遵守国家信息安全法律法规，

仅限个人学习使用！！！

内部学习资料，请勿随意传播！

计算机科学进展

（信息安全）

钱 权

qqian@shu.edu.cn

上海大学计算机学院

2025年3月

课程安排

- 1、加密与认证
- 2、软件破解（EXE文件破解）
- 3、常见攻击及预防（SQL注入与XSS）
- 4、网络安全（防火墙/VPN/IDS）
- 5、移动平台安全（Android/iOS）

课程考核

- 出勤及课堂表现：　　10次，10%
- 演讲+PPT　　5次，　40%
- 课程报告　　1份，　50%
- **分组：**　　**3人**

第1讲 加密与认证

主要内容

- 消息摘要
- 消息认证
- 数字签名
- 数字证书

研讨要求

- 1、程序演示和PPT展示；
- 2、针对加密、认证（原理与技术）的PPT宣讲；
- 3、每组3人；

程序演示

- 采用Java/Python语言编写一个较为完整的加密与认证程序，要求具有：
  - 具有较完整的图形化界面；
  - 使用MD5、SHA系列算法，实现消息摘要，确保消息的完整性；
  - 使用DES、AES等算法实现对称加密，确保消息的机密性；
  - 使用RSA算法，实现公钥加密，且用私钥解密，比较不对称加密和对称加密的性能；
  - 实现基于数字证书的数字签名和验证（含证书的生成和创建）；

研讨内容

- 每组3人，内容包括：
  - 设计思想、主要数据结构和算法；
  - 所用到的主要函数及其含义；
  - 程序界面和运行情况；
- 每组做PPT交流（**每位组员上台介绍自己做的内容**）实现的情况和经验分享。

消息摘要

- 消息摘要的作用
- 单向散列函数
- MD5算法
- SHA安全散列算法

消息摘要的作用

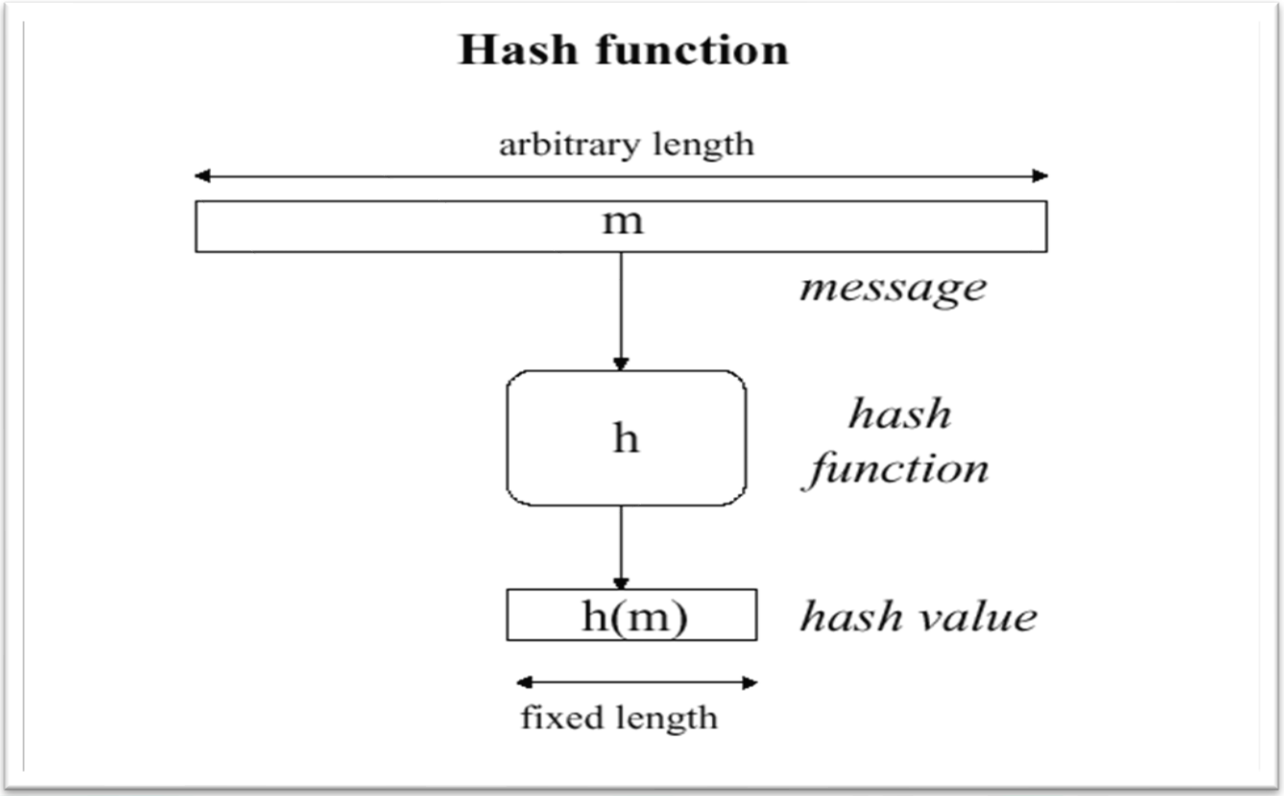
- 在网络安全目标中，要求信息在生成、存储或传输过程中保证不被偶然或蓄意地删除、修改、伪造、乱序、重放、插入等破坏和丢失，因此需要一个较为安全的标准和算法，以保证数据的完整性。
- 常见的消息摘要算法有：
  - Ron Rivest设计的MD（Standard For Message Digest，消息摘要标准）算法
  - NIST设计的SHA（Secure Hash Algorithm，安全散列算法）

单向散列函数

- 消息摘要算法采用单向散列（hash）函数从明文产生摘要密文。
- 摘要密文又称为哈希函数、数字指纹（Digital Fingerprint）、压缩（Compression）函数、紧缩（Contraction ）函数、数据认证码DAC（Data authentication code）、篡改检验码MDC（Manipulation detection code）。
- 散列函数的输出值有固定的长度，该散列值是消息M的所有位的函数并提供错误检测能力，消息中的任何一位或多位的变化都将导致该散列值的变化。从散列值不可能推导出消息M，也很难通过伪造消息M’来生成相同的散列值。

HASH函数

- 也称哈希函数、散列函数
- 是一种由不定长的自变量到定长的函数值的单向映射



HASH函数

- Hash函数的值称为作为自变量的消息的“散列值”或“消息摘要”、“数字指纹”

变长，称为“原像”　　单向“映射”　　定长，称为“映像”

- 不同的原像映射出相同的映像称为“碰撞”或“冲突”
- 映像相同的两个原像互称为“等价原像”

单向散列函数的特点

- 单向散列函数 H(M)作用于一个**任意长度**的数据M，它返回一个**固定长度**的散列h，其中h的长度为m，h称为数据M的摘要。单向散列函数有以下特点：
  - 给定M，很容易计算h；
  - 给定h，无法推算出M；
  - 除了单向性的特点外，消息摘要还要求散列函数具有“防碰撞性”的特点：
    - 给定M，很难找到另一个数据N，满足H(M)=H(N)。

单向散列函数的抗碰撞性

- 抗碰撞性的能力体现出单向散列函数对抗生日攻击和伪造的能力。
- 弱抗碰撞性（Weak collision resistance）：
  - 对于任意给定的M，找到满足M≠N且H(M)=H(N)的N，在计算上是不可行的；
- 强抗碰撞性（Strong collision resistance）：
  - 找到任何满足H（x）=H（y） 的偶对（x，y）在计算上是不可行的。

哈希函数分类

- 根据安全水平
  - 弱无碰撞
  - 强无碰撞

    注：强无碰撞自然含弱无碰撞！
- 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(Message Authentication Code)**
  - 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(Message Detection Code)**

HASH函数的应用

- 由Hash函数产生消息的散列值
- 以消息的散列值来判别消息的完整性
- 用加密消息的散列值来产生数字签名
- 用口令的散列值来安全存储口令（认证系统中的口令列表中仅存储口令的Hash函数值，以避免口令被窃取。认证时用输入口令的Hash函数值与其比较）

哈希函数-生日攻击

- 如果采用传输加密的散列值和不加密的报文M，攻击者需要找到M^-，使得$H(M^-)=H(M)$，以便使用替代报文来欺骗接收者。
- 一种基于生日悖论的攻击可能做到这一点，生日问题：一个教室中，最少应有多少个学生，才使至少有两人具有相同生日的概率不小于1/2?
  - 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**

附：生日问题和生日攻击

- 生日问题
  - 一个教室中至少有几个学生才能使有两个学生生日相同的概率不小于1/2；
- 等价于“球匣问题”
  - 设J个球随机扔进N个匣子，存在一个匣子中至少有两个球的概率为p，则可以推导出: $J^2 \approx -2N\ln(1-p)$或 $p \approx 1-e^{-J^2/2/N}$
- 答案
  - 将365个生日看作N=365个匣子，将学生看作球，p=0.5，则由上式可算出J≈23，即23个学生中有两个学生生日相同的概率不小于1/2；

附：生日问题和生日攻击

- 80人的班级中至少有两个学生生日相同的概率约为0.99984422904
- 散列碰撞问题：
  - 几个m位的比特串中发生碰撞（至少两个串相同）的概率不小于1/2：在上述公式中，令$N=2^m$，p=0.5，则可估算出$J \approx 2^{m/2}$。即$2^{m/2}$个m位比特串中发生碰撞的概率不小于1/2。如：2^4个8位比特串中发生碰撞的概率不小于1/2

生日攻击举例

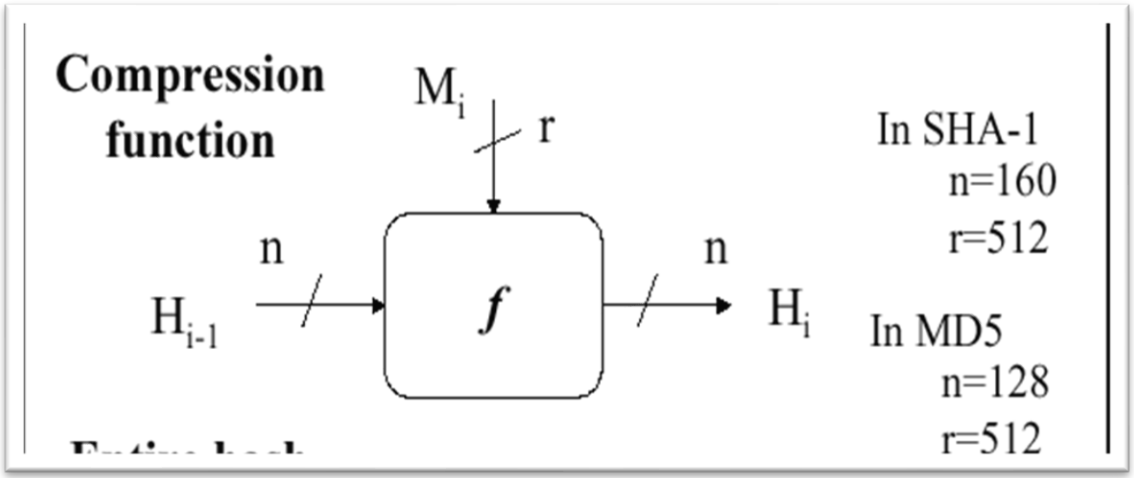
- 假设张三承诺支付李四100万，约定由李四负责起草合同，并通过8位的散列码H(M)实施信息认证。聪明而无德的李四先起草一个100万的版本，并通过变化其中3个无关紧要之处以得到2^3=8个不同的消息明文并计算它们的H(M)，形成集合A；然后再起草一个200万的版本，用同样方法又得到2^3=8 个不同的消息明文及其H(M)，形成集合B。
- 由生日问题知：2^4个8位比特串中发生碰撞的概率不小于1/2，故在A和B共2^4 =16个H(M)中有可能存在相同的一对，并极有可能一个在A中而另一个在B中。假设与它们对应的明文为M_A （100万版） 和M_B （200万版） 。于是李四用M_A让张三签署并公证，而在传送时偷偷地用M_B替代M_A 。由于H(M_A)= H(M_B)，故张三确信签署的文件未被篡改。当李四要求张三支付200万时，法院根据M_B判李四胜诉，而张三因此损失100万。

生日问题与生日攻击

- 生日攻击的前提就是存在碰撞
- 为有效抵御生日攻击，必须使散列码的位数充分大，使得获得碰撞在计算上是不可能的。
- 著名的SHA-1散列算法的散列码取160位，由生日原理攻击者至少得算出$2^{80}\approx1.2\times10^{25}$个散列码才有机会遇到碰撞（如以每秒算1000万个散列码计算，需380亿年）

安全HASH函数的一般结构

- 输入数据分成L个长度固定为r的分组：$M=(M_1,M_2,...,M_L)$
- 末组附加消息的长度值并通过填充凑足r位
- 压缩函数 f使用n位的链接变量H_i ,其初值H_0=IV可任意指定
- 压缩函数 f的最后n位输出H_L取作散列值

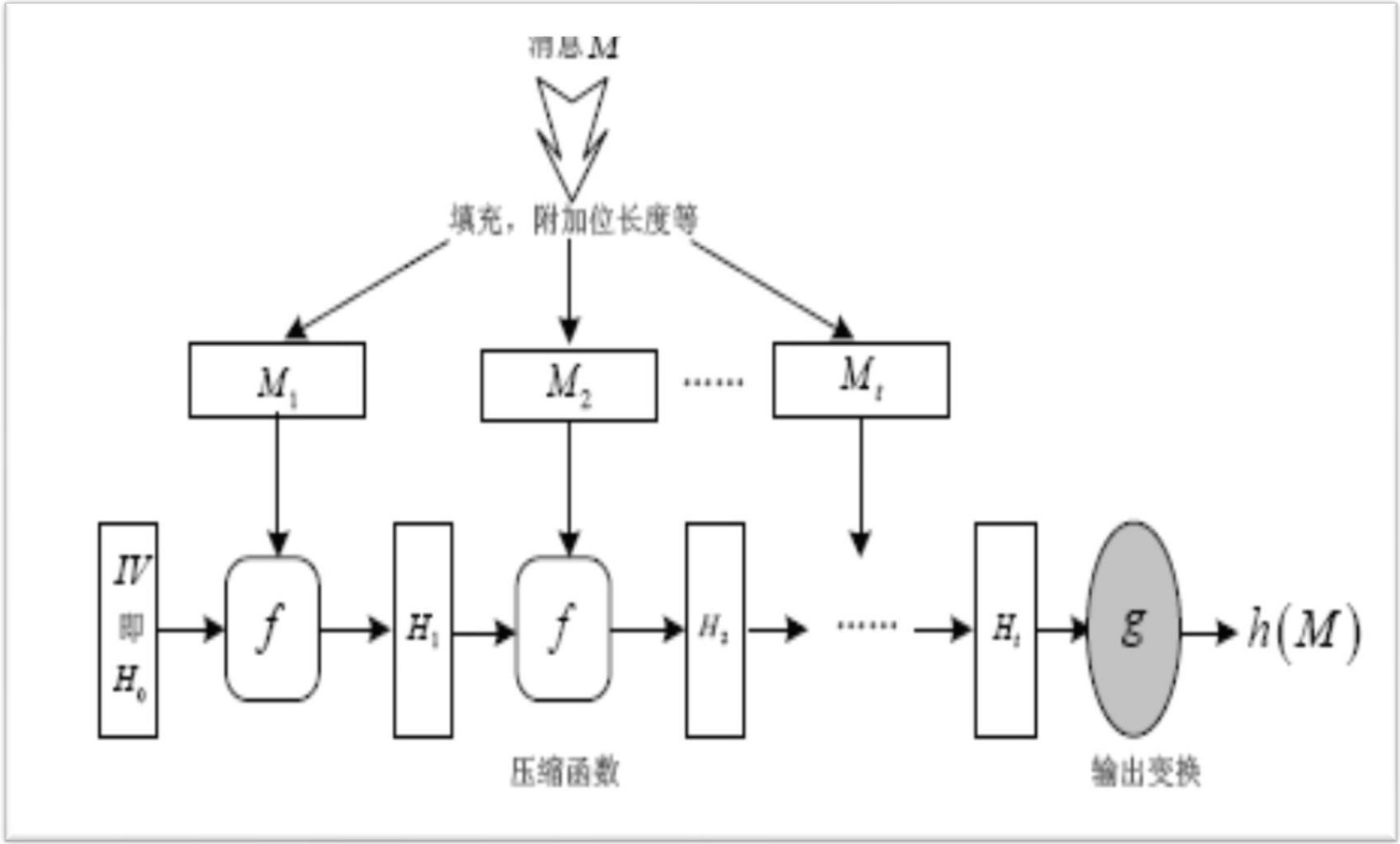


H_0=IV

$H_i=f(H_{i-1},M_i)$, $(1<=i<=L)$

$h=H(M)=H_L$

安全HASH函数的一般过程



MD5算法

- Merkle于1989年提出hash function模型
- Ron Rivest于1990年提出MD4
- 1992年， Ron Rivest提出MD5（RFC 1321）
- 在最近数年之前，MD5是最主要的hash算法
- 现行美国标准SHA-1以MD5的前身MD4为基础

- 输入：任意长度消息
- 输出：128bit消息摘要（16字节编码，32字符）
- 处理：以512bit输入数据块为单位

```
C:\Users\use>certutil -hashfile message.txt MD5
MD5 的 message.txt 哈希:
bc8dac5f66fa5854e9ba0972ef3d51c8
CertUtil: -hashfile 命令成功完成。
```

SHA安全散列算法

- 1992年NIST制定了SHA（128位）
- 1993年SHA成为标准（FIPS PUB 180）
- 1994年修改产生SHA-1（160位）
- 1995年SHA-1成为新的标准，作为SHA-1（FIPS PUB 180-1/RFC 3174），为兼容AES的安全性，NIST发布FIPS PUB 180-2，标准化SHA-256， SHA-384和SHA-512

- 输入：消息长度<2^{64}
- 输出：160bit消息摘要
- 处理：以512bit输入数据块为单位
- 基础是MD4

SHA算法的扩展

- SHA-256
  - 摘要大小由SHA-1的160位扩大到256位
- SHA-384
  - 消息大小由SHA-1的2^{64}位扩大到2^{128}位
  - 分组大小由SHA-1的512位扩大到1024位
  - 字长由SHA-1的32位（双字）扩大到64位（4字）
  - 摘要大小由SHA-1的160位扩大到384位
- SHA-512
  - 摘要大小由SHA-384的384位扩大到512位

消息摘要的安全隐患

- 隐患：无法完全阻止数据的修改。
- 如果在数据传递过程中，窃取者将数据窃取出来，并且修改数据，再重新生成一次摘要，将改后的数据和重新计算的摘要发送给接收者，接收者利用算法对修改过的数据进行验证时，生成的消息摘要和收到的消息摘要仍然相同，消息被判断为“没有被修改”。
- 做法：除了需要知道消息和消息摘要之外，还需要知道发送者身份---消息验证码。

消息验证码

- 单向加密（Hash）的结果也叫做消息摘要，因为不同的数据加密得到的结果不同，因此可以较好地验证数据的完整性。
- 利用MD5算法生成消息摘要，可以验证数据是否被修改，方法是：根据收到的数据，重新利用MD5算法生成摘要，和原来的摘要相比较，如果相同，说明数据没有被修改，反之，说明数据被修改了。

消息验证码

- 消息验证码和MD5/SHA1算法不同的地方是：
  - 在生成摘要时，发送者和接收者都拥有一个共同的密钥。
  - 该密钥可以是通过对称密码体系生成的，事先被双方共有，在生成消息验证码时，还必须要有密钥的参与。
  - 只有同样的密钥才能生成同样的消息验证码。

消息验证码的局限性

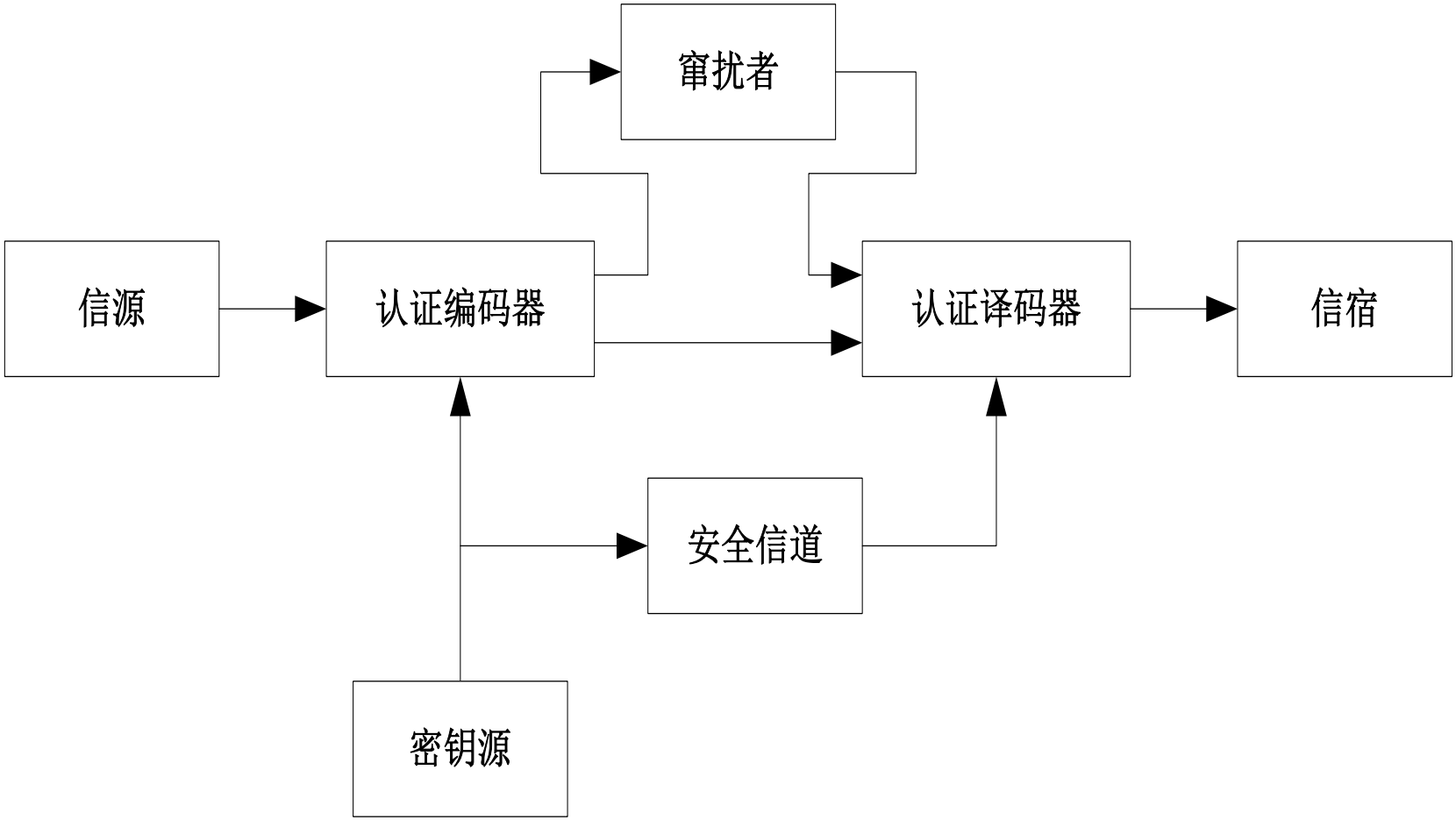
- 消息验证码可以保护信息交换双方不受第三方的攻击，但是它不能处理通信双方的相互攻击
  - 信宿方可以伪造消息并称消息发自信源方，信源方产生一条消息，并用和信宿方共享的密钥产生认证码，并将认证码附于消息之后
  - 信源方可以否认曾发送过某消息，因为信宿方可以伪造消息，所以无法证明信源方确实发送过该消息

- 在收发双方不能完全信任的情况下，引入数字签名来解决上述问题
  - 数字签名的作用相当于手写签名

消息认证

- 用于对抗信息主动攻击之一：消息伪造或篡改
- 目的之一：验证信息来源的真实性
- 目的之二：验证信息的完整性

消息认证的模型

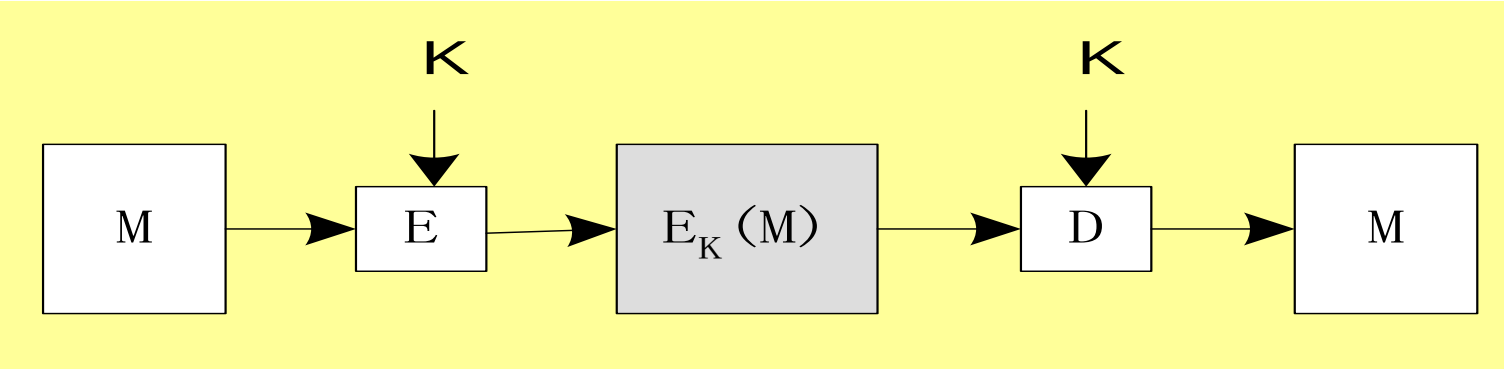


消息认证的方式

- **加密认证**——用消息的密文本身充当认证信息
  - 消息加密的认证；私钥加密公钥解密；公钥私钥双重加解密
- **消息认证码**MAC(Message Authentication Code)——由以消息和密钥作为输入的公开函数产生的认证信息
  - 简单MAC认证；基于明文认证；基于密文认证
- **散列值**——由以消息作为唯一输入的散列函数产生的认证信息（无需密钥）
  - 6种常用的方式

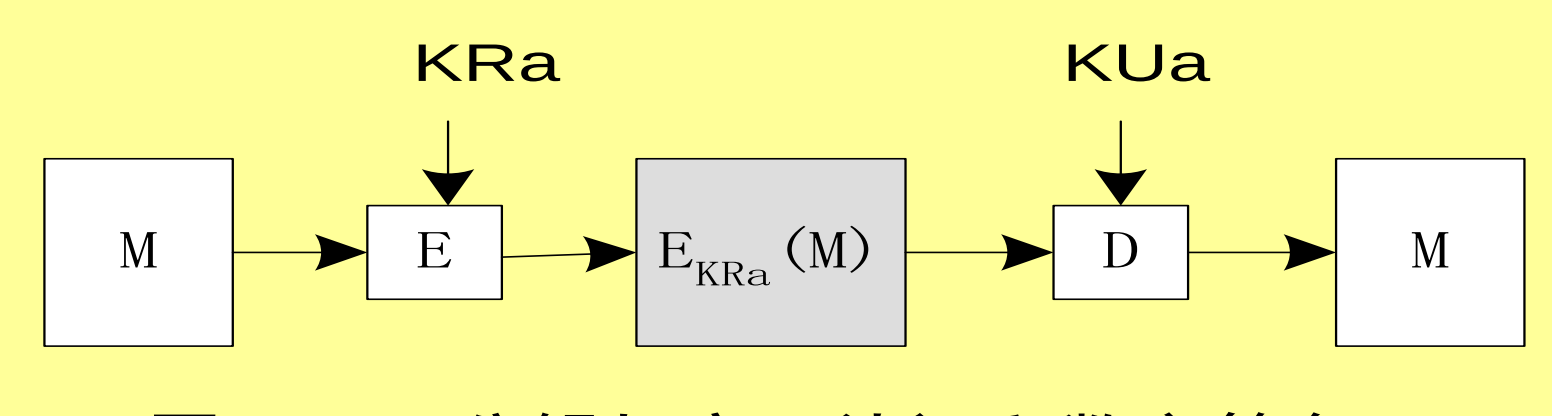
基于消息加密的认证

- 用对称密码体制进行加密认证
  - **过程**——用同一密钥加密、解密消息
  - **作用**——认证+保密
  - **原理**——攻击者无法通过改变密文来产生所期望的明文变化
  - **特点**——接收方需要判别消息本身的逻辑性或合法性。**“我请你吃饭”被乱改成“我请你謯斸”**



私钥加密，公钥解密

- **过程**——发送者用自己的私钥加密明文、接收者用发送者的公钥解密密文
- **作用**——认证及签名，但不保密
- **原理**——因不知发送者的私钥，故其他人无法产生密文或伪造签名

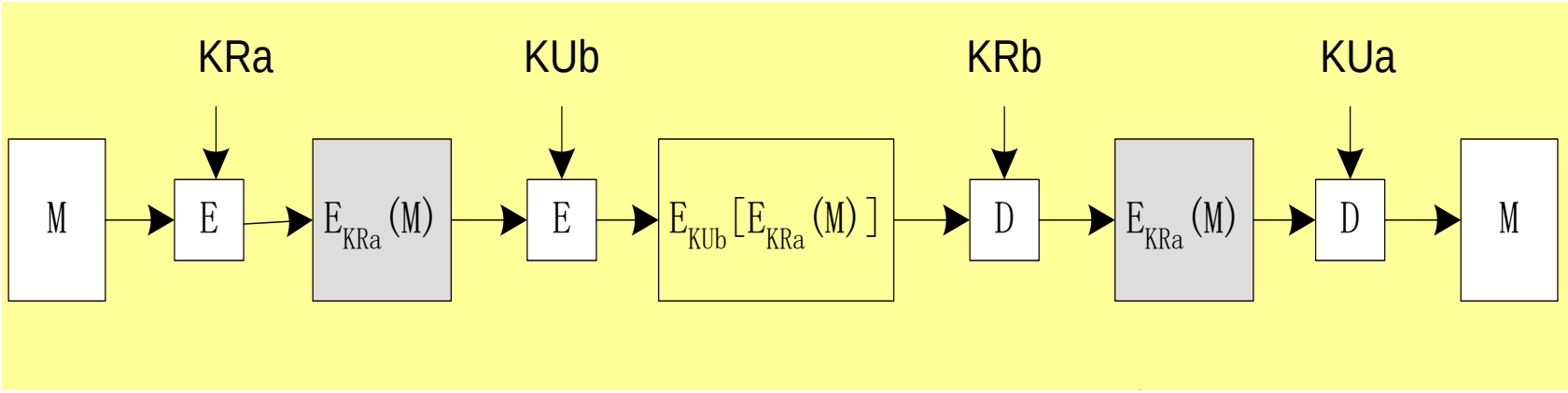


私钥加密，公钥解密

- 注意：若用公钥加密、私钥解密，则无法起到认证的作用。因为知道公钥的人都可以通过产生伪造的密文来篡改消息。

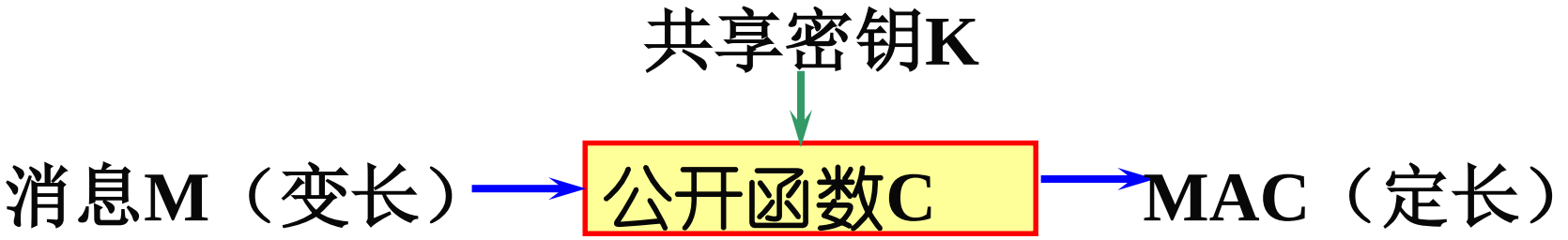
用私钥、公钥双重加密、解密

- **过程**——发送者先用自己的私钥加密明文，再用接收者的公钥加密一次；接收者先用自己的私钥解密密文，再用发送者的公钥解密一次
- **作用**——认证、签名，且保密
- **原理**——认证、签名由发送者的私钥加密实现；保密性由接收者的公钥加密保证

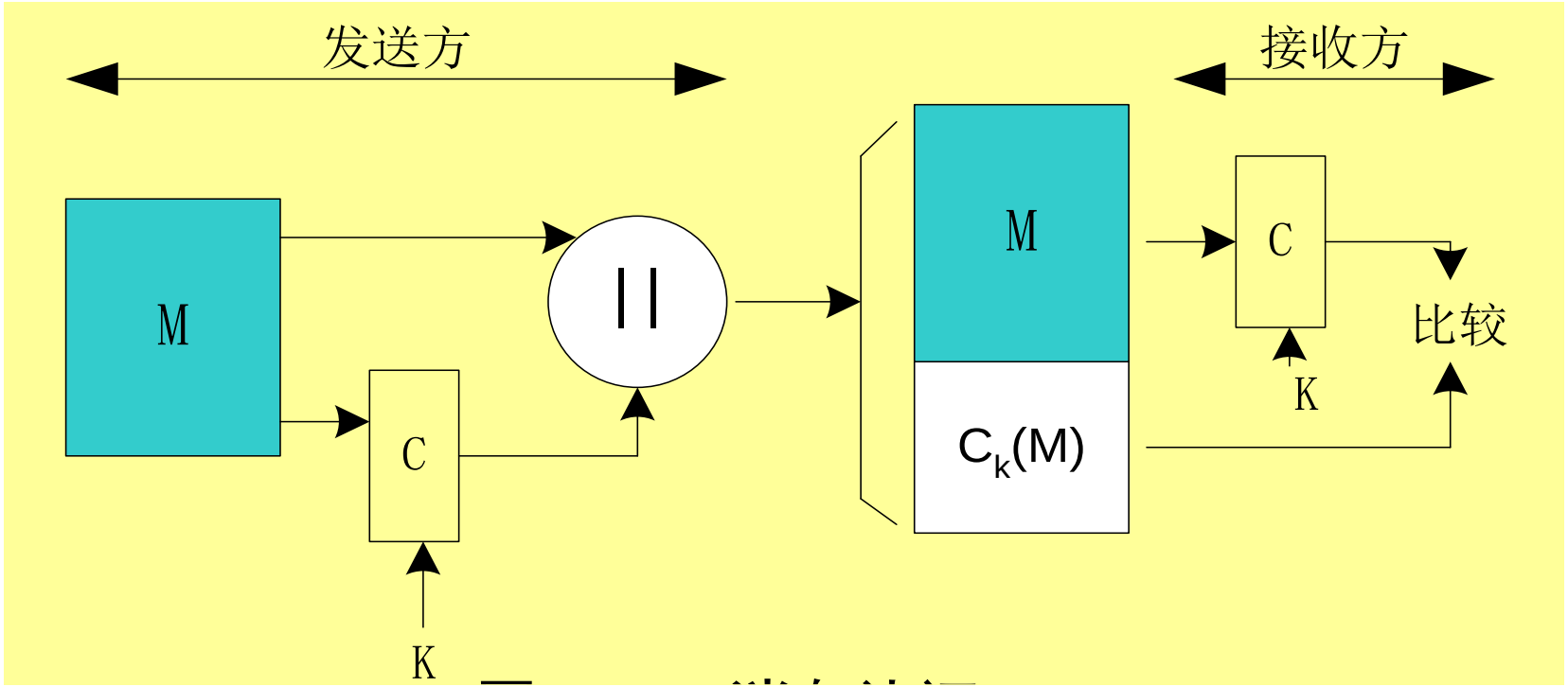


基于消息验证码MAC的认证

- **产生**——发送者以消息M和与接收者共享的密钥K为输入，通过某公开函数C进行加密运算得到MAC
- **传送并接收**——M+MAC
- **认证**——接收者以接收到的M和共享密钥K为输入，用C重新加密算得MAC’，若MAC’=MAC，则可确信M未被篡改
- **作用**——认证，但不保密



基于消息验证码MAC的认证



为保密而改进的方案——基于明文的认证

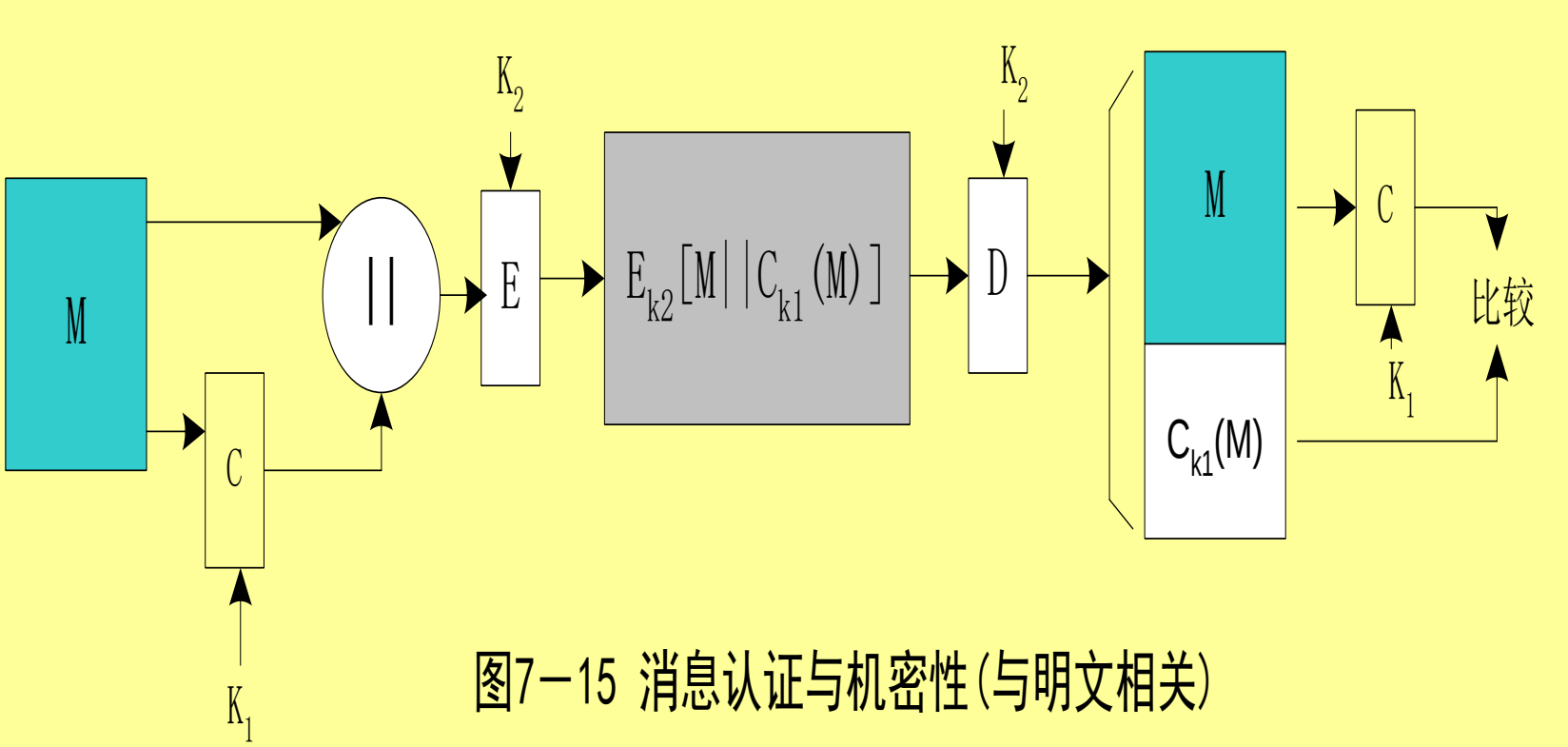
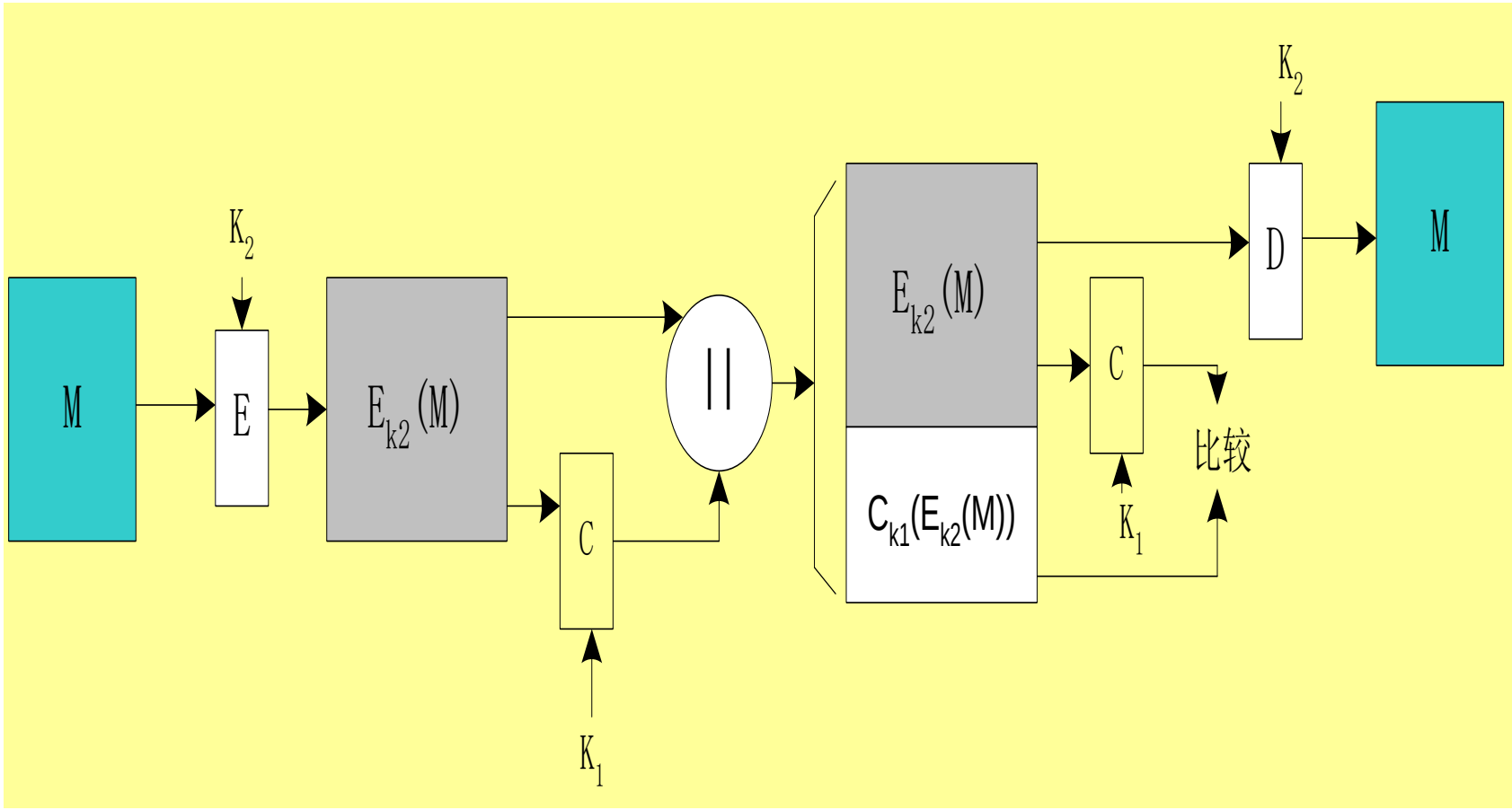


图7－15 消息认证与机密性(与明文相关)

为保密而改进的方案——基于密文的认证

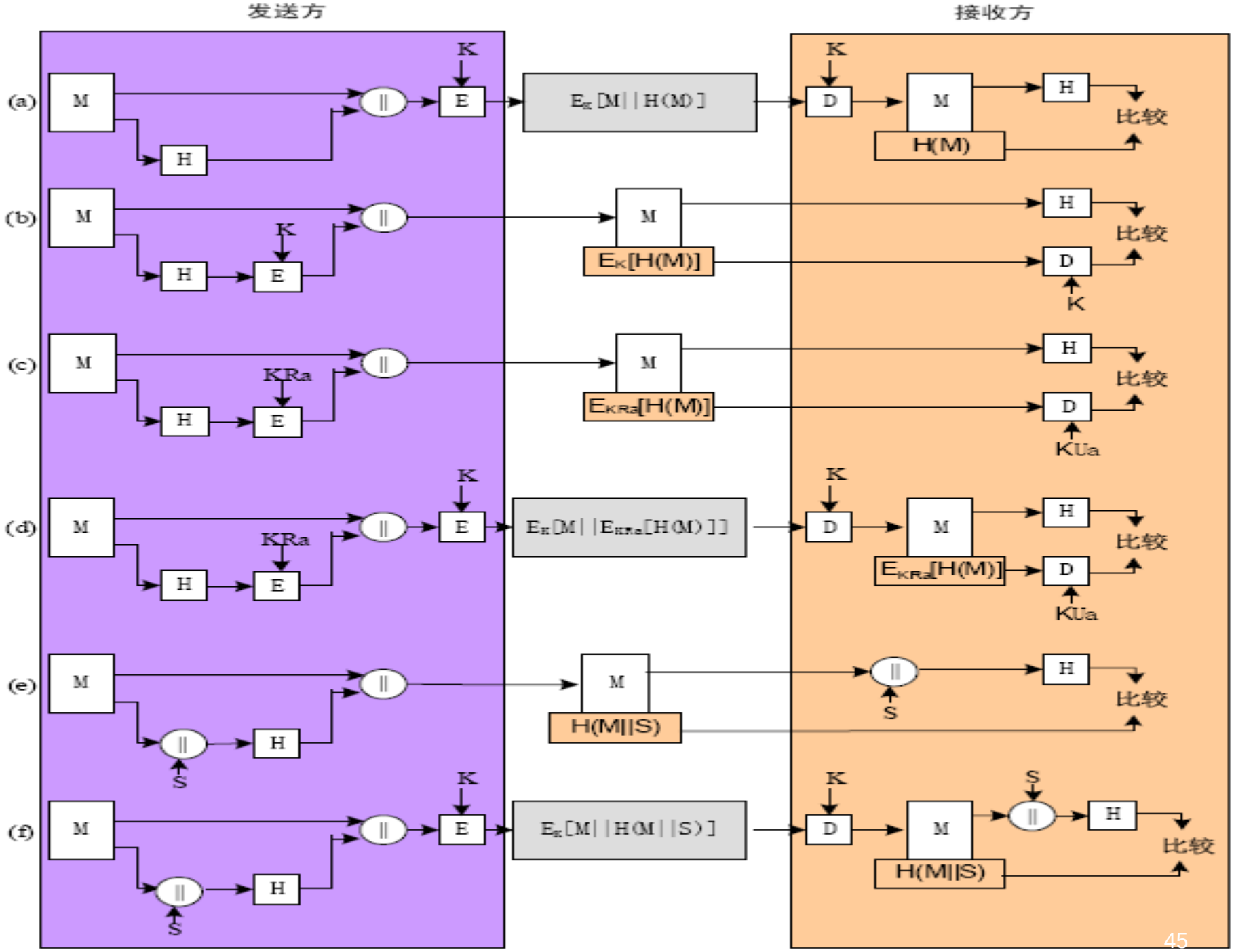


基于散列值的认证

- 1、对附加了散列值的消息实施对称加密，得到并发送$E_k(M+H(M))$
  - 认证+保密
- 2、仅对散列值实施对称加密，得到$E_k(H(M))$，并与M一起发送
  - 认证+不保密
- 3、对散列值实施私钥加密，得到$E_{KRa}(H(M))$并与M一起发送
  - 认证+签名，不保密

基于散列值的认证

- 4、将消息与用私钥加密后的散列值一起再用共享密钥加密，最后得到$E_k(M+E_{KRa}(H(M)))$并发送
  - 认证+签名+保密
- 5、将消息串接一个由通信各方共享的密值S后计算散列值，得到H(M+S)并与M一起发送
  - 认证，不保密
- 6、先将消息串接一个由通信各方共享的密值S后计算散列值，再将它与消息M一起用共享密钥加密，最后得到$E_k(M+H(M+S))$并发送
  - 认证+保密


发送方
接收方
(a)
(b)
(c)
(d)
(e)
(f)
M
H
E
D
K
KRa
KUa
S
||
比较
H(M)
H(M||S)
E_K[M||H(M)]
E_K[H(M)]
E_KRa[H(M)]
E_K[M||E_KRa[H(M)]]
E_K[M||H(M||S)]

消息验证码Java代码示例

```
1. import javax.crypto.KeyGenerator;
2. import javax.crypto.Mac;
3. import javax.crypto.SecretKey;
4. import javax.crypto.spec.SecretKeySpec;
5. public class P12_07
6. {
7.     public static void main(String[] args)
8.     {
9.         //要计算消息验证码的字符串
10.            String str="Java_安全编程技术";
11.            System.out.println("明文是:" + str);
12.            try
13.            {
14.                //用 DES 算法得到计算验证码的密钥
15.                KeyGenerator keyGen=KeyGenerator.getInstance("DESede");
16.                SecretKey key=keyGen.generateKey();
17.                byte[] keyByte=key.getEncoded();
18.                //生成 MAC 对象
19.                SecretKeySpec SKS=new SecretKeySpec(keyByte,"HMACMD5");
20.                Mac mac=Mac.getInstance("HMACMD5");
21.                mac.init(SKS);
22.                //传入要计算验证码的字符串
23.                mac.update(str.getBytes("UTF8"));
24.                //计算验证码
25.                byte[] certifyCode=mac.doFinal();
26.                System.out.println("密文
是:" + newString(certifyCode));
27.            }
28.            catch (Exception e)
29.            {
30.                e.printStackTrace();
31.            }
32.        }
```

数字签名

- 数字签名的概念和作用
- 数字签名的特点
- 数字签名与消息认证的区别
- 数字签名分类与常用算法

数字签名的概念和作用

- 在公钥体制中，用接受者的公钥加密消息得到密文，接受者用自己的私钥解密密文得到消息。加密过程任何人都能完成，解密过程只有接受者能够完成。
- 考虑一种相反的过程，发送者用自己的私钥“加密”消息得到“密文”，然后利用发送者的公钥“解密”密文得到消息。
- 很显然，加密只有发送者能够完成，而解密任何人都可以完成。 所以，任何人都可相信是特定的发送者产生了该消息，这就相当于**“签名”**，证明一个消息的所属。

数字签名的概念和作用

- 随着计算机通信网的发展，人们希望通过电子设备实现快速、远距离的交易，数字(或电子)签名法应运而生，并开始用于商业通信系统，如电子邮递、电子转账和办公自动化等系统。
- 随着计算机网络的发展，过去依赖于手书签名的各种业务都可用这种电子数字签名代替，它是实现电子贸易、电子支票、电子货币、电子购物、电子出版及知识产权保护等系统安全的重要保证。

数字签名的特点

- 传统签名的基本特点
  - 与被签的文件在物理上不可分割
  - 签名者不能否认自己的签名
  - 签名不能被伪造
  - 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

数字签名的主要特征

- 数字签名必须具有下述特征
  - 收方能够确认或证实发方的签名，但不能伪造，简记为**R1-条件（unforgeable)**
  - 发方发出签名的消息给收方后，就不能再否认他所签发的消息，简记为**S-条件(non-repudiation)**
  - 收方对已收到的签名消息不能否认，即有收到认证，简记作**R2-条件**
  - 第三者可以确认收发双方之间的消息传送，但不能伪造这一过程，简记作**T-条件**

数字签名与消息认证的区别

- **与手书签名的区别：**手书签名是模拟的，且因人而异。数字签名是0和1的数字串，因消息而异。
- **与消息认证的的区别：**消息认证使收方能验证消息发送者及所发消息内容是否被窜改过。当收发者之间没有利害冲突时，这对于防止第三者的破坏来说是足够了。但当收者和发者之间有利害冲突时，就无法解决他们之间的纠纷，此时须借助满足前述要求的数字签名技术。
- **与消息加密区别：**消息加密和解密可能是一次性的，它要求在解密之前是安全的；而一个签名的消息可能作为一个法律上的文件，如合同等，很可能在对消息签署多年之后才验证其签名，且可能**需要多次验证**此签名。因此，签名的安全性和防伪造的要求更高些，且要求证实速度比签名速度还要快，特别是联机在线实时验证。

数字签名的分类

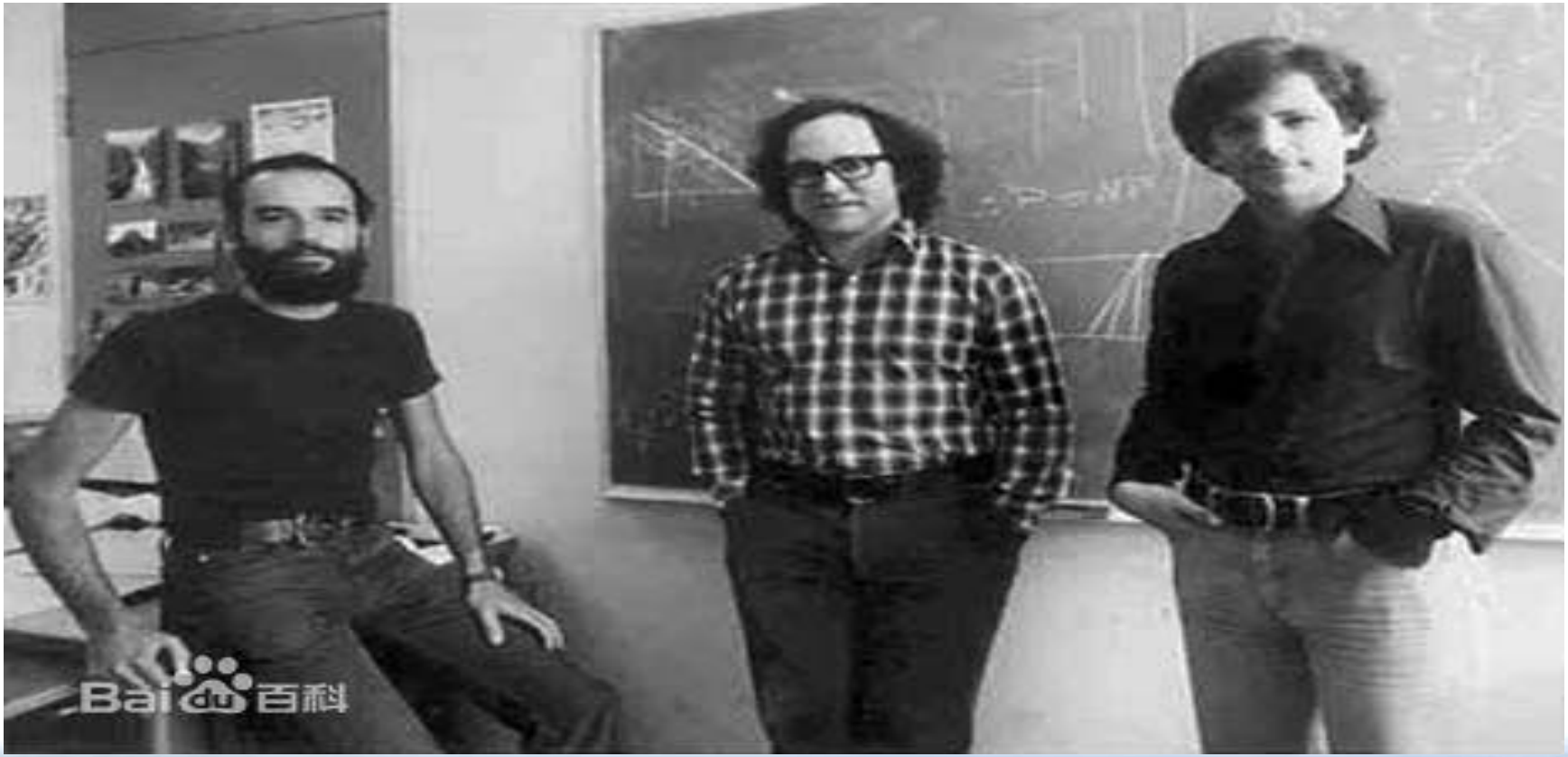
- 根据签名的内容分
  - 对整体消息的签名
  - 对压缩消息的签名
- 按明、密文的对应关系划分
  - 确定性(Deterministic)数字签名，其明文与密文一一对应，它对一特定消息的签名不变化，如RSA、Rabin等签名；
  - 随机化的(Randomized)或概率式数字签名

数字签名常见算法

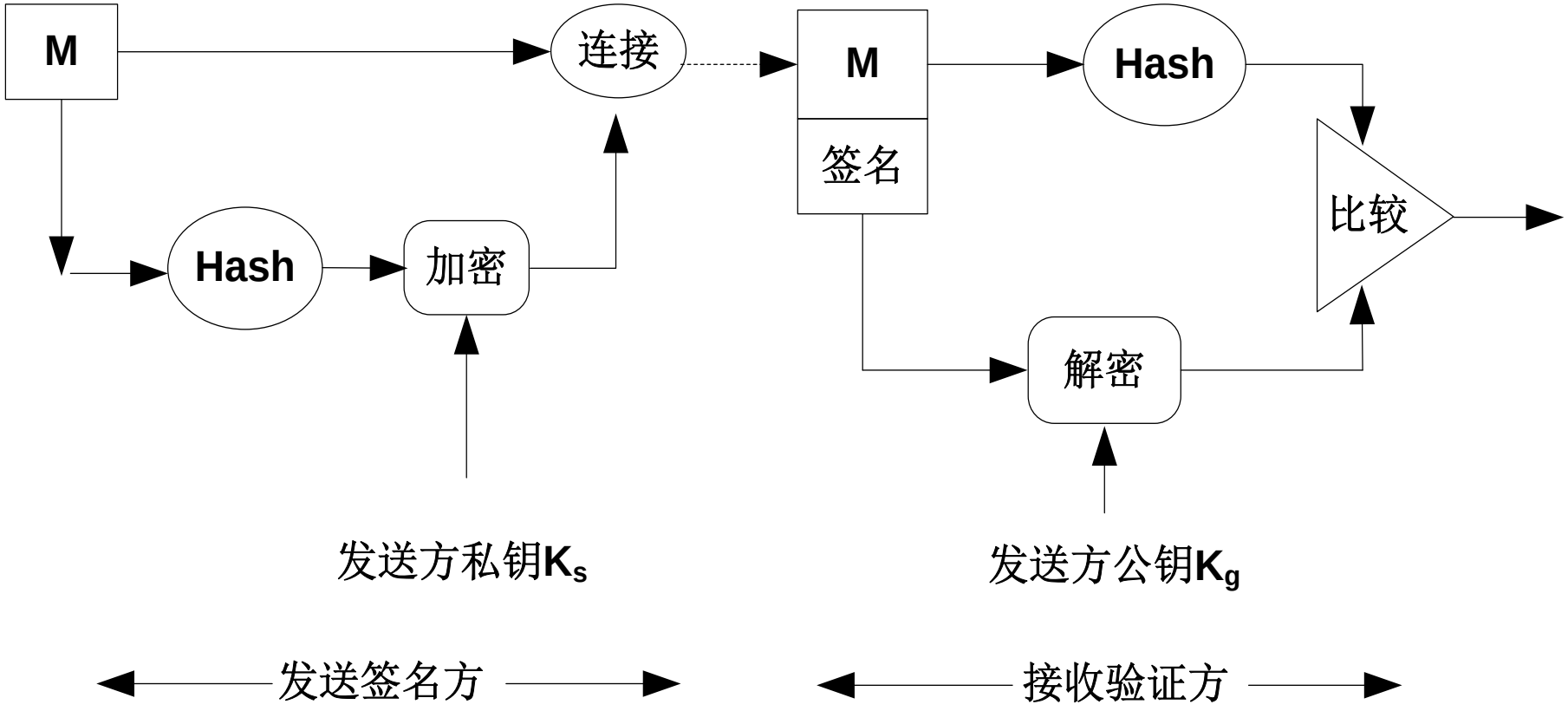
- 普通数字签名算法
  - RSA
  - ElGamal /DSS/DSA
  - ECDSA
- 盲签名算法
- 群签名算法

RSA

- RSA是1977年由麻省理工学院的罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）一起提出的。



RSA算法的签名和验证过程



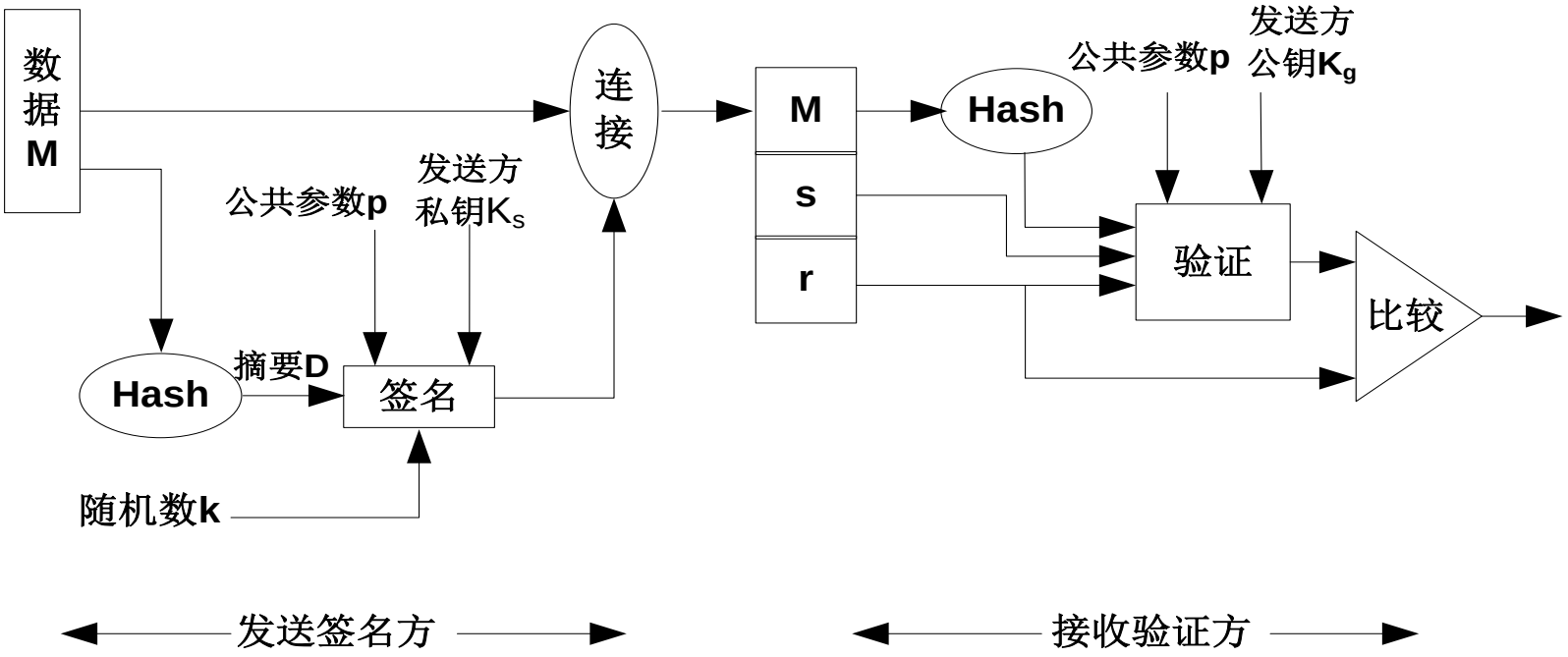
RSA签名方案

- 密钥的生成（同加密系统）
  - 公钥Pk={e, n}；私钥Sk={d, n}
- 签名过程 (用私钥d, n)
  - 明文：M<n　　密文：$S=M^d \bmod n$
- 验证过程 (用公钥e, n)
  - 给定M，S，Ver (M, S)为真，当且仅当$M=S^e \bmod n$
- 安全性
  - 由于只有签名者知道*d*，其他人不能伪造签名，但可易于证实所给任意（*M*，*S*）对是否是消息*M*和相应签名构成的合法对

数字签名标准DSS

- 美国国家标准与技术研究所NIST发布的联邦信息处理标准FIPS186，称为数字签名标准DSS (Digital Signature Standard)
- 于1991年最初提出，1994年12月1日采纳为标准DSS，2000年发布DSS的扩充版，即FIPS 186-2
- DSS为EIGamal和Schnorr签名方案改进，其使用算法记为DSA（Digital Signature Algorithm)。此算法由D. W. Kravitz设计。DSS使用了SHA，安全性是基于求离散对数的困难性
- DSS和DSA是有所不同的：前者是一个标准，后者是标准中使用的算法
- 只能用于数字签名，不能用于加密或密钥分配
- 安全性基于基于离散对数难题
- DSS是一个产生签名比验证签名快得多的方案，验证签名太慢!

DSS的签名和验证过程



- DSS采用的是SHA散列函数计算消息摘要D。签名方在签名时，将消息摘要D和一个随机数k一起作为签名函数的输入。签名函数使用发送方的私钥Ks和一组公共参数P，产生两个输出（s，r）作为签名结果。验证方就是通过比较接收到的签名结果与自己计算出来的签名结果，根据它们是否相等来判断签名的有效性。

数字证书

- 数字证书的作用
- 数字证书的定义
- 数字证书的内容
- 认证中心

数字证书的作用

- 任何的密码体制都不是坚不可摧的，公开密钥体制也不例外。由于公开密钥体制的公钥是对所有人公开的，从而免去了密钥的传递，简化了密钥的管理。
- 但是这个公开性在给人们带来便利的同时，也给攻击者冒充身份篡改公钥有可乘之机。所以，密钥也需要认证，在拿到某人的公钥时，需要先辨别一下它的真伪。这时就需要一个认证机构，将身份证书作为密钥管理的载体，并配套建立各种密钥管理设施。

数字证书的定义

- 数字证书（Digital Certificate）又称为数字标识（Digital ID）。它提供一种在Internet上验证身份的方式，是用来标志和证明网络通信双方身份的数字信息文件。

- 数字安全证书是由权威公正的第三方机构即CA中心签发的。它是在证书申请被认证中心批准后，通过登记服务机构将其发放给申请者。

数字证书的内容

- 最简单的证书包含一个公开密钥、名称以及证书授权中心的数字签名。一般情况下证书中还包括密钥的有效时间，发证机关(证书授权中心)的名称，该证书的序列号等信息，证书的格式遵循ITU-T X.509国际标准。

- 一个标准的X.509数字安全证书包含以下一些内容：
  （1）证书的版本号。不同的版本的证书格式也不同，在读取证书时首先需要检查版本号。
  （2）证书的序列号。每个证书都有一个唯一的证书序列号。
  （3）证书所使用的签名算法标识符。签名算法标识符表明数字签名所采用的算法以及使用的参数。
  （4）证书的发行机构名称。创建并签署证书的CA的名称，命名规则一般采用X.500格式。
  （5）证书的有效期。证书的有效期由证书有效起始时间和终止时间来定义。
  （6）证书所有人的名称。命名规则一般采用X.500格式；
  （7）证书所有人的公开密钥及相关参数。相关参数包括加密算法的标识符及参数等
  （8）证书发行机构ID。这是版本2中增加的可选字段。
  （9）证书所有人ID。这是版本2中增加的可选字段。
  （10）扩展域。这是版本3中增加的字段，它是一个包含若干扩展字段的集合。
  （11）证书发行机构对证书的签名，即CA对证书内除本签名字段以外的所有字段的数字签名。

认证中心

- CA（Certificate Authority，认证中心）作为权威的、可信赖的、公正的第三方机构，专门负责发放并管理所有参与网上交易的实体所需的数字证书。

- CA作为一个权威机构，对密钥进行有效地管理，颁发证书证明密钥的有效性，并将公开密钥同某一个实体（消费者、商户、银行）联系在一起。

CA的主要职责

(1) 颁发证书：如密钥对的生成、私钥的保护等，并保证证书持有者应有不同的密钥对。

(2) 管理证书：记录所有颁发过的证书，以及所有被吊销的证书。

(3) 用户管理：对于每一个新提交的申请，都要和列表中现存的标识名相比较，如出现重复，就给予拒绝。

(4) 吊销证书：在证书有效期内使其无效，并发表CRL（Certificate Revocation List，被吊销的证书列表）

(5) 验证申请者身份：对每一个申请者进行必要的身份认证。

(6) 保护证书服务器：证书服务器必须安全的，CA应采取相应措施保证其安全性。

(7) 保护CA私钥和用户私钥：CA签发证书所用的私钥要受到严格的保护，不能被毁坏，也不能被非法使用。同时，根据用户密钥对的产生方式，CA在某些情况下有保护用户私钥的责任。

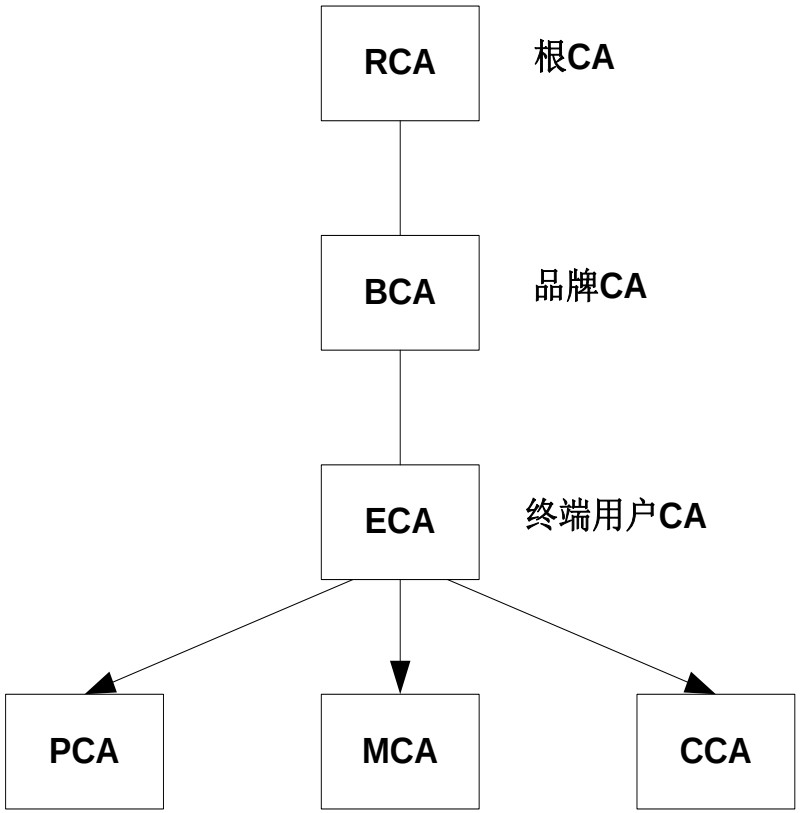
(8) 审计和日志检查：为了安全起见，CA对一些重要的操作应记入系统日志。在CA发生事故后，要根据系统日志做善后追踪处理——审计，CA管理员要定期检查日志文件，尽早发现可能的隐患。

CA的基本组成

- 认证中心主要有三个部分组成
  - 注册服务器（RS）：面向用户，包括计算机系统和功能接口；
  - 注册中心（RA）：负责证书的审批；
  - 认证中心（CA）：负责证书的颁发，是被信任的部门
- 一个完整的安全解决方案除了有认证中心外，一般还包括以下几个方面：
  - 密码体制的选择
  - 安全协议的选择
    - SSL（Secure Socket Layer 安全套接字层）
    - S-HTTP（Secure HTTP，安全的http协议）
    - SET（Secure Electonic Transaction，安全电子交易协议）

CA的三层体系结构

- 第一层为RCA（Root Certificate Authority，根认证中心）。它的职责是负责制定和审批CA的总政策，签发并管理第二层CA的证书，与其它根CA进行交叉认证。
- 第二层为BCA（Brand Certificate Authority，品牌认证中心）。它的职责是根据RCA的规定，制定具体政策、管理制度及运行规范；签发第三层证书并进行证书管理。
- 第三层为ECA（End user CA，终端用户CA）。它为参与电子商务的各实体颁发证书。签发的证书可分为三类：分别是支付网关（Payment Gateway）、持卡人（Cardholder）和商家（Merchant）签发的证书；签发这三种证书的CA对应的可称之为PCA、CCA和MCA。



本讲结束！