第 4 章
数据认证

为什么需要数据认证?

- 验证数据的来源
- 确认数据没有被篡改或伪造
- 一个使用预共享秘密的简单认证方案:
  - Alice发送消息M 和密文$C = E_k(M)$ 给 Bob
  - Bob接收到消息后，用密钥 K 解密密文C 得到 M'
  - 如果$M' = M$ ，那么Bob 确认消息M 来自于Alice
- 公钥密码体系能够提供数据认证和抗抵赖功能
- 当需要认证一个很长的消息M时，只需要计算代表此消息的短字符串h，并进行加密

数字指纹

- 不需要使用秘密密钥而生成的一个长数据的短的表示，称为数字摘要或数字指纹
- 数字指纹可以使用密码散列函数得到，又称为单向散列函数
- 使用秘密密钥产生的数据的短表示，称为消息认证码 (*MAC*) 或标签
- MAC 可以通过加密的校验和算法得到
- 带密钥的散列消息认证码(*HMAC*) 是密码散列函数和密码校验和算法的组合

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

密码散列函数

- 散列函数以一个长的消息为输入，分成若干部分后，进行“打碎重组”，产生一个新的短字符串。
- *不是每一个散列函数都适合用于生成数字指纹*，例如：

$$M = M_1\ M_2\ \ldots\ M_k$$

其中 M_i 是一个 16-bit 的二进制串

定义下面的散列函数$H_\oplus$：

$$H_\oplus(M) = M_1 \oplus M_2 \oplus \ldots \oplus M_k$$

- 对于上面的散列函数，很容易举出一些例子，使得不同的明文消息具有相同的杂凑值
  - S_1: “He likes you but I hate you” 和S_2: “He hates you but I like you”
  - 把上面消息的英文字符用8-bit ASCII码进行编码，并去掉单词之间的空格，则得到 $H_\oplus(S_1) = H_\oplus(S_2)$

设计要求

设H 表示一个散列函数, Γ 是输入长度的上界, γ 是比Γ小得多的固定的输出长度

- 单向性: 计算一个给定消息串的数字指纹是容易的, 但是找到一个数字指纹对应的消息是困难的
  - 对于任意的二进制串x， 其中 $|x| \le \Gamma$, 计算 $H(x)$是容易的, 但是对于一个给定的散列值h ($|h| = \gamma$), 很难找到它对应的原像x使得$h = H(x)$

设计要求

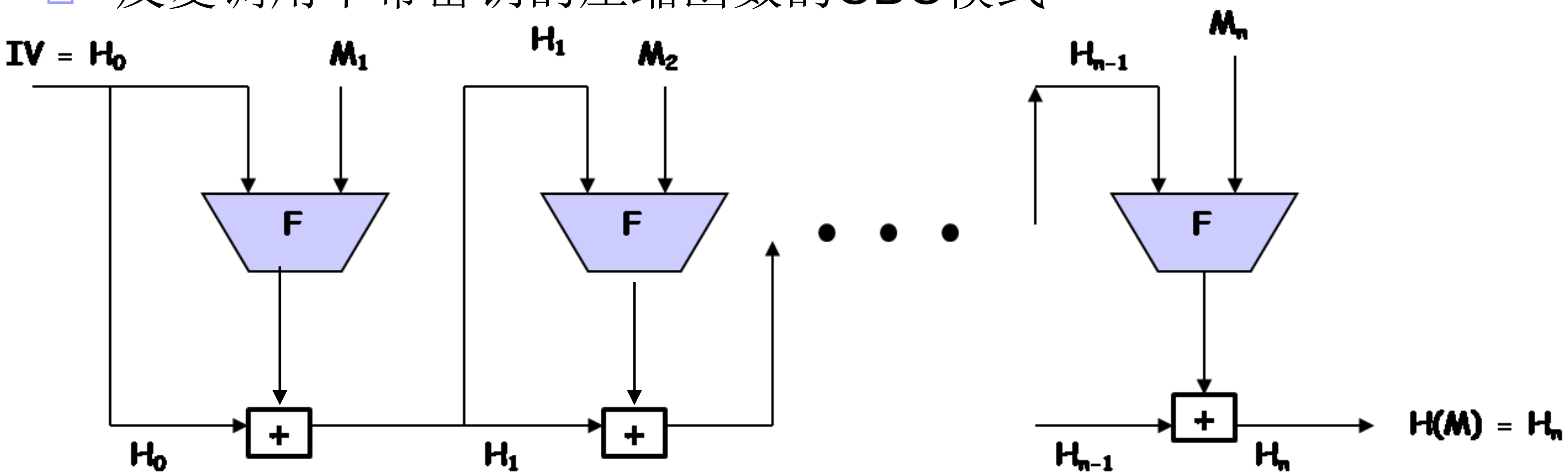
- 计算唯一性：找到两个不同的消息具有相同的数字指纹是困难的
  - 无碰撞性– 给定一个消息 x，满足$|x| \leq \Gamma$, 找到另一个不同的消息y ($|y| \leq \Gamma$)与x具有相同的散列值，即$H(x) = H(y)$，是困难的
  - 强无碰撞性– 很难找到两个消息x和y具有相同的散列，即$H(x) = H(y)$
- 注意：不满足强无碰撞性，并不意味着不满足无碰撞性

密码散列函数的探索

- 寻求密码散列函数
  - 尚未知是否存在单向的、计算唯一的密码散列函数
  - 几个曾经被认为是安全的密码散列函数, 已被证明不能满足强无碰撞性
  - 另一个常用的散列函数SHA-1算法，已经被证明其安全性低于预期
  - 本节介绍两个标准的密码散列函数SHA-512和WHIRLPOOL

基本结构

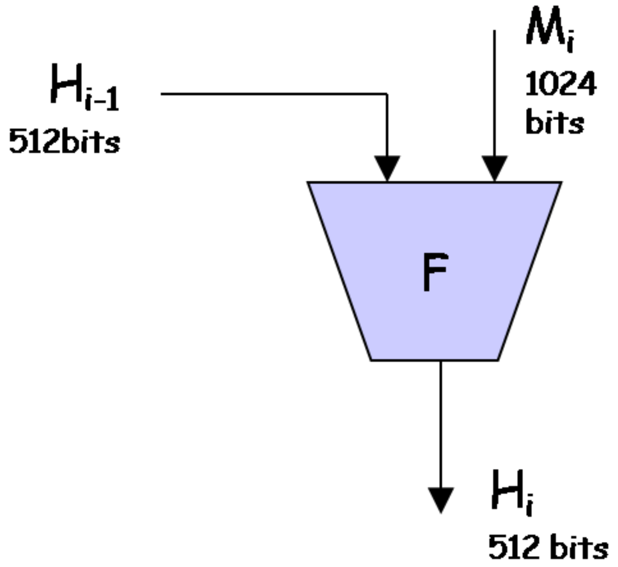
- SHA-1、SHA-2 系列散列函数，以及 WHIRLPOOL都具有相同的基本结构
- 这种结构的核心是压缩函数 *F*
  - 不同的散列函数使用不同的压缩函数
  - 反复调用不带密钥的压缩函数的CBC模式



M 是明文消息块, *IV* 是初始化向量, *F* 是压缩函数, “+” 是模加运算

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

SHA-512 初始化过程 (I)



- SHA-512 使用512-bit 初始化向量*IV*
- 设 $r_1, r_2, r_3, r_4, r_5, r_6, r_7,$ and r_8 be 是8个64-bit 寄存器
  - 开始时，这8个寄存器被设为8个素数的平方根的小数部分:

    $\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7}, \sqrt{11}, \sqrt{13}, \sqrt{17}, \sqrt{19},$

SHA-512 初始化过程 (II)

- 设 $\Gamma = 2^{128} - 1$，$\gamma = 512$
- M 是二进制消息，其中$|M| = L \le \Gamma$
- 设L 是128-bit 二进制串, 标记为 $b_{128}(L)$
- 将消息 M 进行填充，得到新的二进制字符串M' :

$$M' = M \,||\, 10^{\ell} \,||\, b_{128}(L),\ \ell \ge 0$$

其中$|M'|$ (记为L') 是1024的倍数，则有

$$L' = L + (1 + \ell) + 128 = L + \ell + 129$$

- L 可以表示为

$$L = 1024 \cdot \left\lfloor \frac{L}{1024} \right\rfloor + L \bmod 1024$$

- 于是，我们得到 :

$$\ell = \begin{cases} 895 - L \bmod 1024, & \text{if } 895 \ge L \bmod 1024, \\ 895 + (1024 - L \bmod 1024), & \text{if } 895 < L \bmod 1024. \end{cases}$$

- 因此，, L' 是1024的倍数. 设 $L' = 1024N$，则可以记作一系列1024-bit 的消息块: $M' = M_1M_2\ldots M_N$

SHA-512 压缩函数(I)

- 两个输入:
  - 一个1024-bit明文消息块 M_i
  - 一个512-bit 二进制串 H_{i-1}, 其中 $1 \le i \le N$， H_{i-1} 是$r_1r_2r_3r_4r_5r_6r_7r_8$的当前值

$$M_i = W_0 \cdot W_1 \cdots W_{15}, |W_i| = 64$$

$$\text{generate } W_{16} \cdot W_{17} \cdots W_{79} \text{ as follows}$$

$$W_t = [\sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}] \bmod 2^{64}$$

$$t = 16, \cdots, 79,$$

$$\sigma_0(W) = (W >>> 1) \oplus (W >>> 8) \oplus (W << 7)$$

$$\sigma_1(W) = (W >>> 19) \oplus (W >>> 61) \oplus (W << 6)$$

$W>>>n$: 循环右移n 位

$W<<n$: 线性左移 n 位

SHA-512 压缩函数(II)

Define :

logical conjunction : $X \wedge Y = (x_1 \wedge y_1)(x_2 \wedge y_2)\cdots(x_l \wedge y_l)$

logical disjunction : $X \vee Y = (x_1 \vee y_1)(x_2 \vee y_2)\cdots(x_l \vee y_l)$

logical negation : $\overline{X} = \overline{x}_1\overline{x}_2\cdots\overline{x}_l$

conditional predicate : $ch(X,Y,Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$

majority predicate : $maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$

设 $K_0,K_1,\ldots K_{79}$ 表示SHA-512 常量, 其中每一个常量是一个64-bit二进制 串，T_1 和 T_2 表示 64-bit 的临时变量，r 是一个 64-bit 寄存器. 设

$$\Delta_0(r) = (r >>> 28) \oplus (r >>> 34) \oplus (r >>> 39)$$

$$\Delta_1(r) = (r >>> 14) \oplus (r >>> 18) \oplus (r >>> 41)$$

SHA-512 压缩函数(III)

进行80 轮下面的运算:

$$T_1 \leftarrow [r_8 + ch(r_5, r_6, r_7) + \Delta_1(r_5) + W_t + K_t] \bmod 2^{64},$$

$$T_2 \leftarrow [\Delta_0(r_1) + maj(r_1, r_2, r_3)] \bmod 2^{64},$$

$$r_1 \leftarrow (T_1 + T_2) \bmod 2^{64},$$

$$r_2 \leftarrow r_1,$$

$$r_3 \leftarrow r_2,$$

$$r_4 \leftarrow r_3,$$

$$r_5 \leftarrow (r_4 + T_1) \bmod 2^{64},$$

$$r_6 \leftarrow r_5,$$

$$r_7 \leftarrow r_6,$$

$$r_8 \leftarrow r_7,$$

然后, $r_1r_2r_3r_4r_5r_6r_7r_8$ 中的512-bit 字符串 就是$F(M_i, H_{i-1})$的输出

SHA-512 算法

- 设 $X = X_1X_2\ldots X_k$, $Y = Y_1Y_2\ldots Y_k$ 是二进制串, 其中 X_i, Y_i 是 l-bit 字符串. 设l-bit的下列运算:

$$X \oplus_l Y = [(X_1 + Y_1) \bmod 2^l][(X_2 + Y_2) \bmod 2^l]\cdots[(X_k + Y_k) \bmod 2^l]$$

- 消息 M的数字指纹是 $H(M) = H_N$, 其中

$$\begin{aligned} H_0 &= \mathrm{IV}, \\ H_i &= H_{i-1} \oplus_{64} F(M_i, H_{i-1}), \\ i &= 1, 2, \ldots, N. \end{aligned}$$

WHIRLPOOL初始化过程

- 在Whirlpool中, $\Gamma = 2^{256} - 1$, $\gamma = 512$
- 消息M 是二进制串， $|M| = L \le \Gamma$，将L 表示为一个 256-bit 二进制串, 记为$b_{256}(L)$. 进行消息填充:

$$M' = M \,||\, 10^{\ell} \,||\, b_{256}(L), \ell \ge 0$$

满足$L' = |M'|$ 是512的倍数. 则有

$$L' = L + (1 + \ell) + 256 = L + \ell + 257$$

L 表示为

$$L = 512 \cdot \left\lfloor \frac{L}{512} \right\rfloor + L \bmod 512$$

于是, 我们有:

$$\ell = \begin{cases} 255 - L \bmod 512, & \text{if } 255 \ge L \bmod 512, \\ 255 + (512 - L \bmod 512), & \text{if } 255 < L \bmod 512. \end{cases}$$

L' 是512的倍数, 即 $L' = 512N$. 记为

$$M' = M_1 M_2 \ldots M_N$$

其中M_i 是一个512-bit 二进制串

WHIRLPOOL 压缩函数

- WHIRLPOOL的压缩函数定义为:

$$F(X,K) = W(X,K) \oplus X$$

- *W*(*X*, *K*) 类似于AES的加密算法
  - 输入: 512-bit 明文块*X*和 512-bit密钥*K*
  - 输出: 512-bit输出

- 消息*M*的数字指纹 $H(M) = H_N$, 是通过使用CBC模式得到的:

$$\begin{aligned} H_0 &= 0^{512}, \\ H_i &= H_{i-1} \oplus F(M_i, H_{i-1}) \\ &= H_{i-1} \oplus W(M_i, H_{i-1}) \oplus M_i, \\ i &= 1, 2, \ldots, N, \end{aligned}$$

W(X, K)的构造

- 由密钥K共生成11个 512-bit 的轮密钥, 记为$K_0, K_1, \ldots, K_{10}$.
  - $K_0 = K$
  - K_i $(1 \le i \le 10)$ 是由K_{i-1} 经过4个基本运算得到的
    - 字节替代 (*sub*)
    - 列移位 (*shc*)
    - 行混淆 (*mir*)
    - 常数相加 (*arc*)

$$K_i = arc(mir(shc(sub(K_{i-1}))), RC_i)$$

其中RC_i 是512-bit 常数，由WHIRLOOL S盒产生:

$$RC_i[j] = \begin{cases} s_{8(i-1)+j}, & \text{if } 0 \le j \le 7, \\ 00000000, & \text{if } 8 \le j \le 63, \end{cases}$$

其中 $i = 1, 2, \ldots, 10$

- 字节替代(*sub*)
  - WHIRLPOOL的字节替代使用 16 x 16 S盒
  - 设A = $(a_{i,j})_{8\times 8}$ 是8 x 8 的字节状态矩阵
  - 设$x = x_0x_1x_2x_3x_4x_5x_6x_7$ 是8-bit字符串, 其中每一个$x_i \in \{0,1\}$
  - 设$\pi_1(x)$ 是二进制串$x_0x_1x_2x_3$ 的十进制数值，$\pi_2(x)$ 是二进制串$x_4x_5x_6x_7$ 的十进制值
  - 定义替代函数*S* 为

$$S(x) = s_{\pi_1(x), \pi_2(x)}$$

其中$s_{u,v}$ 是WHIRLPOOL's S盒中第u行、第 v列上的元素，$0 \le u, v \le 7$

  - WHIRLPOOL的字节替代操作定义为:

$$sub(\mathrm{A}) = (S(a_{i,j}))_{8\times 8}$$

- 列移位(shc)

  类似于AES的行移位操作.

  第j列将向下循环位移j个字节.

- 行混淆(mir)
  - 类似于AES中的列混淆
  - 使用下面的常数矩阵：

$$\Delta = \begin{bmatrix} 01 & 01 & 04 & 01 & 08 & 05 & 02 & 09 \\ 09 & 01 & 01 & 04 & 01 & 08 & 05 & 02 \\ 02 & 09 & 01 & 01 & 04 & 01 & 08 & 05 \\ 05 & 02 & 09 & 01 & 01 & 04 & 01 & 08 \\ 08 & 05 & 02 & 09 & 01 & 01 & 04 & 01 \\ 01 & 08 & 05 & 02 & 09 & 01 & 01 & 04 \\ 04 & 01 & 08 & 05 & 02 & 09 & 01 & 01 \\ 01 & 04 & 01 & 08 & 05 & 02 & 09 & 01 \end{bmatrix}$$

  - 行混淆定义为:

$$mir(\mathrm{A}) = \mathrm{A} \bullet \triangle$$

- 常数加(arc) 和轮密钥加(ark)
  - 类似于AES的轮密钥加

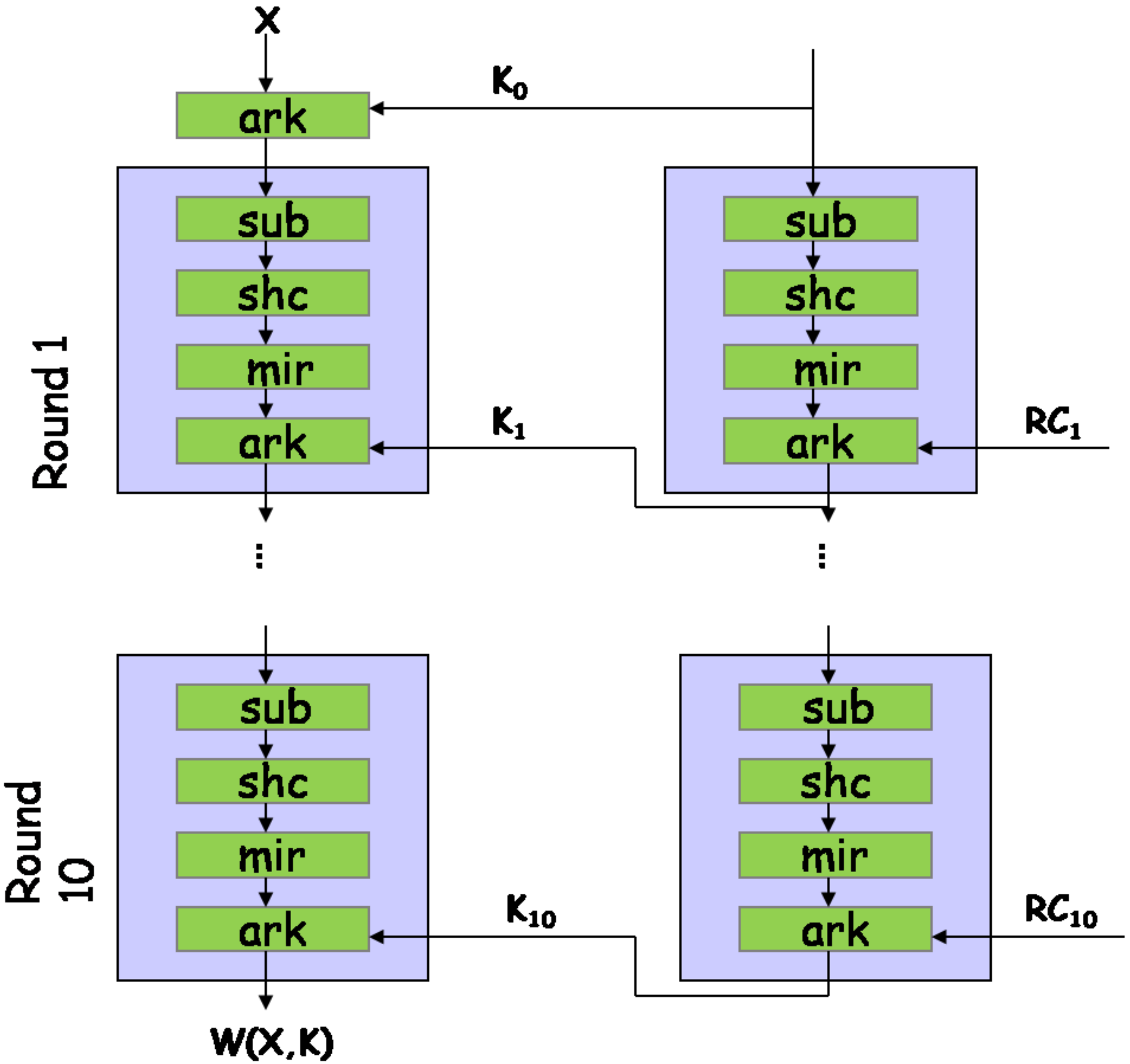
$$arc(\mathrm{A}, \mathrm{RC}_i) = \mathrm{A} \oplus \mathrm{RC}_i$$

$$ark(\mathrm{A}, \mathrm{K}_i) = \mathrm{A} \oplus \mathrm{K}_i$$

- 加密结构
  - 当轮密钥生成后，算法*W*将64字节字符串X表示成状态矩阵的形式
    $A = (a_{u,v})_{8\times8}$, 其中
    $$a_{u,v} = x_{8u+v}, \quad u,v = 0, 1, \ldots ,7$$
  - 然后，将A和K_0进行轮密钥加运算，生成新的状态矩阵A_0
  - 进行10轮的运算, 每一轮计算($1\le i \le 10$)
    $$A_i = ark(mir(shc(sub(A_{i-1}))), K_i),$$
    最后，$W(X, K) = A_{10}$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

W的流程图



《计算机网络安全的理论与实践（第2版）》．【美】王杰，高等教育出版社，2011年.

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

密码校验和

- 校验和通常用于检测网络通信中的传输错误
  - 但是，这些校验和不能用于数据认证或数字指纹，因为很容易找到不同的消息具有相同的校验和
- 我们可以用对称密码算法生成密码校验和，用于数据认证
- 密码校验和又称为消息认证码 (MAC)

异或密码校验和

设 E 表示AES-128 加密算法，K 是AES-128 秘密密钥

$$H_{\oplus}(M) = M_1 \oplus M_2 \oplus \cdots \oplus M_k$$
$$MAC(M) = E_K(H_{\oplus}(M))$$

这种方法是不安全的，容易受到中间人攻击.

例如, 假设Alice和Bob 共享一个AES-128 密钥K.

如果 Alice 发送M, $E_K(H_{\oplus}(M))$ 给Bob 来对消息M 进行认证，Malice 监听了这个密文,

那么Malice 能够用$E_K(H_{\oplus}(M))$ 假冒Alice的身份 .

中间人攻击

设 M’ = $Y_1Y_2\ldots Y_l$ 是任意消息, 其中Y_i 是128-bit 二进制串.

$$Y = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_l \oplus H_{\oplus}(M)$$
$$M'' = M' || Y$$

Malice 发送给Bob:

$$(M'', E_K(H_{\oplus}(M)))$$

Bob 先计算

$$H_{\oplus}(M'') = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_l \oplus Y = H_{\oplus}(M)$$

然后，解密 $E_K(H_{\oplus}(M))$， 得到 $H_{\oplus}(M) = H_{\oplus}(M'')$

这样Bob 认为消息 *M”* 来自于Alice，但实际上不是.

密码校验和的设计要求

- 设 $MAC_K(M)$ 为 M的消息认证码,其中K是秘密密钥，我们需要$MAC_K(M)$满足下列要求:

1. *正向易解性*: 计算$MAC_K(M)$ 是容易的.
2. *反向难解性*: 从$MAC_K(M)$计算出M是困难的.
3. *计算唯一性*: 从$(M, MAC_K(M))$找到 $M' \neq M$ 使得$MAC_K(M') = MAC_K(M)$是困难的.
4. *均匀分布*: 设 k 是消息认证码的长度，M是随机选取的数据. 设M'($M' \neq M$) 是随机选取或由M变换而来的，则$MAC_K(M') = MAC_K(M)$的概率是 2^{-k} .

密码校验和的构造

- 尚没有已知的算法能满足这四条准则

- 构造密码校验和的一般性方法:
  - 加密算法和单向散列函数

- 这种方法满足实际应用的需要

数据认证算法

- 1985年, NIST制定了一种数据认证码标准，称为DAC, 是基于DES算法的CBC模式
- 设M = $M_1M_2\ldots M_k$, 其中M_i 是64-bit 二进制串，K 是 DES 密钥，E 是DES 密码算法. 令

$$C_1 = E_K(M_1)$$
$$C_i = E_K(M_i \oplus C_{i-1})$$
$$i = 2,\cdots,k$$

则 DAC = C_k.

- 当 DES 的安全性受到威胁后，DAC 被新的消息认证码方案HMAC替代

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

密钥散列消息认证码-HMAC

- HMAC 是一种算法方案
- 使用散列函数和对称密码算法生成认证码
- HMAC的设计要求

1. 任何散列函数可以直接嵌入使用
2. HMAC中的密码散列函数应满足单向性、计算唯一性等基本性质
3. 秘密密钥的使用是简单易行的
4. HMAC 认证码的安全性强度与所采用的散列函数的安全性强度密切相关

HMAC参数

H: 嵌入散列函数 (例如SHA-512 和 WHIRLPOOL)
IV: 散列函数的初始化向量
M: 输入的要认证的消息
L: 消息块的个数
l: 散列函数的输出长度
b: 一个消息块的比特数, 其是8的倍数，且满足$b \geq l$
K: 秘密密钥，其长度≤b
K’: $K' = 0^{b-|K|}\ K$，是K的前缀填充，有$|K'| = b$
ipad: $ipad = (00110110)^{b/8}$
opad: $opad = (01011100)^{b/8}$
K'_0: $K'_0 = K' \oplus ipad$.
K'_1: $K'_1 = K' \oplus opad$.

HMAC算法

- HMAC算法描述如下:

$$\mathrm{HMAC}(K, M) = H(K_1^{'} \| H(K_0^{'} \| M))$$

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

第4章

数据认证

Part II

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

生日攻击基础

在一个23 个人的集体中, 至少两个人具有同一天生日的概率大于1/2.

证明. 23个人的生日互不相同的概率为:

$$\left(\frac{364}{365}\right)\times\left(\frac{363}{365}\right)\times\cdots\times\left(\frac{343}{365}\right) < 0.493$$

因此, 1 – 0.493 > ½.

强无碰撞性的复杂性上界

- 抗强无碰撞性的复杂性上界
  - 设 H 是一个输出长度为l的密码散列函数，则 H 至多有$n = 2^l$ 个不同的输出值
  - 问题: 2^l 是抗强无碰撞性的复杂性上界吗?
  - 答案是否定的，我们可以用生日攻击将复杂性降低为 $2^{l/2}$，成功概率在50%以上
- **生日悖论**:

  一个篮子里装了n个不同颜色的球，均匀地、独立随机地从篮子里取 k (k<n) 个球（一次取一个，记录颜色后放回），如果

  $$1.17\sqrt{n} < k < 1.25\sqrt{n}$$

  则一个球被取了两次的概率不小于1/2.
- SHA-1的复杂性上界: $2^{160/2} = 2^{80}$; SHA-512: $2^{512/2} = 2^{256}$

集合相交攻击

- 独立随机地从$\{1,2,\ldots,n\}$中选取两个各包含k个整数的集合，其中$k < n$
- 这两个集合有交集的概率$Q(n,k)$是多少？
  - 这两个集合不相交的概率为

$$\left[\left(1-\frac{1}{n}\right)^{k}\right]^{k}=\left(1-\frac{1}{n}\right)^{k^2}<\left(e^{-1/n}\right)^{k^2}$$

  - 因此,

$$Q(n,k)=1-\left(1-\frac{1}{n}\right)^{k^2}>1-e^{-k^2/n}$$

  - 如果 $0.69\sqrt{n}<k<0.7\sqrt{n}$ ，则

$$Q(n,k)>1/2$$

集合相交攻击的例子

- 集合相交攻击是生日攻击的一种形式
- 例如: Malice可以先用一个合法文档D得到认证者AU的签名

$$C = \langle H(D) \rangle_{K_{AU}^{r}}$$

- 然后，Malice产生一个新的文档 F， 其与D的内容不同，但是有$H(F)=H(D)$
- Malice用 (F,C) 表示文件F是由AU签署的

如何找到文档F?

- Malice 首先准备一个文档集合S_1，其包含$2^{l/2}$ 个不同的文档，这些文档均与文档D具有相同的意思. 这些文档可以用下面方法得到：
  a) 替换一个单词或句子
  b) 重新整理、调整句子的描述
  c) 使用不同的标点符号
  d) 重新组织文档的结构
  e) 改变语气，比如把被动语句改为主动语句
- Malice准备另一个文档集合S_2，其包含$2^{l/2}$个不同的文档，这些文档均与文档F具有相同的意思, 并计算

$$I(S_1) = \{H(X) \mid X \in S_1\}$$

$$I(S_2) = \{H(X) \mid X \in S_2\}$$

因此, $\Pr\{H(D') = H(F') \mid D' \in S_1, F' \in S_2\} > 1/2$

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

数字签名标准 (DSS)

- 对消息M的数字签名:

$$(M, E_{K_{rA}}(H(M)))$$

- 公钥密码系统
  - 产生数字签名的最有效的机制
  - RSA (直到2000年前专利保护)
- 数字签名标准-DSS
  - 1991年颁布
  - 2000年以后，RSA 和ECC 被制定为数字签名标准
  - 只生成数字签名, 不加密数据

DSS的构造

- H: SHA-1 (160 bit)
- L: $512 < L < 1024$

参数:

- P: 大素数; $2^{L-1} < p < 2^{L}$
- q: $p-1$的素因子; $2^{159} < q < 2^{160}$
- g: $g = h^{(p-1)/q} \bmod p$; $1 < h < p-1$, $g > 1$

DSS签名

- Alice 要对消息M进行签名
- 随机选取一个私钥, $0 < x_A < q$
- 计算公钥: $y_A = g^{xA} \bmod p$
- 随机选取一个整数: $0 < k_A < q$
- $r_A = (g^{kA} \bmod p) \bmod q$
- $k_A^{-1} = k_A^{q-2} \bmod q$
- $s_A = k_A^{-1}(H(M)+x_A r_A) \bmod q$
- M的数字签名为: (r_A, s_A)

DSS验证签名

- Bob 得到签名值(M', (r_A', S_A')) 和公钥证书 CA[y_A]
- 解析公钥证书CA[y_A]，得到Alice的公钥 y_A
- 验证签名:
  - $w = (S_A')^{-1} \bmod q = (S_A')^{q-1} \bmod q$
  - $u1 = (H(M')\ w) \bmod q$
  - $u2 = (r_A'\ w) \bmod q$
  - $v = [(g^{u1} y_A^{u2}) \bmod p] \bmod q$
  - 如果 $v = r_A'$，则签名验证通过

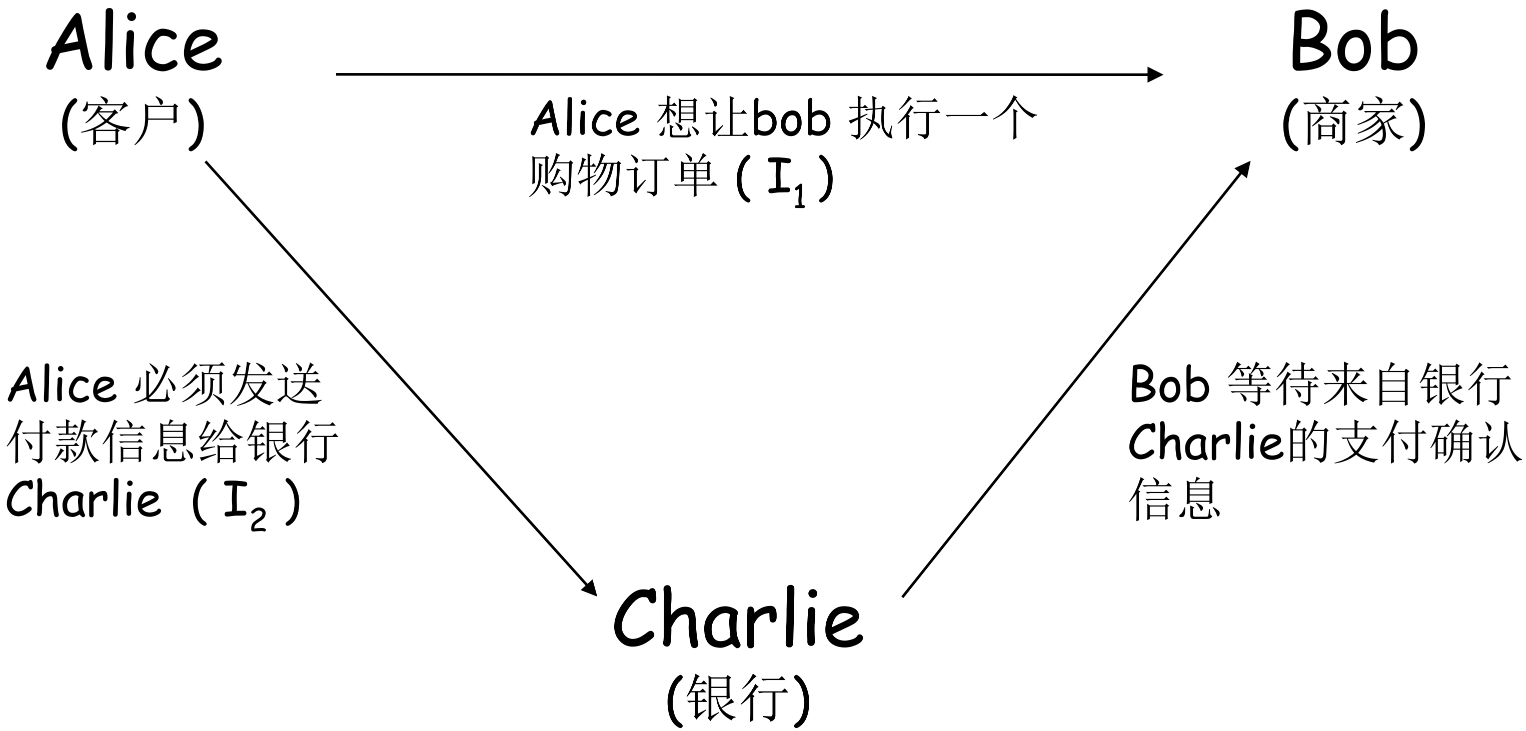
DSS的安全强度

- 基于SHA-1的安全强度和求解离散对数的困难性
  - 攻破SHA-1散列函数的强无碰撞性的已经由 2^{80} 降低到2^{63}
  - 攻破无碰撞性更困难
  - 离散对数的难解性表明，从r_A 和 s_A 计算出 k_A 或 x_A 是困难的

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

双签名与电子交易



Alice
(客户)
Bob
(商家)
Alice 想让bob 执行一个购物订单 (I_1)
Alice 必须发送付款信息给银行Charlie (I_2)
Bob 等待来自银行Charlie的支付确认信息
Charlie
(银行)

双签名

- 我们不想让Bob 看到I_2，不想让Charlie 看到I_1
- 在Bob 得到 I_1 之前，Charlie 不能发送I_2 给Bob
- I_1 和I_2 必须是绑定的，以防止付款与订单不一致
- 所有的消息必须被认证和加密，即没有有用的消息被泄露、修改或伪造

双签名

- 一种用于电子交易的交互式认证协议
- 提供安全和隐私性保护
- 在***SET协议*** (Secure Electronic Transactions)中使用, 此协议是由Visa 和 MasterCard 在1996年设计的，但是没有在实际中推广
- 需要
  - Alice, Bob和Charlie 协商确定使用的散列函数H 和公钥加密算法E
  - Alice, Bob和Charlie 必须各自有一对RSA密钥对: (K_A^u, K_A^r), (K_B^u, K_B^r), (K_C^u, K_C^r)

SET协议: 客户Alice

- 计算下面的值:

$$s_B = E_{K_B^u}(I_1), s_C = E_{K_C^u}(I_2),$$

$$h_B = H(s_B), h_C = H(s_C),$$

$$ds = D_{K_A^r}(H(h_B \| h_C))$$

- 发送 (s_B, s_C, ds)给Bob
- 等待 Bob的收据 $R_B = E_{K_A^u}(D_{K_B^r}(R_B))$
- 用$K_A{}^r$解密 R_B，得到$D_{K_B^r}(R_B)$；用$K_B{}^u$ 验证 Bob 的签名

SET协议: 商家Bob

- 验证Alice的签名：

  比较 $H(H(s_B) \parallel H(s_C))$ 与 $E_{K_A^u}(ds)$
- 解密 $I_1 = D_{K_B^r}(s_B)$
- 发送 (s_B, s_C, ds) 给 Charlie
- 等待Charlie的收据$R_C = E_{K_B^u}(D_{K_C^r}(R_C))$
- 用$K_B{}^r$ 解密 R_C 得到 $D_{K_C^r}(R_C)$，用$K_C{}^u$验证Charlie的签名
- 发送签名的收据 R_B，$E_{K_A^u}(D_{K_B^r}(R_B))$ 给Alice

SET协议: 银行Charlie

- 验证Alice的签名：

  比较 $H(H(s_B) \| H(s_C))$ 和 $E_{K_A^u}(ds)$

- 解密 $I_2 = D_{K_C^r}(s_C)$

- 如果I_2包含了有效的付款信息，则执行正确的付款交易操作，并发送签名的收据R_C，$E_{K_B^u}(D_{K_C^r}(R_C))$给Bob

第4章 内容概要

- 4.1 密码散列函数
- 4.2 密码校验和
- 4.3 HMAC
- 4.4 密码本偏移操作模式
- 4.5 生日攻击
- 4.6 数字签名标准
- 4.7 双签名与电子交易
- 4.8 盲签名与电子现金

盲签名

- 让签名者进行签名，但不向签名者泄露所签文件的内容

- 需要签名的文档与一个掩盖因子绑定，以防止签名者看到文档的信息，但是掩盖因子可以在以后被去掉

RSA盲签名

- 随机产生$r < n$ (掩盖因子)， 满足$\gcd(r, n) = 1$
- 令 $M_r = M r^e \bmod n$
- 签名者对消息M_r 进行签名，得到$s_r = M_r^{\,d} \bmod n$
- 可以通过以下方法去除掩盖因子 r :

$$s_M = s_r\, r^{-1} \bmod n$$
$$= M^d \bmod n$$

证明：

- 掩盖因子被去除：

$$
\begin{aligned}
s_M &= (s_r\, r^{-1}) \bmod n \\
&= (M^d\, r^{ed}\, r^{-1}) \bmod n
\end{aligned}
$$

- 因为

$$ed \equiv 1 \bmod \phi(n))$$

$$r^{ed} \equiv r \bmod n$$ (Fermat’s little theorem)

- 所以

$$s_M = M^d \bmod n$$

电子现金

- 真正的现金具有以下特性:
  - 匿名性
  - 转让性
  - 可分割性
  - 很难伪造
- 电子现金能具备这些特性吗?

理想的电子现金协议

- 一个理想的电子现金协议应具有以下性质:
  - 匿名性和不可追踪性
  - 安全性: 不能被修改或伪造
  - 便捷性: 允许离线交易
  - 单一性: 不能被复制或重用
  - 转让性: 可以转让
  - 可分割性: 可以被分割成小的数值.
- 尚没有满足以上所有性质的电子现金协议

电子现金-eCash

- 1980年代提出
- 满足电子现金的很多重要性质
- 使用盲签名来保证匿名性和不可追踪性
- 设 $\mathcal{B}$ 表示金融机构
- 设$\mathcal{B}$的 RSA 参数为 (n, d, e)

购买电子现金

- Alice 进行下面的操作:
  - 生成一个序列号 m 来代表要购买的电子现金
  - 产生一个随机数 $r < n$ (掩盖因子)， 并计算$x = mr^e \bmod n$
  - 发送 x 和她的账户号给银行$\mathcal{B}$
  - 银行$\mathcal{B}$ 从 Alice的账户支取 \$1，并发送 $y = x^d \bmod n$ 给 Alice (获得银行的签名)
  - Alice 计算 $z \equiv y\, r^{-1} \equiv m^d \bmod n$
  - Alice 得到电子现金(m, z)

兑换电子现金

- Bob 从Alice收到了电子现金, 并想进行兑换
  - 他发送(m, z)和账户号给银行$\mathcal{B}$.
  - 如果签名是有效的，而且序列号为 m的电子现金没有被兑现过, 则银行记录下序列号 m ，并支付$1给Bob的账户
- 问题: 既然复制(m, z)是容易的, Bob如何能够防止别人在他之前兑换电子现金呢?