



# 本章内容

## 一、消息认证概述

- 12.1 对消息认证的要求
- 12.2 消息认证函数
- 12.3 对消息认证码的要求
- 12.4 MAC的安全性

## 二、两类MAC算法

- 12.5 基于Hash函数的MAC: HMAC
- 12.6 基于分组密码的MAC: DAA 和 CMAC

## 三、其它应用

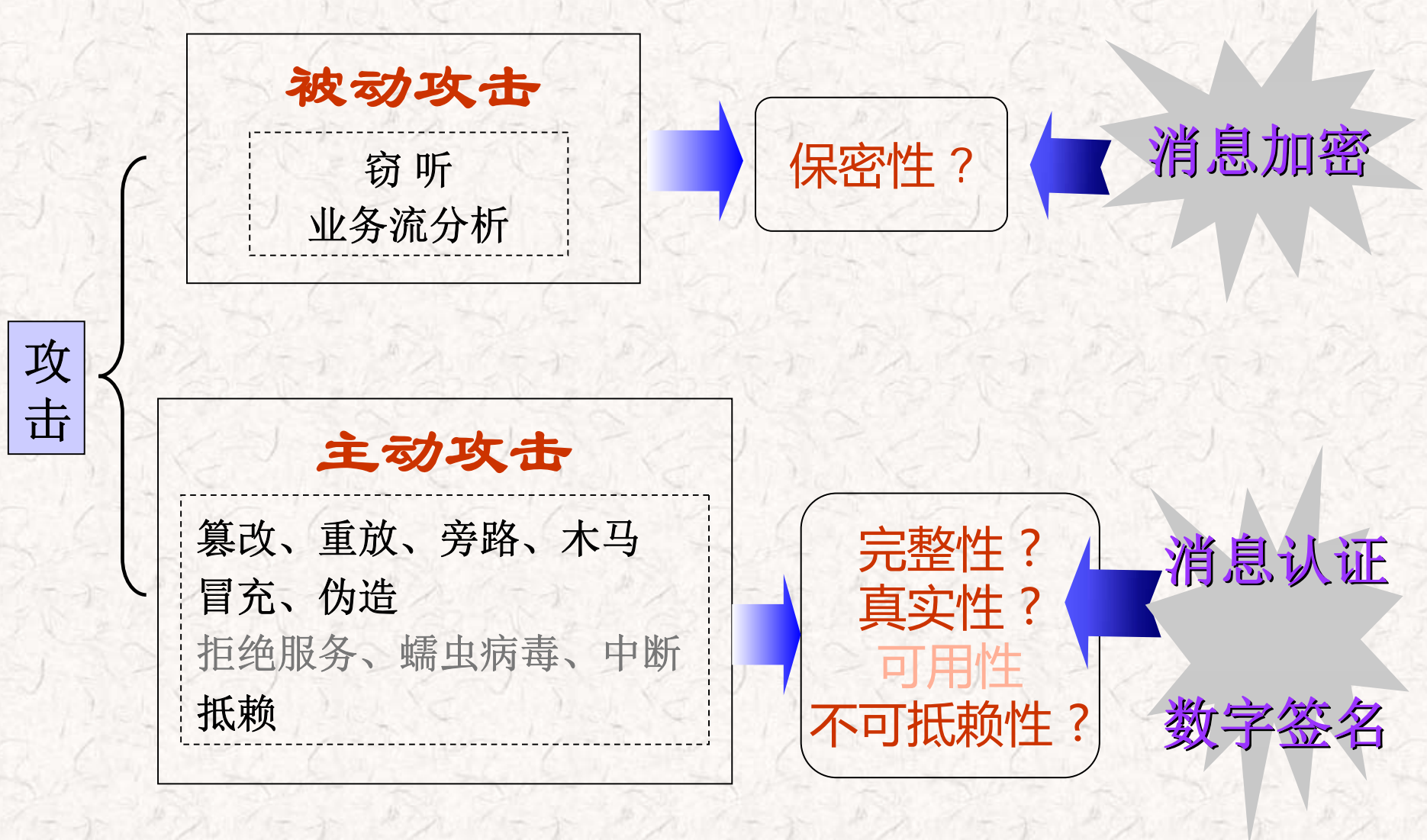
- 12.7 认证加密: CCM 和 GCM
- 12.8 使用Hash函数和MAC产生伪随机数



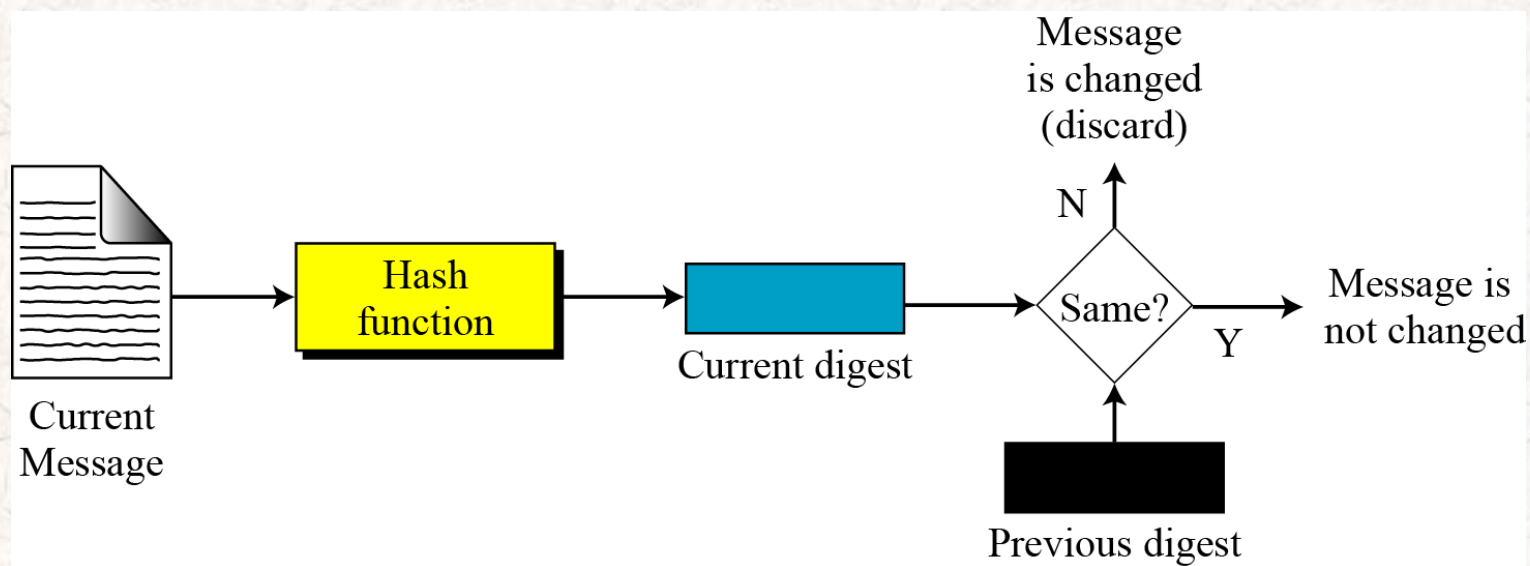
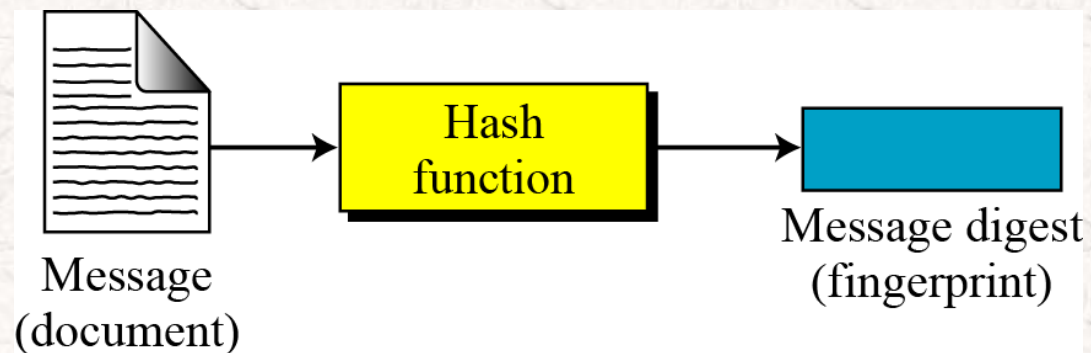
## 参考阅读

- (1) 杨波, 《现代密码学 (第2版) 》, 第6章
- (2) William Stallings, [Cryptography and Network Security : Principle and Practice \(ed. 5\)](#), 第11、12章
- (3) 陈鲁生、沈昌镒, 《现代密码学》, 第8章
- (4) 张焕国, 《密码学》课件, 第13、14讲

# 回顾： 攻击 vs. 安全要求



# 回顾： 第11章 密码学Hash函数



密码学Hash函数可用于实现消息认证和数字签名



## 12.1 对消息认证的要求



# 对消息认证的要求

- 在网络通信环境中，可能有下述攻击：

① 泄密

② 传输分析

加解密

③ 伪装

④ 内容修改

⑤ 顺序修改

⑥ 计时修改

消息认证

⑦ 发送方否认

⑧ 接收方否认

数字签名



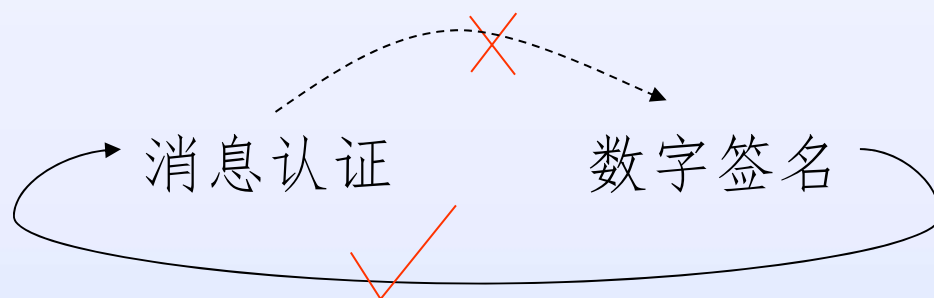
# 消息认证、数字签名

- 消息认证:

- 验证所收到的消息确实是真实的，且是未被修改的消息，它也可验证消息的顺序和及时性。

- 数字签名:

- 是一种**认证技术**，其中的一些方法可用来抵抗发送方否认攻击。（接收者可验证但不能伪造）





## 12.2 消息认证函数





# 消息认证函数

- 任何消息认证或数字签名机制在功能上分两层：
  - 下层：函数 → 产生一个用来认证消息的值（认证符）
  - 上层：协议 → 将该函数作为原语使接收方可以验证消息的真实性

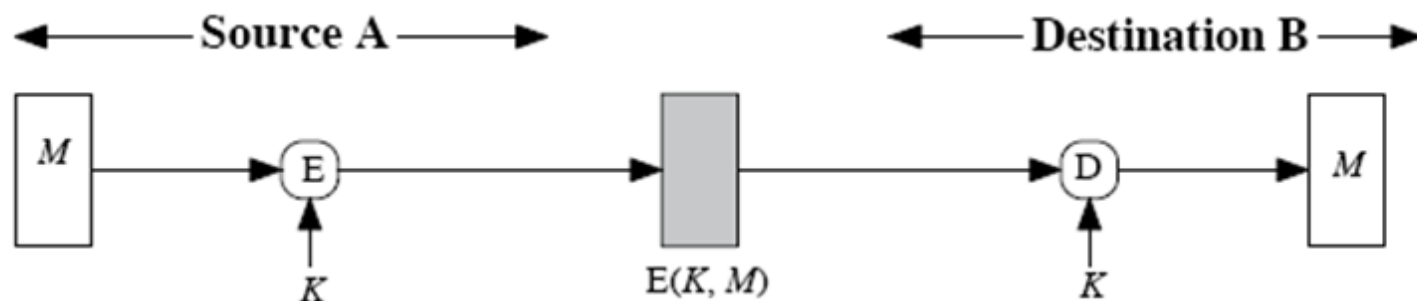


# 消息认证函数的类型

- 可用来产生认证符的函数类型，大致分为三类：
  - 消息加密：消息的密文作为认证符
  - Hash函数：将任意长消息映射为定长的hash值作为认证符
  - 消息认证码（MAC）：消息和密钥的公开函数，产生定长的值作为认证符。 *MAC: message authentication code*



## 消息的对称加密认证



- 对称密码，既可以提供认证又可以提供保密性，*但不是绝对的。*
  - 加密方：设想发送者A用只有他与接收者B知道的**密钥K**加密信息发送给接收者。
  - 解密方：因为A是除B以外惟一拥有密钥K的一方，而敌手不知道如何改变密文来产生明文中所期望的变化。因此，B知道解密得到的明文是否被改变。



## 消息的对称加密认证：考虑解密方

- 现在考虑如下情况：

给定解密函数 $D$ 和密钥 $K$ ，接收者将接受任何输入 $X$ ，并产生输出 $Y=D_K[X]$ ，

如果 $X$ 是合法信息 $M$ 的密文，则 $Y$ 是明文信息 $M$ ，否则 $Y$ 将是毫无意义的二进制比特序列。

- 接收方需要某些自动化措施，以确定 $Y$ 是否是合法的明文



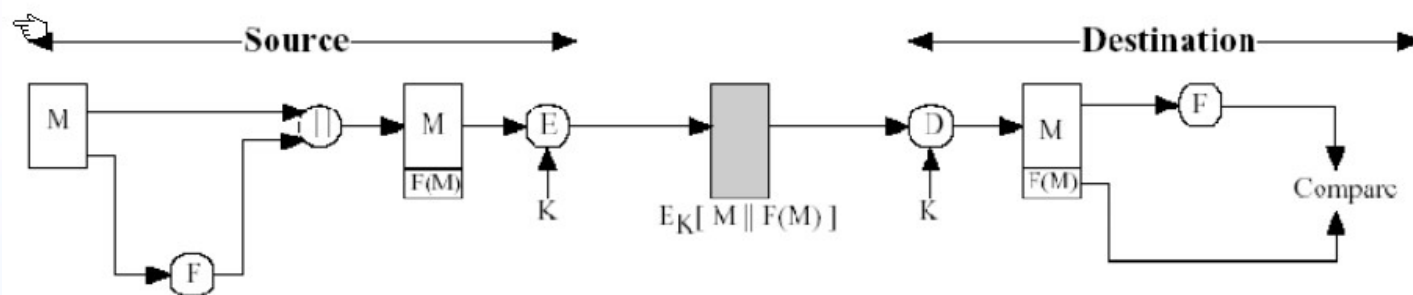
## 消息的对称加密认证：考虑解密方

- 当明文信息可以是任意比特的组合时：

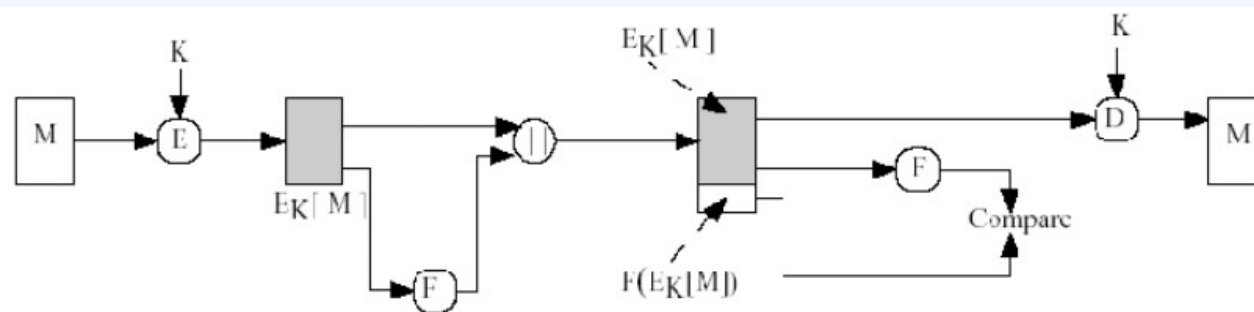
为了实现用**自动的**方法来确定解密得到的是否是合法信息的密文，可以采用的一种方法是**强制明文有某种结构**，这种结构易于识别但不能复制并且不影响加密。

- 例如：可以在加密以前对消息附加纠错码(校验和, checksum)，如下页所示。

# 加密过程中的内部和外部错误控制



(a) Internal error control

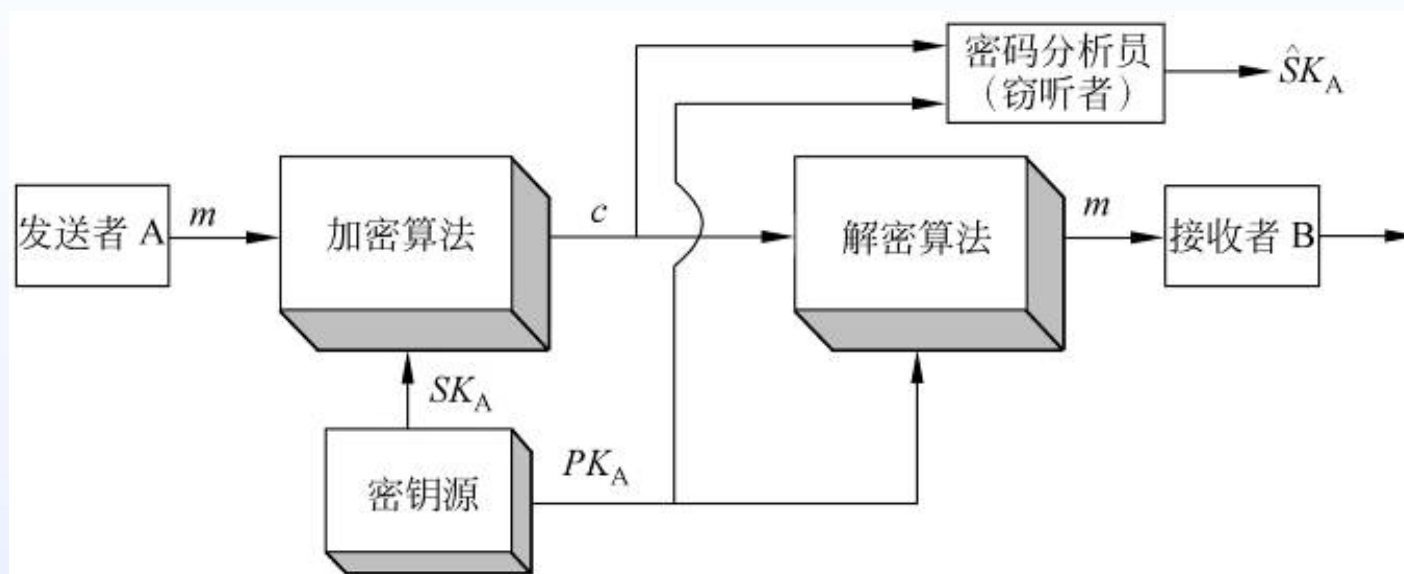


(b) External error control



## 消息的公钥加密认证

- 为了提供认证，发送者A用私钥对信息的明文进行加密，任意接收者都可以用A的公钥解密。



- 要求在明文中有某种内部结构，使得接收者能够识别正常的明文和随机的比特串。



## 消息的公钥加密认证

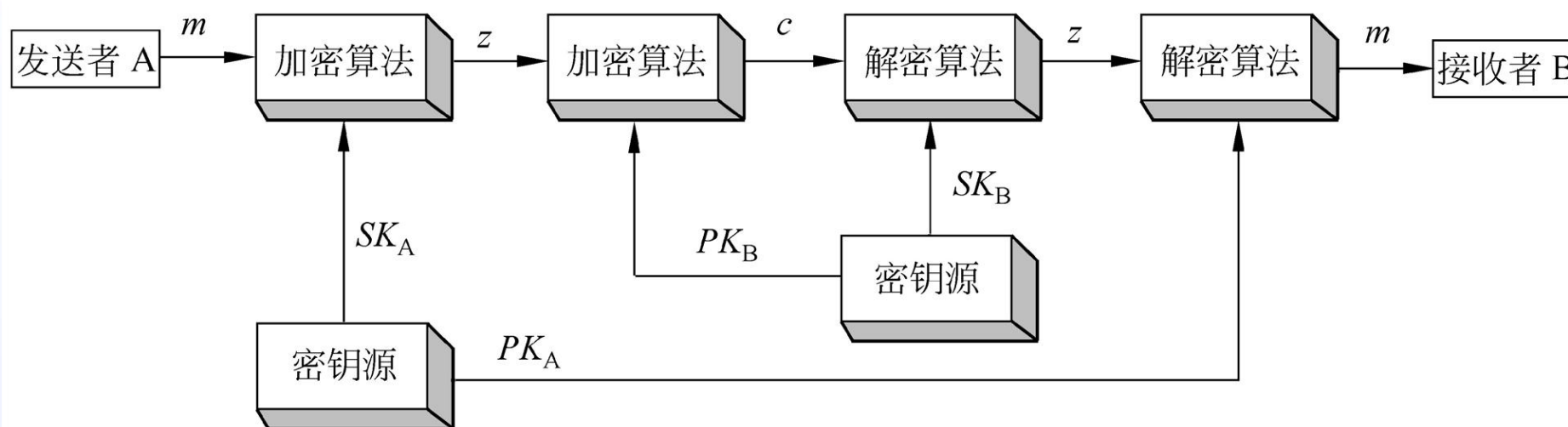
- 采用这样的结构既可提供了认证，也可提供数字签名：
  - 因为只有A能够产生该密文，其它任何一方都不能产生该密文。
  - 从效果上看，A已经用私钥对信息的明文进行了签名。
- 应当注意，**只用私钥加密不能提供保密性**，因为，任何人只要有A的公开密钥就能够对该密文进行解密。





# 消息的公钥加密认证

- 兼顾保密和认证：





# 消息认证码

(MAC: message authentication code)

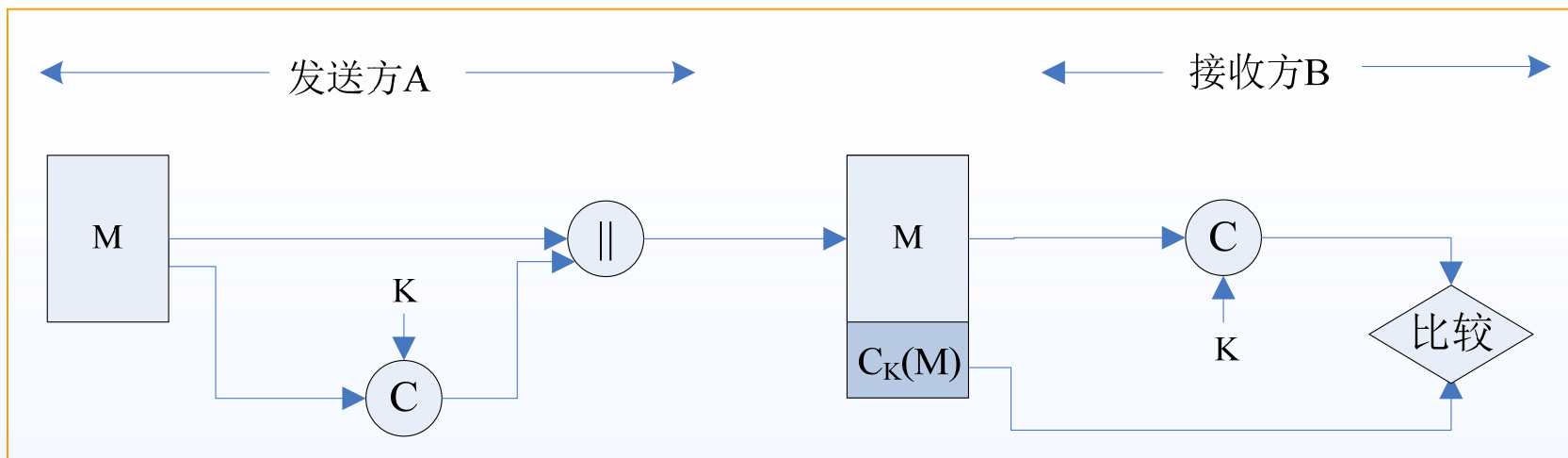
- 消息认证码，是一种认证技术，它利用密钥来生成一个**固定长度**的短数据块，并将该数据块附加在消息之后。
- MAC的值依赖于消息和密钥：

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

- MAC函数于加密类似，但MAC算法**不要求**可逆性，而加密算法必须是可逆的。
- 接收方进行同样的MAC码计算，并验证是否与发送过来的MAC码相同



# MAC的使用方式: 实现消息认证



- 通信双方共享秘密钥 $K$ 。
- A计算MAC并将消息 $M$ 和认证码MAC发送给接收方B:  
 $A \rightarrow B: M \parallel MAC$
- 接收方收到消息 $M$ 后用相同的秘密钥 $K$ 重新计算得出新的MAC, 并将其与接收到的MAC进行比较, 若二者相等, 则认为消息正确且真实。 (问: why? )



## MAC的使用方式： 实现消息认证

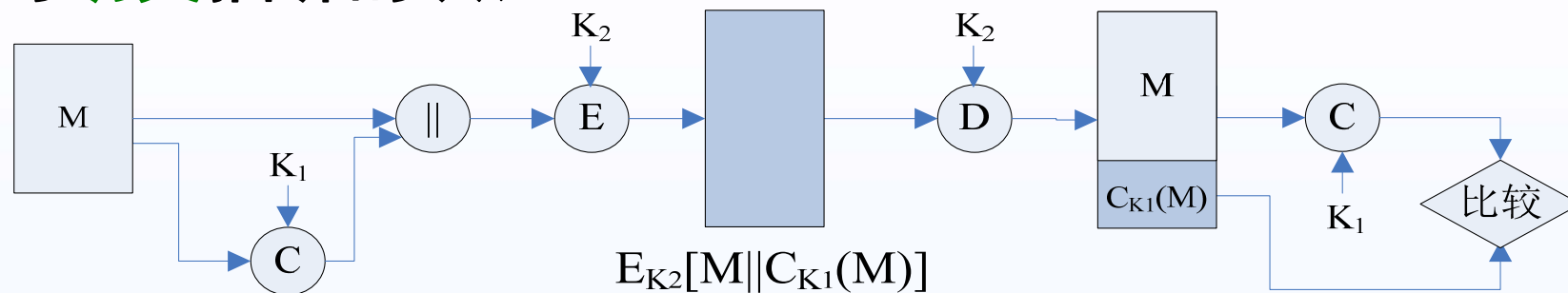
- 如果收到的MAC与计算得出的MAC相同，则接收者可以认为：
  - (1) 消息未被更改过  
因为任意更改消息而没有更改MAC的行为，都将导致收到的MAC与用更改过的消息计算出的MAC不相同；而且，由于使用的密钥只有收发双方知道，其它方无法更改MAC与更改后的消息对应。
  - (2) 消息来自与他共享密钥的发送者  
由于使用的密钥只有收发双方知道，其它方无法产生对应的MAC。
  - (3) 若消息中含有序列号，那么接收方可以相信消息顺序是正确的，因为攻击者无法成功地修改序列号。



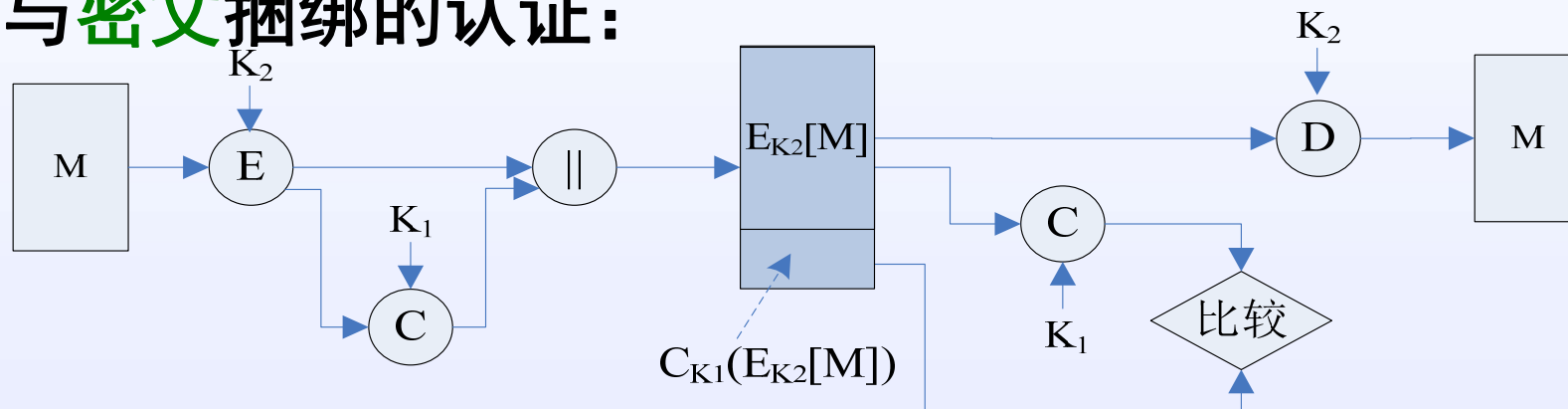
# MAC的使用方式： 实现认证与保密

- 与明文捆绑的认证：

这种方式更常用



- 与密文捆绑的认证：



秘密钥  $K_1$  可提供 认证

秘密钥  $K_2$  可提供 保密



# MAC的特性

- MAC码是一个**密码校验和**(cryptographic checksum )：
$$\text{MAC} = C_K(M)$$
  - 消息M的长度是可变的
  - 密钥K是要保密的，且收发双方共享。
  - 产生固定长度的认证码
- MAC算法是一个**多对一**函数
  - 多个消息对应同一MAC值
  - 但是找到同一MAC值所对应的多个消息是**困难**的。
  - 可证明：与加密相比，认证函数更不易被攻破



## 哪些情况下需要使用MAC?

- 同一消息广播给多人，仅需一个接收者负责验证过程，若出错则由他通知他人。
- 通信某一方处理负荷很大，没时间解密收到的所有消息，他可随机选择消息并对其进行认证。
- 对明文形式的计算机程序进行认证，而无需每次都先解密再运行。
- 有些简单网络管理协议可只关心消息认证，如SNMPv3对收到的消息进行认证，而不必加密SNMP传输。
- 认证和保密性分离提供更加灵活层次结构，例如，在应用层希望对消息认证，而在传输层希望提供保密性。
- 认证可提供更持续的保护。就消息加密而言，消息仅在传输中可以使消息不被修改，而不是在接收方系统中保护消息不被修改。



## 12.3 对消息认证码的要求





## 对MAC的要求

$$MAC = C_k(M)$$

### 从理论上说:

- 对不同的M应产生不同的MAC。因为若  $M_1 \neq M_2$ ，而  $MAC_1 = MAC_2$ ，则攻击者可将  $M_1$  篡改为  $M_2$ ，而接收方不能发现。
- 但是，要使函数C具备上述性质，将要求消息认证码MAC至少和消息M一样长 (why?)，这是不方便的。



## 对MAC的要求

**实际应用时，要求函数C具有以下性质：**

- (1) 对已知 $M_1$ 和 $MAC_1$ ，构造满足 $MAC_2=MAC_1$ 的 $M_2$ ，在计算上是不可行的；
- (2) MAC函数应是均匀分布的，即对任何随机的消息 $M_1$ 和 $M_2$ ， $MAC_1=MAC_2$ 的概率是 $2^{-n}$ ，其中 $n$ 是MAC的位数；
- (3) 设 $M_2$ 是 $M_1$ 的某个已知的变换，即 $M_2=f(M_1)$ ，如  $f$  改变  $M_1$  的一位或多位，那么 $MAC_1=MAC_2$ 的概率为 $2^{-n}$ 。



## 12.4 MAC的安全性



# MAC的安全性

- 同加密算法和Hash函数一样，对MAC的攻击可分为两类：**穷举攻击** 和 **密码分析**。
- MAC算法应具有的安全性质：
  - **抗计算性：**  
给定一个或多个<消息,MAC>对  $\{<x_i, \text{MAC}(K, x_i)>\}$ ,  
对任何新的输入  $x \neq x_i$ ,  
计算<消息,MAC>对  $<x, \text{MAC}(K, x)>$ 在计算上是不可行的。



# 对MAC的穷举攻击

对于给定消息 $x$ ，可通过攻击密钥空间、MAC值这两种方法来找出其MAC值，所需代价为  $\min(2^k, 2^n)$ 。

- 穷举密钥空间：

- 假设密钥长度： $k$ 位，MAC长度： $n$ 位，且  $k > n$
- 给定  $T_1 = C_k(M_1)$ : 穷举所有  $2^k$  个密钥，平均有  $2^{(k-n)}$  个密钥可产生  $T_1$
- 给定  $T_2 = C_k(M_2)$ : 穷举这  $2^{(k-n)}$  个密钥，平均有  $2^{(k-2n)}$  个密钥可产生  $T_2$
- .....
- 平均来讲，若  $k = j \cdot n$ ，则需  $j$  次上述循环可找到密钥

- 穷举MAC值：

- 平均代价  $2^n$
- 需有  $\langle \text{消息}, \text{MAC} \rangle$  对或密钥信息，所以这种攻击不能离线进行。



## 对MAC的密码分析

- 利用MAC算法的某些性质
- 理想的MAC算法要求：  
密码分析攻击所需的代价  $\geq$  穷举攻击所需的代价
- 与Hash函数相比，MAC的结构种类更多，对MAC的密码分析攻击的研究较少。



## 12.5 基于Hash函数的MAC: HMAC



# HMAC算法概述

- 构造MAC的传统的方法： 基于分组密码
- 新方法： 基于密码Hash函数 (RFC2104)  
(Keyed-Hashing for Message Authentication)
  - 密码Hash函数（如MD5、SHA）的软件实现快于分组密码（如DES）的软件实现
  - 密码Hash函数的库代码来源广泛
  - 密码Hash函数没有出口限制，而分组密码即使用于MAC也有出口限制。





# HMAC 设计目标

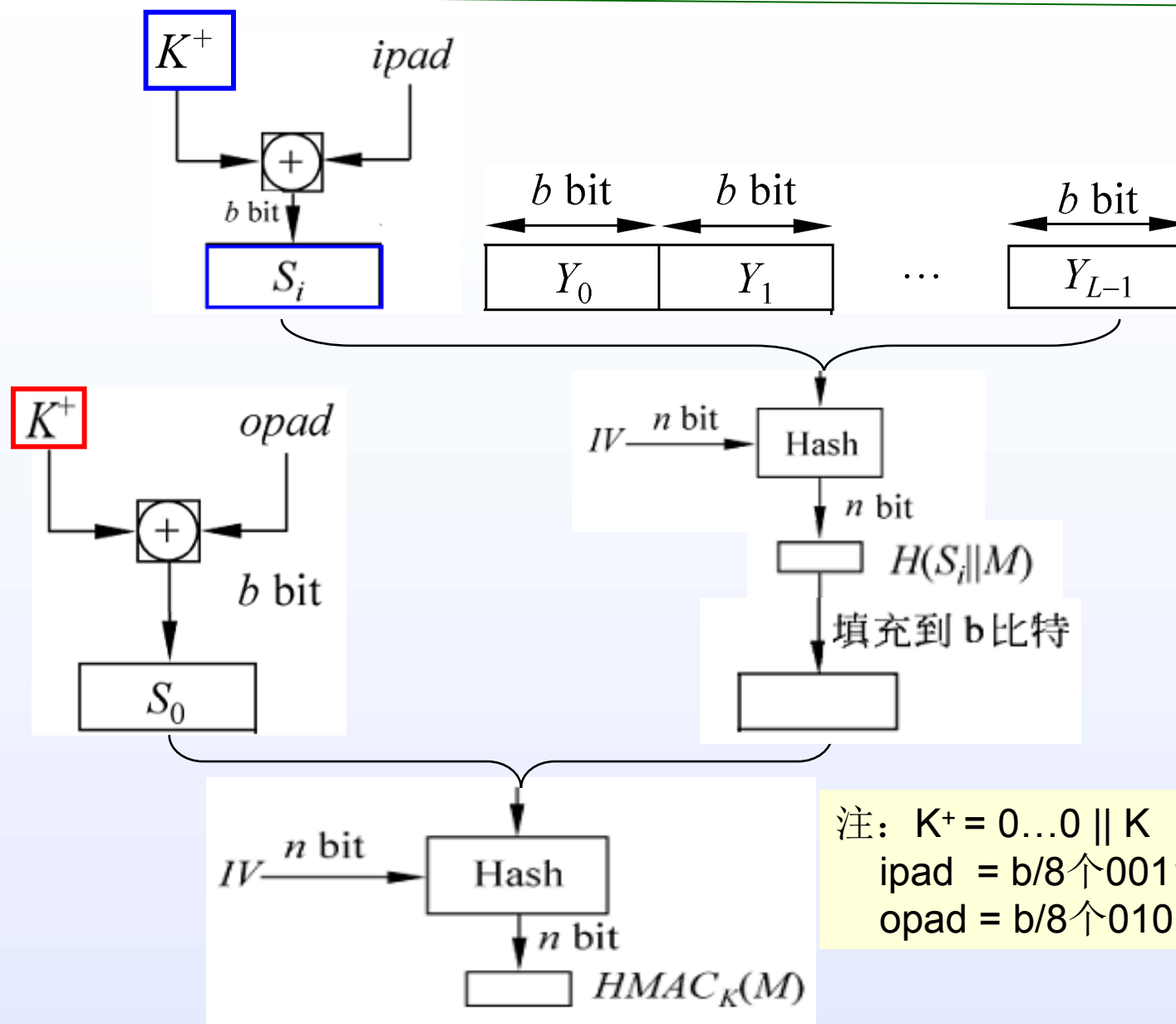
- RFC 2104 给出了HMAC的设计目标：
  - a)可以无需修改地使用现有的Hash函数。
  - b)当出现新的更快或更安全的Hash函数时，要能轻易地替换。
  - c)保持Hash函数的原有性能，不因用于HMAC而使其性能降低。
  - d)使用和处理密钥的方式简单。
  - e)对认证机制的密码安全强度容易分析，与Hash函数有同等的  
安全性。

(a)、(b)→实用性，(e)→安全性



$$HMAC_K(M) = H[ \underbrace{(K^+ \oplus opad)}_{b \text{ bit}} \parallel \underbrace{H[(K^+ \oplus ipad) \parallel M]}_{n \text{ bit}} ]$$

## HMAC 算法 框图





# HMAC算法描述

$$HMAC_K(M) = H[(K^+ \oplus opad) \parallel H[(K^+ \oplus ipad) \parallel M]]$$

- ① K的左边填充0以产生一个b比特长的K+
  - 例如K的长为160比特，b=512，则需填充44个零字节0x00。
- ② K+与ipad 逐比特异或以产生b比特的分组S<sub>i</sub>。
- ③ 将M链接到S<sub>i</sub>后。
- ④ 将H作用于步骤③产生的数据流。
- ⑤ K+与opad逐比特异或，以产生b比特长的分组S<sub>0</sub>。
- ⑥ 将步骤④得到的杂凑值链接在S<sub>0</sub>后。
- ⑦ 将H作用于步骤⑥产生的数据流并输出最终结果。



## 12.6 基于分组密码的MAC: DAA 和 CMAC

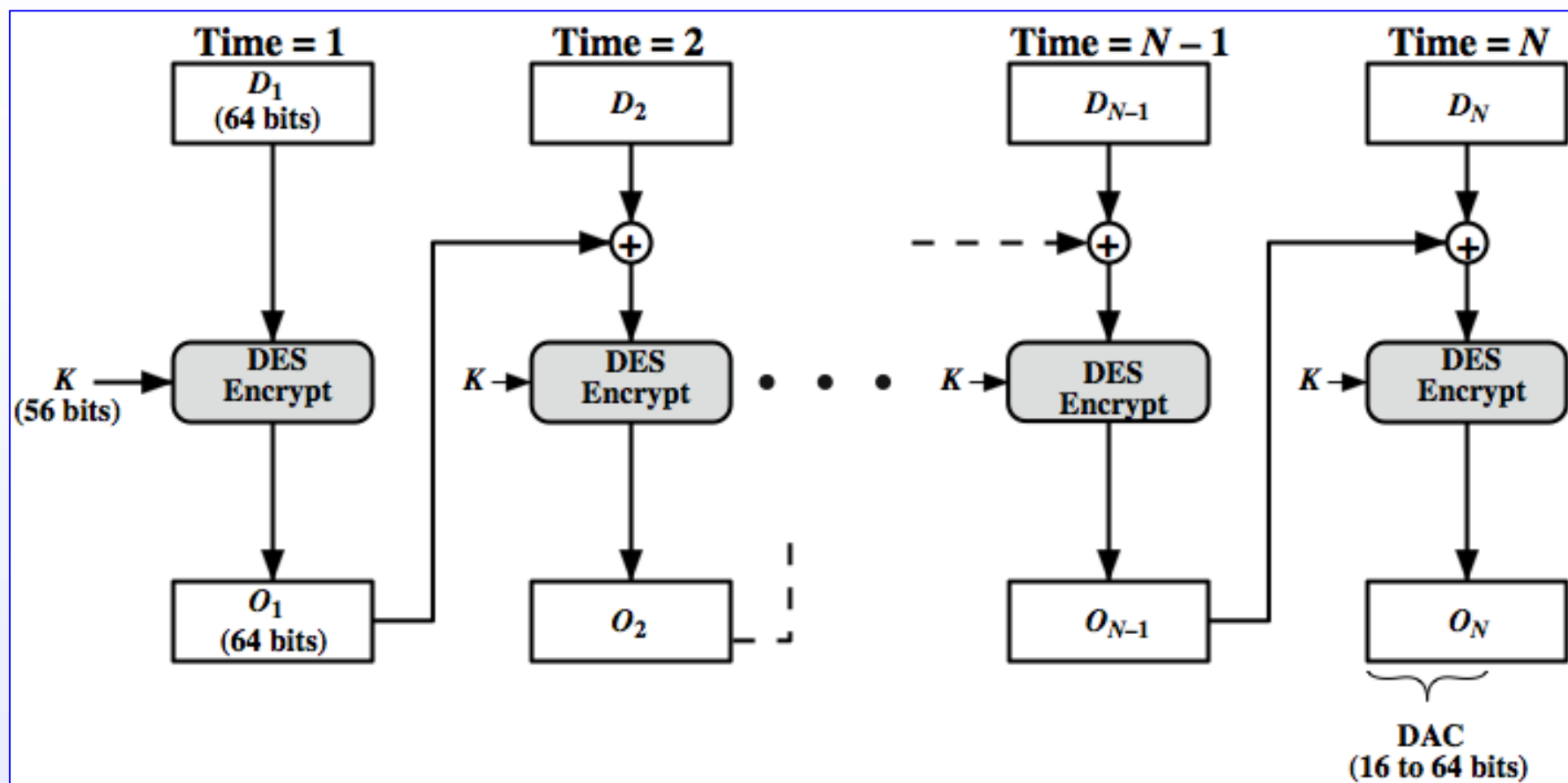


## 数据认证算法（DAA）

- 数据认证算法（DAA: Data Authentication Algorithm）是最广泛的MAC算法之一。
- FIPS标准（FIPS PUB 113）和ANSI标准（X9.17）
- DAA采用DES运算的密文块链接（CBC）方式，最后一个密文分组作为MAC。



# DAA





## DAA的安全性

- 安全性问题：  
若  $T = \text{MAC}(K, X)$ ，  
则可知  $X \parallel (X \oplus T)$  的MAC码也为T
- DAC取最后一个分组或其最左边的m位 ( $16 \leq m \leq 64$ )  
→ 不够安全



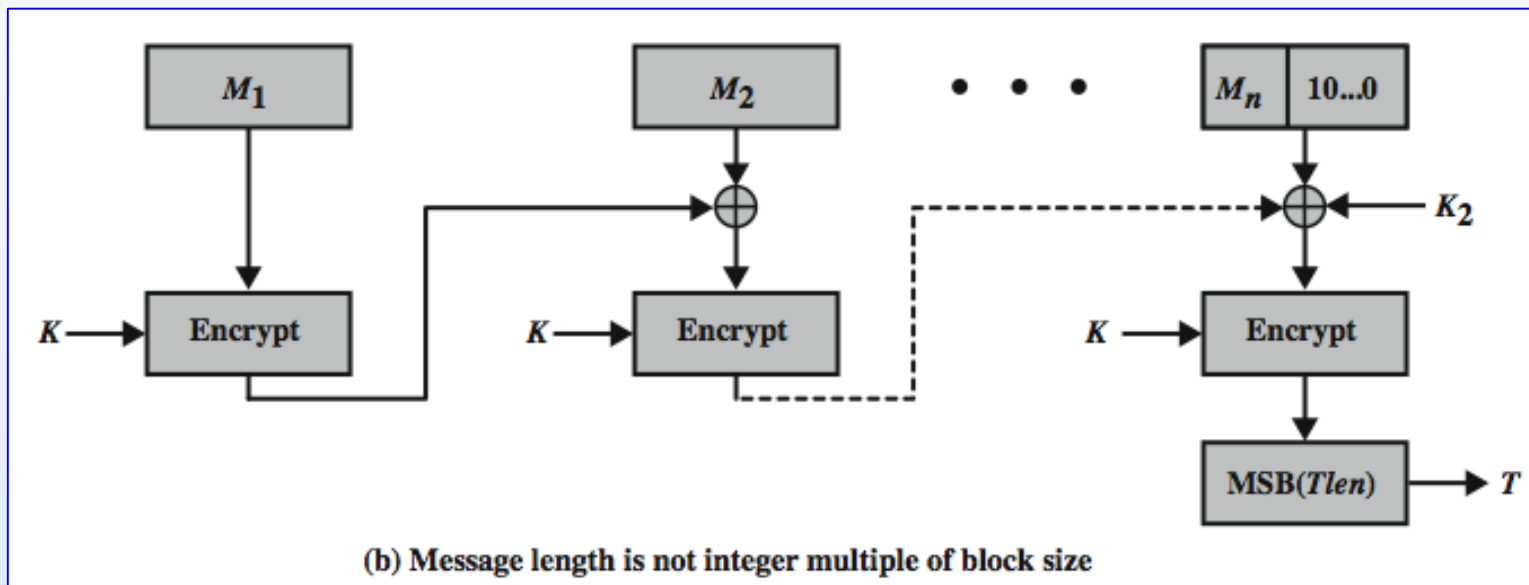
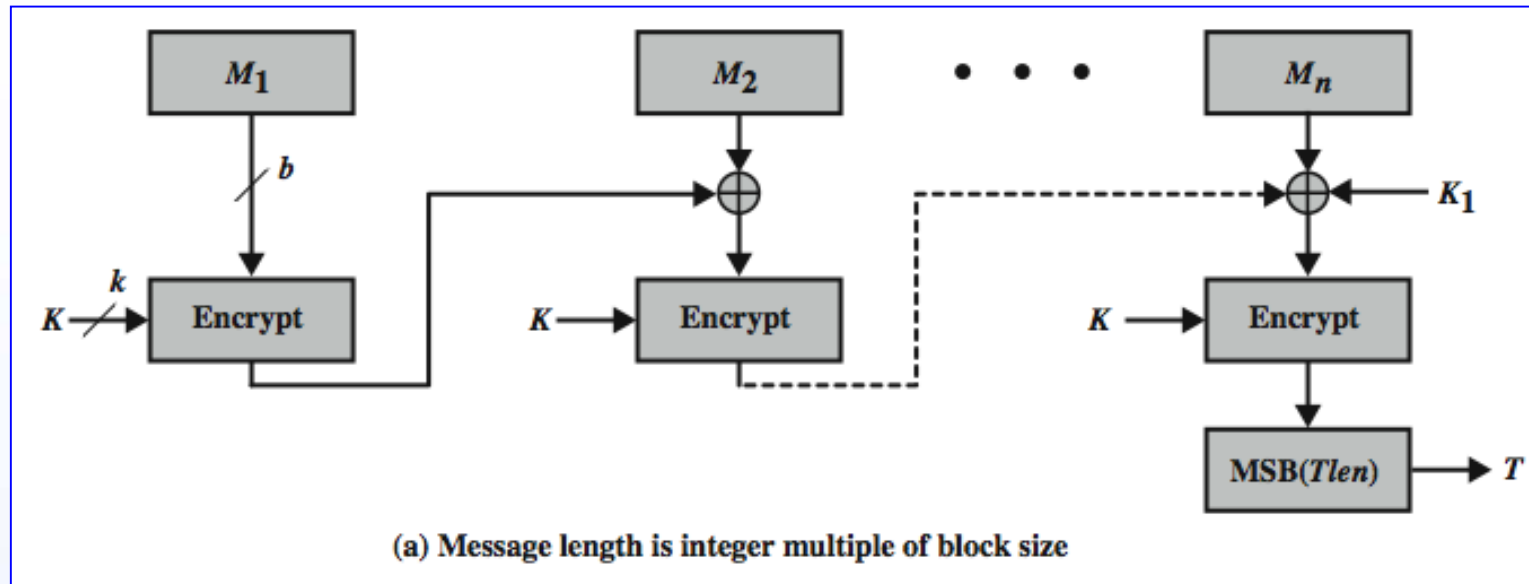
## 基于密码的消息认证码（CMAC）

- DAA的不足可通过使用两个密钥来克服，一个密钥长为 $k$ ，用来密文分组链接的每一步，另一个长度为 $n$ ，用在最后一个分组。此即CMAC。
- CMAC: Cipher-based Message Authentication Code
- 被采用为 NIST SP800-38B





# CMAC





## 12.7 认证加密：CCM 和 GCM



## 认证加密 (Authenticated Encryption)

- 认证加密 (AE: Authenticated Encryption) 是指在通信中同时提供保密性和认证 (完整性) 的加密系统。
- 许多应用和协议中都同时需要这两种形式的安全性保证，但这两类安全系统一直分离设计，直到近几年才合并在一起考虑。




# AE的四种通用方案

- HtE, Hash-then-encrypt:  $E(K, (M \parallel H(M)))$ 
  - WEP协议使用
- MtE, MAC-then-encrypt:  $E(K_2, (M \parallel \text{MAC}(K_1, M)))$ 
  - SSL/TLS协议使用
- EtM, Encrypt-then-MAC:  $(C=E(K_2, M), T=\text{MAC}(K_1, C))$ 
  - IPsec协议使用
- E&M, Encrypt-and-MAC:  $(C=E(K_2, M), T=\text{MAC}(K_1, M))$ 
  - SSH协议使用
- HtE, MtE, E&M: 先解密后验证; EtM: 先验证后解密
- 这些方案都存在安全缺陷

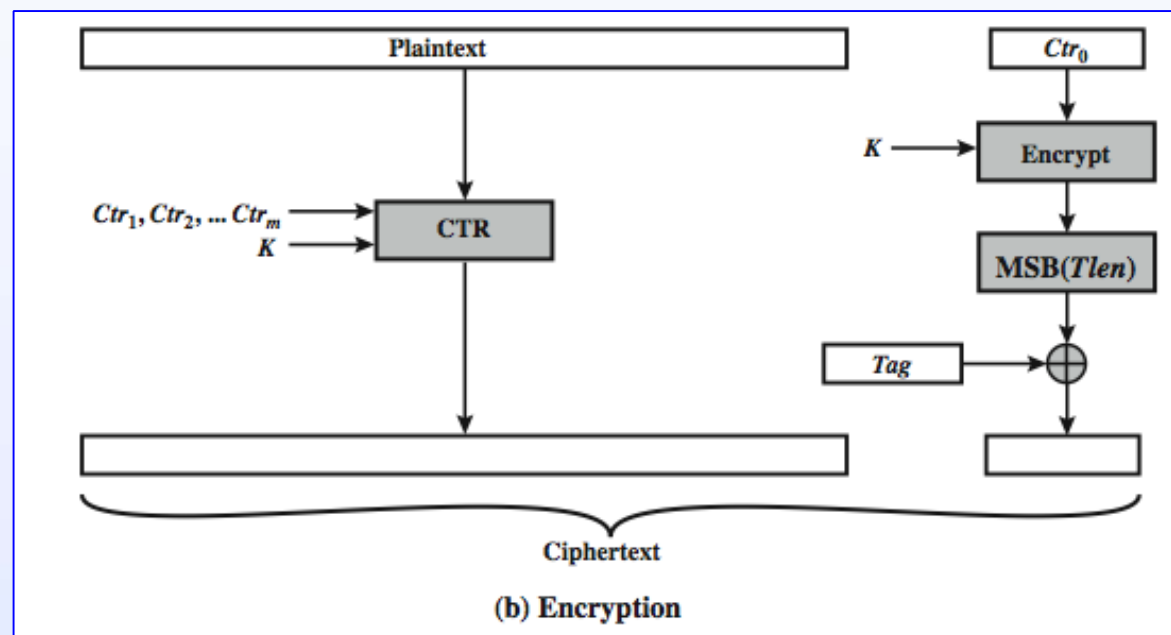
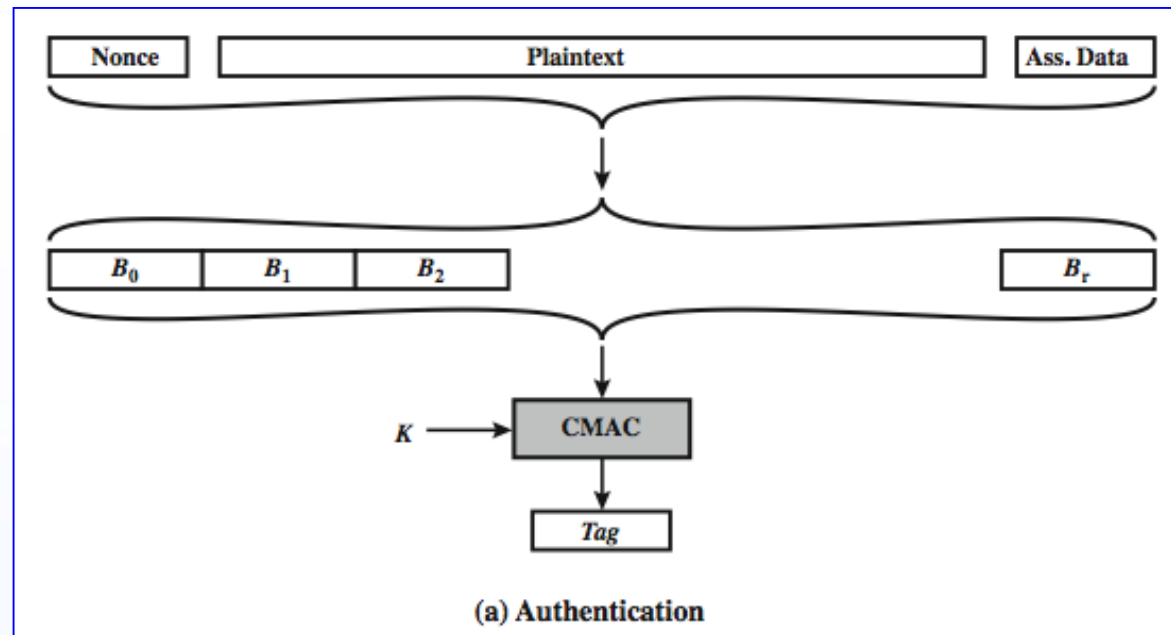


## 分组密码链接 - 消息认证码 (CCM)

- Counter with Cipher Block Chaining - Message Authentication Code (CCM)
- 对 E&M 方案的改进，NIST 标准 SP 800-38C，用于保护 IEEE 802.11 WiFi
- CCM的组成：
  - AES加密算法
  - CTR工作模式
  - CMAC认证算法
- 注：CCM只使用一个密钥



# CCM Operation



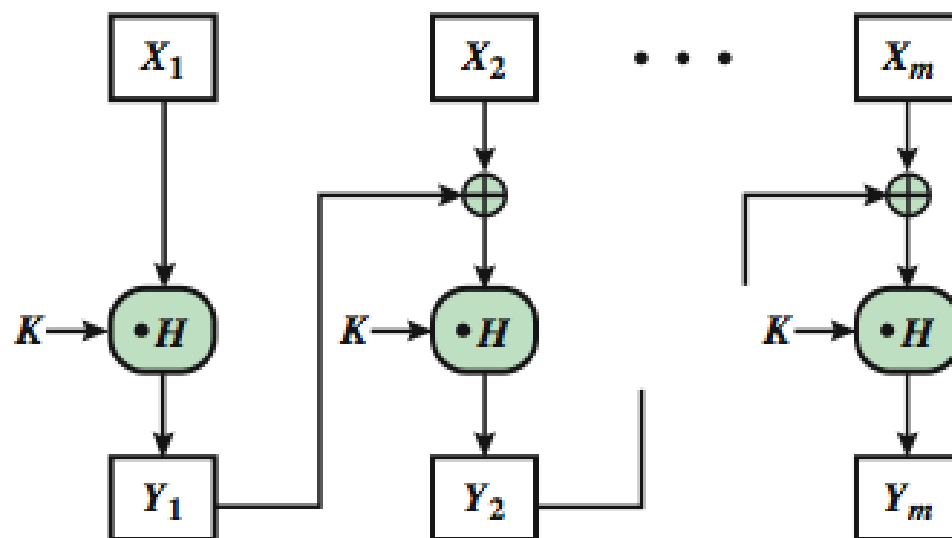


# Galois/计数器模式 (GCM)

- NIST 标准 SP 800-38D, 基于并行化设计
- 本质是：消息在变型的CTR模式下加密，密文结果与密钥以及消息长度信息在GF ( $2^{128}$ ) 上相乘产生认证码
- 该标准同时还制订了仅支持MAC的工作模式，即GMAC
- GCM模式使用两个函数：
  - 带密钥的Hash函数 GHASH
  - 计数器每次增1的CTR模式 GCTR



# GHASH 函数

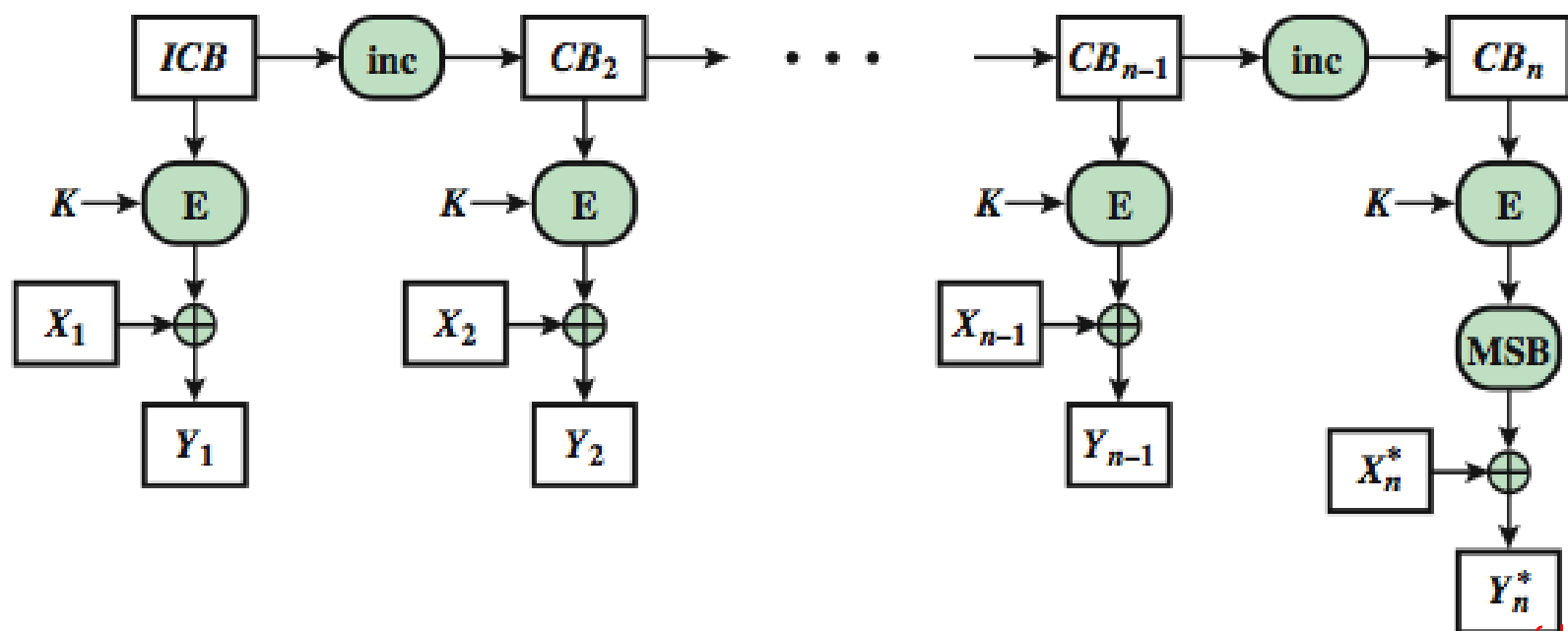


(a)  $\text{GHASH}_H(X_1 \parallel X_2 \parallel \dots \parallel X_m) = Y_m$





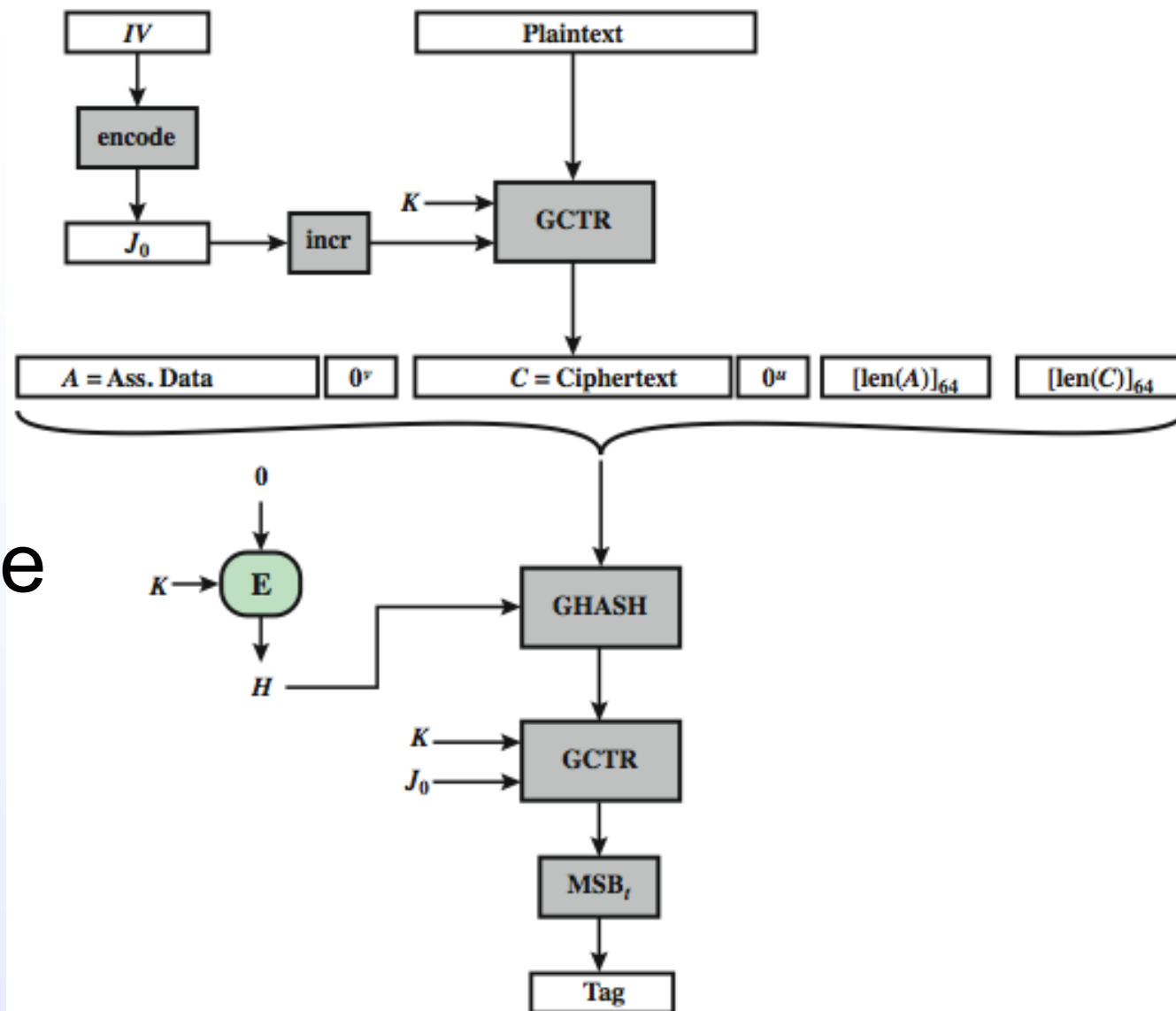
# GCTR 函数



(b)  $\text{GCTR}_K(\text{ICB}, X_1 \parallel X_2 \parallel \dots \parallel X_n^*) = Y_n^*$  y<sub>1</sub> || ... || y<sub>n-1</sub> || y<sub>n</sub>\*



# GCM Mode





## 12.8 使用Hash函数和MAC产生伪随机数



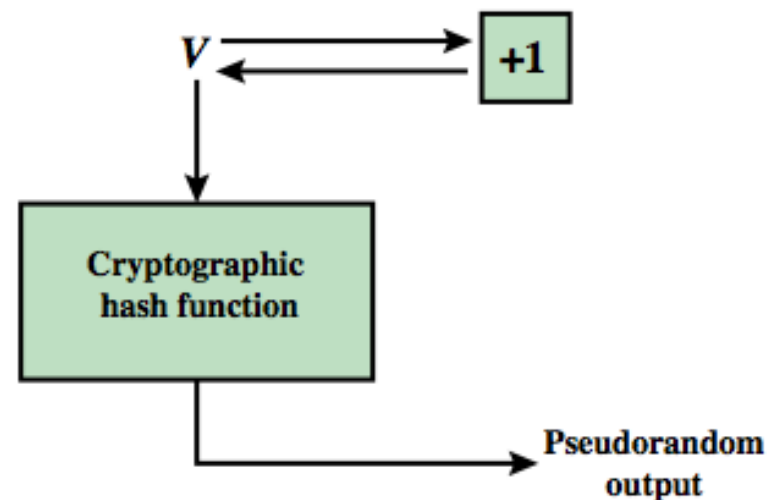
# 使用Hash函数和MAC产生伪随机数

- PRNG的关键组成：
  - 随机种子值
  - 生成伪随机位流的确定性算法
- 构造PRNG的基础可以是：
  - 加密算法（第7、10章）
  - Hash函数（ISO18031 & NIST SP 800-90）
  - MAC (NIST SP 800-90)



# 基于Hash函数的PRNG

- NIST SP800-90 和 ISO18031 中基于Hash的PRNG的基本方案：
  - 种子  $V$
  - 周期性更新 $V$ （比如，加1）
  - 计算 $V$ 的hash值
  - 使用hash值中的 $n$ 位作为随机值
- hash函数的安全性保证了该方案的安全性

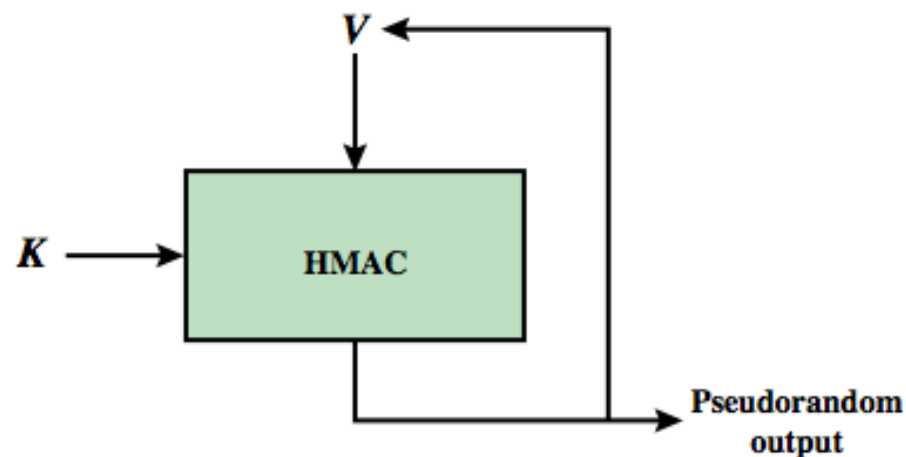


(a) PRNG using cryptographic hash function



# 基于MAC的PRNG

- NIST SP800-90, IEEE 802.11i, TLS/WTLS 中基于MAC的PRNG方案：
  - 采用HMAC
  - 密钥 $K$
  - 种子 $V$
  - 输入等于前一分组的MAC值
  - 每个分组的输入等于前一分组的MAC值
- 相比hash-PRNG，可提供更高的安全性



(b) PRNG using HMAC



## 本章小结

- **消息认证**是用来验证消息完整性的一种机制或服务。
- 对称密码在那些相互共享密钥的用户间提供认证。
- **消息认证码 (MAC)** 是一种需要使用密钥的算法，以可变长度的消息和密钥作为输入，参数一个认证码。拥有密钥的接收方能够计算认证码来**验证消息的完整性**。
- 构造MAC的方法之一，是将密码学Hash函数以某种方式和密钥捆绑起来。
- 构造MAC的另外一种方法是使用对称分组密码，将可变长度的输入转为固定长度的输出。