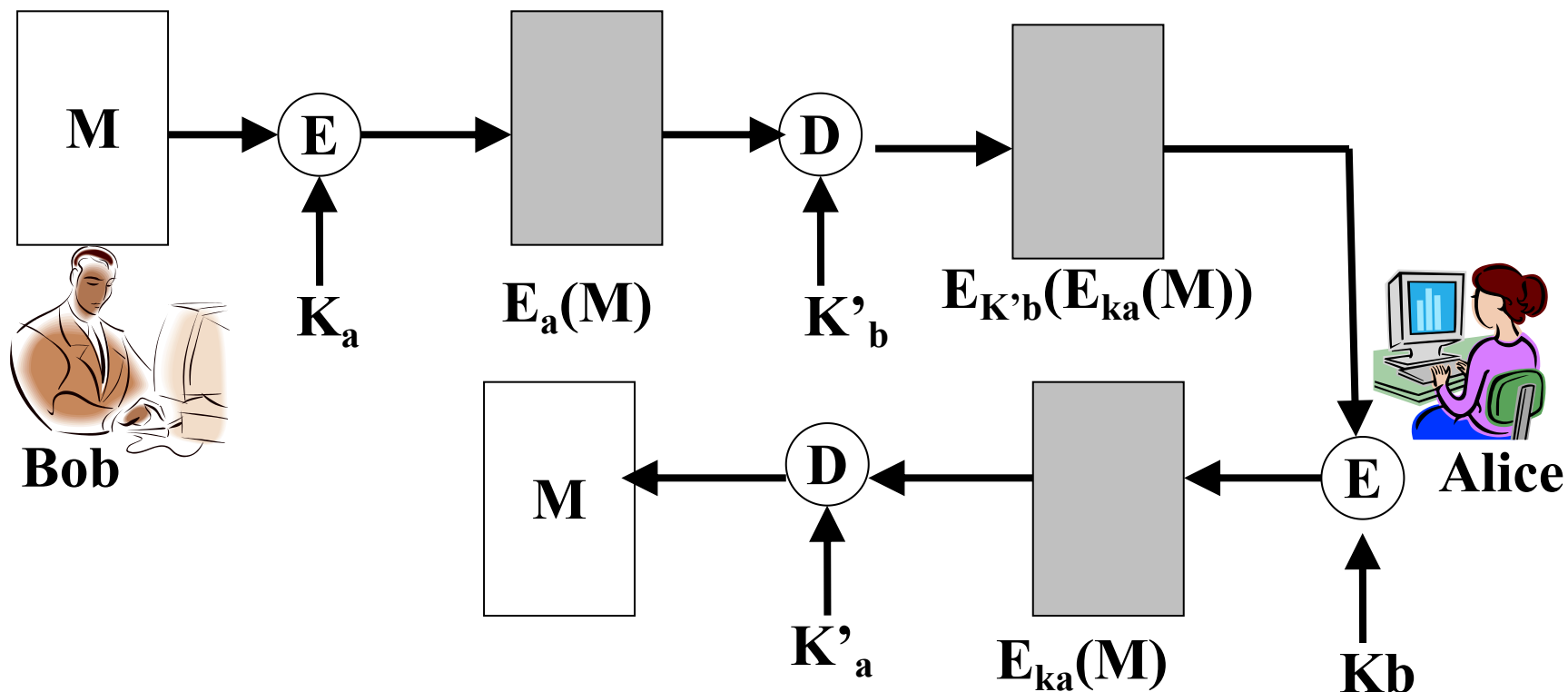
□ 公钥加密：保密、认证与签名



提供认证
提供保密性

认证函数：加密函数（续）

□ 公钥加密：保密、认证与签名??



提供认证
提供保密性

认证函数：消息认证码

□ 消息认证码：

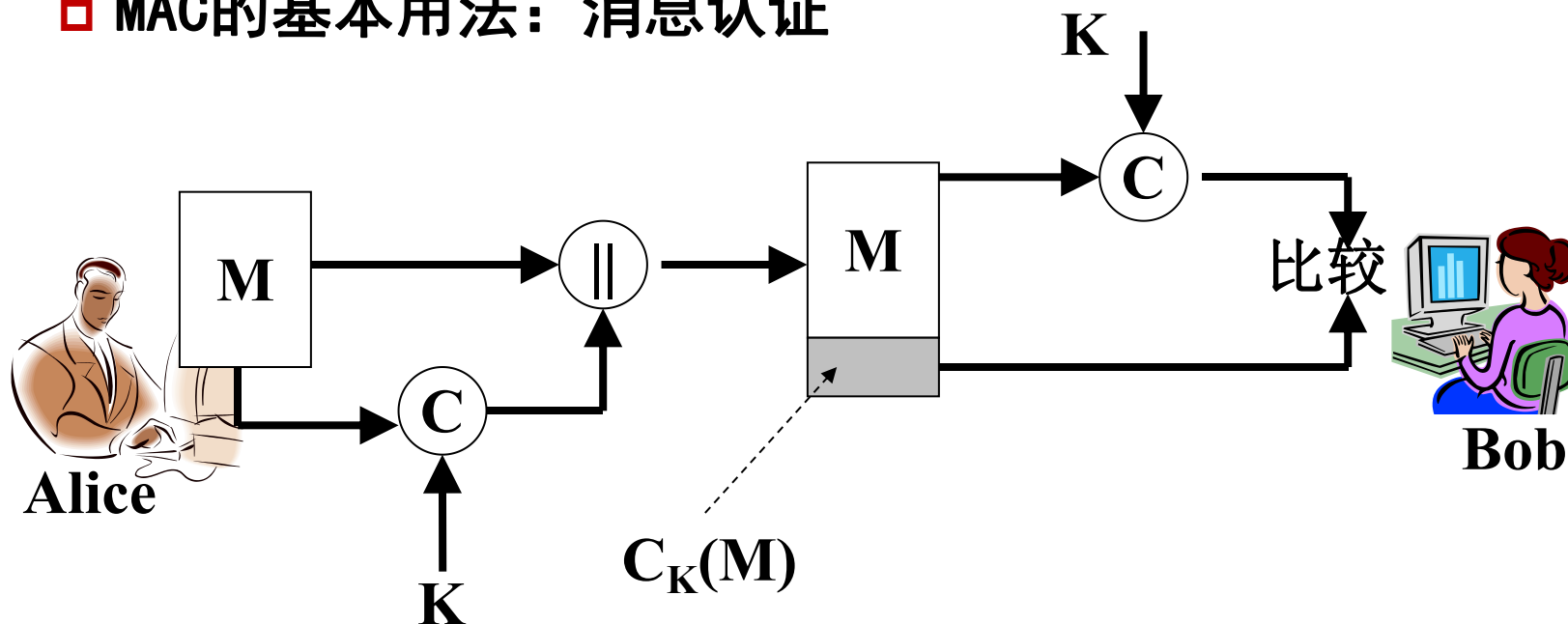
- 使用一个密钥生成一个固定大小的短数据块，并将该数据块加载到消息后面，称MAC（或密码校验和）

$$\text{MAC} = C_k(M)$$

- MAC函数类似于加密函数，但不需要可逆性。因此在数学上比加密算法被攻击的弱点要少

认证函数：消息认证码（续）

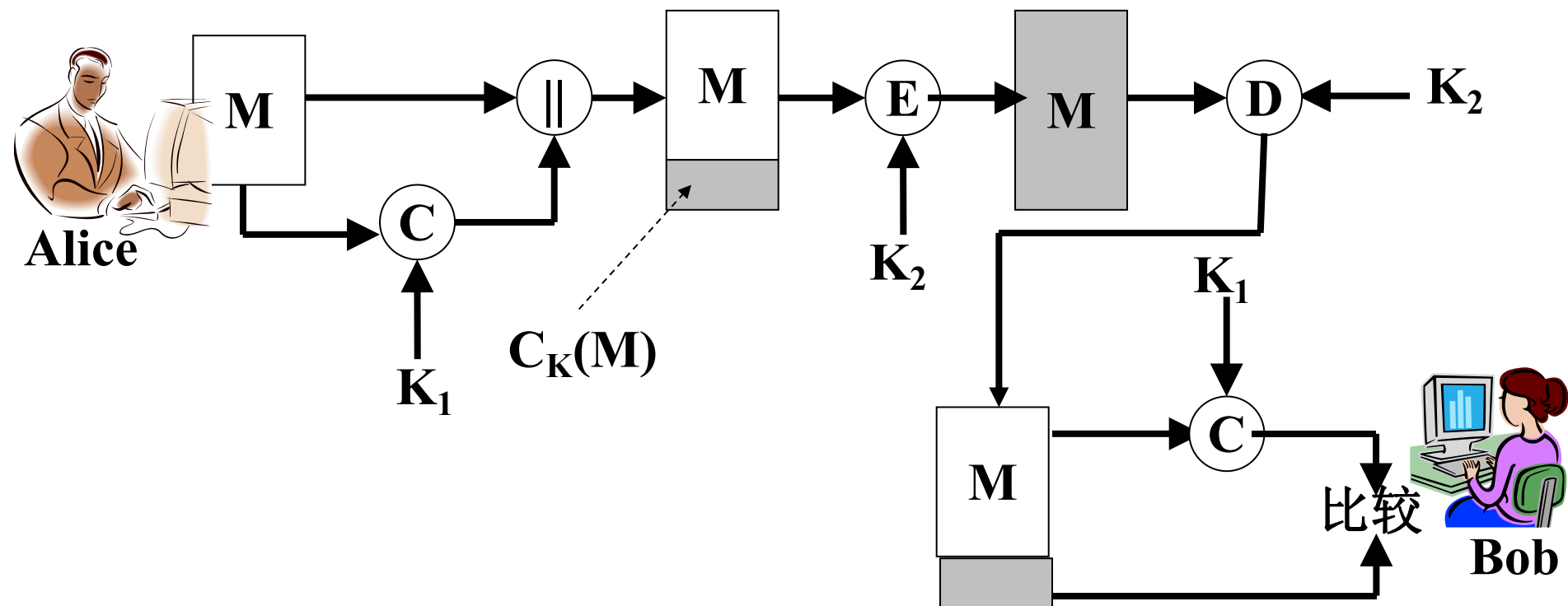
□ MAC的基本用法：消息认证



提供认证

认证函数：消息认证码（续）

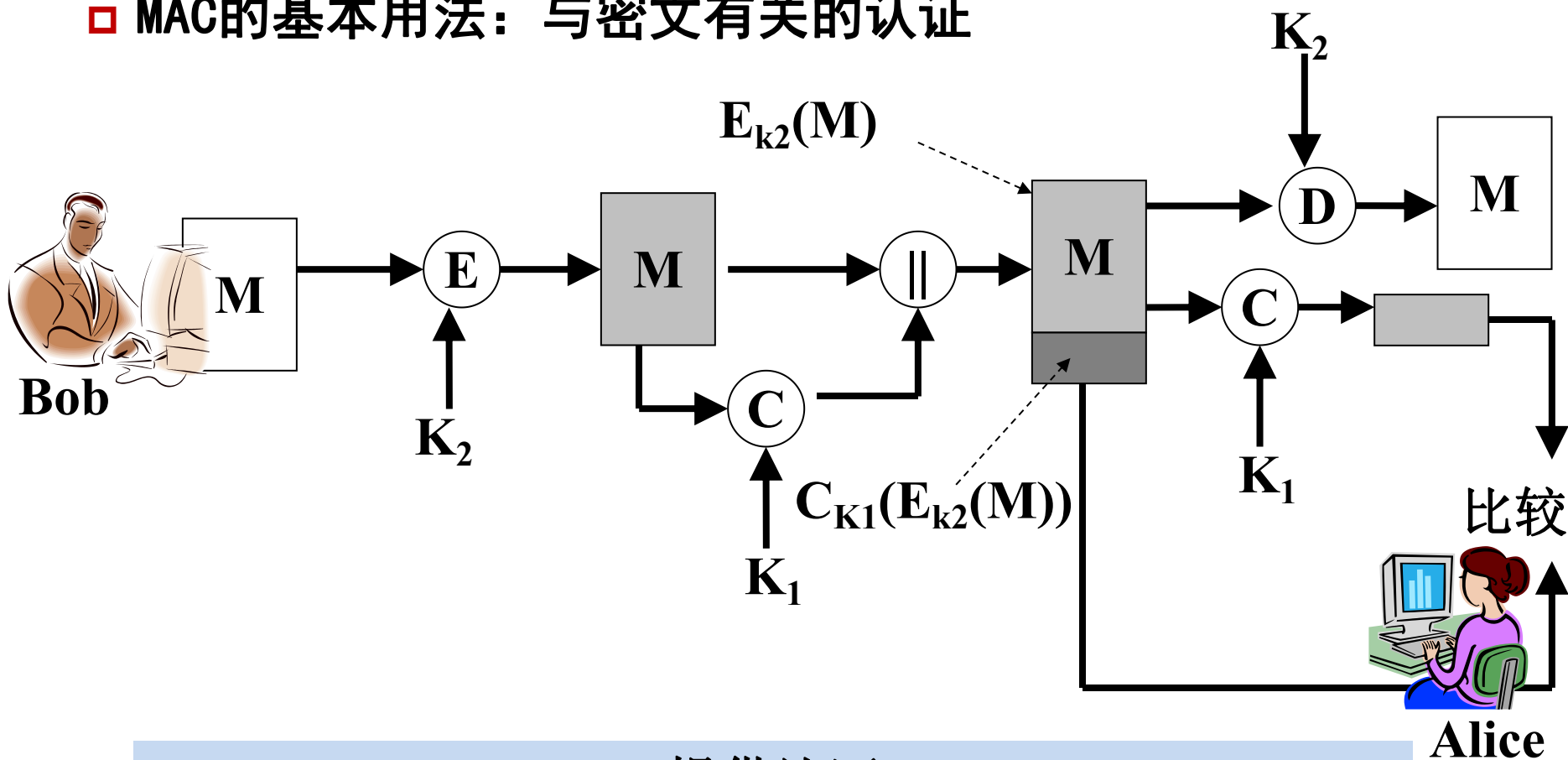
□ MAC的基本用法：与明文有关的认证



提供认证
提供保密

认证函数：消息认证码（续）

□ MAC的基本用法：与密文有关的认证



提供认证
提供保密

认证函数：Hash函数

□ Hash Function

- 哈希函数、摘要函数
- 输入：任意长度的消息报文 M
- 输出：一个固定长度的散列码值 $H(M)$
- 是报文中所有比特的函数值
- 单向函数

认证函数：Hash函数（续）

□ 根据是否使用密钥

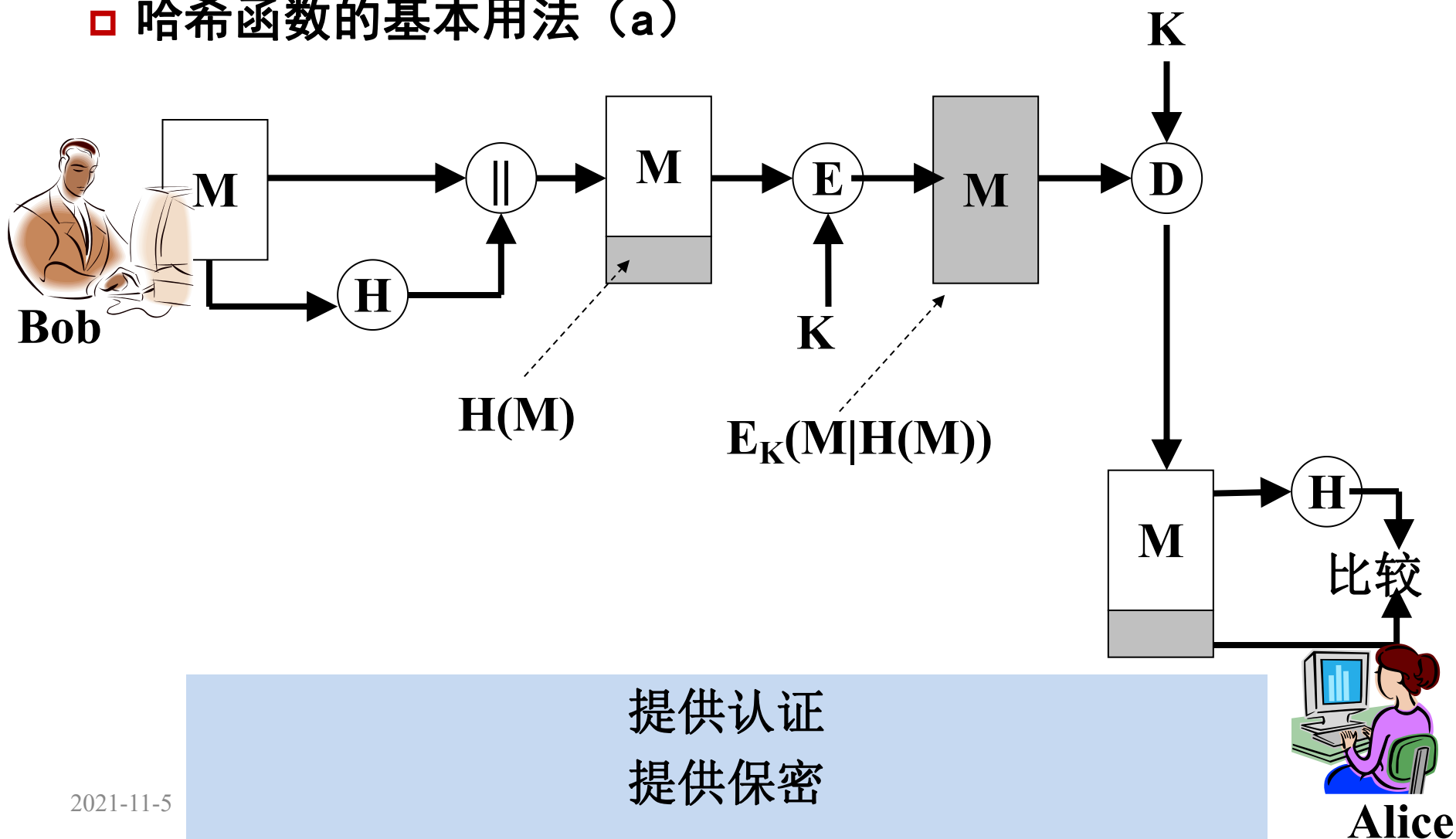
- 带秘密密钥的Hash函数:消息的散列值由只有通信双方知道的秘密密钥K来控制。此时，散列值称作MAC。
- 不带秘密密钥的Hash函数：消息的散列值的产生无需使用密钥。此时，散列值称作MDC。

□ Hash函数需满足以下条件：

- 输入x可以为任意长度，输出为固定长度
- 正向计算容易，反向计算困难
- 抗冲突性(无冲突性)

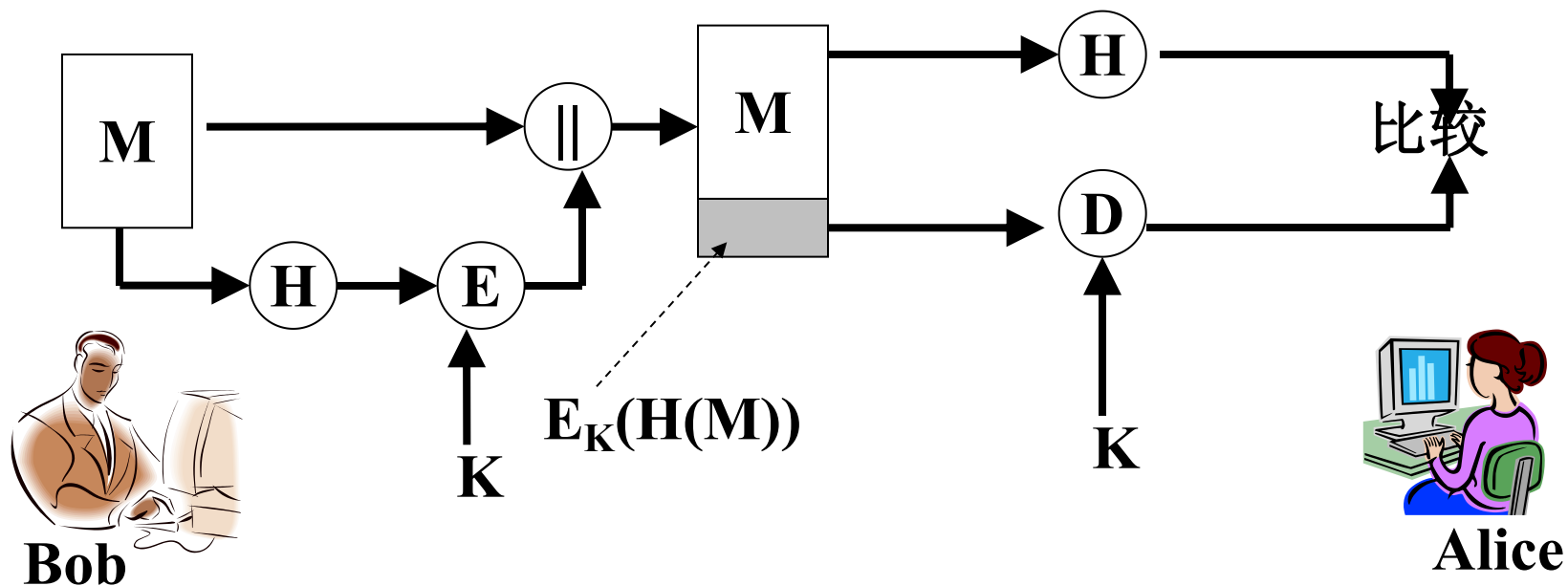
认证函数：Hash函数（续）

□ 哈希函数的基本用法（a）



认证函数：Hash函数（续）

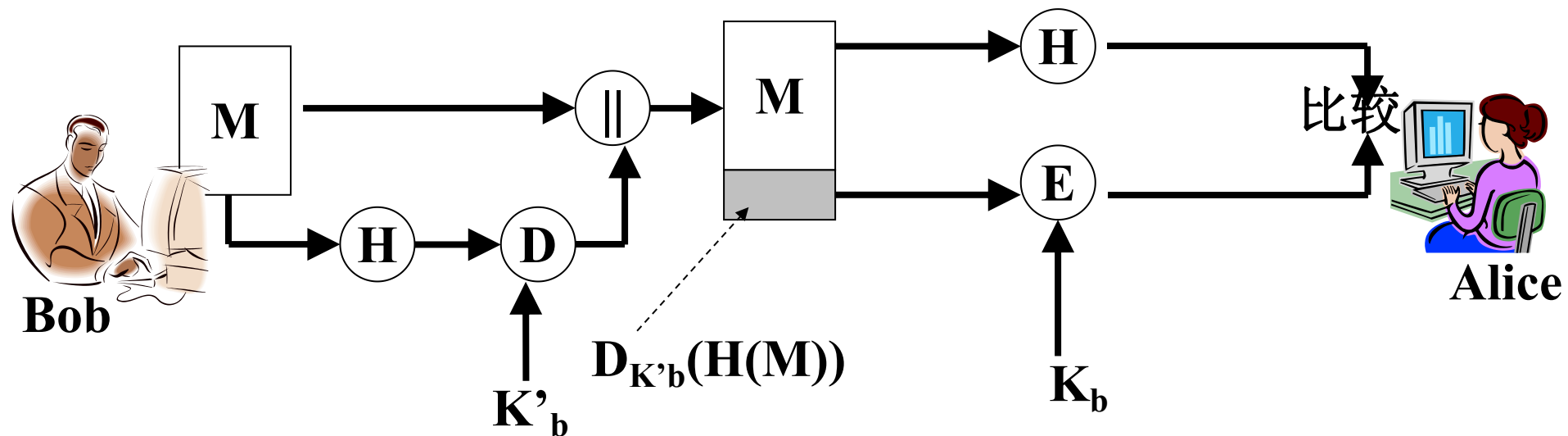
□ 哈希函数的基本用法（b）



提供认证

认证函数：Hash函数（续）

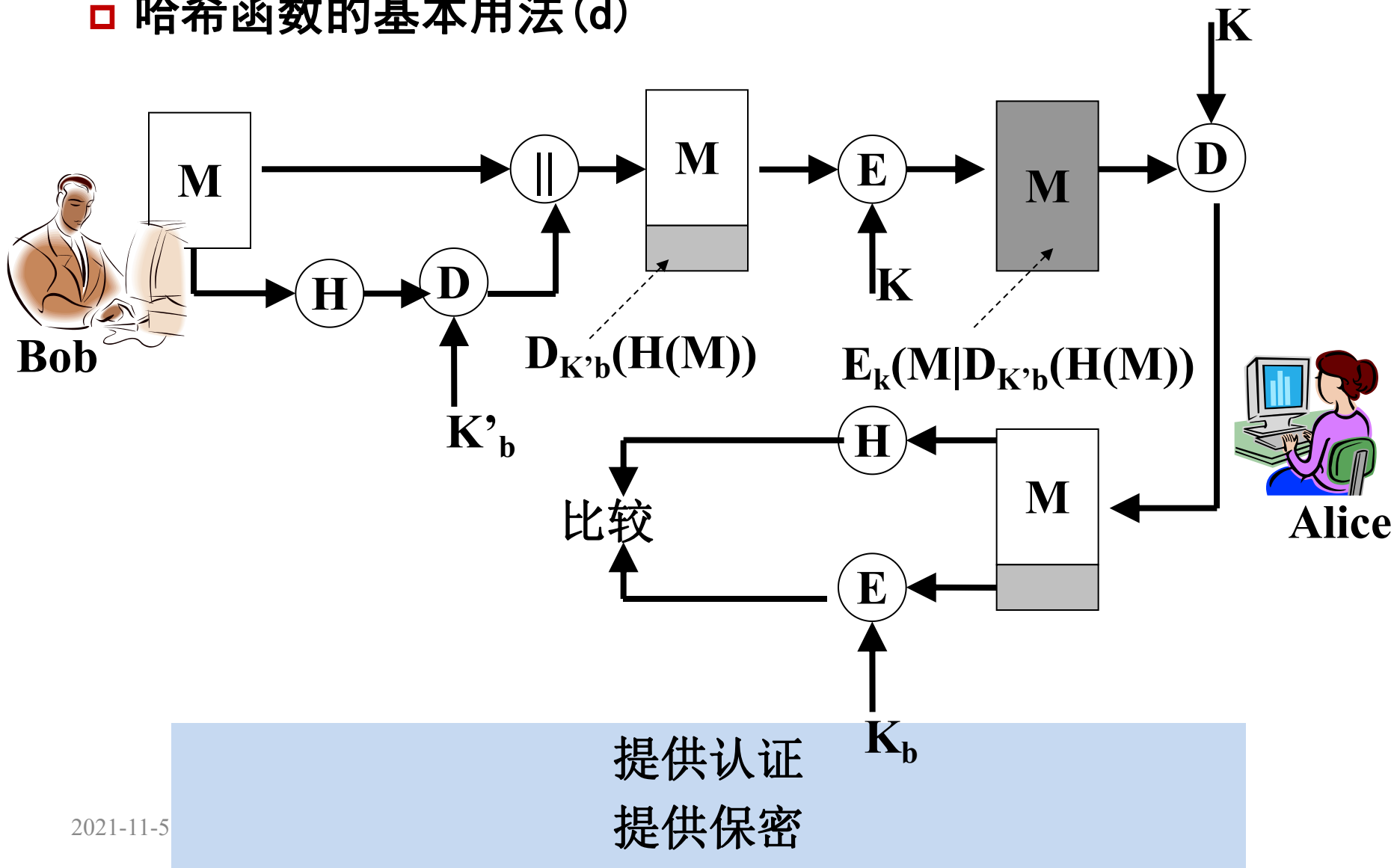
□ 哈希函数的基本用法（c）



提供认证

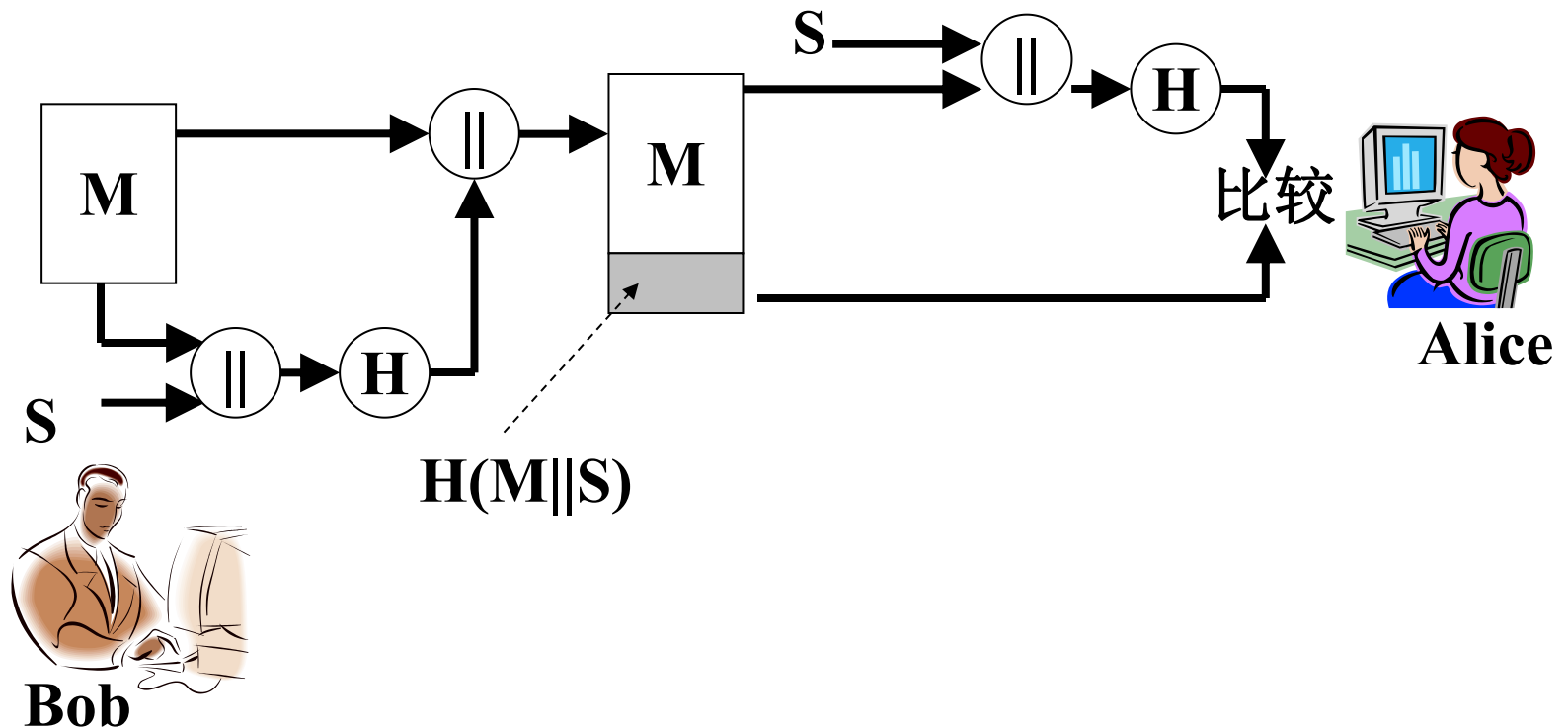
认证函数：Hash函数（续）

□ 哈希函数的基本用法 (d)



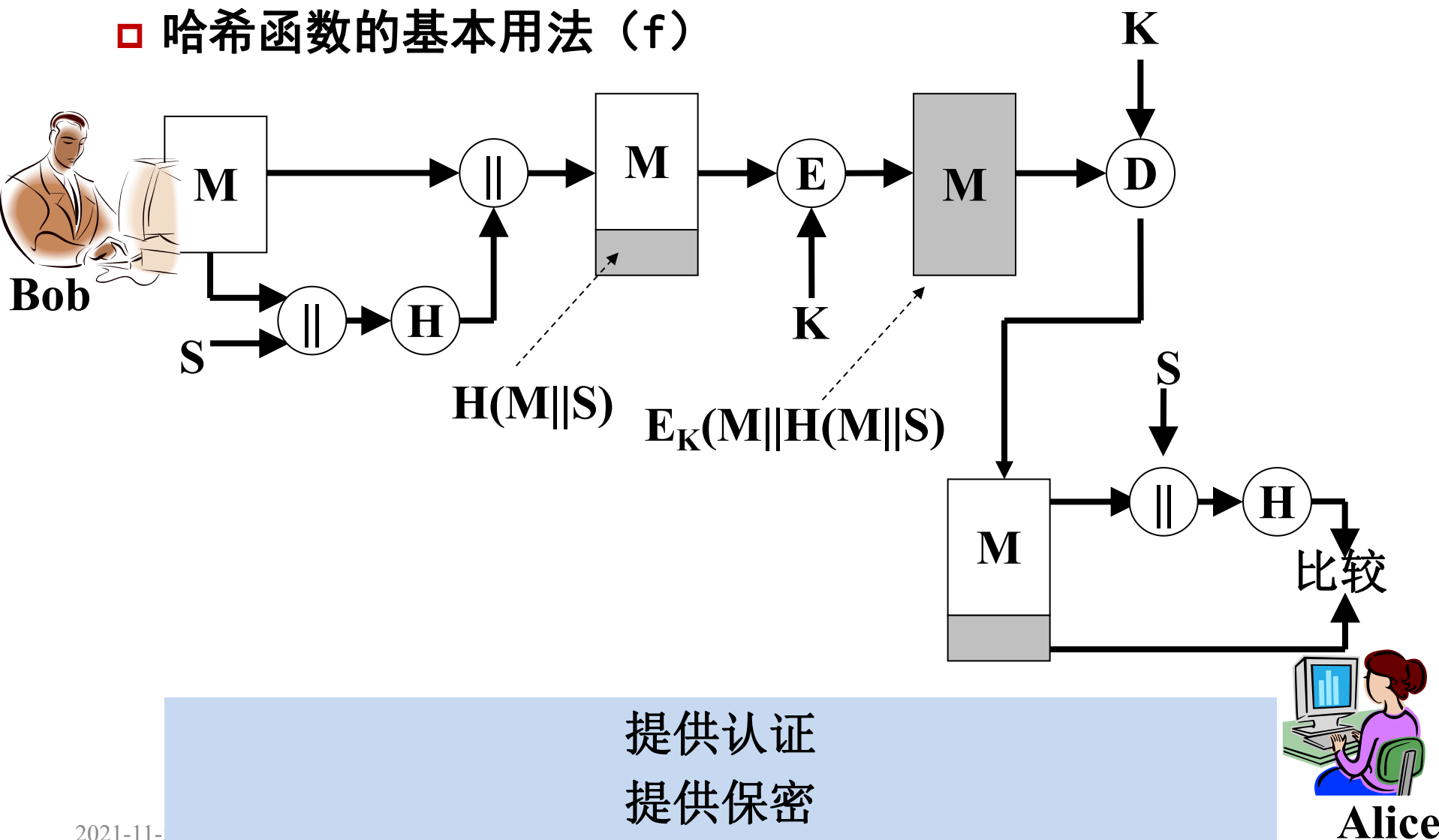
认证函数：Hash函数（续）

□ 哈希函数的基本用法（e）



认证函数：Hash函数（续）

□ 哈希函数的基本用法 (f)





提 纲

CONTENTS



1. 加密算法

2. 消息认证

3. 数字签名

数字签名

□ 数字签名：Digital Signature

□ 传统签名的基本特点

- 签名是可信的：能与被签的文件在物理上不可分割
- 签名是不可抵赖的：签名者不能否认自己的签名
- 签名不能被伪造：除了合法者外，其他任何人不能伪造其签名
- 签名是不可复制的：对一个消息的签名不能通过复制的方式变为另外一个消息的签名
- 签名是不可改变的：经签名的消息不能被篡改
- 容易被验证

□ 数字签名是传统签名的数字化

- 能与所签文件“绑定”
- 签名者不能否认自己的签名
- 容易被自动验证
- 签名不能被伪造

数字签名（续）

□ 五元组 (P, A, K, S, V)

- P是所有消息组成的有限集
- A是所有可能的签名组成的有限集
- K是所有可能的密钥组成的有限集
- S签名算法
- D验证算法

$$S_k : P \rightarrow A$$

$$V_k : P \times A \leftrightarrow \{0, 1\} :$$

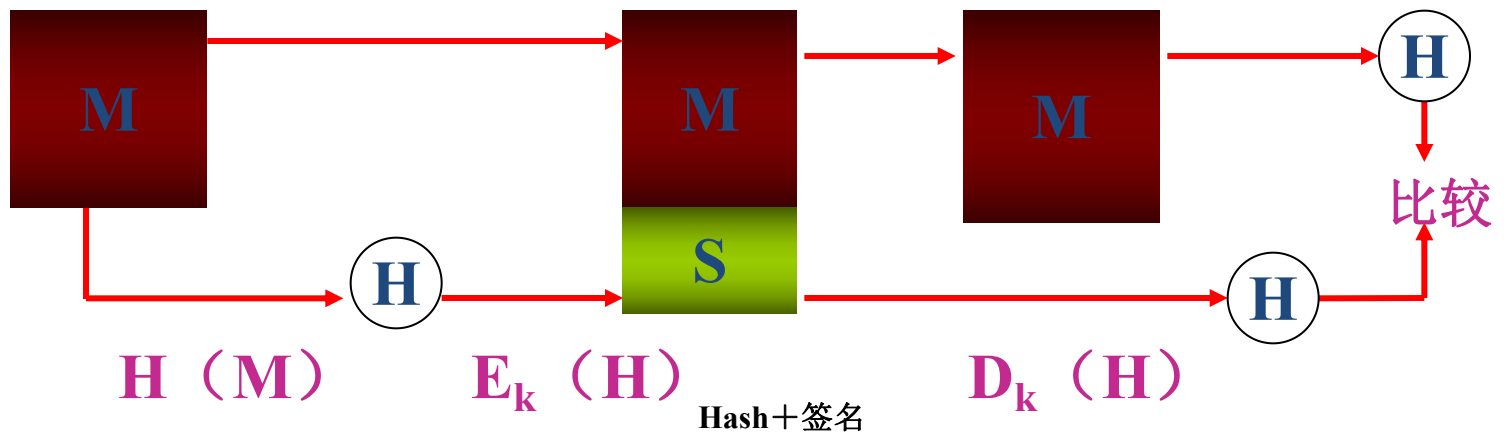
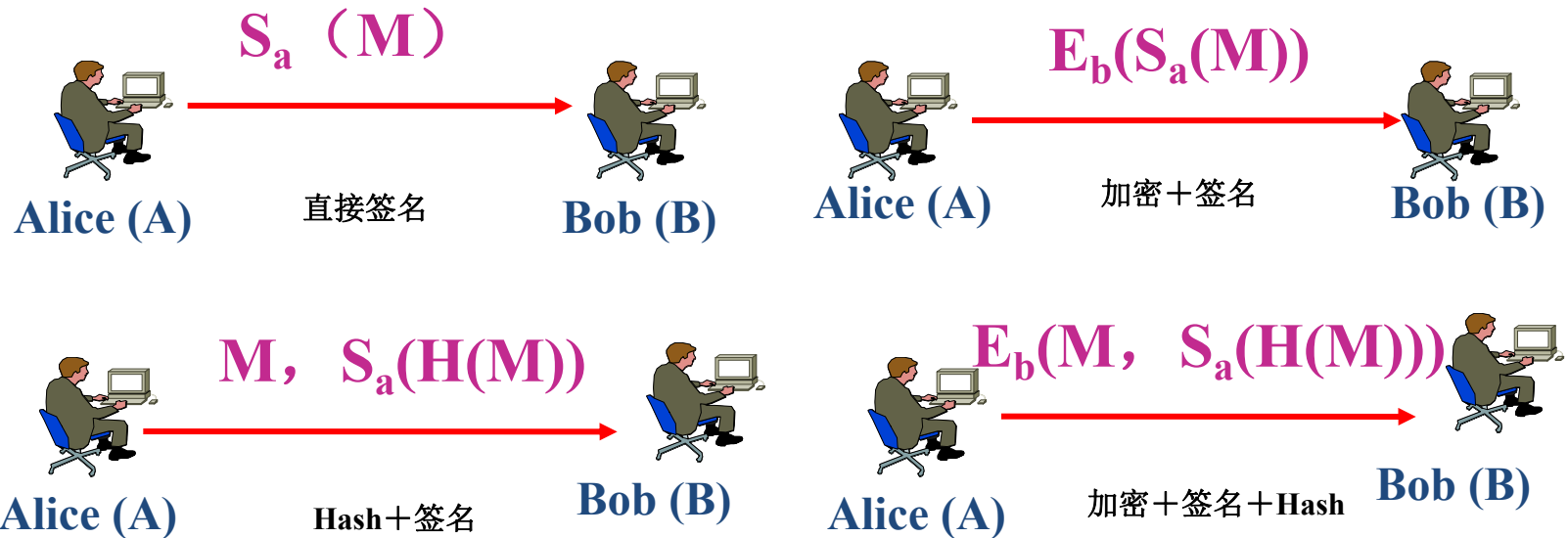
$$y = S_k(x), V_k(x, y) = 1, \text{ 否则 } V_k(x, y) = 0$$

数字签名（续）

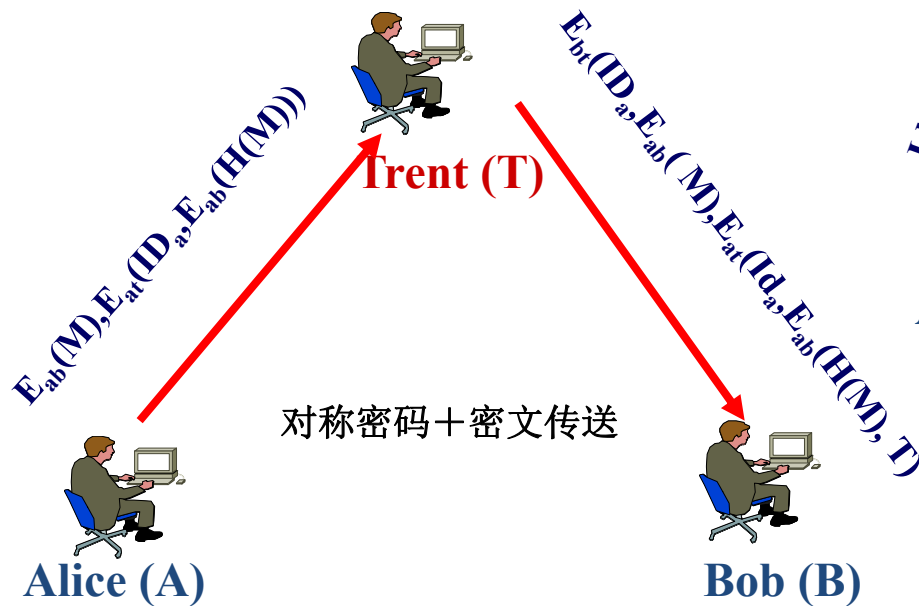
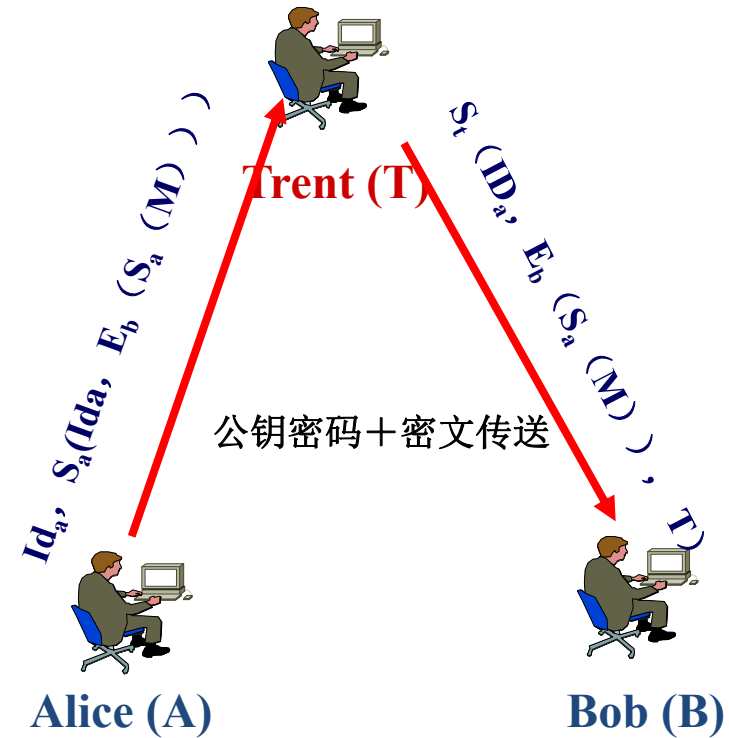
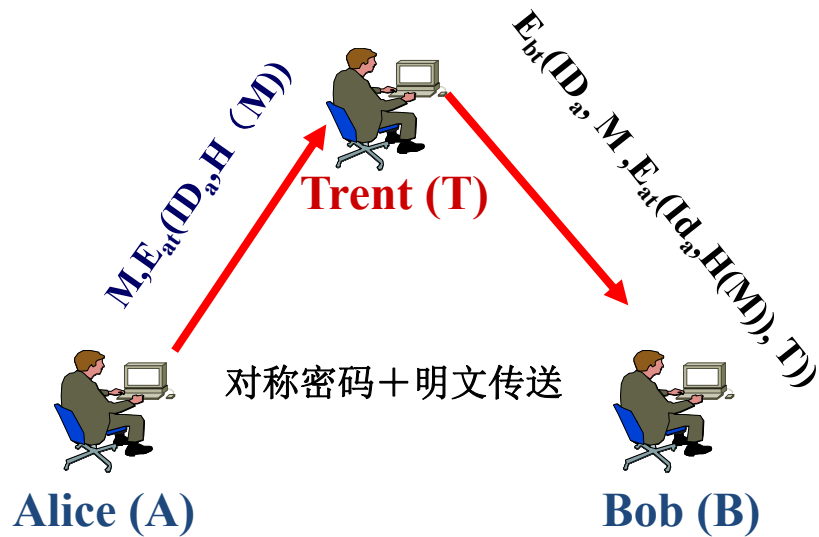
□ 可分为两类：

- 直接数字签名方案（**direct digital signature**）；
- 基于仲裁的数字签名方案（**arbitrated digital signature**）。

直接数字签名



仲裁数字签名



数字签名算法

□ 通用数字签名算法

- RSA
- ElGamal
- DSS/DSA/ECDSA

□ 不可否认的数字签名算法

□ 群签名算法

□ 环签名算法

□ 盲签名算法

□ 门限签名算法

□ . . .

RSA数字签名体制：Toy example

- 签名算法

设消息为 x ，则 x 的RAS签名是

$$y = x^d \bmod n.$$

- 验证算法

当接收方收到签名 (x, y) 后，计算

$$x' = y^e \bmod n.$$

如果 $x = x'$ ，则 y 是 x 的RAS签名。

RSA数字签名的安全性

□ 一般攻击

- 攻击方法:

设 n 与 e 为用户A的公钥，攻击者首先随意选择一个数据 y ，并用A的公钥计算

$$x = y^e \bmod n,$$

于是可以伪造A的一个RSA数字签名 (x, y) 。因为

$$x^d = (y^e)^d = y^{ed} = y \bmod n,$$

所以用户A对 x 的RSA数字签名是 y 。

- 这种攻击实际上成功的概率是不高的

因为对于选择的数据 y ，得到的 $x = y^e \bmod n$ 具有正确语义的概率是很低的。

- 抵抗措施

仔细设计数据格式

对数据的Hash值进行签名

RSA数字签名的安全性(2)

❑ 选择消息攻击

- 攻击方法:

假设攻击者**E**想伪造消息**x**的签名, 他容易找到两个数据**x₁**和**x₂**, 使得

$$x = x_1 \times x_2 \pmod n.$$

攻击者**E**设法让用户**A**分别对**x₁**和**x₂**进行签名, 得到

$$y_1 = x_1^d \pmod n, \quad y_2 = x_2^d \pmod n.$$

然后**E**可以计算

$$y = y_1 \times y_2 = x_1^d \times x_2^d = (x_1 \times x_2)^d = x^d \pmod n.$$

于是攻击者**E**得到了用户**A**对消息**x**的**RSA**数字签名**y**

- 抵抗措施

用户不要輕易地对其他人提供的随机数据进行签名
对数据的Hash值进行签名

RSA数字签名的安全性(3)

□ 利用签名进行攻击从而获得明文

● 攻击方法:

假设攻击者**E**已截获了密文 $c = x^e \bmod n$ ，他想求出明文 x 。于是，他选择一个小的随机数 r ，并计算

$$s = r^e \bmod n,$$

$$l = s \times c \bmod n,$$

$$t = r^{-1} \bmod n.$$

因为 $s = r^e$ ，所以 $s^d = (r^e)^d = r^{ed} \bmod n$ ， $r = s^d \bmod n$ 。然后**E**设法让签名者对 l 签名，于是**E**又获得 $k = l^d \bmod n$ 。攻击者**E**再计算：

$$t \times k = r^{-1} \times l^d = r^{-1} \times s^d \times c^d = r^{-1} \times r \times c^d = c^d = x \bmod n.$$

于是，获得了明文 x 。

● 抵抗措施

用户不要輕易地对其他人提供的随机数据进行签名

对数据的Hash值进行签名

RSA数字签名的安全性(4)

□ 对先加密后签名方案的攻击

● 攻击方法:

假设签名者**A**采用先加密后签名的方案把消息**x**发送给接收者**B**:

$$(x^{e_B} \bmod n_B)^{d_A} \bmod n_A.$$

如果**B**是不诚实的, 那么**B**可能伪造**A**的签名。例如, 假设**B**想抵赖收到**A**发的消息**x**, 谎称收到的是**x₁**。因为**n_B**是**B**的模数, 所以**B**知道**n_B**的分解, 于是能够计算模**n_B**的离散对数。即他能找到**k**满足:

$$(x_1)^k = x \bmod n_B.$$

然后, **B**再公布他的新公开密钥为**ke_B**。现在**B**宣布他收到的消息是**x₁**, 而不是**x**。由于下式成立, 所以**A**无法争辩。

$$(x_1^{ke_B} \bmod n_B)^{d_A} \bmod n_A = (x^{e_B} \bmod n_B)^{d_A} \bmod n_A$$

● 抵抗措施






签名者**A**应当在发送的数据中加入时间戳, 从而可证明是用公开密钥**e_B**对**x**加密, 而不是用新公开密钥**ke_B**对**x₁**加密。

对数据的Hash值进行签名。

RSA数字签名的安全性:小结

- ❑ 综合上述讨论，对于RSA数字签名系统，必须采取如下安全措施：
 - 不直接对消息进行签名，而应该对消息的Hash值进行签名。
 - 要采用先签名后加密的方式，而不要采用先加密后签名的方式。
- ❑ 可实用的安全RSA数字签名方案 RSA-PSS
 - 应用广泛：ISO/IEC 9796和ANSI X.30—199X以及美国联邦信息处理标准FIPS 186—2已经将RSA作为推荐的数字签名标准算法之一。
 - 美国RSA数据安全公司所开发的安全标准PKCS #1也是以RSA数字签名体制作为其推荐算法。

Certificate data:

	Name	Value
	Validity starts	2016/10/25 1
	Serial number	4E B2 00 67
	Issuer	cn=SwissSigi
	Subject	cn=SwissSigi
	Signature algorithm	SHA1 RSA

RSA数字签名方案RSA-PSS

密钥参数

假设 $(N, e, d, G, H, k_0, k_1) \leftarrow_U \text{Gen}(1^k)$, 其中 (N, e, d) 是 RSA 密钥材料, (N, e) 是公钥, $d = e^{-1} \pmod{\phi(N)}$ 是私钥; $k = |N| = k_0 + k_1$, 2^{-k_0} 和 2^{-k_1} 是可忽略的量; G, H 是杂凑函数, 满足:

$$G: \{0, 1\}^{k_1} \mapsto \{0, 1\}^{k-k_1-1}, H: \{0, 1\}^* \mapsto \{0, 1\}^{k_1}$$

(* G 输出的比特串被分成两个子串, 一个记为 G_1 , 它包括前面最重要的 k_0 比特, 另一个记为 G_2 , 它包括其余 $k - k_1 - k_0 - 1$ 比特 *)

签名生成 $\text{SignPSS}(M, d, N)$:

$$r \leftarrow_U \{0, 1\}^{k_0}; w \leftarrow H(\underline{M} \parallel r); r^* \leftarrow G_1(w) \oplus r;$$

$$y \leftarrow \underline{0 \parallel w \parallel r^* \parallel G_2(w)};$$

$$\text{return}(y^d \pmod{N}).$$

RSA数字签名方案RSA-PSS (2)

密钥参数

假设 $(N, e, d, G, H, k_0, k_1) \leftarrow_U \text{Gen}(1^k)$, 其中 (N, e, d) 是 RSA 密钥材料, (N, e) 是公钥, $d = e^{-1} \pmod{\phi(N)}$ 是私钥; $k = |N| = k_0 + k_1$, 2^{-k_0} 和 2^{-k_1} 是可忽略的量; G, H 是杂凑函数, 满足:

$$G: \{0,1\}^{k_1} \mapsto \{0,1\}^{k-k_1-1}, H: \{0,1\}^* \mapsto \{0,1\}^{k_1}$$

(* G 输出的比特串被分成两个子串, 一个记为 G_1 , 它包括前面最重要的 k_0 比特, 另一个记为 G_2 , 它包括其余 $k - k_1 - k_0 - 1$ 比特 *)

签名验证 $\text{Verify}(M, U, e, N)$:

$$y \leftarrow U^e \pmod{N};$$

$$\text{Parse } y \text{ as } b \parallel w \parallel r^* \parallel \gamma;$$

(* 也就是说, 令 b 是 y 的第一个比特, w 是后续的 k_1 比特, r^* 是再接下来的 k_0 比特, γ 是其余比特 *)

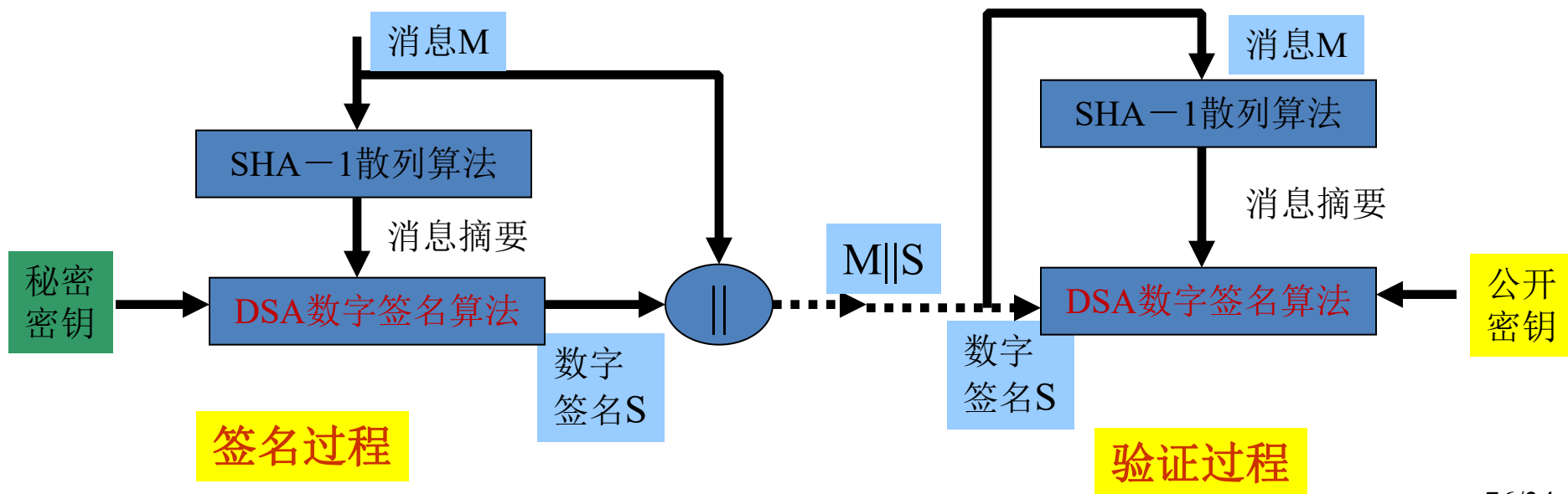
$$r \leftarrow r^* \oplus G_1(w);$$

$$\text{if}(H(M \parallel r) = w \wedge G_2(w) = \gamma \wedge b = 0) \text{ return}(\text{True})$$

$$\text{else return}(\text{False}).$$

数字签名标准 DSS: Digital Signature Standard

- ❑ DSS是美国国家标准与技术研究所（NIST）于1994年5月19日正式公布，同年12月1日正式作为美国联邦信息处理标准FIPS 186颁布（该标准后来经过了一些修改，目前的标准称为FIPS 186-2）。DSS中所采用算法通常称为**DSA(Digital Signature Algorithm)**。
- ❑ DSA已经在许多数字签名标准中得到推荐使用，除了联邦信息处理标准则186-2外，IEEE的P1363标准中，数字签名体制也推荐使用了DSA等算法。
- ❑ DSS中规定使用了安全散列函数(SHA-1)，DSA可以看作是ElGamal数字签名体制的一个变体，它也是基于离散对数问题，下图是DSS的签名与验证过程。



基于椭圆曲线密码的数字签名算法ECDSA

❑ 椭圆曲线数字签名算法（ECDSA）是使用椭圆曲线密码（ECC）对数字签名算法（DSA）的模拟。ECDSA于1998年成为ISO标准，1999年成为ANSI标准，并于2000年成为IEEE和NIST标准。

❑ 为什么要讲ECDSA？

- **Bitcoin** （大多数人了解密码学从Bitcoin开始）

比特币使用数字签名算法ECDSA对货币的所有权进行判定，依托的曲线是secp256k1。

- **区块链** （2019-10-25中央政治局第十八次集体学习）

Bitcoin 2008年最早开始使用ECDSA，随后Ethereum、Hyperledger Fabric以及主要区块链平台都开始采用。

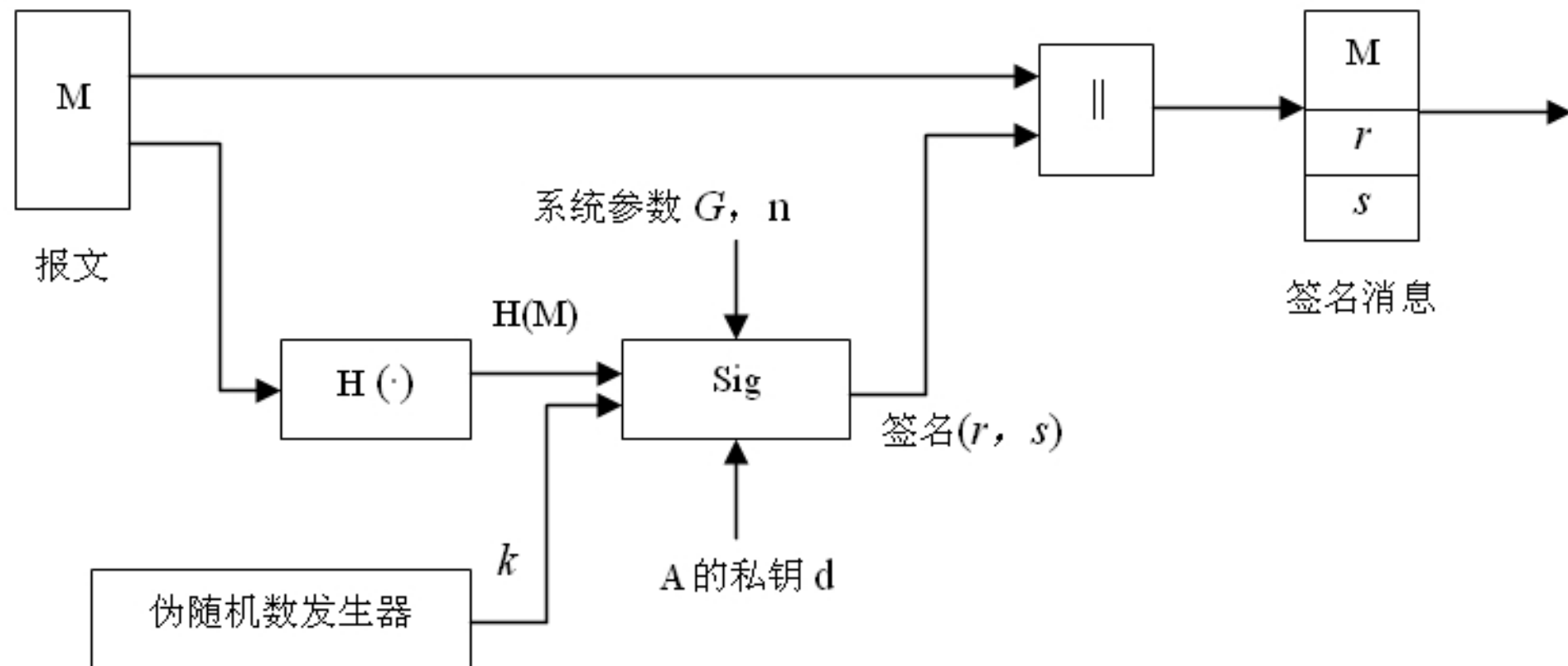
ECDSA在区块链应用中的七宗罪：<https://www.jianshu.com/p/7e3>

- **轻量高效** （移动设备早已超过PC)

与普通的离散对数问题DLP和大数分解问题 IFP不同，椭圆曲线离散对数问题（Elliptic curve discrete logarithm problem, ECDLP）没有亚指数时间的解决方法。因此椭圆曲线密码的单位比特强度要高于其他公钥体制。



ECDSA的签名过程概览



ECDSA算法的参数

设 E 为定义在有限域 F_p 上的椭圆曲线（ p 为大素数）。以 $E(F_p)$ 记椭圆曲线 E 在 F_p 中的有理点集，它是一个有限群。在 $E(F_p)$ 中选一个阶（order）为 n 的基点 G ，通常要求 n 是一个大素数。

每个用户选取一个大整数 d ($1 < d < n$) 作为签名私钥（严格保密），而以点 $Q = d \cdot G$ 作为其公钥，这样便形成一个椭圆曲线密码系统（ECC）。

该系统的公开参数有：

- ① 基域 F_p ；
- ② 椭圆曲线 E （如 $p > 3$ 时，由 $y^2 = x^3 + ax + b$, $4a^3 + 27b^2 \neq 0$, 定义的曲线）；
- ③ 基点 G 及其阶 n ；
- ④ 每个用户的公钥 $Q = d G$ 。

ECDSA算法的签名和验证过程

□ 签名时，用户A进行以下操作：

- ① 选择一个随机数 k ， $1 < k < n - 1$ ；
- ② 计算： $kG = (x_1, y_1)$ ，取 $r = x_1 \bmod n$ 。若 $r = 0$ ，回到①；
- ③ 计算： $s = k^{-1}(h(M) + d \cdot r) \bmod n$ ，其中 d 为签名方A的私钥。

若 $s=0$ ，回到①否则用户A把 (r, s) 作为消息 M 的数字签名，与 M 一起发送给接收方 B。

□ 用户B收到 (r, s, M) ，验签时进行以下操作：

- ① 验证 r 和 s 是否是区间 $[1, n-1]$ 中的整数，若不是则拒绝签名；
- ② 计算 $u_1 = h(M) \cdot s^{-1} \bmod n$ ； $u_2 = rs^{-1} \bmod n$
- ③ 利用签名方A的公钥 Q 计算：
 $X(x_2, y_2) = u_1 \cdot G + u_2 \cdot Q$ ，取 $v = x_2 \bmod n$ 。
- ④ 若 $v = r$ ，接受签名，否则拒绝签名。

ECDSA的正确性证明

□ 对ECDSA数字签名算法正确性的证明如下：

$$\begin{aligned} X(x_2, y_2) &= u_1 \cdot G + u_2 \cdot Q \\ &= h(M)s^{-1} G + r \cdot s^{-1} Q \bmod n \\ &= h(M) \cdot s^{-1} G + r \cdot s^{-1} d G \bmod n \\ &= s^{-1} G \cdot (h(M) + d \cdot r) \bmod n \end{aligned}$$

而由 $s = k^{-1} (h(M) + d \cdot r) \bmod n$

代入上式，有

$$\begin{aligned} X(x_2, y_2) &= k \cdot (h(M) + d \cdot r)^{-1} \cdot G \cdot (h(M) + d \cdot r) \bmod n \\ &= k \cdot G = (x_1, y_1) \end{aligned}$$

$\therefore x_2 = x_1$ ，即 $v = r$

密码算法间强度的等价性

□ 对称密码算法 vs. 非对称密码算法

Table 2: Comparable strengths

Security Strength	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
≤ 80	2TDEA ²¹	$L = 1024$ $N = 160$	$k = 1024$	$f = 160-223$
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224-255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256-383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384-511$
256	AES-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$

实用的密钥长度

- ❑ 攻击者的计算能力不断增强：云计算，超级计算机
- ❑ 美国国家机构的推荐做法
 - 红色：禁止使用
 - 黄色：不推荐使用（效率低）
 - 棕色：NSA不推荐使用

Table 4.1: NIST's recommendation for practical applications revision 4 [130]

Security level in bits	Block cipher	\mathbb{F}_p	\mathbb{F}_2^m	RSA
80	SKIPJACK	192	163	1024
112	Triple-DES	224	233	2048
128	AES Small	256	283	3072
192	AES Medium	384	409	7680
256	AES Large	521	571	15360

量子计算机对密码算法的影响

- ❑ 对密码算法有影响，对密钥无影响
- ❑ 50%削弱对称密码算法的强度
- ❑ 颠覆非对称密码算法的安全性

Table 1: Security Comparison						
Type of Attack	Symmetric Encryption			Public Key Encryption		
		Key Length	Bits of Security		Key Length	Bits of Security
Classical Computers	AES-128	128	128	RSA-2048	2048	112
	AES-256	256	256	RSA-15360	15,360	256
Quantum Computers	AES-128	128	64	RSA-2048	2048	25
	AES-256	256	128	RSA-15360	15,360	31

--<https://techbeacon.com/security/waiting-quantum-computing-why-encryption-has-nothing-worry-about>

欢迎各位同学交流指正！



Woo-Lam身份认证协议

- **预配置：** Trent作为可信第三方（TTP），Alice和Trent共享对称密钥 K_{AT} ，Bob和Trent共享对称密钥 K_{BT} ；
- **目标：** 即使Alice和Bob开始互不相识，Alice仍然能够向Bob证明自己。
 - ① Alice→Bob: Alice; **【我是Alice】**
 - ② Bob→Alice: NB; **【你证明一下吧】**
 - ③ Alice→Bob: {NB}KAT; **【你可以去验证，Trent可以证明】**
 - ④ Bob→Trent: {Alice, {NB}KAT}KBT; **【问Trent，X说她是Alice】**
 - ⑤ Trent→Bob: {NB}KBT; **【你如果看到NB，那就是了】**

Bob使用密钥KBT解密密文分组，如果解密后正确地返回Bob的一次性随机数，Bob就接受此次运行，否则拒绝。协议依赖TTP的可信性。

对Woo-Lam协议的攻击

□ Abadi和Needham对上述Woo-Lam协议的攻击（并行攻击！）

- ① Malice(“Alice”)→Bob: Alice; 【冒充Alice对Bob, 我是Alice】
1'. Malice→Bob: Malice; 【同时执行Malice本身的认证协议——我是Malice】
- ② Bob→Malice(“Alice”): N_B ; 【Bob执行Woo-Lam认证过程, 证明给我】
2'. Bob→Malice: N'_B ; 【Bob执行Woo-lam认证过程, 证明给我】
- ③ Malice(“Alice”)→Bob: $\{N_B\}K_{MT}$; 【执行Woo-lam, 注意 N_B , 你问TTP?】
3'. Malice→Bob: $\{N_B\}K_{MT}$; 【并行执行Woo-lam, 注意 N_B 不是 N'_B 】
- ④ Bob→Trent: $\{Alice, \{N_B\}K_{MT}\}K_{BT}$; 【Bob问Trent关于Alice的身份】
4'. Bob→Trent: $\{Malice, \{N_B\}K_{MT}\}K_{BT}$; 【Bob问Trent关于Malice的身份】
- ⑤ Trent→Bob: {“垃圾”} K_{BT} ; 【按照协议执行, N_b 被破坏了】
(产生“垃圾”是因为Trent使用 K_{AT} 对密文组 $\{N_B\}K_{MT}$ 进行了解密)
5'. Trent→Bob: $\{N_B\}K_{BT}$; 【按照协议, N_b 返回了, Alice的身份被确认】
- ⑥ Bob拒绝和Malice的运行;
(因为解密后返回的是“垃圾”而不是一次性随机数 N'_B)
6'. Bob接受“和Alice的运行”, 但实际上是和Malice的运行。
(因为解密返回的 N_B 是正确的)