数字签名方案

清华大学计算机系
于红波
2023年5月24日

数字签名与消息认证的区别

- 消息认证码：当收发者之间没有利害冲突时，这对于防止第三者的破坏已经足够了。
  - 收方能够验证消息发送者身份是否被篡改
  - 收方能够验证所发消息内容是否被篡改
- 数字签名：当收发双方存在利害冲突时，单纯用消息认证技术就无法解决他们之间的纠纷。必须采用数字签名技术。
  - 数字签名能确定消息来源的真实性
  - 数字签名能保证实体身份的真实性
  - 数字签名是不可否认的

数字签名与公钥加密的区别

- 公钥加密：
  - A采用B的公开密钥对信息加密，A将密文发给B；
  - B用自己的私钥对收到的密文解密，恢复出明文。
- 数字签名：
  - **A采用自己的私钥**对消息m签名，A将m和签名发给B；
  - B收到A的签名后，**采用A的公钥**来验证签名的有效性
  - 一个签名的消息很可能在多年之后才验证其真实性；
  - 数字签名可能需要多次验证；
  - 对数字签名的安全性和防伪造要求很高；
  - 要求签名速度比验证速度更快。

数字签名的分类

- 按照消息是否被压缩
  - 对整体消息进行签名；
  - 对压缩的消息进行签名。
- 按照消息/签名的对应关系划分
  - **确定性（deterministic）数字签名：** 消息与签名一一对应，对同一消息的签名永不变化，如RSA和Rabin算法；
  - **随机化（randomized）或概率式数字签名：** 对同一消息的签名是变化的。因此，此类签名取决于算法中的随机参数的取值，如ElGamal算法。

数字签名方案

- 定义：一个数字签名方案由一个概率多项式时间算法组成的三元组(Gen, Sign, Vrfy)，满足下列条件：

① 密钥生成算法Gen以安全参数1^n为参数，并输出一对密钥（pk, sk），分别称为公钥和私钥。为了方便，假设pk和sk长度至少为n, 并且n可以由pk, sk确定。

② 签名算法Sign以一个私钥sk和消息 $m \in \{0,1\}^*$ 作为输入，输出一个签名 σ，表示为 $\sigma \leftarrow Sign_{sk}(m)$

③ 确定的验证算法Vrfy以一个公钥pk, 消息m和一个签名 σ 为输入，输出一个位b, 当b=1时，签名有效；b=1时，签名无效。将其表示为 $b := Vrfy_{pk}(m, \sigma)$。

数字签名实验

- 数字签名实验 $\text{Sig\_forge}_{A,\Pi}(n)$

(1) 运行Gen(1^n)得到密钥（pk, sk)

(2) 敌手已知pk, 可以访问签名预言机Sign(·)。此预言机对敌手选定的任意消息m返回一个签名Sign_{sk}(m)。敌手输出(m,σ)。用Ω表示已经问询过的签名消息的集合.

(3) 当且仅当： $\text{Vrfy}(m, \sigma) = 1$，且$m \notin \Omega$，输出1.

- 数字签名方案安全定义：

对签名方案Π = (Gen, Sign, Vrfy)，如果所有的多项式时间敌手A，存在一个可忽略函数negl，满足

$$\Pr[\text{Sig\_forge}_{A,\Pi}(n) = 1] \le \text{negl}(n)$$

则此签名方案在适应性选择消息攻击下是存在性不可伪造的。

“教科书式”RSA数字签名体制

- 参数生成：
  - 令 $n = p_1 \times p_2$ ，p_1和p_2是大素数；
  - 令$m, s \in Z_n$（整数域）
  - 选e，并计算出d，使 $ed \equiv 1 \bmod \varphi(n)$
  - 将n, e公开（公钥），将p_1、p_2和d保密（私钥）。
- 签名过程：

  对$m \in Z_n$，定义签名：$s = Sig_k(m) = m^d \bmod n$
- 验证过程：

  给定m, s，验证：$m \equiv s^e \bmod n$?

"教科书式" RSA签名体制的安全性

- 无消息伪造：

  给定公钥pk=<N,e>, 任意选择一个 $\sigma \in Z_N^*$ 并计算 $m := [\sigma^e \bmod N]$，然后输出一个伪造 (m, σ)

  容易验证，σ 是m的一个有效签名。

- 利用乘法同态伪造对任意消息伪造签名：

  假设敌手希望伪造 $m \in Z_N^*$ 关于公钥pk=<N, e>的签名：敌手随机选择一个消息 $m_1 \in Z_N^*$, 设置 $m_2 = [m / m_1 \bmod N]$ 分别得到m_1和m_2的签名 σ_1和 σ_2 。伪造m的有效签名为 $\sigma := [\sigma_1 \cdot \sigma_2 \bmod N]$

Hash-and-Sign (先散列后签名)

- 设Π=(Gen, Sign, Vrfy)是一个长度为l(n)的标准签名方案, Π_H=(Gen_H,H)是一个输出长度为l(n)的Hash函数。构造一个具有任意输入长度的签名方案 Π’=(Gen’, Sign’, Vrfy’)如下：
  - Gen’：输入参数1^n, 运行$Gen_S(1^n)$产生<pk,sk>, 运行$Gen_H(1^n)$产生s; 其中公钥是pk’=<pk,s>, 私钥是 sk’=<sk,s>.
  - Sign’：输入私钥<sk,s>和一个消息m $\in\{0,1\}^*$，计算 $\sigma \leftarrow \mathrm{Sign}_{sk}(H^s(m))$.
  - Vrfy ’:输入公钥<pk,s>和消息m $\in\{0,1\}^*$, MAC码 σ' ,当且仅当$\mathrm{Vrfy}_{pk}(H^s(m),\sigma)=1$时输出1.
- 如果Π是适应性选择消息攻击下是存在性不可伪造的, Π_H是一个抗碰撞的Hash函数，上述构造是适应性选择消息攻击下是存在性不可伪造的。

ElGamal签名体制

- 参数生成
  - p：一个大素数，可使Z_p中求解离散对数为困难问题；
  - g：是群Z_P^*的一个生成元或本原元素；
  - M：消息空间，为Z_P^*；
  - S：签名空间，为Z_{P-1}；
  - x：用户秘密钥，$x \in Z_P^*$
  - $y \equiv g^x \bmod p$
  - p, g, y为公钥，x为秘密钥。
- 签名过程
  - 选择秘密随机数$k \in Z_P^*$，$m \in M$
  - 计算：H(m)
  - 计算：$r = g^k \bmod p$
  - 计算：$s = [H(m) - xr]k^{-1} \bmod (p-1)$
  - 签名为$Sig_k(m) = (r, s)$，将m和(r, s)送给对方。

ElGamal签名体制

- 验证过程
  - 收信人收到m和(r, s)；
  - 计算：H(m);
  - 验证：$Ver_k(H(m), (r, s))$=真⇔ $y^r r^s \equiv g^{H(m)} \bmod p$;

    左边：$y^r r^s \equiv g^{xr} g^{sk} \bmod p \equiv g^{(rx+sk)} \bmod p$

    而$(rx+sk) \equiv H(m) \bmod p-1$

    故：$y^r r^s \equiv g^{H(m)} \bmod p$

ElGamal签名体制

- 例：
  - 选择p=467, g=2, x=127, 则有$y \equiv g^x \equiv 2^{127} \equiv 132 \bmod 467$
  - 若待送消息m的杂凑值H(m)=100，选随机数k=213

    注意：(213, 466)=1, 且$213^{-1} \bmod 466=431$
  - 则：$r=2^{213}=29 \bmod 467$, $s=(100-127*29)431=51 \bmod 466$。
  - 验证：(1)收信人计算H(m)=100,

    (2)验证：$132^{29}29^{51}=189 \bmod 467$

    $2^{100}=189 \bmod 467$

ElGamal签名体制的安全性

- 安全性：
  - 在不知{消息,签名}对时，伪造签名相当于求离散对数；
  - 如果攻击者掌握了同一随机数**k**下的两个消息m_1, m_2的合法签名（r_1, s_1）（r_2, s_2），就会构造如下的方程：

    $m_1=r_1\mathbf{x}+s_1\mathbf{k}$

    $m_2=r_2\mathbf{x}+s_2\mathbf{k}$

    可见：攻击者解此方程可以求出x和k。
- 要确保此签名体制的安全性，就必须保证每次签名时，选择不同的随机数k。

Schnorr签名体制

参数生成

- p, q：大素数, q|p-1, 确保Z_p中求解离散对数为困难问题；
- g：是Z_p中乘群Z_P^*的一个元素，且$g^q \equiv 1 \bmod p$；
- M：消息空间，为Z_P^*；
- S：签名空间，为$Z_P^* \times Z_{P-1}$；
- x：用户秘密钥，$1<x<q$
- y：用户公钥，$y \equiv g^x \bmod p$
- p, q, g, y为公钥，x为秘密钥。

签名过程

- 用户选择秘密随机数$k \in Z_q$， $m \in M$
- 计算：$w=g^k \bmod p$
- 计算：$r=H(w||m)$
- 计算：$s=k+xr \bmod q$
- 签名：$Sig_k(m)=(r, s)$作为签名，将m和(r, s)送给对方。

Schnorr签名体制

□签名验证

收信人收到消息m和签名(r, s)

计算：$w' = g^s y^{-r} \bmod p$

计算：$H(w' \| m)$

验证$H(w' \| m) = r$？即 $Ver(y, (e, s), m) =$ 真

Schnorr签名和ElGamal签名的区别

- 在ElGamal体制中，g为Z_p的本原元素；在Schnorr体制中，g为Z_p^*中的子集Z_q^*的本原元，它不是Z_p^*的本原元。
- Schnorr的签名长度要比ElGamal短，由|q|及|H(m)|决定。
- $w=g^k \bmod p$可以预先计算，签名只需1次乘法和1次加法，所以签名速度非常快，适用于智能卡应用。

DSS签名体制

DSS概况

- 1991年8月由NIST公布
- 1994年5月19日由NIST正式公布
- 1994年12月1日正式成为美国联邦信息处理标准
- 它是基于ElGamal和Schnorr签名体制设计的
- 该签名体制有较好的兼容性和适用性，已经成为网络安全体系的基本构件之一。

DSA

- DSA是DSS签名标准中所采用的数字签名算法；
- 此算法由D. W. Kravitz设计。

DSS签名算法——DSA

参数

- p：大素数，$2^{L}-1<p<2^{L}$，512≦L≦1024；
- q：(p-1)的素因子，且$2^{159}<q<2^{160}$，即字长160b
- g：$g \equiv h^{(p-1)/q} \bmod p$，且1<h<(p-1)，$h^{(p-1)/q} \bmod p >1$
- x：选择用户私钥，1<x<q
- y：计算用户公钥，$y \equiv g^{x} \bmod p$
- p, q, g, y为公钥，x为私钥。

签名过程

- 用户选择秘密随机数k，0<k<q
- 计算：H(m)
- 计算：$r=(g^{k} \bmod p) \bmod q$
- 计算：$s=[k^{-1}(H(m)+xr)] \bmod q$
- 签名：$Sig_k(m)=(r, s)$，将m和(r, s)送给对方。

DSS签名算法——DSA

验证过程

- 收信人收到m和(r, s)；
- 计算：H(m)；
- 计算：$w=s^{-1} \bmod q$
- 计算：$u1=[H(m)w] \bmod q$
- 计算：$u2=rw \bmod q$
- 计算：$v=[(g^{u1}y^{u2}) \bmod p] \bmod q$
- 验证：$v \equiv r$?

证明：

$v=[(g^{u1}y^{u2}) \bmod p] \bmod q$

$=[g^{H(m)w}y^{rw} \bmod p] \bmod q$

$=[g^{H(m)w}g^{xrw} \bmod p] \bmod q$

$=[g^{[H(M)+xr]w} \bmod p\ ] \bmod q$

而： $[H(m)+xr]w = [H(m)+xr]s^{-1}=k \bmod q$

所以：$v=g^k \bmod q= r$

Lamport的“一次性签名方案”

构造方法：设f 为一单向函数，构造一个长度为$l=l(n)$的签名方案：

- Gen: 输入1n, 当 $i\in\{0,1,\ldots,l\}$ 时，按下面方式进行

  (1)随机选择 $x_{i,0},x_{1,0}\leftarrow\{0,1\}^n$

  (2) 计算 $y_{i,0}=f(x_{i,0}),y_{i,1}=f(x_{i,1})$
- 公钥pk和私钥sk为

$$pk:=\begin{pmatrix} y_{1,0} & y_{2,1} & \cdots & y_{l,0} \\ y_{1,1} & y_{2,2} & \cdots & y_{l,1} \end{pmatrix}, sk:=\begin{pmatrix} x_{1,0} & x_{2,1} & \cdots & x_{l,0} \\ x_{1,1} & x_{2,2} & \cdots & x_{l,1} \end{pmatrix}$$

- Sign: 输入上面的私钥和消息 $m\in\{0,1\}^l$,其中 $m=m_1m_2\cdots m_l$，输出签名$(x_{1,m_1},x_{2,m_2},\cdots,x_{l,m_l})$ 。
- Vrfy:输入上面的公钥钥和消息$m\in\{0,1\}^l$ 和签名$\sigma=(x_1,x_2,\cdots,x_l)$

  对于$1\le i\le l$, 当且仅当 $f(x_i)=y_{i,m_i}$ 时输出1.

Lamport的“一次性签名方案（OTS）”

- 基于单向函数的一次性签名方案

例子： 对3比特的消息m=$m_0m_1m_2$进行签名。

私钥：随机选择$x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}$

公钥：

$$y_{1.0} = f(x_{1,0}), y_{1,1} = f(x_{1,1}),$$

$$y_{2,0} = f(x_{2,0}), y_{2,1} = f(x_{2,1}),$$

$$y_{3,0} = f(x_{3,0}), y_{3,1} = f(x_{3,1})$$

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix}$$

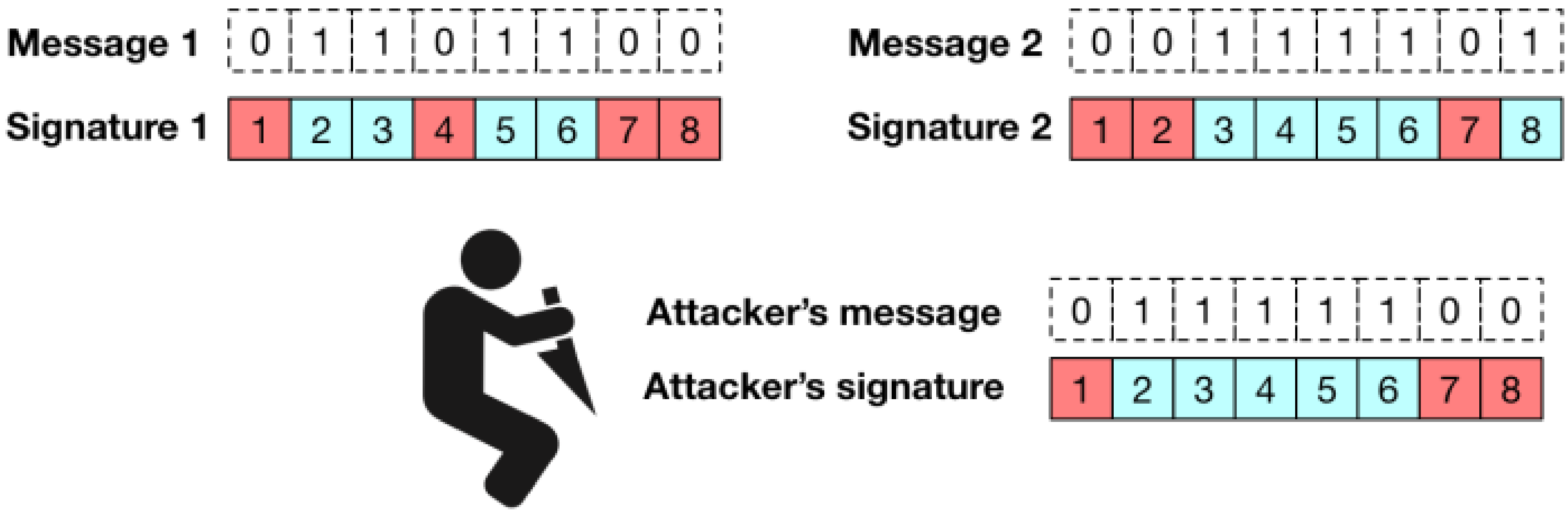
Lamport的“一次性签名方案”

- 签名消息m=011

$$sk=\begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma=(x_{1,0},x_{2,1},x_{3,1})$$

- 验证 m=011和 $\sigma=(x_{1,0},x_{2,1},x_{3,1})$

$$pk=\begin{pmatrix} \boxed{y_{1,0}} & y_{2,0} & y_{3,0} \\ y_{1,1} & \boxed{y_{2,1}} & \boxed{y_{3,1}} \end{pmatrix} \Rightarrow \begin{matrix} f(x_{1,0})=y_{1,0}\,? \\ f(x_{2,1})=y_{2,1}\,? \\ f(x_{3,1})=y_{3,1}\,? \end{matrix}$$


Message 1
0 1 1 0 1 1 0 0
Signature 1
1 2 3 4 5 6 7 8
Message 2
0 0 1 1 1 1 0 1
Signature 2
1 2 3 4 5 6 7 8
Attacker's message
0 1 1 1 1 1 0 0
Attacker's signature
1 2 3 4 5 6 7 8

Cost

- 假设使用160位的Hash，密钥长度160*2*k=320k, 取k=160，密钥长度160*2*160=51200bits=6400bytes. 是1024-RSA公钥长度的50倍。签名长度为160*k=160*160=25600bits=3200bytes. 是1024-RSA签名长度的25倍。

Winternitz一次签名：时间换取空间

- 思想：一次对一个字节（或者4，8，16比特）进行签名。
- 假设将消息分成每4比特一组。每一组为0-15的整数。

设F为Hash函数，定义 $F^b(x)=\begin{cases} x, b=0 \\ F^{b-1}(F(x)), b>0 \end{cases}$

私钥：对每一组消息w_i，随机选择一个u比特的串X_i, 作为私钥，公钥$F^{16}(X_i)$。

签名： w_i的签名 $C_i=F^{wi}(X_i)$

验签：$F^{16-wi}(C_i)$与公钥$F^{16}(X_i)$相比较

Winternitz 一次签名（WOTS）

- 想法：签名大小与时间的折衷
- $sk = [\dots, x_i, \dots]$
- $pk = [\dots, H^{w-1}(x_i), \dots]$

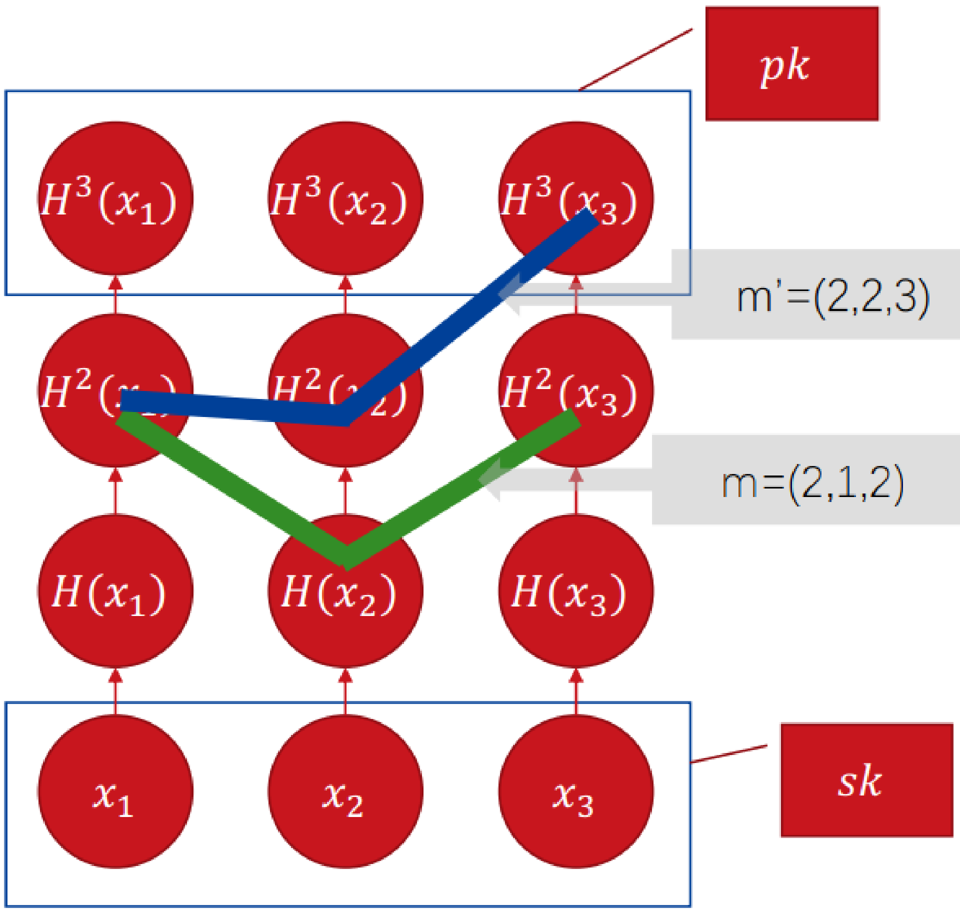
  其中$H^i(x) = H\left(H^{i-1}(x)\right)$
- sign(sk,m, σ):

  m=$[m_1, \dots, m_l] \in \{0,1, \dots, w-1\}^l$

  $\sigma = [\dots, H^{m_i}(x_i), \dots]$
- verify(pk, m, σ):

  $H^{w-1-m_i}(\sigma_i) = H^{w-1}(x_i) = pk_i$

安全性：并不安全!



Winternitz一次签名：时间换取空间

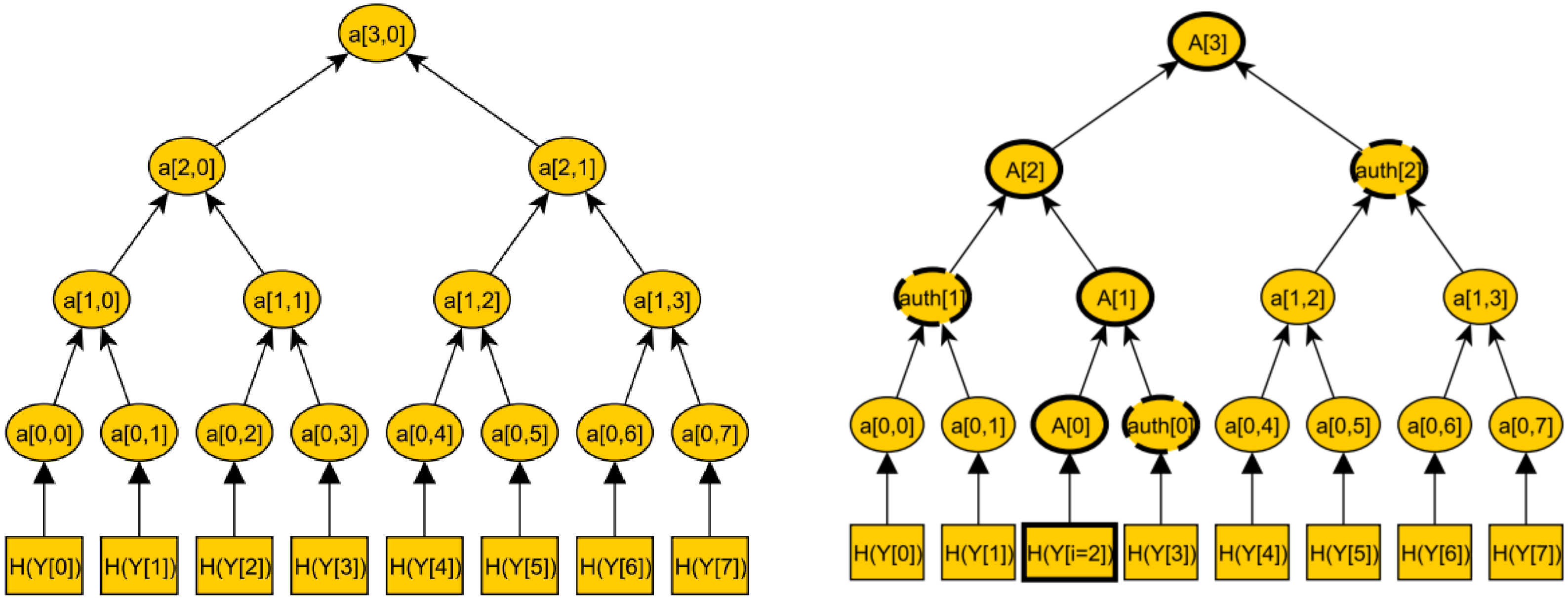
- 缺点：伪造w_i-1的签名很难，但是伪造w_i+1的签名容易
- 解决方法：将16-w_i进行签名，作为校验码。

WOTS$^+$: 在签名中引入随机盐值

$$F^b(x)=\begin{cases} x, b=0 \\ F^{b-1}(F(x)+r_b), b>0 \end{cases}$$

Merkle signature scheme

- 多次签名算法（不是任意次）



Merkle signature scheme

- 密钥产生：使用一个随机数生成器和一个随机种子，产生2^h个一次性密钥， 构造一个高度为h的二叉树，树的叶子节点是2^h个一次性签名的公钥。
- 签名：给一个消息签名时，从没有用过的一次性密钥中随机选择一个，用该密钥给消息签名，然后找到该公钥到根节点的路径，纪录路径上每个节点的兄弟节点的值。与一次性签名一起发布。
- 验签： 接收者根据消息和签名，验证签名及路径。

Chain-based Signature （有状态的）

签名：$(pk_{i+1}, \sigma_i, \{m_j, pk_{j+1}, \sigma_j\}_{j=1}^{i-1})$

验签 $\mathsf{Vrfy}_{pk_j}(m_j \| pk_{j+1}, \sigma_j) \stackrel{?}{=} 1$ for all $j \in \{1, \ldots, i\}$.

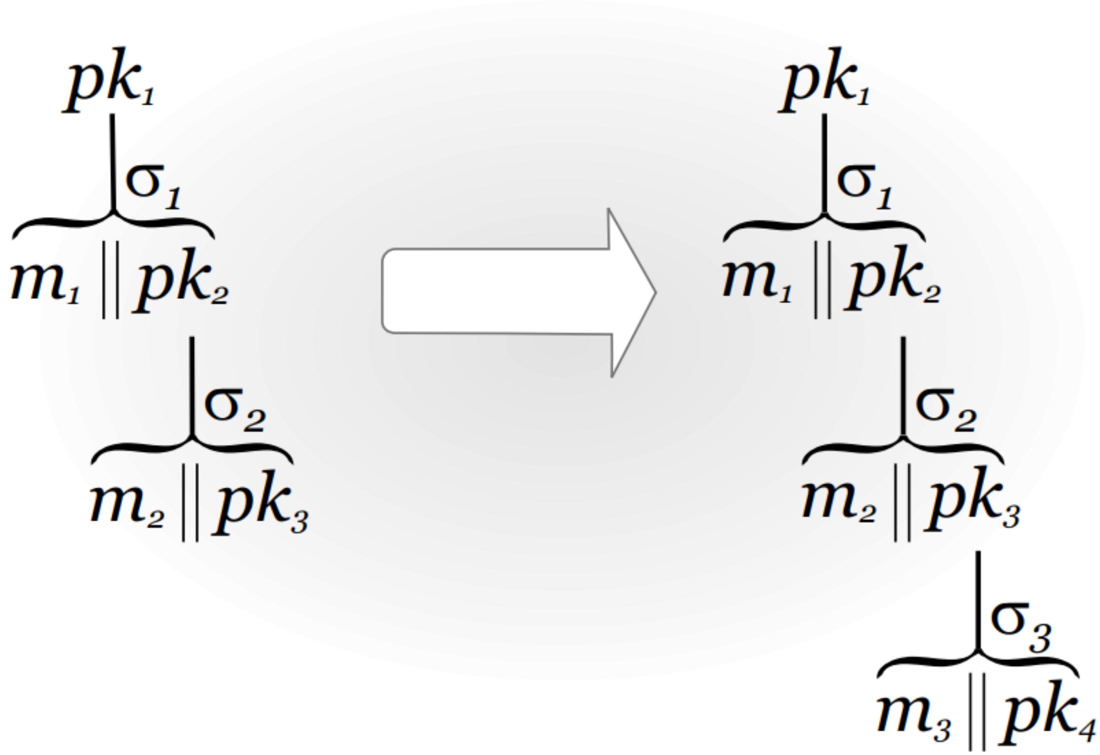
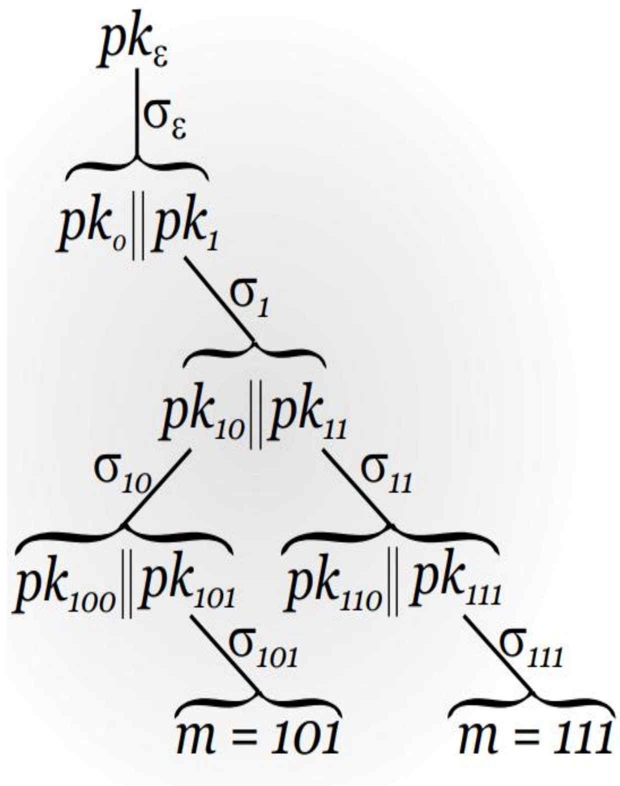


FIGURE 12.4: Chain-based signatures: the situation before and after signing the third message m_3.

Goldreich树签名

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. For a binary string m, let $m|_i \stackrel{\text{def}}{=} m_1 \cdots m_i$ denote the i-bit prefix of m (with $m|_0 \stackrel{\text{def}}{=} \varepsilon$, the empty string). Construct the scheme $\Pi^* = (\mathsf{Gen}^*, \mathsf{Sign}^*, \mathsf{Vrfy}^*)$ as follows:

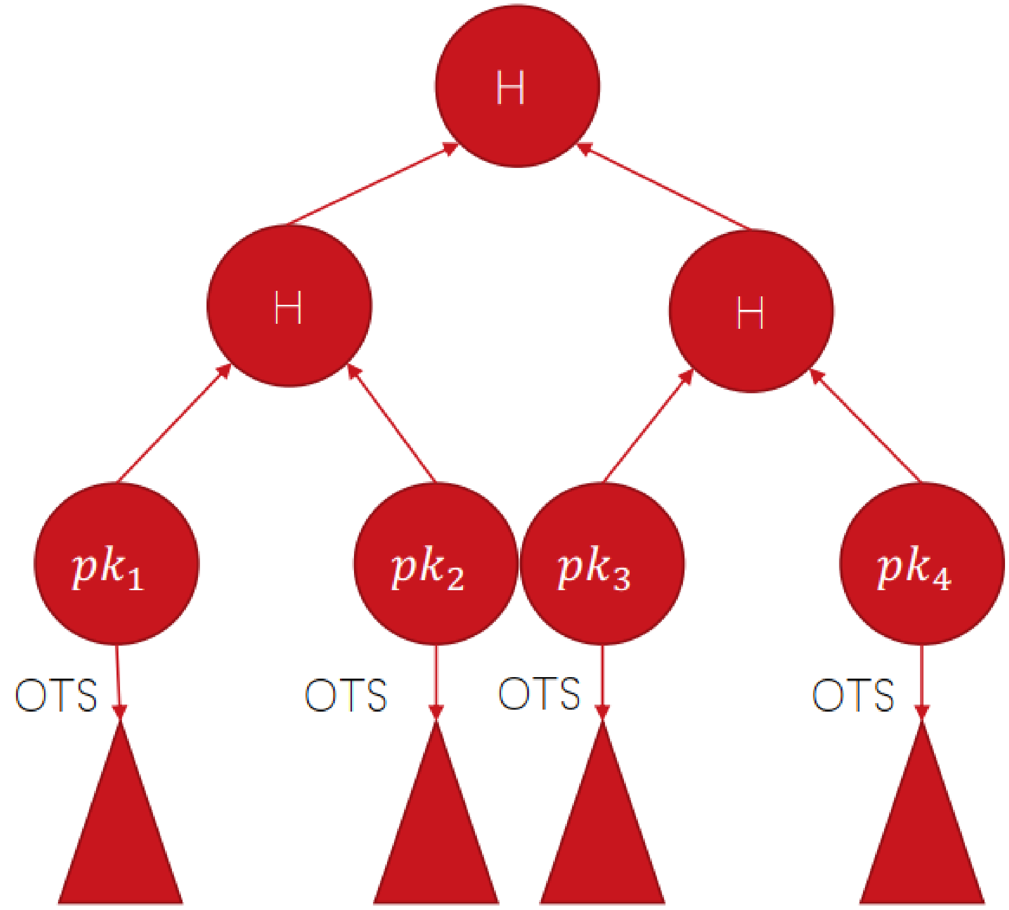
- Gen^*: on input 1^n, compute $(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(1^n)$ and output the public key pk_ε. The private key and initial state are sk_ε.
- Sign^*: on input a message $m \in \{0,1\}^n$, carry out the following.
  1. For $i = 0$ to $n-1$:
     - If $pk_{m|_i0}, pk_{m|_i1}$, and $\sigma_{m|_i}$ are not in the state, compute $(pk_{m|_i0}, sk_{m|_i0}) \leftarrow \mathsf{Gen}(1^n)$, $(pk_{m|_i1}, sk_{m|_i1}) \leftarrow \mathsf{Gen}(1^n)$, and $\sigma_{m|_i} \leftarrow \mathsf{Sign}_{sk_{m|_i}}(pk_{m|_i0} \,\|\, pk_{m|_i1})$. In addition, add all of these values to the state.
  2. If σ_m is not yet included in the state, compute $\sigma_m \leftarrow \mathsf{Sign}_{sk_m}(m)$ and store it as part of the state.
  3. Output the signature $\left(\{\sigma_{m|_i}, pk_{m|_i0}, pk_{m|_i1}\}_{i=0}^{n-1}, \sigma_m\right)$.
- Vrfy^*: on input a public key pk_ε, message m, and signature $\left(\{\sigma_{m|_i}, pk_{m|_i0}, pk_{m|_i1}\}_{i=0}^{n-1}, \sigma_m\right)$, output 1 if and only if:
  1. $\mathsf{Vrfy}_{pk_{m|_i}}(pk_{m|_i0} \,\|\, pk_{m|_i1}, \sigma_{m|_i}) \stackrel{?}{=} 1$ for all $i \in \{0, \ldots, n-1\}$.
  2. $\mathsf{Vrfy}_{pk_m}(m, \sigma_m) \stackrel{?}{=} 1$.



超树（Hyper tree ）

- [Gol04]结构的推广
- [Gol04]： 签名一次认证2个OTS
- Merkle树: 一次认证 2^h个OTS
- 叶节点的一次签名认证其他Mekle树

- 签名大小和时间的取舍
- 减轻树高，增加时间



少次签名（FORS）

- 一次签名的变种（1次->少数次）
- 多个Merkle树，每个树认证 a 比特，消息分成多个a比特
- 敌手伪造 m的签名：如果 $H(m)$ 可以被此前的签名覆盖（概率可量化分析）

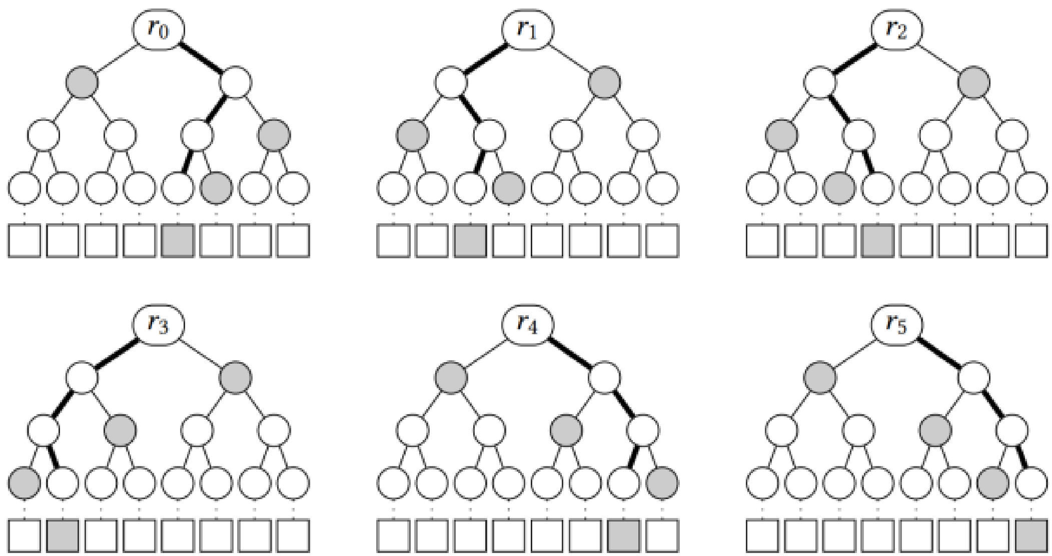


Figure 3: An illustration of a FORS signature with k = 6 and a = 3, for the message `100 010 011 001 110 111`.

SPHINCS+ 全景图

- SPHINCS+组合了： Winternitz一次签名、超树、少次签名等

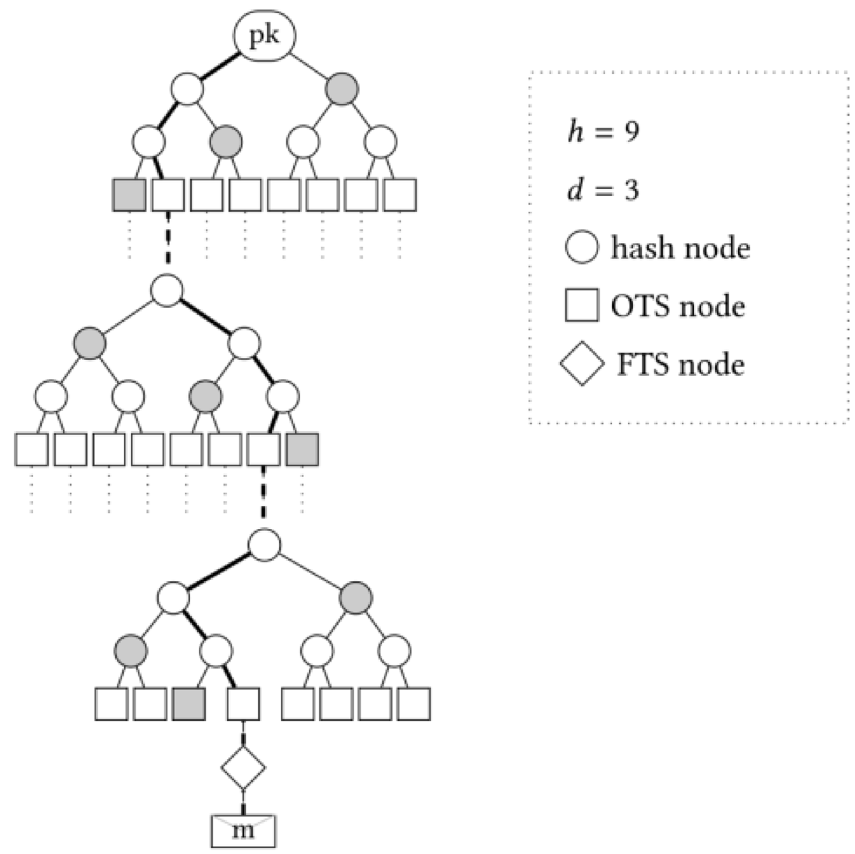
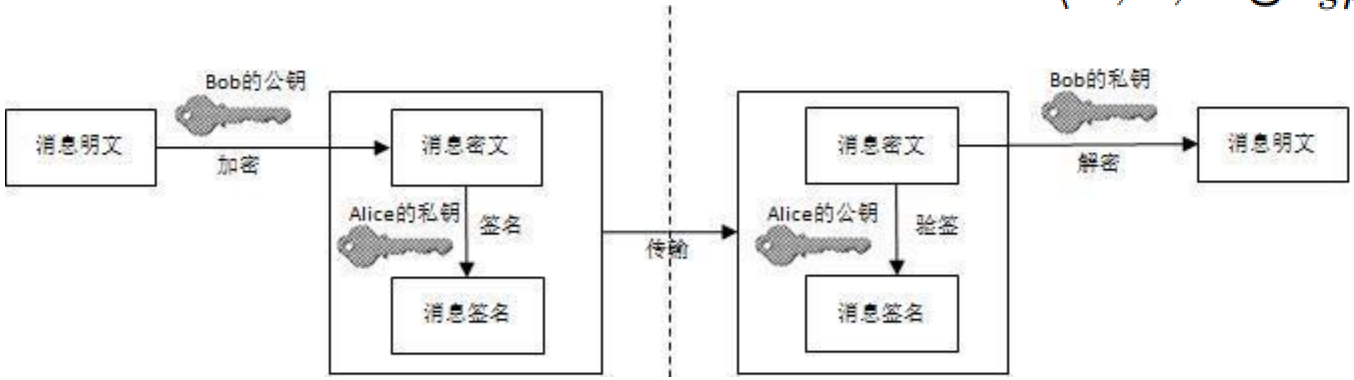


Figure 1: An illustration of a (small) SPHINCS structure.

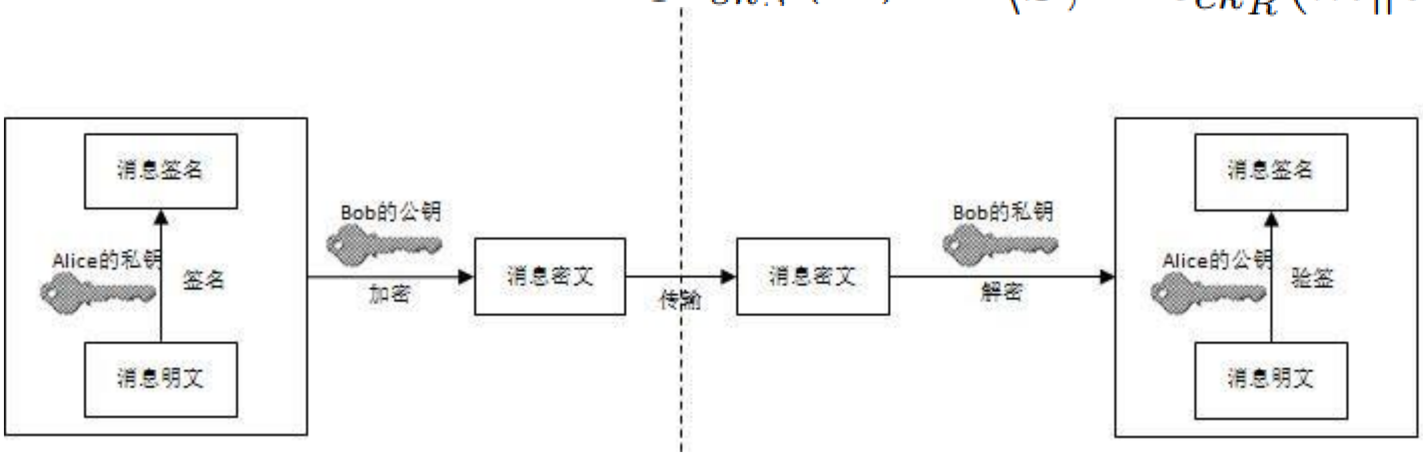
签密

- 先加密后签名？ S发送给R：$\langle S, c, \mathsf{Sign}_{sk_S}(c)\rangle$

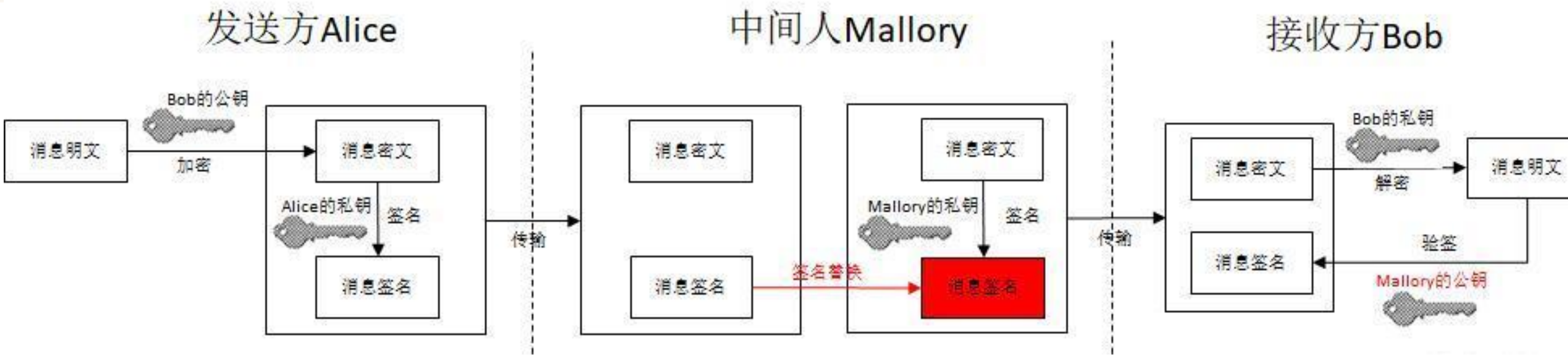


- 先签名后加密？

$$\sigma \leftarrow \mathsf{Sign}_{sk_S}(m) \qquad \langle S, \mathsf{Enc}_{ek_R}(m\|\sigma)\rangle.$$



签密



解决方案

$$\sigma \leftarrow Sign_{SKs}(m \| R)$$

$$Send < S, Enc_{ekR}(S \| m \| \sigma) >$$

谢谢！