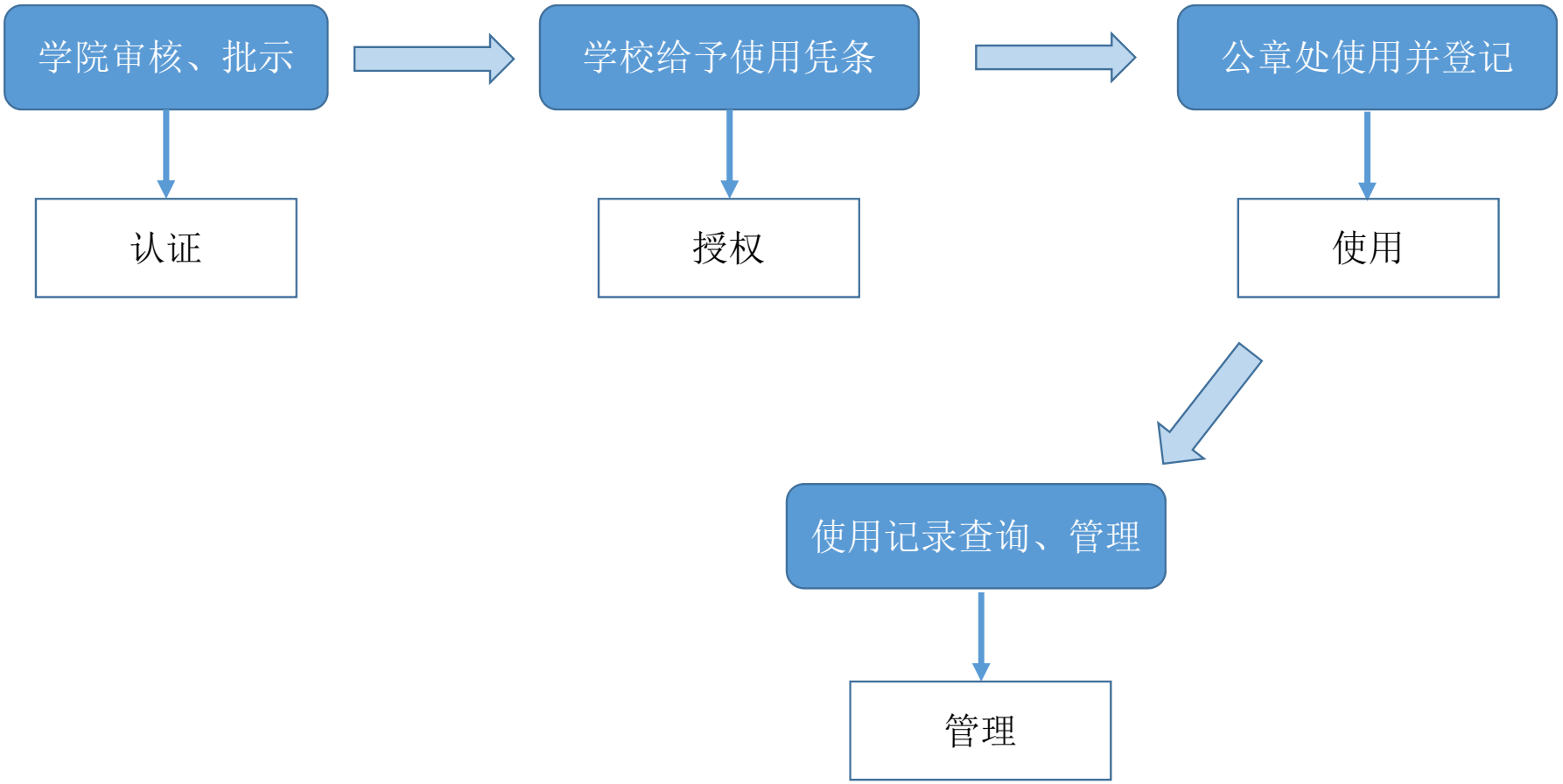
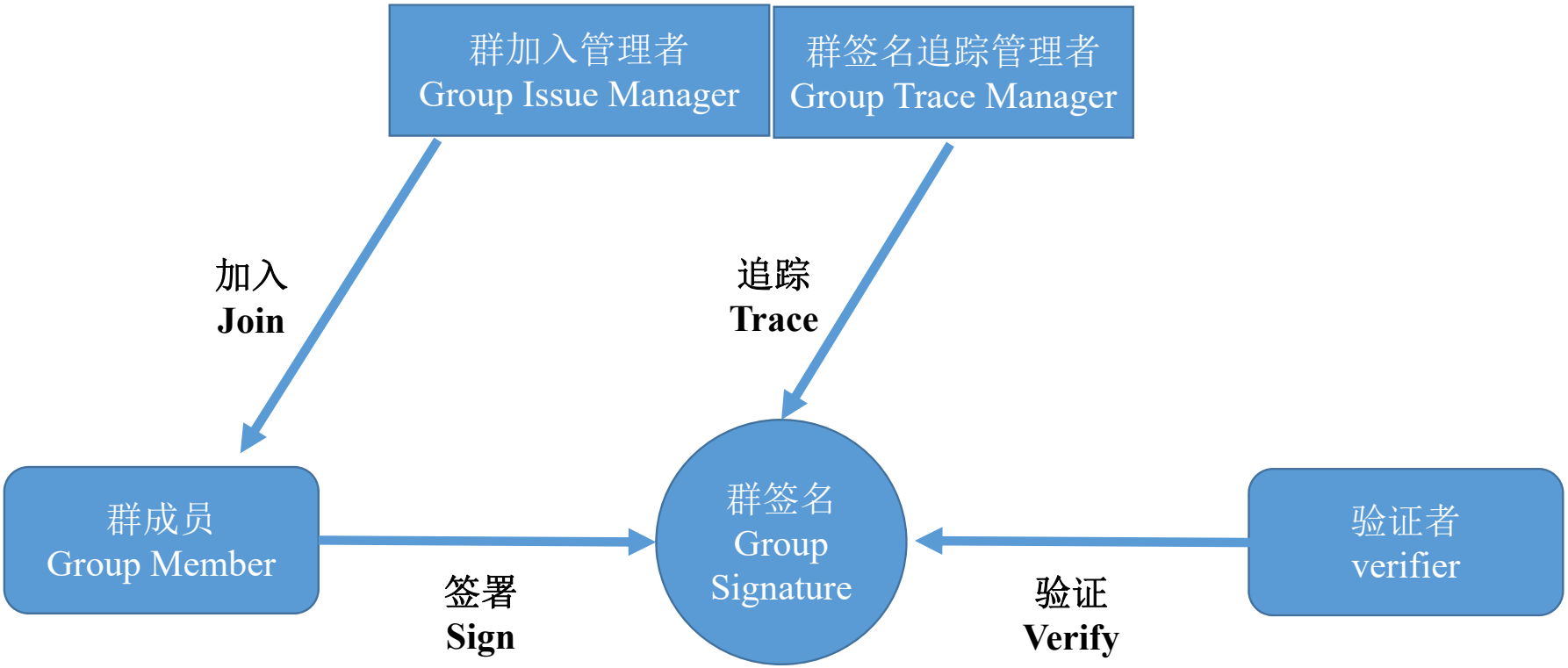
第九章、群签名及其在区块链中的应用

目　录

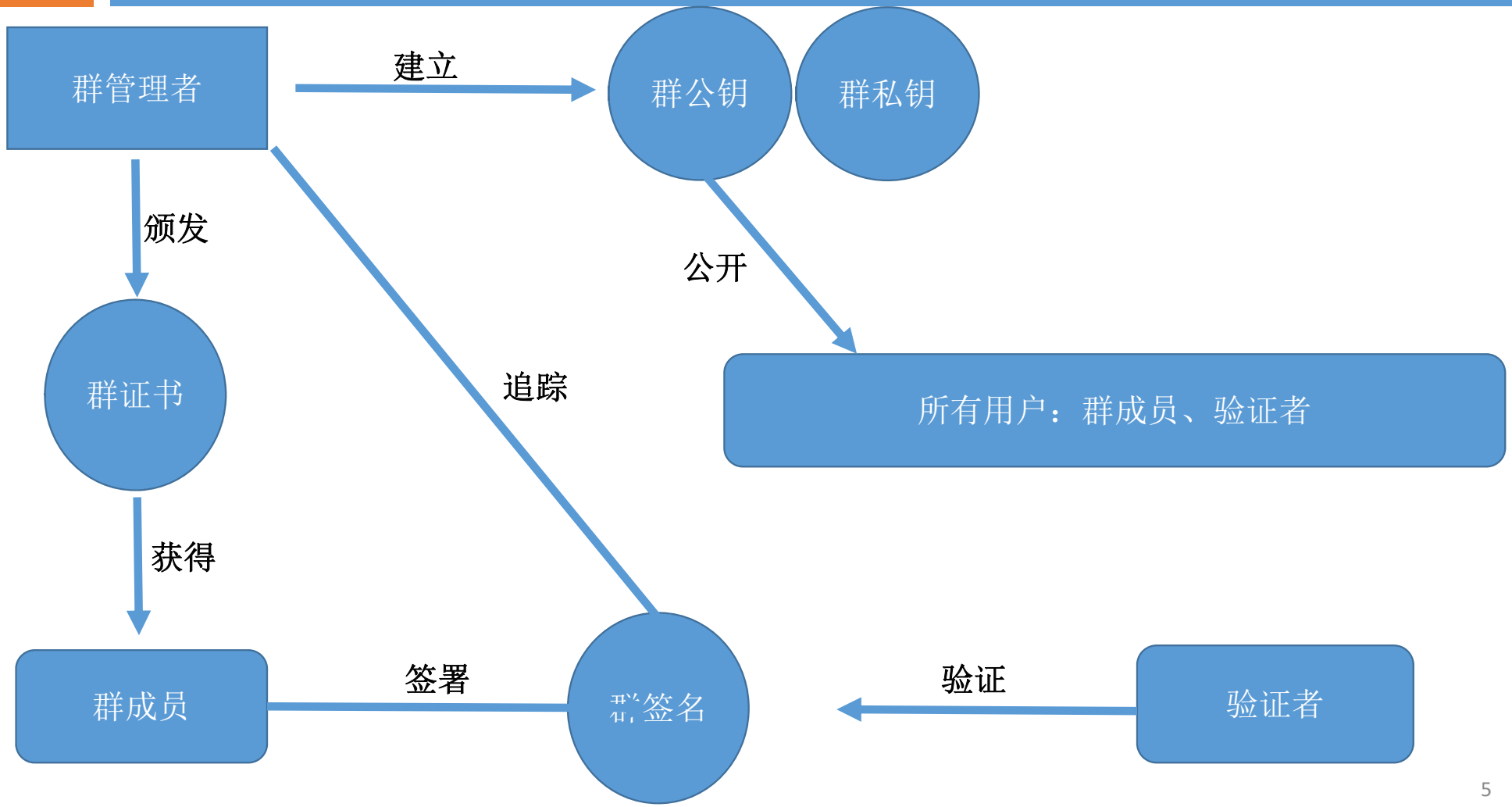
9.1 群签名概述



9.1 群签名概述


群加入管理者
Group Issue Manager
群签名追踪管理者
Group Trace Manager
加入
Join
追踪
Trace
群成员
Group Member
群签名
Group
Signature
验证者
verifier
签署
Sign
验证
Verify

9.1 群签名概述



9.1 群签名概述

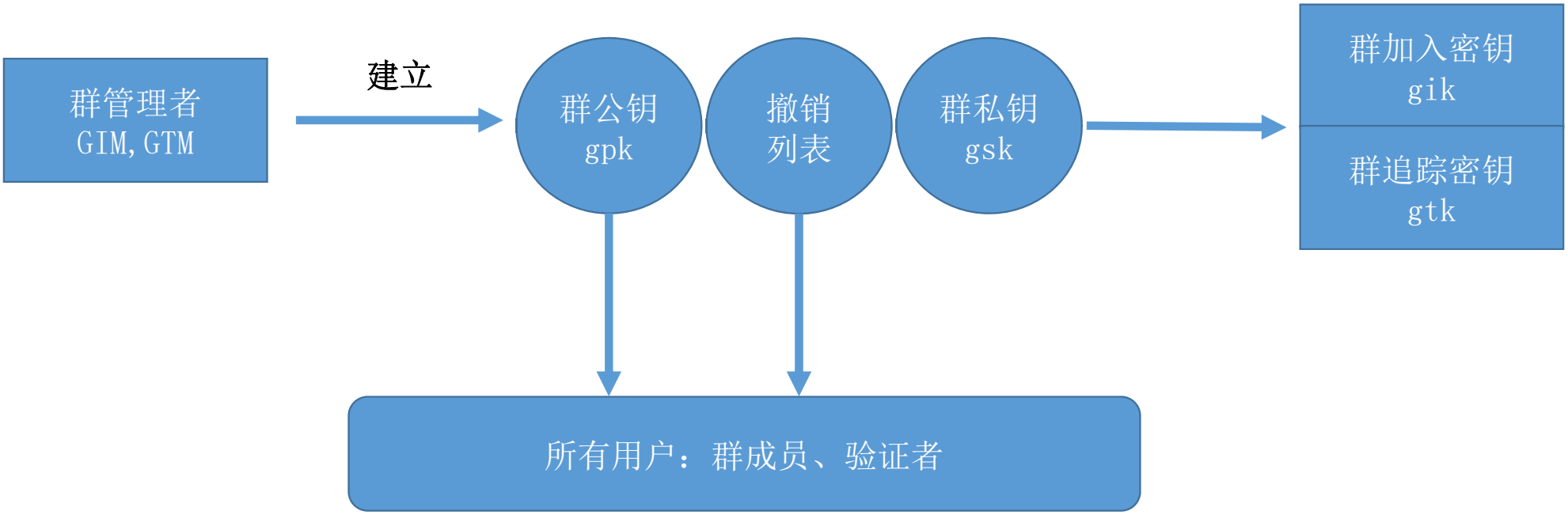
1. 群签名的形式化定义

群签名方案流程：

- 系统建立 $Setup$
- 加入 $Enroll(GIM, U_i)$
- 签名 $Sign(gpk, usk_i, m)$
- 验证 $Verify(gpk, RL, m, s)$
- 追踪 $Open(gtk, Y, m, s)$
- 撤销 $Revoke(grk, U_i)$

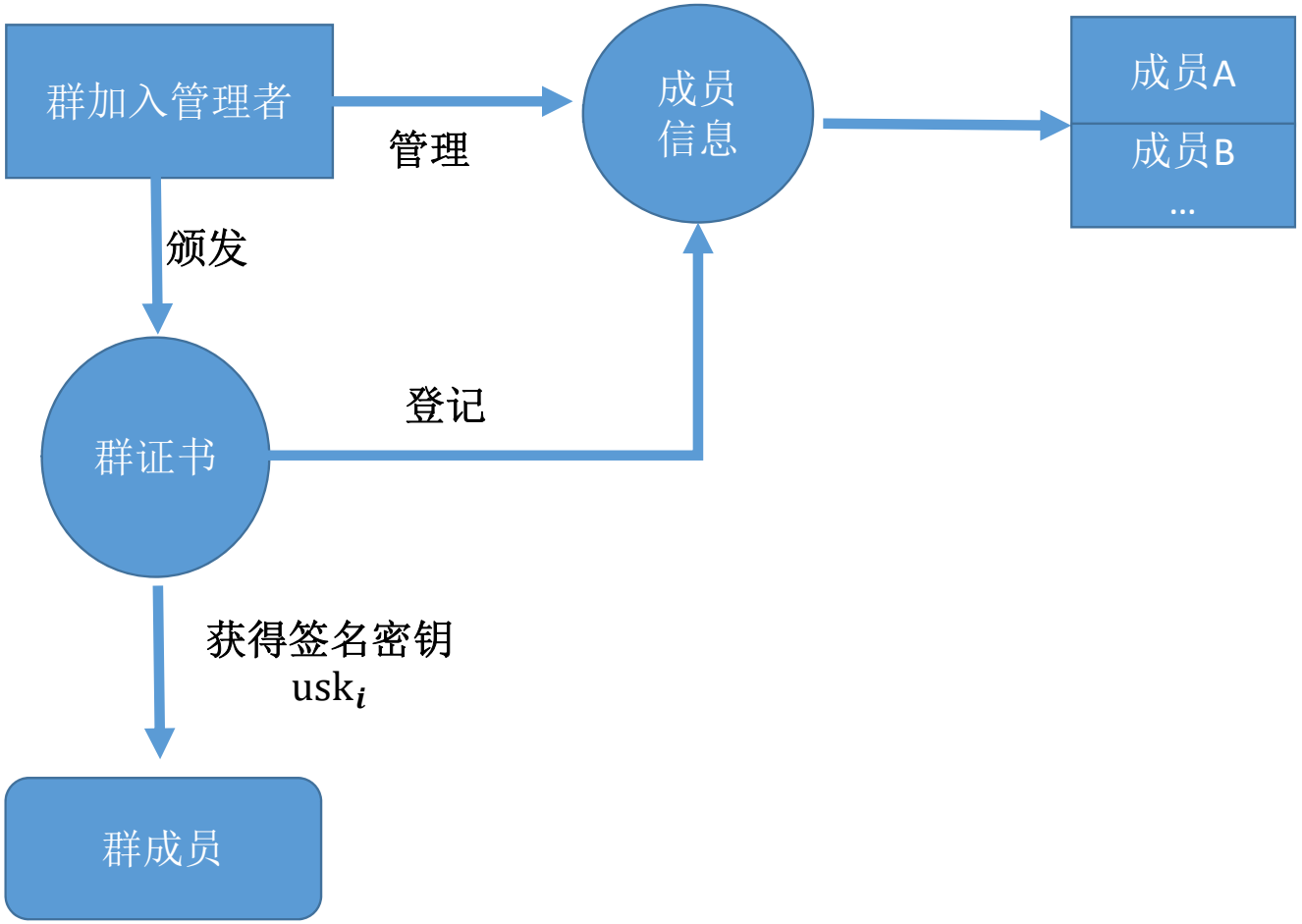
9.1 群签名概述

- **第一步，系统建立 *Setup***



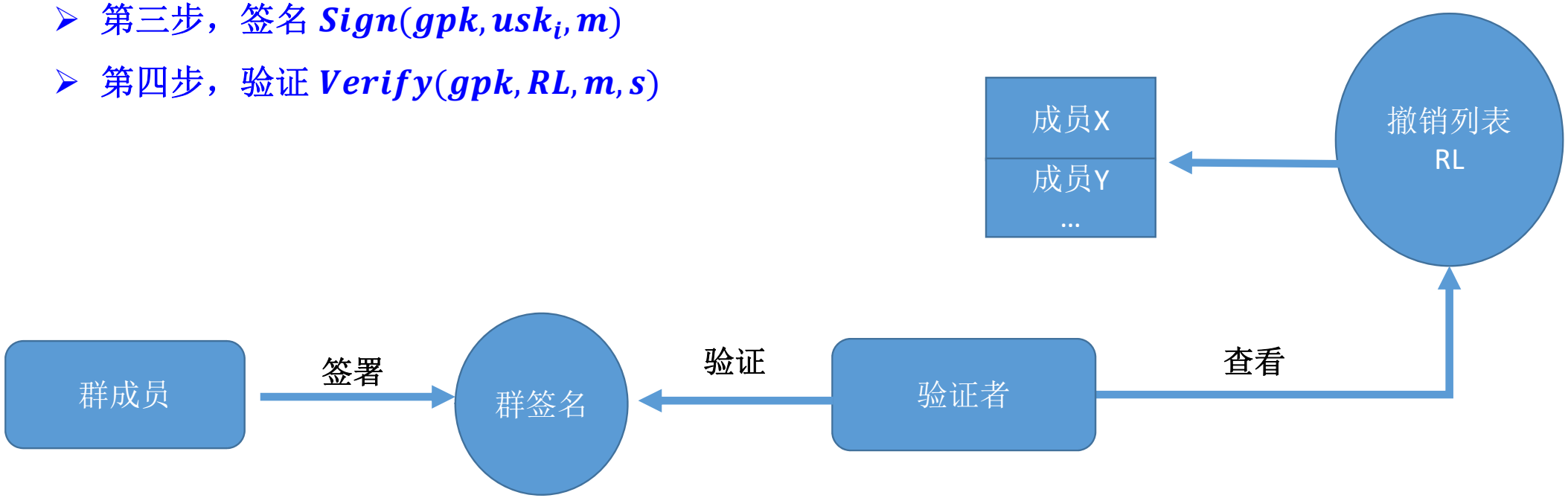
9.1 群签名概述

➢ 第二步，加入 $Enroll(GIM,U_i)$



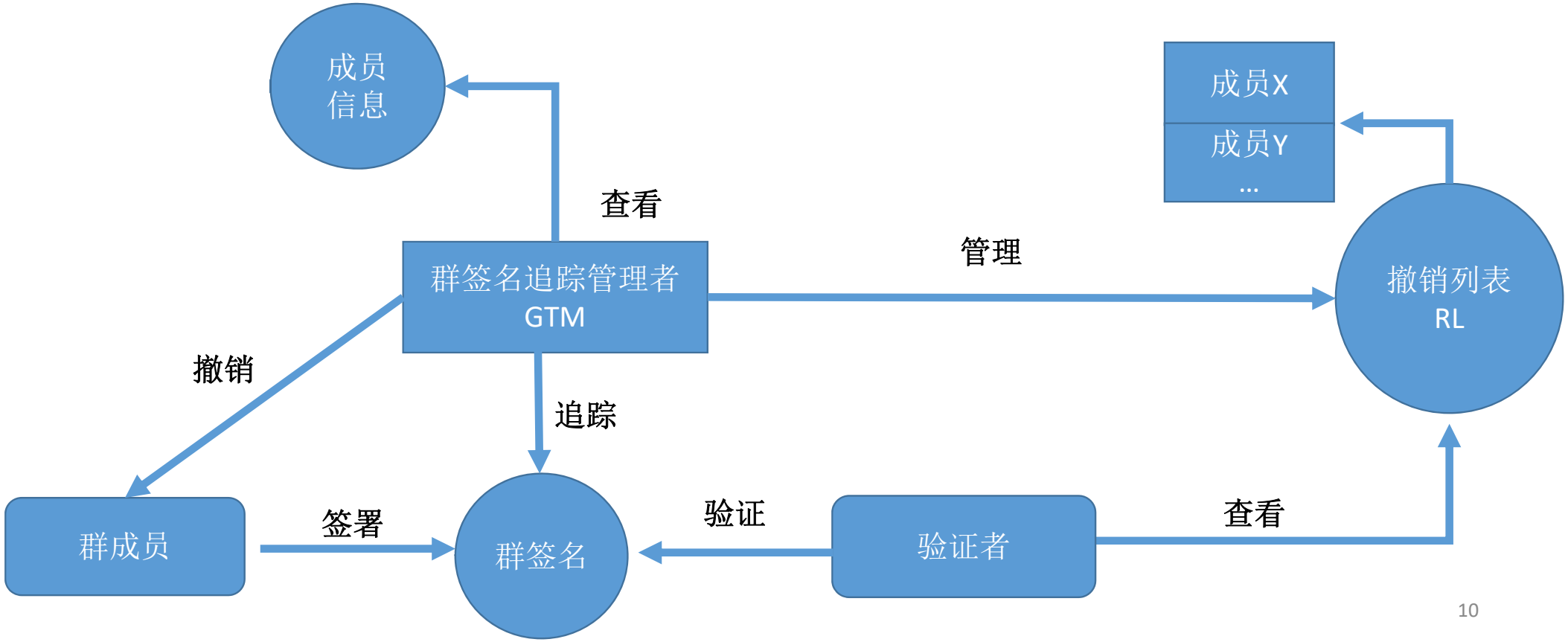
9.1 群签名概述

- 第三步，签名 $Sign(gpk, usk_i, m)$
- 第四步，验证 $Verify(gpk, RL, m, s)$



9.1 群签名概述

- 第五步，追踪 $Open(gtk, Y, m, s)$
- 第六步，撤销 $Revoke(grk, U_i)$



9.1 群签名概述

群签名方案的安全性要求：

① 正确性

② 不可伪造性

③ 匿名性

④ 可追踪性

⑤ 不可连接性

⑥ 不可替代性

⑦ 防陷害攻击

⑧ 防联合攻击

9.1 群签名概述

正确性Correctness. 方案的正确性是保证整个系统的一致性，保证合法群用户签署出的群签名能被正确验证，并且该群签名被追踪到原始签署者。用更清楚的语言来说，正确性保证如下三个方面：

1. 登记的群成员信息列表总是一致的。比如在群成员信息Y_i中包含用户个人的公钥以及对于私钥的拥有证明，那么这些关系都应该一直是正确有效的。
2. 只要群成员没有被群管理者撤销，那么任何由签名算法正确产生的群签名都必须有效，即

$$Verify(gpk, \mathbf{RL}, m, Sign(gpk, usk_i, m)) = 1.$$

3. 任何正确产生的群签名必须被追踪到原始签署者，即

$$Open(grk, \mathbf{Y}, m, Sign(gpk, usk_i, m)) = U_i.$$

9.1 群签名概述

- **不可伪造性**：只有获得群成员证书和签名密钥的群成员才能够生成合法的群签名。
- **匿名性**：接收签名的人只能验证签名的合法性但不能判定生成签名的群成员的身份，即使群中其它的群成员也不能判定。
- **可跟踪性**：当需要揭示签名群成员的身份时，有且仅有管理员可以打开签名，找到签名的群成员。
- **不可链接性**：对于没有打开的签名，要判断两个签名是否由同一个群成员签署，是计算上不可行的。
- **防陷害攻击**：即使别的群成员包括群管理员联合在一起，也不能代替另一个群成员签名。
- **防联合攻击**：即使群成员联合在一起，也不能产生一个无法跟踪的群签名。

9.1 群签名概述

群签名的效率：

① 群公钥的大小

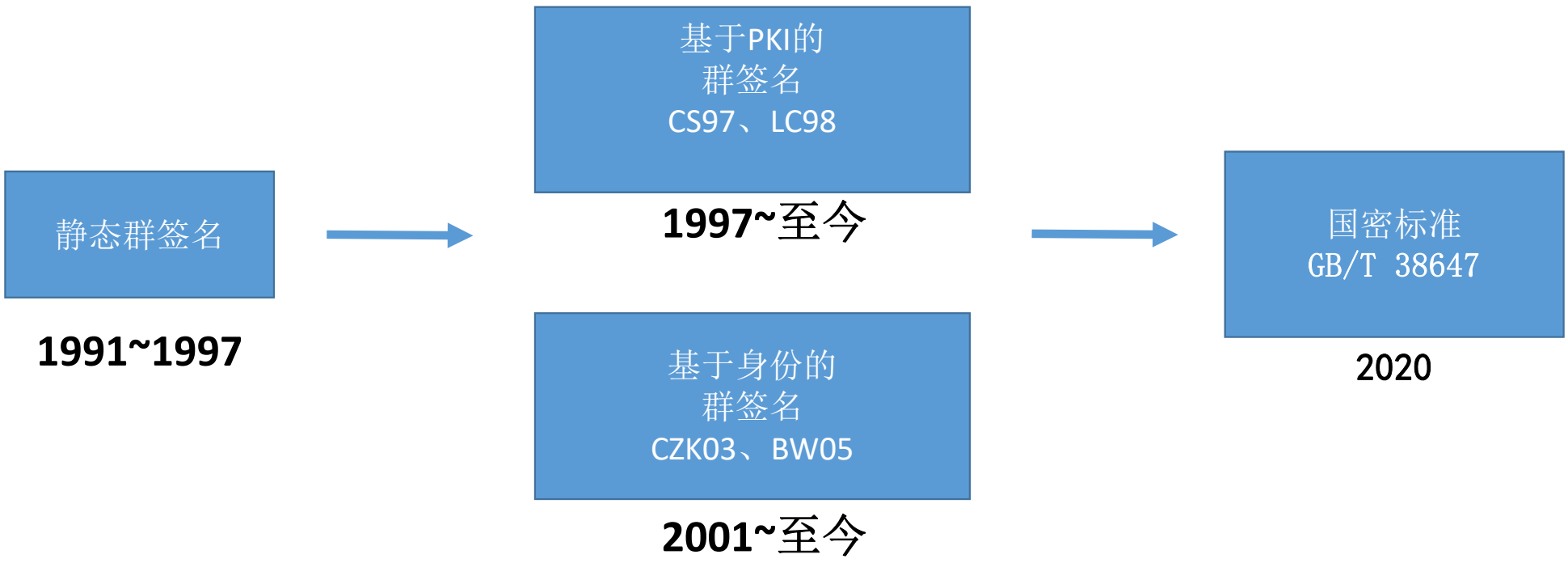
② 群签名的长度

③ 群签名算法和验证算法的效率

④ 创建算法，加入以及打开算法的效率

9.1 群签名概述

- 群签名方案的发展：



9.1 群签名概述

- 群签名应用：电子商务、电子银行、电子投票、电子拍卖等

9.2.1 LC98群签名方案

目　录

9.2.1 LC98群签名方案

LC98群签名流程

1) Setup():

群管理者执行:

- 选择两个大素数p和q，且$q|p-1$
- 选择$GF(p)$中阶为q的生成元g
- 随机选择$x_T \in [1, q-1]$为群的主私钥，
- 计算$y_T = g^{x_T} \bmod p$，
- 群管理者公钥为$Y = (p, q, g, y_T)$。

9.2.1 LC98群签名方案

2) Enroll():

群成员U_i

- 随机选择$x_i \in [1, q-1]$
- 计算$y_i = g^{x_i} \bmod p$

y_i

$(r_i,\ s_i)$

群管理员

- 随机选择$k_i \in [1, q-1]$
- 计算:

$r_i = g^{-k_i} {y_i}^{k_i} \ mod\ p$

$s_i = k_i - r_i x_T \ mod\ p$

- 验证以下等式是否成立:

$$g^{s_i} y_T^{r_i} r_i = {y_i}^{s_i} y_T^{r_i x_i} \ mod\ p$$

- 若成立，则将$(r_i,\ s_i)$作为自己的群证书

9.2.1 LC98群签名方案

3) Sign()

群成员U_i

- 计算$\alpha_i = y_T^{r_i} g^{s_i} \bmod p$
- 随机选择t ∈ $[1, q-1]$
- 计算$r = \alpha_i{}^t \ mod\ p$
- 计算s, 由等式:

$$h(m) = rx_i + ts \ mod\ q$$

推导出.

- (r, s)为群成员对消息m的签名

$(r_i,\ s_i)$

(r, s)

4) Verify()

接收者

- 计算:

$$\alpha_i = y_T^{r_i} g^{s_i} \bmod p$$

$$DH_i = \alpha_i r_i \ mod\ p$$

- 验证以下等式是否成立:

$$\alpha_i^{h(m)} = r^s DH_i{}^r \ mod\ p$$

若成立，则为合法签名

9.2.1 LC98群签名方案

5) Open():

由于群管理者保存了用户U_i的身份，证书(r_i, s_i)，参数(y_i, k_i),则可以根据证书判断签名的群成员身份

9.2.1 LC98群签名方案

LC98方案安全性是基于离散对数问题，与最早需要使用知识签名的群签名方案CS97相比，LC98方案大大提高了效率，减小了通信量，但是安全性存在着以下缺陷：

- 由于签名结果包含成员证书（r_i,s_i）,同一群成员的签名具有相同的（r_i,s_i）部分，签名具有可连接性;
- 可以通过“参数选取法”伪造签名

9.2.1 LC98群签名方案

“参数选取法”

(1) 选择随机数 v,w,v_i,w_i;

(2) 计算 $r = y_T^v g^w \bmod p$ 和 $r_i = y_T^{v_i} g^{w_i} \bmod p$;

(3) 为了满足验证等式(3-2-1)，须从从下面等式中分别解出 s 和 s_i

$$r_i\ h(m) = vs + (r_i + v_i)r \mod q$$

$$s_i\ h(m) = ws + (s_i + w_i)r \mod q$$

(4) 则对于消息 m 的 LC98 签名是(r, s, r_i, s_i)。

9.2.1 LC98群签名方案

证明：

将(r, s, r_i, s_i)代入等式分别有：

$$\alpha_i^{h(m)} = (y_T^{r_i} g^{s_i})^{h(m)} \mod p$$

$$r^s DH_i^r = (y_T^v g^w)^s \left(y_T^{r_i} g^{s_i} y_T^{v_i} g^{w_i}\right)^r = y_T^{(r_i+v_i)r+vs} g^{(s_i+w_i)r+ws} \mod p$$

即伪造的签名能 通过验证等式

$$\alpha_i^{h(m)} = r^s DH_i^r \mod p$$

9.2.1 LC98群签名方案

LC98方案之后，又有很多人提出了针对该方案的改良，我们称之为类LC98方案，但是几乎所有的类LC98方案都可以通过“参数选取法”进行伪造。

总结而言，判断一个类LC98方案是否安全，要检查其是否解决了α_i和DH_i的合法性验证问题。

9.2.1 LC98群签名方案

要保证α_i和DH_i的合法性,也可以要求签名人在群管理员的辅助下生成签名,或者由管理员为每一个群成员生成并公布一系列(α_i, DH_i)二元组,每次签名使用一个(α_i, DH_i)二元组,以便验证者检验(α_i, DH_i)的合法性。但是这些做法是以牺牲群管理员的效率为代价的,同时也违背了群签名由群成员代替群进行签名的初衷。

9.2.2 GB/T 38647 采用群组公钥的机制

目　录

9.2.2 GB/T 38647 采用群组公钥的机制

根据2020年4月28日国家市场监督管理总局、国家标准化管理委员会发布的中华人民共和国国家标准公告（2020年第8号），全国信息安全标准化技术委员会归口的GB/T 20281-2020《信息安全技术 防火墙安全技术要求和测试评价方法》等26项国家标准正式发布（其中7项密码标准）

21	GB/T 38647.1-2020	信息技术 安全技术 匿名数字签名 第1部分：总则		2020/11/1
22	GB/T 38647.2-2020	信息技术 安全技术 匿名数字签名 第2部分：采用群组公钥的机制		2020/11/1

9.2.2 GB/T 38647 采用群组公钥的机制

术语及定义

辅助签名方 assistant signer
可以帮助主要签名方去创建匿名签名，但不可以独立地产生匿名签名的实体

成员列表 member-list
包含群组成员身份及其对应的群组成员证书的列表。

主要签名方 principal signer
拥有群组成员签名密钥的实体，主签名方可以使用该密钥来创建匿名签名。

密钥种子值 secret seed value
群组成员所知的并且用来导出群组成员私钥的保密数据。

安全参数 security parameter
决定一个机制安全强度的变量。

9.2.2 GB/T 38647 采用群组公钥的机制

符号

bsn	连接基，既可以是一个特殊符号⊥也可以是一个任意字符串
e	一个双线性映射函数$e: G1 \times G2 \rightarrow GT$ 使得对所有$P \in G1, Q \in G2$, 所有正整数a, b,该等式$e([a]P, [b]Q) = e(P, Q)^{ab}$ 成立。该函数也叫做对函数
$\gcd(a, b)$	整数a 和b的最大公因子
$G1$	一个椭圆曲线上的阶为p 的加法循环群
$G2$	一个椭圆曲线上的阶为p 的加法循环群
GT	一个阶为p 的乘法循环群
H	密码杂凑函数
m	未签消息
n	一个RSA模块，其中$n = pq$
OE	椭圆曲线上的无穷大点
P	一个素数
$P1$	$G1$的生成元
$P2$	$G2$的生成元
q	素数
$Q1+Q2$	椭圆曲线点 $Q1$ 和 $Q2$ 的总和

9.2.2 GB/T 38647 采用群组公钥的机制

符号

QR(n)	模n的二次剩余群组
Zn^*	Zn 中可逆元素的乘法群
Zp	[0, p-1]中整数的集合
Zp^*	[1, p-1]中整数的集合
$(a\|p)$	a 和p的勒让德符号，其中 a 是一个整数p是一个偶素数
$[n]P$	一个正整数和一个在椭圆曲线E上的点P之间的乘法运算，n 和P作为输入并产生另外椭圆曲线E上的点Q ，其中$Q = [n]P = P + P + \ldots + P$, i.e。, n 个P的总和。 该运算满足$[0]P = OE$ 和$[-n]P = [n](-P)$
$[x, y]$	从 x 到 y 之间包括 x 和 y 的整数集合，如果 x, y 是整数满足 $x \le y$
||	Y || Z 表示数据项 Y 和 Z 以特定的顺序级联的结果。在级联两个或多个数据项的结果作为文本中的一个机制的输出的情况下，例如该结果应该被拆分以便可以独立的组成连续的数据串，使得它不可能有模糊不清的解释。后一种性质可以根据具体的应用以不同的方式来呈现。例如，它可以在整个机制的使用时确保一个子字符串的固定长度，或确保使用唯一的编码方式来对级联的字符串序列进行编码。主要的编码规则在 ISO/IEC 8825-1 [1] 中定义

9.2.2 GB/T 38647 采用群组公钥的机制

一般模型和要求

采用群组公钥机制的过程：

1）密钥生成过程；
2）签名过程；
3）验证过程；
4）打开过程（如果机制支持打开能力）；
5）连接过程（如果机制支持连接能力）；
6）撤销过程。

9.2.2 GB/T 38647 采用群组公钥的机制

具有打开、链接、撤销功能的机制

下列符号适用于本机制：

- Q, Q_1, Q_2, U, W, D, V, A, Z, WID, Q_1', Q_2', U', W', D', $\tilde{A}$, D_1, D_2, D_3, R_1, R_2, R_3, K_{open}, W_{open}, V_{open}, $Y_{1,i}$, $Y_{2,j}$, $X_{1,i}$, $X_{2,i}$, $S_{1,j}$, $S_{2,j}$, $S_{3,j}$, $S_{4,j}$, $S_{5,j}$：$G1$中的元素。
- B_1, B_θ: $G2$中的元素。
- L_1, L_2, L_3, L_4, L_A, L_1', L_2', L_3', L_4': GT中的元素。
- η, ξ, θ, x, y, z, r_{ID}, c_{ID}, s_{ID}, α, γ, r_α, r_x, r_γ, r_y, c, s_α, s_x, s_γ, s_y, c_{open}, s_{open}, x_1, x_2, x_j, υ_j：Z_p^*中的整数。
- i, j, λ, ρ, κ：长度为t bit的整数，这里t是一个固定的非负整数。
- Hp：输出元素在Zp^*内的密码杂凑函数。

9. 2.1 GB/T 38647 采用群组公钥的机制

具有打开、链接、撤销功能的机制

该机制流程：

- 密钥产生过程（包括建立过程和群组成员发布过程）
- 签名过程
- 验证过程
- 打开过程
- 连接过程
- 撤销过程。

其中撤销过程包括：

- 撤销列表RL
- 更新群公钥gpk
- 更新用户私钥usk。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

密钥产生过程由两部分组成：建立过程和群组成员发布过程。

建立过程输入安全参数并产生群组公钥*gpk*和它对应的群组成员发布密钥*gmik*，群组成员打开密钥*gmok*和群组签名连接密钥*gslk*。

产生群组公共参数如下：

a）产生三个阶为p 的素数群组G_1, G_2, G_T 和一个双线性映射 $e: G_1 \times G_2 \rightarrow G_T$。假设群组是乘法群

b）选取$B_1 \leftarrow G_2$ 和 Q_1, Q_2, Q, $U \leftarrow G_1$。

c）选取一个密码密码杂凑函数H_p: $\{0, 1\}^* \rightarrow Z_p^*$, 其中 $H_p(M)$ 是消息$M \in \{0, 1\}^*$的散列码 。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

产生密钥如下：

a）选取 $\eta, \xi, \theta \leftarrow Z_p^*$。

b）计算 $W = [\eta]U$, $D = [\xi]U$, $B_\theta = [\theta]B_1$, $V = [\xi]B_1$。

c）计算 $L_1 = e(W, B_1)$, $L_2 = e(W, B_\theta)$, $L_3 = e(Q_1, B_1)$和$L_4 = e(Q_2, B_1)$。

d）输出群组公共参数：$((e, G_1, G_2, G_T), Q, B_1, B_\theta, H_p)$。

e）输出初始群组公钥：$gpk = (Q_1, Q_2, U, W, D, (L_1, L_2, L_3, L_4))$。

f）输出 :群组成员发布方私钥$gmik = \theta$, ，群组成员打开方私钥 $gmok = (\eta, \xi)$ 和群组签名连接方私钥$gslk = (V)$

注：L_1, L_2, L_3, 和 L_4 在 gpk内是可选的, 因为他们可以由签名方和验证方通过$Q_1, Q_2, B_1, B_\theta, W$ 计算得出。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

管理密钥gmik, gmok,和 gslk 分别由群组成员发布方，群组成员打开方和群组签名连接方安全存储。

群组成员发布过程如下：群组成员发布方管理一个成员列表LIST=(LIST[1],…,LIST[n])，其中n是目前已注册群组成员的数量。列表的每一条记录包含每一个注册用户的个人信息。两个子过程，用户加入 (由加入用户使用其身份ID进行) 和发布 (由群组成员发布方运行) 交互产生一个群组成员签名密钥如下。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

假设两个子过程通过一个安全的鉴别信道，产生一个群组签名密钥过程如下

用户加入过程如下：

a）选取一个随机群组成员私钥$sk = z \leftarrow Z_p^*$并计算$Z = [z]W$。

b）选取$r_{ID} \leftarrow Z_p^*$并计算$W_{ID} = [r_{ID}]W$。

c）选取$c_{ID} = H_p(ID \| W \| Z \| W_{ID})$。

d）计算$s_{ID} = r_{ID} + c_{ID} z \pmod p$。

e）$T_{ID} = (Z, s_{ID}, c_{ID})$。

f）发送$(Join_Request, ID, T_{ID})$给发布过程

（发布过程执行完之后再继续执行）

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

群成员发布过程如下：

a）接收一个加入请求消息 (*Join_Request*, *ID*, T_{ID})。

b）检查消息的合法性如下：

1）检查是否 *ID* 合法。

2）检查是否 $c_{ID} = H_p(ID \| W \| Z \| [s_{ID}]W\text{-}[c_{ID}]Z)$。

c）检查 LIST[*i*]=(*ID*, …)中的*i*的*ID*是否包含在成员列表LIST= (LIST[1],…, LIST[*n*])中。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程群成员发布过程如下：

d）如果 *ID* 没有被注册，例如，没有匹配的*i*存在，那么进程如下：

1）选取 $x, y \leftarrow Z_p^*$。

2）计算$A = [1/(\theta+x)](Q_1-[y]Q_2-Z)$ $(=[1/(\theta+x)](Q_1-[y]Q_2-[z]W))$。

3）增加LIST[$n+1$] = (ID, $[y]Q$, A, x, y, $upk[i] = Z(=[z]W)$, T_{ID})到LIST。

否则

1) 选取 $x \leftarrow Z_p^*$。

2) 计算$A = [1/(\vartheta+x)](Q_1-[y]Q_2-Z)$ $(=[1/(\vartheta+x)](Q_1-[y]Q_2-[z]W))$。

3) 用新的 LIST[i]= (ID, $[y]Q$, A, x, y, $upk[i] = Z$, T_U) 代替之前的LIST[i]。

e) 发送群组成员证书(i, A, x, y) 到用户加入算法。

9. 2.2 GB/T 38647 采用群组公钥的机制

密钥产生过程

用户加入过程如下{接在之前的加入过程 f) 后面}。

g) 接收消息 (i, A, x, y)。

h) 验证是否$e(A, B_{\vartheta}+[x]B_1)= e(Q_1-[y]Q_2-[z]W, B_1)$，其中$Q_1$, Q_2, W 包含在 gpk内。如果等式不成立，则终止。否则, 第i个群组成员签名密钥usk_{i0} = (0, x, y, z, A = $[1/(\vartheta+x)](Q_1-[y]Q_2-[z]W)$)。值0意味着该密钥是由群组成员发布方发布的初始群组成员签名密钥。该密钥将通过撤销更新并且0将根据更新变为相应的指数。

9. 2.2 GB/T 38647 采用群组公钥的机制

签名过程

群组成员签名密钥包括一个指数κ，κ表示该密钥已经进行了κ次撤销列表指数RI（群组签名撤销过程相关）注册更新。设λ 是RI中最新的的撤销密钥数量。为了产生一个签名，群组成员签名密钥需要λ次RI的注册更新。已产生的签名包括一个λ，λ表示签名是使用已进行λ次RI的注册更新的密钥生成的。该签名可以使用已进行λ次RI的注册更新的群组公钥验证。

该签名过程输入群组公共参数 $(e, G_1, G_2, G_T), Q, B_1, B_\vartheta, H_p)$, 群组公钥 gpk_κ , 群组成员签名密钥$usk_{i\kappa} = (\kappa, x, y, z, A)$, 消息$M \in \{0,1\}^*$ ，其中κ ($\leq \lambda$) 是签名方i最后更新的撤销列表指数i，而λ在所有群组成员中最新的撤销列表指数。该过程执行下列步骤：

9. 2.2 GB/T 38647 采用群组公钥的机制

签名过程

a) 需要使用usk_{ik}更新群组签名撤销过程的，并获取$gpk_{\lambda}=(Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))$和一个已更新的群组成员签名密钥$usk_{i\lambda}=(\lambda, x, y, z, \tilde{A})$，其中$\tilde{A}$已经是第$\lambda$次RI的注册更新。

b) 选取$\alpha \leftarrow Z_p^*$。

c) 计算$D_1=[\alpha]U'$，　$D_2=\tilde{A}+[\alpha]W'$，　$D_3=[y]Q+[\alpha]D'$。

d) 计算$\gamma = x_{\alpha} - z \pmod p$。

e) 选取$r_{\alpha}, r_x, r_{\gamma}, r_y \leftarrow Z_p^*$。

f) 计算$R_1=[r_{\alpha}]U'$。

g) 计算$R_2 = e(D_2, B_1)^{r_x} e(W', B_{\vartheta})^{-r_{\alpha}} e(W', B_1)^{-r_{\gamma}} e(Q_2', B_1)^{r_y}$。

h) 计算$R_3=[r_y]Q+[r_{\alpha}]D'$。

i) 计算$c = H_p(M \,||\, \lambda \,||\, D_1 \,||\, D_2 \,||\, D_3 \,||\, R_1 \,||\, R_2 \,||\, R_3)$。

j) 计算$s_{\alpha} = r_{\alpha} + c_{\alpha} \pmod p$, $s_x = r_x + c_x \pmod p$, $s_{\gamma} = r_{\gamma} + c_{\gamma} \pmod p$, $s_y = r_y + c_y \pmod p$。

k) 输出$\sigma = (\lambda, D_1, D_2, D_3, c, s_{\alpha}, s_x, s_{\gamma}, s_y)$。

9. 2.2 GB/T 38647 采用群组公钥的机制

验证过程

验证签名的指数为ρ，使用的群组公钥已经过ρ次RI的注册更新，应注意ρ可能不是RI中的最新的的撤销密钥数量，因为一些密钥可能在之前的验证中被撤销。

群组签名验证过程输入：

群组公共参数$((e, G_1, G_2, G_T), Q, B_1, B_\theta, H_p)$,

$gpk_0 = (Q_1, Q_2, U, W, D, (L_1, L_2, L_3, L_4))$,

签名$\sigma = (\rho, D_1, D_2, D_3, c, s\alpha, sx, s\gamma, sy)$

消息 $M \in \{0,1\}^*$,

9. 2.2 GB/T 38647 采用群组公钥的机制

验证过程

过程执行如下：这里 ρ为在签名产生过程中的撤销指数。

a） 需要使用gpk，(ρ, RL)更新撤销的upk, 并获取$gpk\rho$ =(Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))。

b） 计算$R1 = [s\alpha]U' - [c]D1$,

c） 计算$R2 = e(D_2, B1)^{s}_{x}\, e(W', B\theta)^{-s}_{\alpha}\, e(W', B1)^{-s}_{\gamma}\, e(Q2', B1)^{s}_{y}\, (e(D2, B\theta)/e(Q1', B1))^{c}$。

d）计算$R3 = [sy]Q + [s\alpha]D' - [c]\, D3$。

e）检查等式$c = Hp(M \,||\, \rho \,||\, D1 \,||\, D2 \,||\, D3 \,||\, R1 \,||\, R2 \,||\, R3)$ 是否成立。

f） 如果等式成立，则输出1 (有效)。

g） 否则，输出 0(无效)。

9. 2.2 GB/T 38647 采用群组公钥的机制

打开过程

群组成员打开过程输入签名$\sigma = (\rho, D_1, D_2, D_3, c, s_\alpha, s_x, s_y, s_y)$, 和群组成员打开密钥$gmok = (\eta, \xi)$，执行过程如下步骤：

a）计算$[y]Q = D3 - [\xi]D1$。

b）从成员列表LIST找到对应i的LIST[i]= (ID, $[y]Q$, A, x, y, $upk[i]=Z(=[z]W)$, ·)。

c) 如果没有相匹配的i， 增输出 (i = 0, *)。

否则, 该过程执行如下：

- 选取 $r \leftarrow Zp^*$。
- 计算 $K_{open}=[\eta]D_1$, $W_{open}=[r]U$, $V_{open}=[r]D_1$, $c_{open}= H_p(\sigma \,||\, g \,||\, K_{open} \,||\, W_{open} \,||\, V_{open})$和 $s_{open}= r + c_{open}\, \eta \pmod p$。
- 输出一个绑定证据 (i, $\tau = (K_{open}, c_{open}, s_{open})$, $upk[i]=Z$, $Y_{1,i}= [y_i]Q_2$, $Y_{2,i}=[y_i]B1$, $X_{1,i}=[x_i]Q$, $X_{2,i}=[x_i]B_1$))。

  如果使用该绑定证据的证据评估过程输出为1, 输出用户ID对应为i。

9. 2.2 GB/T 38647 采用群组公钥的机制

证据评估过程

证据评估过程输入群组公钥gpk, 签名$\sigma = (\rho, D_1, D_2, D_3, c, s_\alpha, s_x, s_y, s_y)$, 绑定证据$(i, \tau = (K_{open}, c_{\text{open}}, s_{\text{open}}), upk[\text{i}] = Z, Y_{1,i}, Y_{2,i}, X_{1,i}, X_{2,i}))$，则执行过程如下：

a) 需要使用gpk，(ρ, RL)更新撤销的upk, 并获取$gpk_\rho = (Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))$。

b) 如果$i = 0$那么输出⊥(失败)。否则, 检查是否下列等式成立：

$c_{\text{open}} = Hp(\sigma \| g \| K_{\text{open}} \| [s_{\text{open}}]U - [c_{\text{open}}]W \| [s_{\text{open}}]D_1 - [c_{\text{open}}]K_{\text{open}})$。

$e(D_2 - K_{open}, X_{2,i} + B_\theta) = e(Q_1 - Y_{1,i} - Z_i, B_1')$，

其中$Q1$包含在gpk并且$\log_{B1}B_1' = \log_{Q1}Q_1'$,如果所有的等式成立，则输出1 (“合法”), 否则, 输出0 (“不合法”)。

9. 2.2 GB/T 38647 采用群组公钥的机制

链接过程

群组签名连接过程输入两个签名σ和σ'，群组签名连接密钥$gslk = (B1, V = [\xi] B1)$，则执行过程如下：

a) 计算$LI_1 = e(D_3, B_1)\, e(D_1, V)^{-1}$和$LI_2 = e(D_3', B_1)\, e(D_1', V)^{-1}$。

b) 如果$LI_1 = LI_2$，则输出1 (连接)，否则，0 (非连接)。

9.2.2 GB/T 38647 采用群组公钥的机制

撤销过程

RI是撤销指数列表，它包含目前已撤销群组成员的所有指数。无论何时密钥被撤销，RI都能马上将它更新进去。 RL是一个包含目前已撤销群组成员隐私信息的列表。假设RL总是更新到最新的撤销指数RI。任何人都可以公开的使用RL和RI。列表RI和RL的初始设置为空。

该撤销过程是一个全局撤销。 它执行三个子过程，产生RL (由发布方执行)，更新gpk (由任意一方执行)和更新usk (由合法签名方或群组成员执行)。

9.3.1 CZK03群签名方案

目　录

9.3.1. CZK03群签名方案

Gap Diffie-Hellman群

令G_1是由生成元P产生的循环加法群，它的阶为素数q,假设G_1中求逆和乘法是有效的，我们将以下问题引入G_1

- DLP:已知两个元素P和Q,找到一个整数$n \in Z_q^*$,使得等式$Q = nP$成立
- $CDHP$:已知P, aP, bP,其中$a, b \in Z_q^*$,计算abP
- $DDHP$:已知P, aP, bP, cP,其中$a, b, c \in Z_q^*$,判断等式$c \equiv ab \ mod \ q$是否成立

如果$DDHP$可以在多项式时间内解出，但没有多项式时间算法可以在不可忽略的概率下解出$CDHP$或DLP，则我们称G_1为Gap Diffie-Hellman群。

9.3.1. CZK03群签名方案

CZK03算法流程:

(1) $Setup()$

- 群管理员令G_1为Gap Diffie-Hellman群，生成元为P且它的阶为素数q。G_2为一个阶为素数q的乘法循环群，一个双线性对映射为 $e: G_1 \times G_2 \rightarrow G_2$。相关hash函数为$H_1$：$\{0,1\}^* \times G_1 \rightarrow Z_q$; $H_2: \{0,1\}^* \times G_1 \rightarrow G_1$。
- 群管理员随机选择一个数s$\in Z_q^*$，设置群管理员公钥$P_{\text{pub}} = sP$。群公钥为$\gamma = \{G_1, G_2, e, q, P, P_{\text{pub}}, H_1, H_2\}$。
- 用户随机选择一个数$r_i \in Z_q^*$，并将$R_i = r_i P$发送给群管理员。群管理员计算用户的部分私钥 $S_{ID} = sH_2(ID||T, R_i)$ 并将它通过安全信道发送给用户。

9.3.1. CZK03群签名方案

2) Enroll():

群成员U_i

- 随机选择$x_i \in Z_q^*$
- 计算$Q_i = r_i x_i P$
- 计算$X_i = x_i P$

Q_i，R_i，X_i，ID，S_{ID}，

S_i

群管理员

- 验证等式是否成立:

$$S_{ID} = sH_2(ID||T，R_i)$$

$$e(Q_i，P) = e(X_i, R_i)$$

- 计算$S_i = sH_2(T，Q_i)$

将$(S_i，Q_i)$作为自己的群证书

9.3.1. CZK03群签名方案

3) Sign()

群成员U_i

- 随机选择$a \in\in Z_q^*$
- 计算$U = aH_2(T, Q_i)$,
  $V = r_i x_i H_2(\mathrm{m}, U)$,
  $h = H_1(\mathrm{m}, U + V)$,
  $W = (a + h)S_i$,
- $\{U, V, W, T, Q_i\}$为群成员对消息m的签名

$\{U, V, W, T, Q_i\}$

4) Verify()

接收者

如果时间戳T是有效的，验证者计算：

- $Y = H_2(T, Q_i)$,
- $H_2(\mathrm{m}, U)$,
- $h = H_1(\mathrm{m}, U + V)$,

若以下等式成立，则验证者接受这个签名：

$$e(W, P) = e\left(U + hY, P_{pub}\right)$$
$$e(V, P) = e(H_2(\mathrm{m}, U), Q_i)$$

9.3.1. CZK03群签名方案

(5) Open ()

对于一个有效的群签名，群管理者能通过$Q_i = \mathrm{r}x_i P$轻松确定签名用户的身份。而且用户无法抵赖，这是因为群管理者能提供等式：

$$e(\mathrm{r}x_i P, P) = e(x_i P，\ \mathrm{r}P)$$

$$e(S_{ID}, P) = e(H_2(ID||T)，\ P_{pub})$$

证明这个签名确实是用户签署的。

CZK03方案是基于*RCDHP* (Reversion of Computation Diffie-Hellman Problem)：

已知P, aP, rP, 计算bP使其满足等式$a \equiv \mathrm{rb}\ mod\ q$

9.3.1. CZK03群签名方案

*RCDHP*问题等同于G_1上的*CDHP*问题

证明:

- 已知P, aP, bP,假设我们可以解决G_1上的*RCDHP*问题，这样我们就通过P和bP可以得到$\mathrm{b}^{-1}P$。由于$a \equiv (\mathrm{ab})\mathrm{b}^{-1} mod\ q$,我们可以通过P, aP，$\mathrm{b}^{-1}P$计算出ab$P$。即我们可以解决G_1上*CDHP*问题。
- 已知P, aP, bP,令$\mathrm{Q} = \mathrm{b}P$, 则$P = \mathrm{b}^{-1}Q$。假设我们可以解决G_1上的*CDHP*问题，这样我们就可以通过Q和$\mathrm{b}^{-1}Q$得到$\mathrm{b}^{-2}Q$，即$\mathrm{b}^{-1}P$。这样我们就能通过P, aP, $\mathrm{b}^{-1}P$得到$a\ \mathrm{b}^{-1}P$。即我们可以解决G_1上*RCDHP*问题。

9.3.1. CZK03群签名方案

安全性分析:

1. 如果存在一个没有串通群管理员的敌手A以不可忽略的概率ϵ，在t时间内伪造一个成员证书，则我们可以在t时间内以不可忽略的概率ϵ解决G_1上*CDHP*问题。

证明:

$Game$:敌手A可以询问H_2最多k次；假设第i次H_2询问的输入为$(T, \mathrm{r}_i P)$,敌手A得到相应的成员证书S_i。最后敌手A可以一对新的(rP, S)。如果rP之前并没有被询问过，且等式$e(S,P) = e(H_2(T,\mathrm{r}P), P_{pub})$成立，则敌手A赢得这个游戏。

如果敌手A输出的(rP, S)为有效的，令$H_2(T,\mathrm{r}P) = \mathrm{aP}, P_{pub} = bP$,则有$S = \mathrm{abP}$，即我们可以解决$G_1$上*CDHP*问题。

9.3.1. CZK03群签名方案

2. 签名方案下的非交互式协议是一个诚实验证者对成员证书和相应身份的零知识证明。我们证明了知识提取器一旦发现两个可接受元组，就可以恢复成员证书

证明：

令$\{U,V,W,T,\mathrm{r}x_iP\}$和$\{U,V',W',T,\mathrm{r}x_iP\}$为两个可被接受的元组，令$h=H_1(\mathrm{m},\ U+V)$。由于$e(W,P)=e(U+hH_2(T,\ \mathrm{r}x_iP),\ P_{pub})$，我们可以得到$W=\mathrm{s}\big(U+hH_2(T,\ \mathrm{r}x_iP)\big)$和$W'=\mathrm{s}\big(U+h'H_2(T,\ \mathrm{r}x_iP)\big)$，因此我们可以得到

$$\mathrm{s}hH_2(T,\ \mathrm{r}x_iP)=(h-h')^{-1}(W-W')$$

又有$e(\mathrm{r}x_iP,P)=e(x_iP,\ \mathrm{r}P)$，则$\mathrm{r}x_iP$与$\mathrm{r}P$，ID相对应。

由于签名者的S_{ID}满足等式$e(S_{ID},P)=\ e(H_2(ID||T),\ P_{pub})$，因此签名者无法抵赖他的签名。

9.3.2. BW05群签名方案

目　录

9.3.2. BW05群签名方案

简单两级签名

在两级签名中，假设身份串长度为 k bits,消息长度为 m bits，且有 $k<<m$，$m \sim \lambda$，其中 λ 为系统的安全参数。g 为 G_p 的生成元，G 与 G_T 都是阶为 p 的循环群,其中的 CDH 问题是难解的。假设存在双线性映射 $e:G\times G\rightarrow G_T$。

9.3.2. BW05群签名方案

1) 系统建立人 CA 建立系统参数：Setup(1^{λ})

(1) 秘密选择$\alpha \in_R Z_p$，令$A = e(g,g)^{\alpha}$；

(2) 秘密选择$y', z' \in_R Z_p$，向量$\vec{y} = (y_1, \cdots, y_k) \in_R Z_p^{\,k}$，$\vec{z} = (z_1, \cdots, z_m) \in_R Z_p^{\,m}$；

(3) 生成系统参数PP和主密钥MK如下：

$$PP = (g, u' = g^{y'}, u_1 = g^{y_1}, \cdots, u_k = g^{y_k},$$
$$v' = g^{z'}, v_1 = g^{z_1}, \cdots, v_m = g^{z_m}, A = e(g,g)^{\alpha}) \in G^{k+m+3} \times G_T, \quad (3\text{-}4\text{-}1)$$
$$MK = g^{\alpha} \in G \quad (3\text{-}4\text{-}2)$$

(4) 公布系统公钥参数：PP，m，k，p，G，G_T，e；秘密保存主密钥MK。

9.3.2. BW05群签名方案

2) 用户密钥提取：Extract(PP，MK，ID)

(1) 用户身份标识为 $ID=(\kappa_1\cdots\kappa_k)\in\{0,1\}^k$，用户提交$ID$给CA；

(2) CA秘密选择 $r\in_R Z_p$，计算

$$K_{ID}=(g^{\alpha}\cdot(u'\prod_{i=1}^{k}u_i^{\kappa_i})^r,g^{-r})\in G^2$$

9.3.2. BW05群签名方案

3) 用户签名：Sign(PP，K_{ID}，M)。其中$K_{ID}=(K_1, K_2)\in G_2$为要执行签名的用户ID的签名密钥，$M=(\mu_1\cdots\mu_2)\in\{01\}^m$为需要签名的消息。用户签名过程为：

(1) 秘密地选取$s\in_R Z_p$；

(2) 计算签名S为

$$\begin{aligned} S &= (K_1\cdot(v'\prod_{j=1}^{m}v_j^{\mu_j})^s, K_2, g^{-s}) \\ &= (g^{\alpha}\cdot(u'\prod_{i=1}^{k}u_i^{\kappa_i})^r(v'\prod_{j=1}^{m}v_j^{\mu_j})^s, g^{-r}, g^{-s}) \\ &= (S_1, S_2, S_3)\in G^3 \end{aligned} \tag{3-4-4}$$

9.3.2. BW05群签名方案

4) 接收者验证签名：Verify(*PP*，*ID*，*M*，*S*)

(1) 检验等式

$$e(S_1,g)\cdot e(S_2,u'\prod_{i=1}^{k}u_i^{\kappa_i})\cdot e(S_3,v'\prod_{j=1}^{m}v_j^{\mu_j})\stackrel{?}{=}A$$

(2) 若式(3-4-5)成立，则接受签名为合法签名；否则，拒绝。

9.3.2. BW05群签名方案

BW05群签名方案

1) 创建：Setup(1^{λ})

群建立人 CA 选取系统的安全参数为 λ，假设身份串长度为 k bits，消息长度为 m bits，且有 $k<<m$，m~ λ。随机选择两个大素数 p，q，并令 $n=pq$。设 G 是阶为 n 的双线性群，Gp，Gq 为 G 的子群，阶分别为 p，q。另外随机选择生成元 $g \in G$ 与 $h \in G_q$，指数 $\alpha \in_R Z_n$，生成元 $u', u_1, \cdots, u_k \in G$ 与 $v', v_1, \cdots, v_k \in G$。

CA 主私钥：$MK = g^{\alpha} \in G$；群管理员 TA 私钥：$TK=q$。

CA 公布群公钥：

$$PP = (g, h, u', u_1, \cdots, u_k, v', v_1, \cdots, v_m, A = e(g,g)^{\alpha}) \in G \times G_q \times G^{k+m+2} \times G_T$$

9.3.2. BW05群签名方案

2) 加入：Enroll(PP，MK，ID)

(1) 用户身份标识为$ID=(\kappa_1\cdots\kappa_k)\in\{0,1\}^k$，用户提交$ID$给CA；

(2) CA秘密选择$s\in_R Z_n$，并生成用户ID的群成员证书K_{ID}，即群签名密钥，秘密地传给用户：

$$K_{ID}=(K_1,K_2,K_3)=(g^{\alpha}\cdot(u'\prod_{i=1}^{k}u_i^{\kappa_i})^s,g^{-s},h^s)\in G^2\times G_q \qquad (3\text{-}4\text{-}7)$$

9.3.2. BW05群签名方案

3) 签名：Sign(PP，K_{ID}，M，ID)，其中$M=(\mu_1\cdots\mu_2)\in\{01\}^m$为需要签名的消息。签名过程为：

(1) 用户秘密选择指数$t_1,\cdots,t_k\in Z_n$，对于所有的i=1,..,k，计算

$$c_i=u_i^{\kappa_i}\cdot h^{t_i},\qquad \pi_i=(u_i^{2\kappa_i-1}\cdot h^{t_i})^{t_i} \tag{3-4-8}$$

并定义

$$t=\sum_{i-1}^{k}t_i \tag{3-4-9}$$

和

$$c=u'\prod_{i=1}^{k}c_i=(u'\prod_{i=1}^{k}u_i^{\kappa_i})\cdot h^t \tag{3-4-10}$$

9.3.2. BW05群签名方案

(2) 令$V=v'\prod_{j=1}^{m}v_j^{\mu_j}$，选择2个指数$\tilde{s}_1$，$s_2\in Z_n$，并计算

$$\sigma_1=K_1\cdot K_3^{t}\cdot c^{\tilde{s}_1}\cdot V^{s_2},\quad \sigma_2=K_2\cdot g^{-\tilde{s}_1},\quad \sigma_3=g^{-s_2}$$

若令$s_1=\tilde{s}_1+s_2$，则有

$$\sigma_1=g^{\alpha}\cdot(u'\prod_{i=1}^{k}u_i^{\kappa_i})^{s_1}\cdot(v'\prod_{i=1}^{m}v_i^{\mu_i})^{s_2}\cdot h^{s_1 t}=g^{\alpha}\cdot c^{s_1}\cdot V^{s_2},$$

$$\sigma_2=g^{-s_1},\qquad \sigma_3=g^{-s_2}$$

(3) 签名输出为：

$$\sigma=(\sigma_1,\sigma_2,\sigma_3,c_1,\cdots,c_k,\pi_1,\cdots,\pi_k,)\in G^{2k+3}$$

9.3.2. BW05群签名方案

4) 验证：Verify(PP，ID，M，s)

(1) 通过检验以下等式，检验 c 是否是合法生成的

$$\forall i=1,\ldots,k: \quad e(c_i,u_i^{-1}c_i) \stackrel{?}{=} e(h,\pi_i)$$

(2) 计算出 $c=u'\prod_{i=1}^{k}u_i^{\kappa_i}$，$V=v'\prod_{j=1}^{m}v_j^{\mu_j}$ 并检查以下等式是否成立

$$e(\sigma_1,g)e(\sigma_2,c)e(\sigma_3,V) \stackrel{?}{=} A$$

若成立，则接受此签名为合法签名；否则，拒绝。

9.3.2. BW05群签名方案

5) 打开：Open(PP, TK, s)

κ_i表示签名人身份ID的第i位，对于$i=1,..,k$，TA计算

$$\kappa_i = \begin{cases} 0 & if\ (c_i)^q = g^0 \\ 1 & otherwise \end{cases}$$

则可恢复出签名人身份$ID = (\kappa_1 \cdots \kappa_k) \in \{0,1\}^k$。

9.3.2. BW05群签名方案

安全性问题

BW05群签名方案可以伪造出签名：

对于群参数$(u',u_1,\cdots,u_k)\in G^k$，设$u_i=g^{x_i},x_i\in Z_n,\forall i=\{1,\cdots,k\}$，若存在：

$$u_a u_b = u_c \qquad a,b,c\in(1,..,k)$$

对于某群成员$ID=(\kappa_1\cdots\kappa_k)\in\{0,1\}^k$，其$\kappa_a=1,\kappa_b=1,\kappa_c=0$，其群签名密钥为：$K_{ID}=(K_1,K_2,K_3)$。可按如下方法构造不可跟踪得群签名：

1) 伪造一身份$ID'=(\kappa'_1\cdots\kappa'_k)\in\{0,1\}^k$：

当$i\neq a,b,c$时 $\kappa'_i=\kappa_i$

当$i=a,b,c$时 $\kappa'_i=1-\kappa_i$

9.3.2. BW05群签名方案

安全性问题

2) 按照 ID' 构造 c_i,π_i，并构造 c。显然这样构造的 c_i,π_i 可以通过验证等式

$$\forall i=1,\ldots,k: \qquad e(c_i,u_i^{-1}c_i) \stackrel{?}{=} e(h,\pi_i)$$

3) 将 c 带入签名式

$$\sigma_1 = K_1 \cdot K_3^{t} \cdot c^{\tilde{s}_1} \cdot V^{s_2}, \quad \sigma_2 = K_2 \cdot g^{-\tilde{s}_1}, \quad \sigma_3 = g^{-s_2}$$

其余的签名步骤都与正常的签名相同

9.3.2. BW05群签名方案

安全性问题

4) 得到签名 $\sigma=(\sigma_1,\sigma_2,\sigma_3,c_1,\cdots,c_k,\pi_1,\cdots,\pi_k,)\in G^{2k+3}$。

由于 $c=u'\prod_{i=1}^{k}c_i=(u'\prod_{i=1}^{k}u_i^{\kappa'_i})\cdot h^t$，且 $u'\prod_{i=1}^{k}u_i^{\kappa'_i}=u'\prod_{i=1}^{k}u_i^{\kappa_i}$，则所得的签名组可以通过验证等式(3-4-15)和式(3-4-16)的验证，接收签名人认为这是一合法的群签名。但当群管理员 TA 执行打开签名算法 Open()时，得到的签名人身份为 $ID'=(\kappa'_1\cdots\kappa'_k)$。因此，群成员成功地伪造除了一个不可跟踪的群签名。

9.4 群签名算法在区块链中的应用

目　录

9.4 群签名算法在区块链中的应用

- 近年来，匿名数字货币发展迅猛:门罗币（Monero）、Zcash、Blindcion…，它们在数字货币交易中保护了用户的隐私。然而,由于它们的匿名性，可能会导致一些新的威胁，如敲诈勒索，走私，逃税和洗钱。更糟糕的是，在匿名的掩护下很难追踪罪犯。

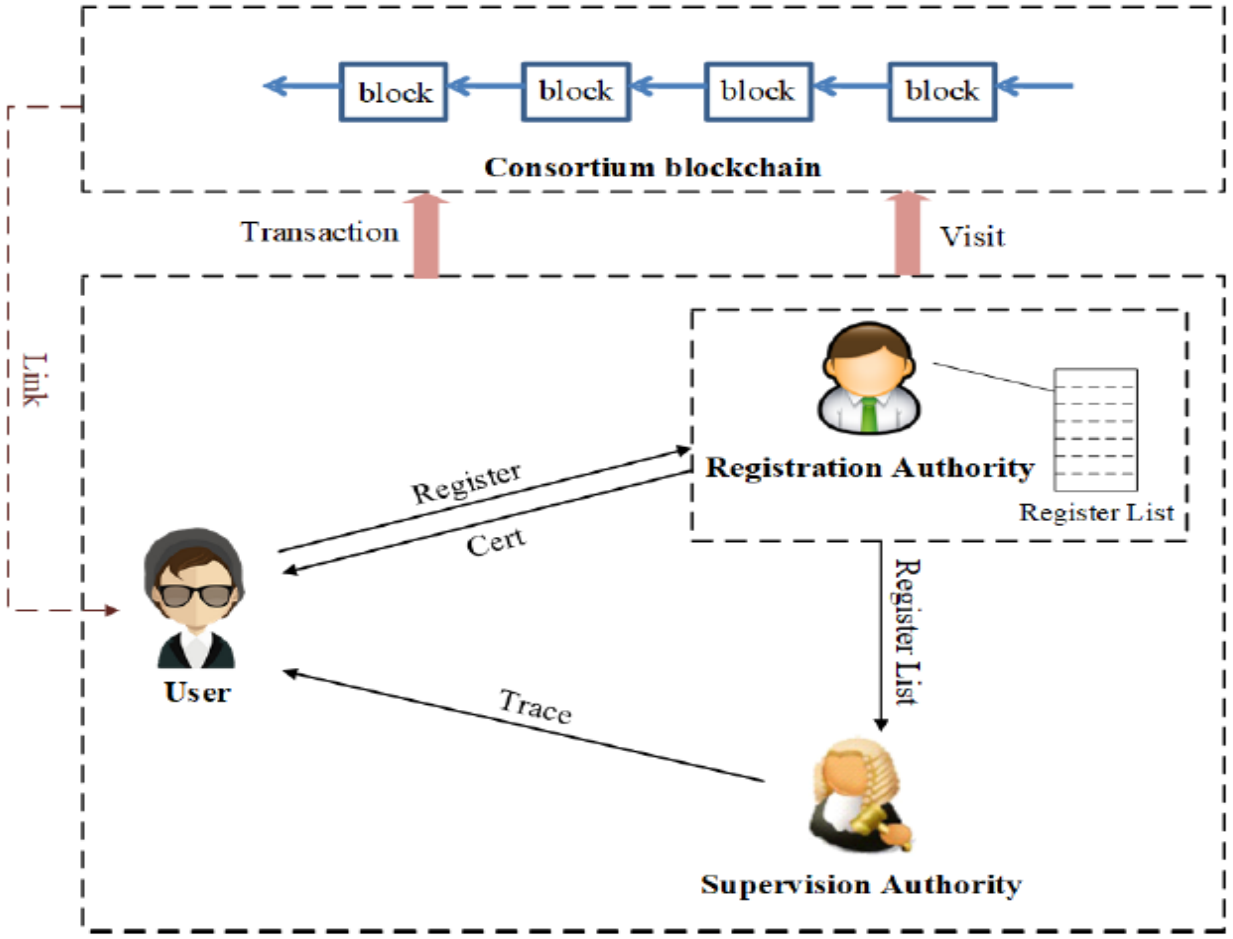
9.5 群签名算法在区块链中的应用

一种在匿名数字货币中跟踪付款人的高效可连接群签名

- 使用可连接群签名来实现支付者的真实身份跟踪。这种方法能够同时保障匿名数字货币交易中的匿名性和可靠性。

- 该方案利用线性加密来帮助群管理器跟踪群成员的身份，并通过生成VR-SDH三元组的零知识证明(ZKPK)来生成群签名。如果一个群成员对同一个消息签名两次，那么两个签名就可以公开链接，这可以用于匿名数字货币的双花检测。

9.5 群签名算法在区块链中的应用



9.5 群签名算法在区块链中的应用

高效可连接群签名安全需求：

- 正确性：保证诚实用户生成的签名始终被接受，使用同一密钥生成的两个签名始终被正确链接，任何有效的签名都可以用SA的跟踪密钥跟踪到实际的签名者。
- 完全匿名：给定两个签名和其中一个签名者，没有人能够确定哪一个签名是由已知的签名者产生的，除了SA。
- 完全可追溯性：完全可追溯性比可追溯性更强，它也可以被看作是一种强大的抗合谋形式。具体来说，一个敌手根据其他组成员的合谋创建了一个签名，甚至持有SA的跟踪密钥，而SA无法跟踪到实际签名者之一，这种情况发生的概率几乎可以忽略不计。
- 可链接性：一个敌手生成了两个具有相同密钥的签名，其amount相同，而其他群成员没有将其链接的概率可以忽略不计。

9.5 群签名算法在区块链中的应用

算法流程：

（1）$Setup()$

- n是安全参数,e^为双线性对。(G_1, G_2)代表计算关系为φ一个双线性组，假设SDH问题在(G_1, G_2)和G_1上都是困难的 。
- 定义HASH函数H_0：$\{0,1\}^* \to G_1$，H_1：$\{0,1\}^* \to Z_P$。
- 随机选择G_2上的生成元g_2，G_1上的生成元g_1, h, u。 且有$g_2 = \varphi(g_1)$。
- RA随机选择一个$\gamma \in Z_P$，令$\omega = g_2^{\gamma}$。RA的公钥$PK_R = \omega$，私钥 $SK_R = \gamma$。
- SA随机选择G_1上的两个生成元v_1, v_2;随机选择$k_1, k_2 \in Z_P$使得$v_1^{k_1} = v_2^{k_2} = u$。SA的公钥$PK_S = u$，私钥 $SK_S = (k_1, k_2)$。
- 群公开参数为：$GPK = (G_1, G_2, e^{\wedge}, p, n, g_1, g_2, h, u, v_1, v_2, \omega, H_0, H_1)$

9.5 群签名算法在区块链中的应用

2) Enroll():

群成员U_i

- 随机选择$y_i \in Z_p^*$
- 计算$Y = h^{y_i}$
- 知识证明$PK\{(y_i): Y = h^{y_i}\}$

Y

$PK\{(y_i): Y = h^{y_i}\}$

(A_i, x_i)

群管理员RA

- 随机选择$\mathrm{x}_i \in Z_p^*$
- 计算$A_i = (g_1 Y^{-1})^{\frac{1}{\gamma + \mathrm{x}_i}}$

- 验证等式$\mathrm{e}(A_i, \omega \mathrm{g}_2{}^{x_i}) = \mathrm{e}(g_1 h^{-y_i}, g_2)$
- 若等式成立，接受群证书$\mathrm{Cert} = (A_i, \mathrm{x}_i)$
- 将三元组(A_i, x_i, y_i)作为用户私钥SK_{U_i}

9.5 群签名算法在区块链中的应用

3) Sign()

群成员U_i

- 随机选择α，$\beta \in Z_p^*$
- 对于指定的amount $\in \{0,1\}^*$,计算$u_0 = H_0(amount)$
- 计算

$$l_1 = v_1^{\alpha}, \quad l_2 = v_2^{\beta}, \quad l_3 = A_i \cdot u^{\alpha+\beta}, \quad l_4 = u_0^{x_i}$$

$$\delta_1 = x_i \cdot \alpha, \quad \delta_2 = x_i \cdot \beta$$

- 对消息m执行非交互式零知识证明得到Π
- 签名为$\sigma = (l_1, l_2, l_3, l_4, \Pi)$,其中$l_4$为链接的tag

9.5 群签名算法在区块链中的应用

3) Sign()

群成员U_i

- $$\Pi = SoK\left\{ \begin{pmatrix} \alpha, \beta, \\ x_i, y_i, \\ \delta_1, \delta_2 \end{pmatrix} : \begin{array}{c} \wedge \\ \wedge \\ \wedge \\ \wedge \\ \wedge \end{array} \begin{array}{c} l_1 = v_1^{\alpha} \\ l_2 = v_2^{\beta} \\ 1_{G_1} = l_1^{x_i} \cdot v_1^{\delta_1} \\ 1_{G_1} = l_1^{x_i} \cdot v_1^{\delta_1} \\ \frac{\hat{e}(g_1,g_2)}{\hat{e}(l_3,\omega)} = \hat{e}(u,\omega)^{-\alpha-\beta} \cdot \hat{e}(l_3,g_2)^{x_i} \cdot \hat{e}(u,g_2)^{-\delta_1-\delta_1} \cdot \hat{e}(h,g_2)^{y_i} \\ l_4 = u_0^{x_i} \end{array} \right\}(m)$$

9.5 群签名算法在区块链中的应用

求Π的具体过程

- 随机选择$\gamma_\alpha, \gamma_\beta, \gamma_x, \gamma_y, \gamma_{\delta_1}, \gamma_{\delta_2} \in Z_p$
- 计算

$$a_1 = v_1^{r_\alpha}, \quad a_2 = v_2^{r_\beta},$$

$$a_3 = \hat{e}(u,\omega)^{-r_\alpha - r_\beta} \cdot \hat{e}(l_3, g_2)^{r_x} \cdot \hat{e}(u, g_2)^{-r_{\delta_1} - r_{\delta_2}} \cdot \hat{e}(h, g_2)^{r_y}$$

$$a_4 = l_1^{r_x} \cdot v_1^{-r_{\delta_1}}, \quad a_5 = l_1^{r_x} \cdot v_2^{-r_{\delta_2}}, \quad a_6 = u_0^{r_x}$$

- 计算$c = H_1(m, l_1, l_2, l_3, l_4, a_1, a_2, a_3, a_4, a_5, a_6)$
- 计算

$$z_\alpha = r_\alpha - c\alpha, \quad z_\beta = r_\beta - c\beta,$$
$$z_x = r_x - cx, \quad z_y = r_y - cy,$$
$$z_{\delta_1} = r_{\delta_1} - c\delta_1, \quad z_{\delta_2} = r_{\delta_2} - c\delta_2$$

则$\Pi = (c, z_\alpha, z_\beta, z_x, z_y, z_{\delta_1}, z_{\delta_2})$

9.5 群签名算法在区块链中的应用

验证者可通过以下等式验证Π是否正确

$$v_1^{z_\alpha} = a_1 \cdot l_1^c$$

$$v_2^{z_\beta} = a_2 \cdot l_2^c$$

$$\hat{e}(u,\omega)^{-z_\alpha - z_\beta} \cdot \hat{e}(l_3, g_2)^{z_x} \cdot \hat{e}(u, g_2)^{-z_{\delta_1} - z_{\delta_1}} \cdot \hat{e}(h, g_2)^{z_y} = a_3 \cdot \left(\frac{\hat{e}(g_1, g_2)}{\hat{e}(l_3, \omega)} \right)^c$$

$$l_1^{z_x} \cdot v_1^{-z_\alpha} = a_4$$

$$l_1^{z_x} \cdot v_2^{-z_{\delta_2}} = a_5$$

$$u_0^{z_x} = a_6 \cdot l_4^c$$

9.5 群签名算法在区块链中的应用

(4) Verify ()

- 计算$u_0 = H_0(amount)$
- 计算

$$\tilde{a}_1 = v_1^{z_\alpha} \cdot l_1^c, \quad \tilde{a}_2 = v_2^{z_\beta} \cdot l_2^c,$$

$$\tilde{a}_3 = \hat{e}(u,\omega)^{-z_\alpha - z_\beta} \cdot \hat{e}(l_3, g_2)^{z_{x_i}} \cdot \hat{e}(u, g_2)^{-z_{\delta_1} - z_{\delta_1}} \cdot \hat{e}(h, g_2)^{z_{y_i}} \cdot \left(\frac{\hat{e}(g_1, g_2)}{\hat{e}(l_3, \omega)}\right)^c$$

$$\tilde{a}_4 = l_1^{z_{x_i}} \cdot v_1^{z_\alpha}, \quad \tilde{a}_5 = l_2^{z_{x_i}} \cdot v_2^{z_\beta}, \quad \tilde{a}_6 = l_4^c \cdot u_0^{z_{x_i}}$$

- 计算 $\tilde{c} = H_1(m, l_1, l_2, l_3, l_4, \tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \tilde{a}_4, \tilde{a}_5, \tilde{a}_6)$
- 若等式

$$c = \tilde{c}$$

成立，输出1；不成立，输出0

9.5 群签名算法在区块链中的应用

(5) *Link* ()

- 给定两个不同的签名$(\mathrm{m},\sigma),(\mathrm{m}',\sigma')$，若两个签名是由同一个签名者在同样的amount下进行的，那么任何一个人都可以公开链接这两个签名。首先，验证者验证这两个签名是否有效。然后，我们可以通过σ得到l_4，σ'得到l_4',若$\mathrm{l}_4=\mathrm{l}_4'$，则可以连接上两个签名，输出1；否则输出0。

9.5 群签名算法在区块链中的应用

（6）Trace（）

给定(m, σ),若签名是有效的，SA可以通过它的私钥$SK_S = (k_1, k_2)$计算

$A_{\mathrm{i}} = \frac{l_3}{l_1^{k_1} l_2^{k_2}}$来找到签名群成员的身份。

谢谢！