HASH 函数

定义

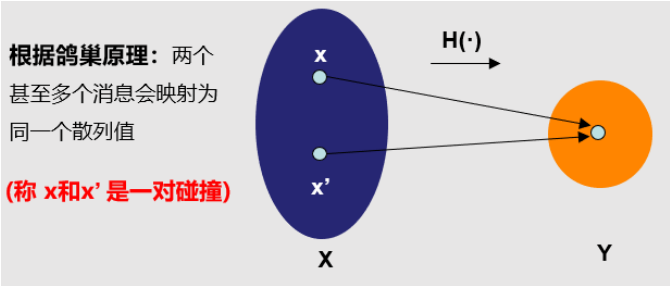
- 又称 哈希函数、杂凑函数、单向散列函数
- 是一个将任意长度的消息映射成固定长度输出的函数

$$H:\{0,1\}^{*}\rightarrow\{0,1\}^{n}$$

输入称为 “消息”
输出称为 “散列值” (Hash 值、消息摘要)
n 是散列值的长度

通常，Hash 函数是一个具有压缩功能的函数

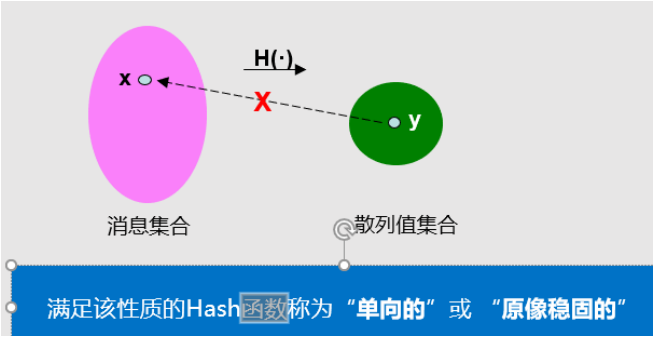
- 设 X 是消息的集合，Y 是散列值的集合
- 我们总是假设|X|≥|Y|，并且经常假设更强的条件|X|≥2|Y|



Hash 函数的安全性

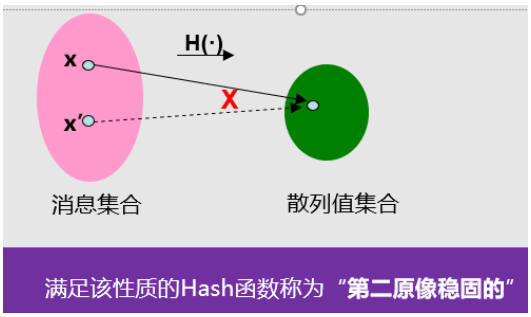
原像稳固

- 给定散列值 y，要找到一个 x，使得 H(x)=y 是计算上不可行的



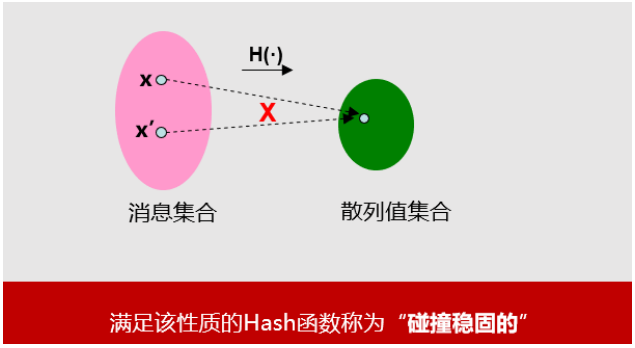
- **第二原像稳固**

  给定消息 x，找到另一个 x’ ，使得 H(x’)=H(x)是计算上不可行的



- **碰撞稳固**

  找到两个不同的消息 x 和 x’ ，使得 H(x)=H(x’)是计算上不可行的。



对 Hash 函数的攻击实际就是 寻找一对碰撞 的过程

如果Hash函数 H 设计得“好”，对给定x，要想求得相应的散列值，必须通过计算H(x)才行。

如果想绕过 H，采用其他方法求出散列值是计算上不可行的。

即使在已知若干散列值$H(x_1),H(x_2),...$的情况下，仍是如此。

因此，我们有

$H(x_1)+H(x_2)\neq H(x_1+x_2)$

$H(x_1)H(x_2)\neq H(x_1x_2)$

...

(这些特点在后面介绍数字签名时非常有用)

生日攻击告诉我们：为了能达到n-bit的安全性，你所选择的Hash函数的散列值长度应该是2n。

- 例如：如果你想让攻击者成功找到碰撞的可能性低于$1/2^{80}$，那么应该使用散列值长度是160-bit的Hash函数。

迭代法构造 Hash 函数的步骤

① **预处理**

用一个填充函数 pad(·)在消息 x 右方追加若干比特，得到比特串 y，使得 y 的长度为 t 的倍数。即有

$y = x \| pad(x) = y_1 \| y_2 \| \cdots \| y_r$，其中 $|y_i| = t$

典型的填充函数是先追加与 x 等长的值，再追加若干比特（如 0）。

该阶段必须保证变换是单射

- 因为如果预处理变换不是单射，则存在 x≠x’ 使得 y=y’，从而 H(x)=H(x’)，能够很容易找到碰撞。

② 迭代处理

设 H_0=IV 是一个长度为 m 的初始比特串，重复使用压缩函数 f，依次计算

$H_i = f(H_{i-1} \| y_i)$ 直到计算出 Hr 为止。

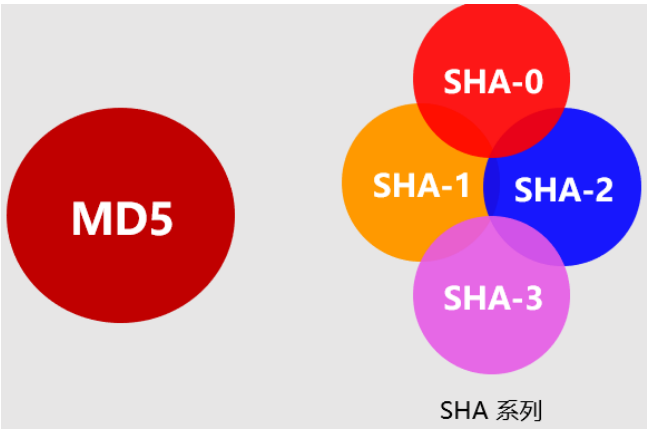
③ 输出变换

③ 输出变换（可选）

设函数 $g: \{0,1\}^m \to \{0,1\}^n$，令

$$H(x) = g(H_r)$$

几个著名的 Hash 函数



SHA 系列

MD5 加密的说法不对

对 MD5,SHA-0，SHA-1 的破译（不建议再使用这些算法）

SHA-2 系列：易遭到“长度扩展攻击”，建议使用:SHA-512/224、SHA-512/256

Q: 消息在传输过程中是否发生改变或被恶意篡改，我们怎么能知道？

使用“数据完整性技术”当消息发生改变时，我们可以检查出来。

Q: Hash函数能否保证数据的完整性?

举例:

某公司经网络发送一张订单m，为防范遭恶意篡改，先用Hash函数计算订单散列值H(m)，再将之与订单一起发送 [m,H(m)]。

接收方计算收到订单m’的散列值H(m’)，若与收到的散列值相同，则确信订单未被篡改。

问此方法是否行得通？为什么?

不行!

原因：攻击者可先篡改订单,再计算假订单的散列值 (接收方无法识别)

关键问题：Hash函数没有密钥 (给定消息，任何人都可以计算)

可以看得出，如果设计一个带有密钥的算法，便可以解决这一问题，也就可以防范攻击者进行伪造

消息认证码(MAC)算法便是这种带密钥的算法	它是实现数据完整性的重要工具
	其产生的输出也相应地被称作MAC

消息认证码 MAC　安全性

MAC 的安全性要求——抗伪造：

1. 在不知道密钥的情况下，给定任何消息，产生相应的 MAC 是计算上不可行的

2. 即使已知很多消息及对应的 MAC，对新消息产生 MAC 仍是计算上不可行的

CBC-MAC

利用已有分组密码间接构造MAC

利用分组密码的CBC模式构造MAC的方法：

设$E_k(\cdot)$是一个分组长度为n的分组密码的加密算法。

对任意消息x，首先对x进行分组，每组的长度为n。设消息x为 $x=x_1\ x_2 \ldots x_r$

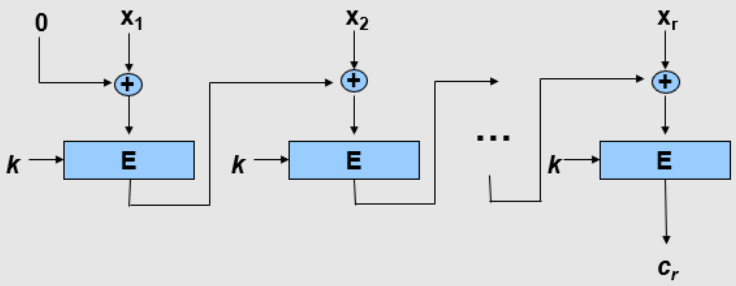
- 基于CBC模式构造MAC

① 选取初始向量: $c_0 = IV = 0$

② 依次计算:

$$c_i = E_k(x_i \oplus c_{i-1}) \quad (1 \le i \le r)$$

③ 输出散列值: c_r (只保留最后一个输出分组)



- **注意**
  - 该方法对于定长消息是安全的。
  - 如果消息长度可变，即使密钥K没泄露，同样不安全
    - 只要攻击者获得任意消息$m=x_1x_2\ldots x_{r-1}$的MAC

$$CBC\text{-}MAC_K(x_1x_2\cdots x_{r-1}) = c_{r-1}$$

    - 他任意找一个消息x_r(长度为一个分组)，便可产生一对碰撞
    - 因为 $m_1 = x_1x_2\cdots x_r, m_2 = c_{r-1} \oplus x_r$

$$CBC\text{-}MAC_K(x_1x_2\cdots x_r) = CBC\text{-}MAC_K(c_{r-1} \oplus x_r) = c_r$$

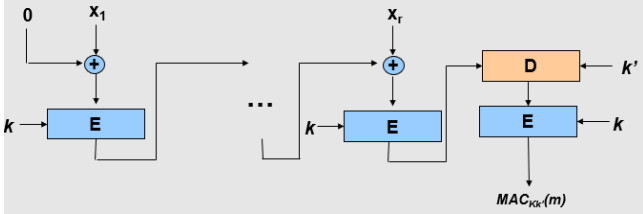
CBC-MAC 只有用于定长消息才能抗伪造

Q: 如何让 CBC-MAC 对于变长消息也能抗伪造？

ISO标准的CBC-MAC （改进方案3的改版——双密钥三重加密）

– 设消息m=x_1...x_r

$MAC_{KK'}(m)=E_K(D_{K'}(CBC\text{-}MAC_K(m)))$

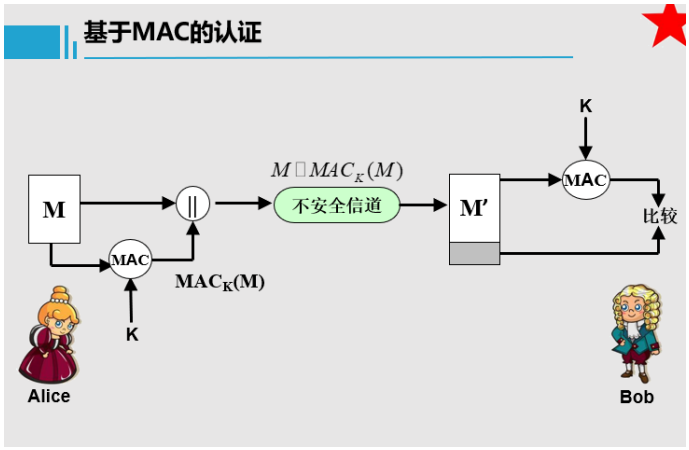


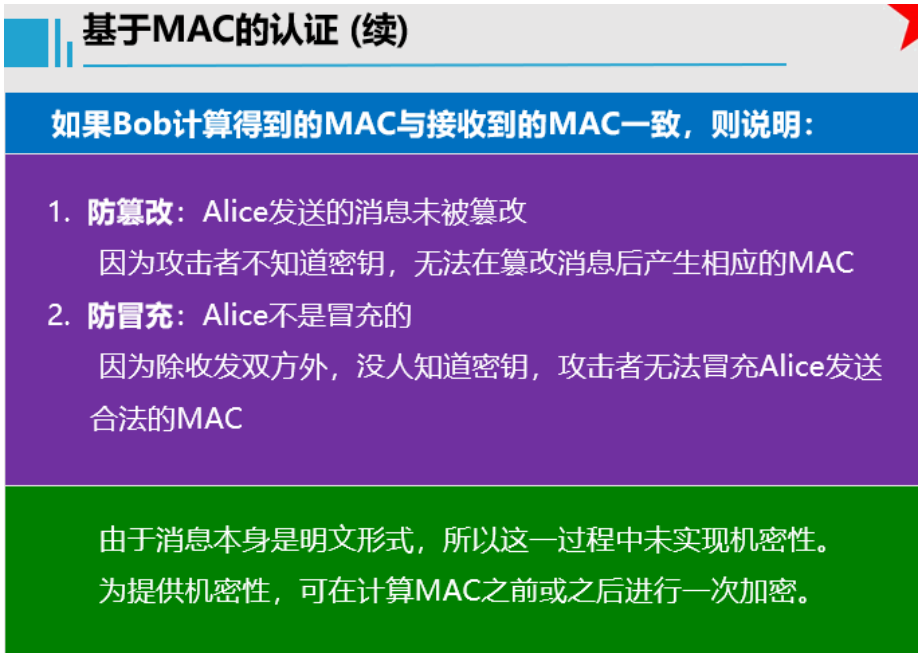
4.3 认证消息

消息认证，又称数据源认证，它的重要目标：

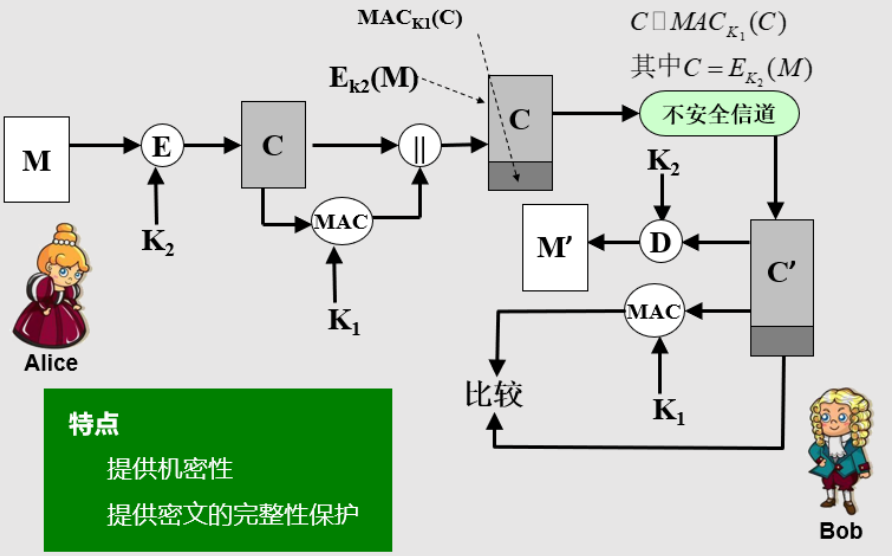
1. 保证传输消息的完整性（检查消息是否被改动过）
2. 保证消息是由指定发送者发来的

主要技术： 基于 MAC 的认证





提供机密性的消息认证 （方案1）

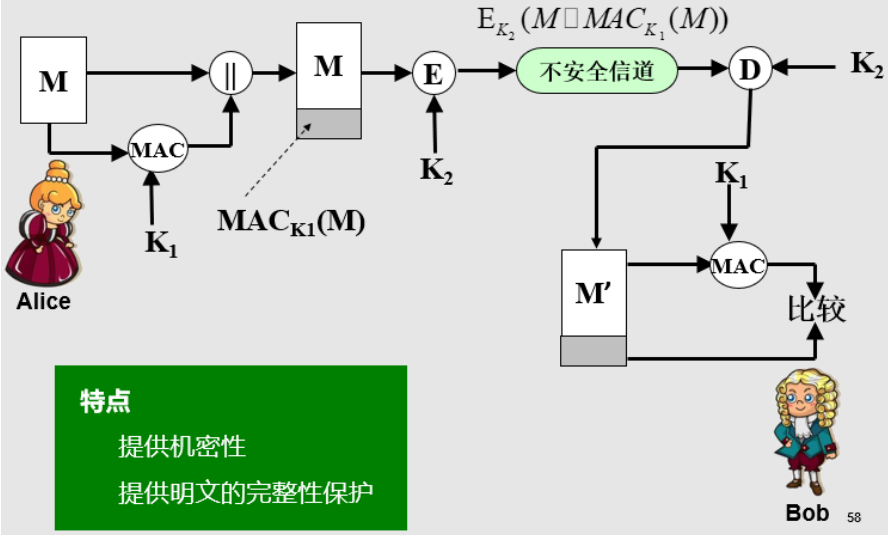


特点

提供机密性

提供密文的完整性保护

提供机密性的消息认证 （方案2）



特点

提供机密性

提供明文的完整性保护

58

两种方案的比较

方案1	方案2
明文M先被加密，再与MAC一起发送	明文M与MAC被一起加密

通常，我们希望直接对明文进行认证，因此 **方案2** 的使用方式更为常用。