8.3　群签名

群签名（group signature）的概念是由 Chaum 和 Heyst 于 1991 年联合提出的。在一个群签名体制中，群体（group）中的成员（member）可代表整个群体进行匿名签名：一方面，验证者只能确定签名是由群体中的某个成员产生的，但不能确定是哪个成员，此即群签名的匿名性；另一方面，在必要的时候（如发生争执的情况下）群管理者（group manager）可以打开（open）签名来揭示签名人的身份，使签名人不能否认自己的签名行为，此即群签名的可追踪性。将两者结合在一起，可以说，群签名是一种同时提供匿名性和可跟踪性的技术，其匿名性为合法用户提供匿名保护，其可跟踪性又使得可信机构可以跟踪非法行为。群签名还具有无关联性（unlinkability），即在不打开群签名的条件下，任何人不能确定两个群签名是否为同一个成员所产生。可撤销匿名性和无关联性使群签名在管理、军事、政治、经济等多个领域有着广泛的应用前景，因此引起了研究者的广泛关注。

8.3.1　群签名的基本概念

在一个群签名方案中，一般包含一个群管理员和若干个群成员，这些成员构成的集合称为群。群管理员负责产生系统参数、群公钥、群私钥，同时要为群成员产生签名私钥或群成员身份证书。群成员用自己掌握的签名私钥代表整个群执行签名操作。在发生争端的情况下，群管理员能够从给定的群签名中识别出产生该签名的成员身份。在有的群签名体制中，存在两个群管理员：一个负责为群成员颁发群成员私钥或群成员证书；另一个执行追踪功能。通常，一个群签名体制由下列算法组成。

1．系统初始化算法

产生群公钥、群成员的公钥和私钥及群管理员用于打开签名的打开私钥。

2．成员加入

一个新用户通过和群管理员的交互协议请求加入，协议执行结束后，合法的新成员完成身份注册并获得一个私钥（有的方案中还会包含一个成员资格证书）。

3．签名生成

群签名产生算法，用群成员的私钥和成员资格证书对消息 m 进行签名。

4．签名验证

验证消息 m 的签名是否是一个合法的群签名。

5．签名打开

群管理员输入消息、消息的签名和自己的私钥，运行打开算法以揭示签名者的真实身份。

8.3.2 群签名的安全性需求

Chaum 和 Heyst 提出群签名概念时所规定的群签名的安全性包括：给定一个群签名，签名验证者不能由此识别出产生该签名的签名人的身份，即匿名性问题；群管理员可以追踪产生该签名的成员的身份，即可追踪性问题。随着对群签名研究的不断深入，要求群签名体制满足的性质逐渐增多。总的来说，一个安全的群签名方案至少应具有如下性质。

（1）正确性：一个合法的群成员按照签名产生算法产生的群签名一定能够通过签名验证算法。

（2）不可伪造性（unforgeability）：非群成员要产生一个通过验证算法的群签名在计算上是不可行的。

（3）匿名性（anonymity）：除群管理员之外，任何人要确定一个给定群签名的实际签名人在计算上是不可行的。

（4）无关联性（unlinkability）：在不打开签名的情况下，确定两个不同的群签名是否为同一个签名人所签是不可能的。

（5）可跟踪性（traceability）：一个正确的签名可以由群管理员揭示签名者的真实身份。

（6）防陷害性（exculpability）：包括群管理员在内的任何成员都不能以其他群成员的名义产生合法的群签名。

（7）抗联合攻击（coalition-resistance）：任意多个群成员勾结或与群管理员勾结都不能伪造其他群成员的签名。

8.3.3 一个简单的群签名体制

Chaum 和 Heyst 在提出群签名概念的同时也描述了几个群签名体制，我们在这里简单介绍其中之一，目的是让读者对群签名有一个直观的认识。假设有 n 个人构成一个群，GM 是该群的群管理员。

1. 系统初始化算法

在这个算法里，群管理员 GM 为群中的每个成员分发一张秘密密钥表，这些表是互不相交的。GM 将各个成员拥有的私钥汇总在一起，将这些私钥对应的公钥以一种随机的次序排成一张表，并将这个公钥表公开。

2. 签名生成

每个群成员每次从自己的私钥表中选取一个没有使用过的私钥，利用这个私钥对消息签名。

3. 签名验证

如果接收者要对某个群成员产生的群签名执行签名验证，他就用公钥表中的每个公钥去验证，只要发现有一个公钥使签名验证通过，就说明这个签名是该群的合法签名。

4. 签名打开

在发生争端的情况下，由于群管理员知道所有群成员的私钥和公钥之间的对应关系，从而可以根据签名、公钥恢复出签名人的身份。

在这个简单的方案里，我们假设群成员都是在系统初始化时固定加入的，所以没有讨论群成员的动态加入问题。此外，每个群成员的任意一个私钥只能使用一次，否则，如果某个群成员使用自己的某个私钥 x_i 同时对消息 m_1,m_2 执行签名操作，则验证者可以利用 x_i 对应的公钥 y_i 验证这两个签名有效，同时验证者可以确定这两个签名是由该群中的同一个成员产生的。这导致方案丧失不可关联性。

在这个方案中，由于群管理员知道每个群成员的私钥，所以能够以任意一个群成员的名义产生有效群签名。不过，如果假设群管理员总是可信的，则可以认为 GM 不会试图假冒群成员伪造群签名。

8.3.4 另一个简单的群签名体制

正如我们分析的那样，在 8.3.3 节介绍的体制中，GM 知道每个群成员的私钥，从而可以伪造群签名。可以采取一些机制使 GM 不知道群成员的私钥，下面的简单体制做到了这一点。设 p 是一个大素数，在 Z_p 上计算离散对数是不可行的，g 是 Z_p^* 的一个生成元。假设有 n 个人构成了一个群，他们的秘密密钥分别为 $s_1,\cdots,s_n$，对应的公钥是 $y_i = g^{s_i} \bmod p, 1 \leqslant i \leqslant n$。

1. 系统初始化

群管理员 GM 有一张群成员的名字与他们的公钥相对应的表。GM 为群成员 i 选取随机数 $r_i \in_R Z_p^*(1 \leqslant i \leqslant n)$，发送 r_i 给对应的群成员，另将 $y_i^{r_i}(1 \leqslant i \leqslant n)$ 以一种随机的次序排成一张“公钥表”，并将这个表公开。

2. 签名生成

每个群成员将 $r_i s_i \bmod (p-1)$ 作为私钥，利用 ElGamal 型数字签名算法对消息产生群签名。

3. 签名验证

如果接收者要对某个群成员产生的群签名执行签名验证操作，他用“公钥表”中的每个“公钥”去验证，只要发现有一个“公钥”使签名验证通过，就说明这个签名是该群的合法签名。

4. 签名打开

在发生争端的情况下，由于群管理员知道 $r_i \in Z_p^*(1 \leqslant i \leqslant n)$，也知道所有群成员的公钥 y_i 和名字之间的对应关系，从而可以根据签名、“公钥”恢复出签名人的身份。

可以看到，在这个方案中，每个群成员的签名私钥也只有一个，即 $r_i s_i \bmod (p-1)$，且群管理员不再拥有群成员执行群签名时所使用的这个签名私钥。此外，群管理者可以定期更新颁发给每个群成员的 $r_i \in Z_p^*(1 \leqslant i \leqslant n)$，以使方案具有更大的灵活性。这个方案的

一个明显缺点是如果有新的群成员加入，所有的群成员都不得不改变他们的密钥，否则接收者可以将旧的群成员和新的群成员区别开来。

8.3.5 短的群签名方案

前面介绍的群签名方案中，群公钥的长度是群成员个数 n 的线性函数，而且每个群成员使用当前的私钥执行签名操作时所能签名的次数都是固定的。因此，前述方案虽然简单，但不能算是有效的。另外，一个群签名的长度依赖于所使用的签名算法。例如，如果 8.3.4 节介绍的方案中签名者使用的是 $Z_p(|p|=1024)$ 上的 ElGamal 数字签名，则最终的签名长度为 2048 比特。这样长的签名长度对某些应用是不合适的。Boneh 等人在 2004 年的美国密码学会上提出了一个基于双线性对的短的群签名体制（short group signature）。该方案假设系统中存在两个群管理者：其中一个被称为 issuer，负责为群成员颁发用于执行群签名操作所需的私钥；另一个群管理者被称为 opener，在发生争端的情况下负责追踪群成员身份。

1. 系统初始化

令 p 为一个大素数，点 P 为 p 阶加法循环群 G_1 的生成元，H 为 G_1 中的任意非单位元的点，G_2 为同阶的乘法循环群，$e:G_1\times G_1\to G_2$ 为双线性映射，$\text{Hash}:\{0,1\}^*\to Z_p$ 为一哈希函数，$\xi_1,\xi_2,\gamma\in_R Z_p^*$，令 $U,V\in G_1$ 使得 $\xi_1U=\xi_2V=H$，置 $W=\gamma P$。将群公钥 $\text{gpk}\triangleq(P,H,U,V,W)$ 公开，而 $\text{gmsk}\triangleq(\xi_1,\xi_2)$ 由群管理员 opener 作为群私钥保密收藏，用于追踪给定群签名的签名人身份，γ由群管理员 issuer 持有。

2. 成员加入

假设系统中有 n 个群成员。群管理员 issuer 为第 i 个群成员选取 $x_i\in_R Z_p^*(1\leqslant i\leqslant n)$，计算 $A_i=\dfrac{1}{\gamma+x_i}P$，第 i 个群成员的私钥就是 (A_i,x_i)。此外，为便于 opener 执行追踪操作，每个群成员将自己的 A_i 也交给 opener。这样，opener 知道群成员的 A_i 与该群成员的身份之间的对应关系。

3. 签名生成

给定群公钥 $\text{gpk}=(P,H,U,V,W)$、某群成员的私钥 (A_i,x_i) 及待签名的消息 $m\in\{0,1\}^*$，持有该私钥的群成员执行下述操作：

（1）选择 $\alpha,\beta\in_R Z_p^*$；

（2）计算 $\delta_1=x_i\alpha$，$\delta_2=x_i\beta$，$T_1=\alpha U$，$T_2=\beta V$，$T_3=A_i+(\alpha+\beta)H$；

（3）选择 $r_\alpha,r_\beta,r_x,r_{\delta_1},r_{\delta_2}\in_R Z_p^*$；

（4）$R_1=r_\alpha U$，$R_2=r_\beta V$，$R_3=e(T_3,P)^{r_x}e(H,W)^{-r_\alpha-r_\beta}e(H,P)^{-r_{\delta_1}-r_{\delta_2}}$，$R_4=r_xT_1-r_{\delta_1}U$，$R_5=r_xT_2-r_{\delta_2}V$；

（5）计算 $c=\text{Hash}(m,T_1,T_2,T_3,R_1,R_2,R_3,R_4,R_5)$；

（6）$s_\alpha=r_\alpha+c\alpha,s_\beta=r_\beta+c\beta,s_x=r_x+cx_i,s_{\delta_1}=r_{\delta_1}+c\delta_1,s_{\delta_2}=r_{\delta_2}+c\delta_2$；

（7）输出 $(T_1,T_2,T_3,c,s_\alpha,s_\beta,s_x,s_{\delta_1},s_{\delta_2})$ 作为群成员代表该群对消息 m 产生的群签名。

4. 签名验证

给定群公钥 $\text{gpk}=(P,H,U,V,W)$、消息 m 和待验证的签名 $(T_1,T_2,T_3,c,s_\alpha,s_\beta,s_x,s_{\delta_1},s_{\delta_2})$，任意接收者通过执行以下操作来检验该签名是否为合法签名。

（1）计算 $R_1'=s_\alpha U-cT_1, R_2'=s_\beta V+cT_2$，$R_4'=s_xT_1-s_{\delta_1}U$，$R_5'=s_xT_2-s_{\delta_2}V$，$R_3'=e(T_3,P)^{s_x}e(H,W)^{-s_\alpha-s_\beta}e(H,P)^{-s_{\delta_1}-s_{\delta_2}}(\dfrac{e(T_3,W)}{e(P,P)})^c$；

（2）当且仅当下式成立时接受该签名为有效群签名：

$$c=H(m,T_1,T_2,T_3,R_1',R_2',R_3',R_4',R_5')$$

这里，方案的正确性成立：$R_1=R_1',R_2=R_2',R_4=R_4',R_5=R_5'$ 均容易说明，下面主要说明 $R_3=R_3'$。

事实上，有

$$\begin{aligned}R_3&=e(T_3,P)^{r_x}e(H,W)^{-r_\alpha-r_\beta}e(H,P)^{-r_{\delta_1}-r_{\delta_2}}\\&=e(T_3,P)^{s_x-cx_i}e(H,W)^{-s_\alpha-s_\beta+c\alpha+c\beta}e(H,P)^{-s_{\delta_1}-s_{\delta_2}+c\delta_1+c\delta_2}\\&=e(T_3,P)^{s_x}e(H,W)^{-s_\alpha-s_\beta}e(H,P)^{-s_{\delta_1}-s_{\delta_2}}e(T_3,P)^{-cx_i}e(H,W)^{c\alpha+c\beta}e(H,P)^{c\delta_1+c\delta_2}\end{aligned}$$

由此，如能说明 $e(T_3,P)^{-cx_i}e(H,W)^{c\alpha+c\beta}e(H,P)^{c\delta_1+c\delta_2}=(\dfrac{e(T_3,W)}{e(P,P)})^c$，就证明了 $R_3=R_3'$，这一点由下面的计算过程可以得到：

$$\begin{aligned}&e(T_3,P)^{-cx_i}e(H,W)^{c\alpha+c\beta}e(H,P)^{c\delta_1+c\delta_2}\\&=\frac{e(H,W)^{c\alpha+c\beta}e(H,P)^{c\delta_1+c\delta_2}}{e(T_3,P)^{cx_i}}=\frac{e((\alpha+\beta)H,W)^ce(H,P)^{c\delta_1+c\delta_2}}{e(T_3,P)^{cx_i}}\\&=\frac{e(T_3-A_i,W)^ce(H,P)^{c\delta_1+c\delta_2}}{e(T_3,P)^{cx_i}}=\frac{e(T_3,W)^ce(-A_i,W)^ce(H,P)^{c\delta_1+c\delta_2}}{e(T_3,P)^{cx_i}}\\&=e(T_3,W)^c\frac{e(H,P)^{c\delta_1+c\delta_2}}{e(A_i,W)^ce(T_3,P)^{cx_i}}=e(T_3,W)^c\frac{e(H,P)^{cx_i(\alpha+\beta)}}{e(A_i,W)^ce(T_3,P)^{cx_i}}\\&=e(T_3,W)^c\frac{e((\alpha+\beta)H,P)^{cx_i}}{e(A_i,W)^ce(T_3,P)^{cx_i}}=e(T_3,W)^c\frac{e((\alpha+\beta)H-T_3,P)^{cx_i}}{e(A_i,W)^c}\\&=e(T_3,W)^c\frac{e(-A_i,P)^{cx_i}}{e(A_i,W)^c}=e(T_3,W)^ce(A_i,x_iP+W)^{-c}=e(T_3,W)^ce(P,P)^{-c}\end{aligned}$$

5. 签名打开

给定群公钥 $\text{gpk}=(P,H,U,V,W)$、消息 m、待追踪的签名 $(T_1,T_2,T_3,c,s_\alpha,s_\beta,s_x,s_{\delta_1},s_{\delta_2})$，持有群私钥 $\text{gmsk}=(\xi_1,\xi_2)$ 的群管理员 opener 执行下述操作即可追踪出产生该签名的群成员的身份：

（1）执行签名验证算法，确保该签名是对消息 m 的有效群签名；

（2）计算 $A=T_3-(\xi_1T_1+\xi_2T_2)$；

（3）检查自己拥有的 A_i 与群成员身份的对应关系表，从而确定产生该签名的群成员的身份。

至此，我们完整描述了 Boneh 等人的短的群签名体制。如同这个体制的名字所揭示的那样，这个方案的最大优点是每个群成员产生的群签名都是固定长度的，而与群成员的个数无关。更进一步，每个群签名由 3 个群 G_1 中的元素和 6 个 Z_p 中的元素构成。以 p 为 170 比特的素数、G_1 中的每个元素为 171 比特为例，群签名的长度为 1533 比特，即 192 字节。该方案产生的签名之所以能够达到这个长度，很重要的一点就是利用了椭圆曲线群上点的压缩存储技术。

注意到方案中的许多量，如 $e(H,W),e(H,P),e(P,P)$，都可以预先计算出来，以便执行签名操作或签名验证操作时直接调用。此外，每个群成员都可以预先计算 $e(A_i,P)$，从而在执行签名操作时无须执行双线性配对就可以得到 $e(T_3,P)$ 的值。总的来说，每产生一个群签名需要计算 8 个指数（或多指数，multi-exponentiation），而不涉及任何双线性对运算。由于 $e(T_3,P)^{s_x}e(T_3,W)^{c}=e(T_3,cW+s_xP)$，在执行签名验证操作时，验证者只需执行 6 个多指数运算和 1 个双线性对运算。

8.3.6　成员撤销

任何一个实用的群签名体制必须考虑到群成员动态流动问题，即群成员不仅可以加入，还可以离开或在任何时间被群管理员取消签名权限，后者就是群签名体制中的群成员撤销问题。所谓安全有效地撤销群成员，是指一种机制，使某个群成员被撤销后，他拥有的私钥和成员证书不能再用于产生有效的群签名。

群成员撤销问题是群签名研究中的一个重要方向，目前研究者已提出了多种群成员撤销机制。其中一类常用的成员撤销方法是群管理员发布一个身份撤销列表（revocation list）给所有群成员和群签名验证者。对于 8.3.5 节的短的群签名体制，可以给出下面的成员撤销方法。

注意到，方案中的群公钥为 gpk=(P,H,U,V,W)，式中 H,U,V 为群 G_1 中的随机元素，$W=\gamma P,\gamma\in_R Z_p^*$。群成员的私钥形如 (A_i,x_i)，其中 $A_i=\dfrac{1}{\gamma+x_i}P$。假设要在不影响其他群成员签名能力的前提下撤销群成员 $1,\cdots,r$ 的签名能力。群管理员公布一个撤销列表 RL，这个列表由所有被撤销成员的私钥构成，即 RL=$\{(A_1,x_1),\cdots,(A_r,x_r)\}$。群管理员将撤销列表发送给所有群成员和系统中的所有群签名验证者。这些人可以根据该列表更新群公钥或签名私钥。群公钥的更新方法如下。

不失一般性，假设第一次撤销群成员 1。撤销群成员 1 后的群公钥构造方法为：任何人都可以计算 $P_1'=A_1,W'=P-x_1A_1$，然后将 gpk$'=(P',H,U,V,W')$ 作为新的群公钥。容易检查，$W'=P-x_1A_1=P-x_1\dfrac{1}{\gamma+x_1}P=\gamma\cdot\dfrac{1}{\gamma+x_1}P=\gamma P'$，即 gpk$'=(P',H,U,V,W')$ 满足群公钥需要具备的形式。这个过程重复 r 次，即可得到撤销群成员 $1,\cdots,r$ 后的群公钥。

对于群成员$(r+1),\cdots,n$，他们按照下述方法更新自己的私钥。假设第一次撤销群成员 1，

此时某个合法成员私钥为 (A,x)，他计算 $A'=\frac{1}{x-x_1}A_1-\frac{1}{x-x_1}A$，置 (A',x) 为自己新的私钥。此时，$(\gamma+x)A'=(\gamma+x)\frac{1}{x-x_1}A_1-(\gamma+x)\frac{1}{x-x_1}A=\frac{\gamma+x}{x-x_1}\frac{1}{\gamma+x_1}P-\frac{\gamma+x}{x-x_1}\frac{1}{\gamma+x}P$ $=\frac{1}{\gamma+x_1}P$，即所得的 (A',x) 满足作为群成员私钥应具备的形式。这个过程重复 r 次，即可得到撤销群成员 $1,\cdots,r$ 后的该合法群成员新的私钥。

容易看到，在这个撤销方法中，每当撤销一个群成员的签名能力，群公钥和剩余群成员的私钥需要相应地更新。根据新成员加入或退出时群公钥或群成员私钥是否变化，可以将群签名方案分为动态（dynamic）群签名和静态（static）群签名两种。当一个成员加入或撤销时，需要更新群公钥或群成员的私钥，则称这样的群签名体制为静态群签名；反之，则称为动态群签名。

8.4　环签名

环签名（ring signature）的概念是 Rivest 等人在 2001 年的亚洲密码学年会上提出的。在一个环签名体制中，签名人可以随意挑选（n–1）个人，这些人连同他自身构成一个含 n 个人的集合，该集合被称为环。然后他可以用自己的私钥和其他（n–1）个人的公钥一起对某个消息 m 执行环签名操作，产生签名 σ。接到消息-签名对 (m,σ) 后，任意一个验证者执行环签名验证算法，如果签名有效，则可以确信该签名是由这个环中某个签名者产生的，但他无法识别该签名人的身份。由此可见，环签名体制能够实现签名人匿名性。与群签名提供的匿名性不同，在环签名体制中不存在一个具有撤销匿名性的管理者，因此，环签名体制提供的是一种不可撤销的匿名性。给定一个环签名，除了签名人外，任何人均无法获知产生该签名的签名人身份。

环签名的这一性质在某些场合是适用的。例如，假设 Bob 是某个国家的内阁成员，他知道一条关于首相的丑闻，并想将这个丑闻泄露给报刊记者。Bob 自然不能对这个丑闻使用普通数字签名，因为这样会暴露自己的身份。Bob 也不能随便让一个平民百姓去告诉记者，因为这样的检举不具有可信性。此时，Bob 可以选择所有内阁成员，连同自己一起构成一个环，然后使用环签名体制，对该丑闻进行环签名，将产生的消息-签名对发送给记者。记者接到后执行环签名验证算法，之后可以确信这个消息是由内阁中的某个成员泄露的，从而具有很大的可信性；但同时记者无从获知检举人的身份，Bob 被猜中的机会只是 $\frac{1}{n}$，从而实现了匿名检举的目的。

8.4.1　环签名的基本概念

给定一个环 $U=\{U_1,U_2,\cdots,U_n\}$，环中每个用户的公钥-私钥对为 $(\mathrm{pk}_i,\mathrm{sk}_i),i=1,\cdots,n$。不失一般性，假设 $U_k(1\leqslant k\leqslant n)$ 是签名人。除密钥生成算法外，一个环签名体制还包含环签名产生算法 ring-sign 和环签名验证算法 ring-verify。

另外，直观上可以看到，任何人都可以任选一个随机数 $r \in_R Z_N^*$，进而构造出 $\mathrm{ID}t^{H(t\|m)}$，但他要计算 $\mathrm{ID}t^{H(t\|m)}$ 的 e 次方根 s（在 mod N 意义下）是困难的，除非他知道 ID 所对应的私钥。而这个唯一的私钥由 PKG 秘密发送给身份信息为 ID 的签名人，因此，如果一个数字签名 (t,s) 通过了算法 Verify 的验证，我们就可以相信这个签名是身份信息为 ID 的签名人产生的。

8.6 民主群签名

群签名和环签名是关系比较密切的两个概念：共同点是它们都实现了某个个体代表一群人对消息进行签名的操作，但签名的验证者不知道签名是由群体中的哪个成员产生的；不同点是，群签名实现的签名人匿名性是可撤销的，而环签名实现的签名人匿名性是不可撤销的。

群签名中的撤销匿名性操作是由该群体的群管理员执行的，在环签名中不存在这样的特殊成员。由于群管理员是一个集权性（centralized）的成员，它比普通群成员具有更高的权限。然而，现实中我们经常希望一个群体中的成员的地位都是平等的，不要出现一个凌驾于其他成员之上的特殊个体，环签名中的群体就具有这样的特点。本节的目的就是要在这样的群体中实现可撤销匿名性。民主群签名（democratic group signature）是具有这种性质的一种特殊群体签名，它是由 Manulis 于 2006 年提出的。在一个民主群签名体制中，群体中任何成员都可以代表该群体产生群签名；群体内任何成员都可以从给定的有效群签名中恢复出产生该签名的群成员的身份。

具体描述民主群签名定义之前，我们先介绍一个应用背景。随着经济全球化浪潮的到来，许多企业都希望将自己的业务不断拓展至新兴市场，而与其他企业（特别是新兴市场中的本土企业）一起建立合资公司是扩大市场份额、占领新市场的有效方式。在这种合资公司中，各个企业提供资金、人员与技术，地位平等。合资公司中的人员负责做出资金流向等决策。为公平起见，合资公司中的其他企业应该有能力检查是谁做出了资金发放的决定。民主群签名适用于这样的场合。

8.6.1 民主群签名的定义

一个民主群签名体制 DGS=(Setup, Join, Leave, Sign, Verify, Trace, VerifyTrace)由以下 7 个算法构成。

1. 群体初始化算法 Setup

假设 n 个用户构成一个群体，他们以协作的方式生成群体的初始信息。算法的公共输出为群公钥 $Y_{[0]}$，秘密输出为各成员的签名钥 $x_{i[0]}, i=1,\cdots,n$ 和用于撤销签名人匿名性的追踪陷门 $\hat{x}_{[0]}$。

2. 成员加入算法 Join

成员加入算法由欲加入该群体的某个用户与构成当前群体的群成员交互执行。令 t 为标记当前群体的计数器（本算法执行之前），t=0 对应的是初始群体，初始群体的群公

钥为 $Y_{[0]}$。令 n 为当前群体中成员个数（本算法执行之前），算法的公开输出为更新后的群公钥 $Y_{[t+1]}$，秘密输出为更新后的各成员签名私钥 $x_{i[t+1]}, i=1,\cdots,(n+1)$ 和用于撤销签名人匿名性的追踪陷门 $\hat{x}_{[t+1]}$。

3. 成员退出算法 Leave

如果群体中某个成员退出，则剩余成员需要协作执行本算法以更新群公钥和签名私钥。令 t 为标记当前群体的计数器（本算法执行之前），t=0 对应的是初始群体，初始群体的群公钥为 $Y_{[0]}$。令 n 为当前群体中成员个数（本算法执行之前），算法的公开输出为更新后的群公钥 $Y_{[t+1]}$，秘密输出为更新后的各成员签名私钥 $x_{i[t+1]}, i=1,\cdots,(n-1)$ 和用于撤销签名人匿名性的追踪陷门 $\hat{x}_{[t+1]}$。

4. 签名生成算法 Sign

假设在 t 时刻，签名人要代表当前群体对消息 m 执行数字签名操作，其输入为签名人的签名私钥 $x_{i[t]}$、消息 m 和当前群公钥 $Y_{[t]}$，输出即为签名人代表当前群体产生的民主群签名，记作 $\sigma \leftarrow \mathrm{Sign}(m, Y_{[t]}, x_{i[t]})$。

5. 签名验证算法 Verify

在任意 t 时刻，给定待验证的消息-签名对 (m,σ) 和当前群公钥 $Y_{[t]}$，算法输出 1 当且仅当 σ 是由某个群成员使用签名私钥执行签名算法产生的。

6. 签名人追踪算法 Trace

在任意 t 时刻，给定消息-签名对 (m,σ)、当前群公钥 $Y_{[t]}$ 和追踪陷门 $\hat{x}_{[t]}$，算法输出产生签名 σ 的群成员的身份 ID_i 和这一事实的证据 π，记作 $(\mathrm{ID}_i, \pi) \leftarrow \mathrm{Trace}(m, \sigma, Y_{[t]}, \hat{x}_{[t]})$。

7. 追踪验证算法 VerifyTrace

在任意 t 时刻，给定消息-签名对 (m,σ)、当前群公钥 $Y_{[t]}$、某一群成员身份 ID_i 和 ID_i 产生 σ 的证据 π，算法输出 1 当且仅当 ID_i 和 π 是由签名人追踪算法 Trace 产生的，即 $(\mathrm{ID}_i, \pi) \leftarrow \mathrm{Trace}(m, \sigma, Y_{[t]}, \hat{x}_{[t]})$。

根据上述定义容易看出，如果当前群体中的某个群成员代表群体产生了一个群签名，在发生争端的情况下群体中任何成员均可执行签名追踪算法揭示该签名人的身份；为保证群体外的用户不具有这样的能力，要求追踪陷门 $\hat{x}_{[t]}$ 随着成员加入和退出群体的动态变化而更新。结合起来说，给定一个所有成员地位平等的群体和该群体的一个签名 σ，群体中的任何成员且只有群体中的成员能够恢复出产生 σ 的签名人身份。这正是本节开头部分所要的结果。根据具体构造方案的不同，群体中群成员的签名私钥可以不随着成员加入和退出的动态变化而改变。

8.6.2 民主群签名的安全性需求

给定一个民主群签名体制 DGS= (Setup, Join, Leave, Sign, Verify, Trace, VerifyTrace)，关于其安全性，我们需要考虑以下几点。

1. 正确性

我们称 DGS 具有正确性是指，在任意 t 时刻，对于算法 Setup、Join 或 Leave 返回的 $Y_{[t]}$、$x_{i[t]}$、$\hat{x}_{[t]}$ 和任意签名 $\sigma=\text{Sign}(m,Y_{[t]},x_{i[t]})$，下面的等式总成立：

$$\text{Verify}(m,\sigma,Y_{[t]})=1$$

$$\text{Trace}(m,\sigma,Y_{[t]},\hat{x}_{[t]})=(\text{ID}_i,\pi)$$

$$\text{VerifyTrace}(m,\sigma,Y_{[t]},\text{ID}_i,\pi)=1$$

2. 不可伪造性

不知道群体中任意一个签名私钥的人无法产生一个能够通过签名验证算法的数字签名。

3. 可追踪性

如果群体中的某个成员对一个消息产生了群签名，则他必定会被群体中任意一个其他成员通过签名追踪算法追踪出来。任何群成员，如 ID_i，要对某消息产生一个群签名 $\sigma=\text{Sign}(m,Y_{[t]},x_{i[t]})$，使得 $\text{Trace}(m,\sigma,Y_{[t]},\hat{x}_{[t]})=(\text{ID}_j,\pi),i\neq j$ 在计算上是不可行的。任何群成员，如 ID_i，要对某消息产生一个群签名 $\sigma=\text{Sign}(m,Y_{[t]},x_{i[t]})$，使得算法 $\text{Trace}(m,\sigma,Y_{[t]},\hat{x}_{[t]})$ 的输出为群体外的用户在计算上也是不可行的。

4. 匿名性

给定一个民主群签名，不知道追踪陷门的任何验证者都无法确定出产生该签名的群成员的身份。

8.6.3 Manulis 民主群签名

下面介绍 Manulis 的民主群签名体制，以便具体了解如何在对等的群体中实现可撤销的匿名性。该方案的主要思想是使用群密钥协商协议来保证群成员动态变化后群公钥、群秘密的相应更新，协商后的群秘密用做追踪陷门，因此当前群体内任何成员均可对群签名执行追踪操作，不在该群体内的任意用户因不知群秘密而无法做到这一点。

为此，假设已经有一个安全的群密钥协商体制 GKA=(S,J,L)，该体制包含三个算法：群体初始化算法 S、成员加入算法 J 和成员退出算法 L。算法 S 以当前群成员的公钥/私钥为输入，通过所有群成员的交互生成一个共享的群密钥；现有群成员与欲加入该群的新成员执行交互算法 J：原有群成员的输入包括自己的原签名私钥、全部原有群成员的公钥，新成员的输入包括自己的公钥/私钥，算法输出为更新后的共享群密钥和（可能更新的）各成员公钥/私钥；算法 L 的输入包括剩余群成员的公钥/私钥、退出成员的公钥，输出为更新后的共享群密钥和（可能更新的）剩余各成员的公钥/私钥。

对这个群密钥协商体制的安全性要求是：协商出的群密钥只能由群成员掌握，不属于群体内的任何人都不知道该密钥的值；新加入群体的成员不知道他加入之前群体的群密钥；退出群体后的成员不知道他退出之后群体的群密钥。满足这些条件的任何方案均可用于将要介绍的 Manulis 的民主群签名体制中，所以我们对具体的群密钥协商体制不做描述，感兴趣的读者可以参见参考文献[12]。

1. 系统初始化算法 Setup

假设初始时有 n 个用户要形成一个群体。令 $G=<g>$ 为一个素数 q 阶循环群，其上的离散对数问题是困难的。每个成员令 $t=0$，选取 $x_{i[0]} \in_R Z_q$ 作为自己的签名私钥，计算对应的签名公钥 $z_{i[0]}=g^{x_{i[0]}}$。n 个用户以自己的公钥/私钥为输入执行一个群密钥协商体制的初始化算法，从而得到一个共享的群秘密 $\hat{x}_{[0]} \in_R Z_q$，计算 $\hat{y}_{[0]}=g^{\hat{x}_{[0]}}$，令 $Z_{[0]}=\{z_{1[0]},\cdots,z_{n[0]}\}$，则群公钥即为 $Y_{[0]}=\{\hat{y}_{[0]},Z_{[0]}\}$。$Y_{[0]}$ 是算法 Setup 的公共输出，$\hat{x}_{[0]}$，$x_{1[0]},\cdots,x_{n[0]}$ 为秘密输出，群秘密即为追踪陷门。

2. 加入算法 Join

假设 t 为标记当前群体的计数器值，n 为当前群体含有成员的个数。另有一用户 ID_u 要加入该群体，因此本算法执行结束后群体所含成员个数为（n+1）。ID_u 选取随机数 $x_{u[t+1]} \in_R Z_q$ 作为自己的签名私钥，计算对应的签名公钥 $z_{u[t+1]}=g^{x_{u[t+1]}}$。新群体的构成可以看作是：在原有的 n 个成员中，每人均以自己的签名私钥 $x_{i[t]}$ 和 $Z_{[t]}=\{z_{1[t]},\cdots,z_{n[t]}\}$ 为输入执行群密钥协商协议的加入算法，获得更新后的群秘密 $\hat{x}_{[t+1]}$ 和签名私钥 $x_{i[t+1]}$；ID_u 以自己的签名私钥 $x_{u[t+1]}$ 和公钥 $z_{u[t+1]}$ 为输入执行密钥协商协议的加入算法，获得群秘密 $\hat{x}_{[t+1]}$。令 $\hat{y}_{[t+1]}=g^{\hat{x}_{[t+1]}}$，$Z_{[t+1]}=\{z_{1[t+1]},\cdots,z_{n[t+1]}\}$，则更新后的群公钥为 $Y_{[t+1]}=\{\hat{y}_{[t+1]},Z_{[t+1]}\}$。最后，所有（$n$+1）个成员计算（$t$+1）以更新标记群体的计数器值。群秘密即为追踪陷门。

3. 退出算法 Leave

假设 t 为标记当前群体的计数器值，n 为当前群体含有成员的个数。某一群成员 ID_u 欲退出该群体，剩余的（n–1）个成员需要执行下述步骤：每个成员以自己的签名私钥 $x_{i[t]}$、ID_u 的签名公钥 $z_{u[t]}$ 和 $Z_{[t]}$ 为输入执行群密钥协商协议的退出算法，得到更新后的群秘密 $\hat{x}_{[t+1]}$ 和签名私钥 $x_{i[t+1]}$。群秘密即为追踪陷门。令 $\hat{y}_{[t+1]}=g^{\hat{x}_{[t+1]}}$，$Z_{[t+1]}=\{z_{1[t+1]},\cdots,z_{n-1[t+1]}\}$，则更新后的群公钥即为 $Y_{[t+1]}=\{\hat{y}_{[t+1]},Z_{[t+1]}\}$。最后，所有（$n$–1）个成员计算（$t$+1）以更新标记群体的计数器值。

4. 签名算法 Sign

假设 $m \in \{0,1\}^*$ 为 t 时刻待签名的消息，算法的输入为 $x_{i[t]}$、m、$Y_{[t]}=\{\hat{y}_{[t]},Z_{[t]}\}$，然后执行如下计算：

（1）选取随机数 $r \in_R Z_q$，计算 $\tilde{g}=g^r$，$\tilde{y}=\hat{y}_{[t]}{}^r z_{i[t]}$；

（2）选取随机数 $r_{u_1},r_{u_2},\cdots,r_{u_n},r_{v_1},r_{v_2},\cdots,r_{v_n} \in_R Z_q$，$c_1,c_2,\cdots,c_{i-1},c_{i+1},c_n \in_R \{0,1\}^k$；

（3）$u_i=\hat{y}_{[t]}{}^{r_{ui}},v_i=g^{r_{vi}},w_i=g^{r_{ui}}$；

（4）$j \neq i$ 时，$u_j=\tilde{y}^{c_j}\hat{y}_{[t]}{}^{r_{uj}}g^{r_{vj}},v_j=z_{j[t]}{}^{c_j}g^{r_{vj}},w_j=\tilde{g}^{c_j}g^{r_{uj}}$；

（5）c_i 满足：$c_1 \oplus \cdots \oplus c_n=H(g,\hat{y}_{[t]},\tilde{g},\tilde{y},u_1,\cdots,u_{i-1},u_iv_i,u_{i+1},\cdots,u_n,v_1,\cdots,v_n,w_1,\cdots,w_n,m)$；

（6）$s_{u_i}=r_{u_i}-c_ir$，$s_{v_i}=r_{v_i}-c_ix_{i[t]}$；

（7）$j \neq i$ 时，$s_{u_j}=r_{u_j},s_{v_j}=r_{v_j}$；

（8）输出 $(\tilde{g},\tilde{y},c_1,\cdots,c_n,s_{u_1},\cdots,s_{u_n},s_{v_1},\cdots,s_{v_n})$ 作为对消息 m 的签名。

5. 签名验证算法 Verify

给定 t 时刻待验证的签名 $(\tilde{g},\tilde{y},c_1,\cdots,c_n,s_{u_1},\cdots,s_{u_n},s_{v_1},\cdots,s_{v_n})$、群公钥 $Y_{[t]}=\{\hat{y}_{[t]},Z_{[t]}\}$ 和对应的消息 m，验证算法检查下面的等式是否成立：

$$c_1\oplus\cdots\oplus c_n$$
$$=H(g,\hat{y}_{[t]},\tilde{g},\tilde{y},\tilde{y}^{c_1}\hat{y}_{[t]}^{s_{u1}}g^{s_{v1}},\cdots,\tilde{y}^{c_n}\hat{y}_{[t]}^{s_{un}}g^{s_{vn}},z_{1[t]}{}^{c_1}g^{s_{v1}},\cdots,z_{n[t]}{}^{c_n}g^{s_{vn}},\tilde{g}^{c_1}g^{s_{u1}},\cdots,\tilde{g}^{c_n}g^{s_{un}},m)$$

如果成立，则接受该签名有效；否则拒绝。

6. 签名追踪算法 Trace

给定 t 时刻待追踪的签名 $(\tilde{g},\tilde{y},c_1,\cdots,c_n,s_{u_1},\cdots,s_{u_n},s_{v_1},\cdots,s_{v_n})$、群公钥 $Y_{[t]}=\{\hat{y}_{[t]},Z_{[t]}\}$、对应的消息 m 及追踪陷门 $\hat{x}_{[t]}$，追踪算法需要完成以下步骤：

（1）检查签名为有效，即 $\text{Verify}(m,\sigma,Y_{[t]})=1$，若无效则拒绝该签名；

（2）计算 $V_1=\tilde{g}^{\hat{x}_{[t]}}$；

（3）$z_{i[t]}=\dfrac{\tilde{y}}{V_1}$；

（4）检查 $z_{i[t]}\in Z_{[t]}$ 是否成立，否则拒绝；

（5）输出签名人的签名公钥 $z_{i[t]}$（或其身份 ID_i）和相应的证据 $\pi=(V_1,V_2)$，式中 V_2 是指关于两个等式的指数相同的知识证明：$\hat{y}_{[t]}=g^{\alpha}$，$V_1=\tilde{g}^{\alpha}$。

7. 验证追踪算法 VerifyTrace

给定 t 时刻的签名 $(\tilde{g},\tilde{y},c_1,\cdots,c_n,s_{u_1},\cdots,s_{u_n},s_{v_1},\cdots,s_{v_n})$、消息 m、群公钥 $Y_{[t]}=\{\hat{y}_{[t]},Z_{[t]}\}$、待验证的签名人身份 ID_i 和相应的证据 $\pi=(V_1,V_2)$，验证追踪算法首先检查签名为有效，即 $\text{Verify}(m,\sigma,Y_{[t]})=1$；然后确认 $\tilde{y}=z_{i[t]}V_1$ 成立；最后检查关于 $\hat{y}_{[t]}=g^{\alpha}$，$V_1=\tilde{g}^{\alpha}$ 的知识证明 V_2 有效。如果三次检查均通过，则表明追踪的签名人身份正确，否则拒绝该身份。

8.7　具有门限追踪性的民主群签名

民主群签名融合了普通群签名和环签名的特点，提供了群体内任意一个成员的签名追踪能力。不过，这种追踪能力有时可能过于自由而导致不受控制的滥用。例如，在合资公司中，如果参与组建该公司的每个企业均要对每笔资金流向进行追踪，并依此对该笔资金决策的正确性做出评断，这与合资公司内每个企业均希望每笔资金流向都带来利益最大化的动机是矛盾的。因此，如果能够提供一种民主群签名体制，若干个群成员（如 t 个）一起才能够进行签名追踪，少于 t 个成员则无法完成这一操作，这种民主群签名将更适用于合资公司的类似应用场景。这种做法具有较大的灵活性，合资公司可以根据具体需要决定 t 的取值。此即本节所要描述的具有门限追踪性的民主群签名。

Manulis 构造的民主群签名体制使用了群密钥协商协议作为子模块，协商出的群秘密作为追踪陷门。与 Manulis 的方法不同，我们将使用公开可验证秘密分享技术来实现门限

追踪性。所设计的体制 DGS=(Setup, KeyGen, Sign, Verify, Trace)具体包括 5 个算法：群体初始化算法、密钥生成算法、群签名生成算法、群签名验证算法、追踪算法及验证追踪算法。

8.7.1 群体初始化

在群体初始化算法 Setup 中，可信中心生成系统公开参数，即根据给定的安全参数λ，系统可信中心生成系统参数G,q,g,h,H，其中，q是长为λ比特的素数，G为q阶乘法循环群，g和h为G上的任意两个生成元，$H:\{0,1\}^* \to Z_q$为密码学意义上安全的哈希函数，其中，$Z_q=\{0,1,\cdots,q-1\}$，G,q,g,h,H一起构成系统的公开参数。

8.7.2 密钥生成

密钥生成算法 KeyGen 生成用户公钥、私钥。假设n个用户构成一个群体，群成员ID_i选取随机数$x_i \in Z_q$作为私钥，计算$y_i=h^{x_i}$作为公钥，该用户将自己的公钥公开注册于可信中心，以便系统中其他用户可以在可信中心处检索。拥有公钥/私钥$(x_i,y_i),i=1,\cdots,n$的n个用户构成一个群体$U=\{\mathrm{ID}_1,\mathrm{ID}_2,\cdots,\mathrm{ID}_n\}$，$U$中的每个用户均有权以群体的名义对任意消息产生民主群签名。

8.7.3 签名生成

在群签名生成算法 Sign 中，某一群成员$\mathrm{ID}_k(1\leqslant k\leqslant n)$作为签名人代表群体$U$对消息$m$产生民主群签名，具体步骤如下。

1. 计算秘密分享

签名人ID_k以秘密分发者的身份执行一个公开可验证的秘密分享方案，实现对秘密值h^s的(t,n)秘密分发，具体如下：

（1）在集合Z_q中选择随机数$s,w_i,1\leqslant i\leqslant n$和一个$q$元域上常数项为$s$的（$t$–1）次随机多项式$p(x)=\sum_{j=0}^{t-1}\alpha_j x^j$，满足条件$\alpha_0=s$，签名人$\mathrm{ID}_k$计算并广播自己对该多项式的承诺，即$\tau=g^{\alpha_0}$，$\tau_j=g^{\alpha_j},1\leqslant j\leqslant t-1$，利用这些承诺值计算$\chi_i=\prod_{j=0}^{t-1}\tau_j^{\ i^j},i=1,2,\cdots,n$，式中$\tau_0=\tau$；

（2）为使群成员最终能够恢复出秘密值h^s，该签名人计算多项式值$p(i)$并用该值加密第i个成员的公钥，即计算并公布$\eta_i=y_i{}^{p(i)},1\leqslant i\leqslant n$；

（3）利用所选择的随机数、所有群成员的公钥及公开参数计算$a_{i1}=g^{w_i}$，$a_{i2}=y_i{}^{w_i}$，利用所选择的哈希函数计算哈希值$e=H(\chi_1,\cdots,\chi_n,\eta_1,\cdots,\eta_n,a_{11},\cdots,a_{n1},a_{12},\cdots,a_{n2})$并由该哈希值和多项式值获得响应值$r_i=w_i-p(i)e$，$1\leqslant i\leqslant n$；

（4）作为秘密分享部分的输出，签名人置$\mathrm{share}=(\tau,\tau_1,\cdots,\tau_{t-1},\eta_1,\cdots,\eta_n,e,r_1,\cdots,r_n)$。

此处，签名人对不同的消息执行民主群签名操作时需要使用不同的随机数s，以便在

群体 U 中分发不同的秘密值 h^s，若分发相同的秘密值，将可能导致意想不到的潜在威胁。

任何人在收到 share 后都能够确信秘密分发者是正确地产生了其输出结果，秘密分发者要想欺骗秘密接收者接受一个假秘密值在计算上是不可行的。

2．计算数字签名

签名者利用所分发的秘密和自己的私钥对消息 m 产生数字签名，具体如下：

（1）计算 $\gamma = g^{x_k}, c = h^s y_k$；

（2）选择 $r_{k1}, r_{k2}, z_{i1}, z_{i2}, \rho_i \in_R Z_q, i = 1,2,\cdots,n, i \neq k$，计算 $l_{i1} = H(m, \tau, \frac{c}{y_i})$，

$l_{i2} = H(m, \gamma, y_i)$，$u_{i1} = (g^{l_{i1}} h)^{z_{i1}} (\tau^{l_{i1}} \frac{c}{y_i})^{\rho_i}$，$u_{i2} = (h^{l_{i2}} g)^{z_{i2}} (y_i^{l_{i2}} \gamma)^{\rho_i}$，$i = 1,2,\cdots,n, i \neq k$；

（3）$l_{k1} = H(m, \tau, h^s), l_{k2} = H(m, \gamma, y_k)$，$u_{k1} = (g^{l_{k1}} h)^{r_{k1}}, u_{k2} = (h^{l_{k2}} g)^{r_{k2}}$，

$\rho_k = H(m, \tau, c, \gamma, u_{11}, \cdots, u_{n1}, u_{12}, \cdots, u_{n2}) - \sum_{j \neq k} \rho_j$，$z_{k1} = r_{k1} - \rho_k s, z_{k2} = r_{k2} - \rho_k x_k$；

（4）输出 $\text{sig} = (\gamma, c, \rho_1, \cdots, \rho_n, z_{11}, \cdots, z_{n1}, z_{12}, \cdots, z_{n2})$。

签名人在群体 U 中分发的秘密值 h^s 及其中的随机数 s 在签名过程中均被使用到，对不同的消息执行签名时使用的整数 s 应该是随机的，否则公开可验证秘密分享部分与签名人使用自己的私钥计算签名部分就是相互孤立开来的，这容易带来潜在攻击。

签名人以自己的私钥和群体 U 中所有成员的公钥计算出签名部分 sig，这种方法使任意签名接收者获得签名 sig 后都能够确信是群体 U 产生了该签名，但是要想精确知晓群体 U 中哪个成员产生了 sig 在计算上是不可行的。由于签名人在计算中使用了自己的私钥，而这个私钥只有他自己才知道，所以不知道该私钥的人无法产生这样的签名。

3．输出民主群签名

将秘密分享部分和数字签名部分组成一个二元组（share,sig）作为群成员 ID_k 代表群体 U 对消息 m 最终产生的民主群签名。

8.7.4　签名验证

在群签名验证算法 Verify 中，任何人均可以判定给定的民主群签名是否确实是由群体 U 中的某个群成员代表整个群体 U 产生的，具体方法如下。

（1）验证秘密分享部分是否有效：计算 $\chi_i = \prod_{j=0}^{t-1} \tau_j^{i^j}, i = 1,2,\cdots,n$，式中 $\tau_0 = \tau$，进而重构 $a_{i1} = g^{r_i} \chi_i^e$，$a_{i2} = y_i^{r_i} \eta_i^e$，最后检查等式 $e = H(\chi_1, \cdots, \chi_n, \eta_1, \cdots, \eta_n, a_{11}, \cdots, a_{n1}, a_{12}, \cdots, a_{n2})$ 是否成立，如果不成立，则意味着原签名人对 h^s 的秘密分享不正确，从而拒绝该签名；否则继续执行；

（2）验证数字签名部分是否有效：计算哈希值 $l_{i1} = H(m, \tau, \frac{c}{y_i}), l_{i2} = H(m, \gamma, y_i)$，进而

重构值 $u_{i1}=(g^{l_{i1}}h)^{z_{i1}}(\tau^{l_{i1}}\frac{c}{y_i})^{\rho_i}$，$u_{i2}=(h^{l_{i2}}g)^{z_{i2}}(y_i^{l_{i2}}\gamma)^{\rho_i}$，$i=1,2,\cdots,n$，最后检查等式 $\sum_{i=1}^{n}\rho_i=H(m,\tau,c,\gamma,u_{11},\cdots,u_{n1},u_{12},\cdots,u_{n2})$ 是否成立，如果不成立，则拒绝该签名；否则接受该民主群签名为有效。

由于只有群体 U 中的所有成员的公钥才能够使得（2）中最后要检查的等式成立，通过这样的计算，验证者能够确信签名来自群体 U。但是，由于在这一验证过程中群体 U 中所有成员的地位都是对称的，所以，验证者并不能确切地知道群体 U 中的哪个成员产生了这个签名。这正是本方案能够提供签名者匿名性的原因。

8.7.5 签名人追踪

签名人追踪算法 Trace 使得在发生争端的情况下（如一个签名通过了签名验证过程，但是群体 U 中的所有成员均否认自己产生了该签名）由群体 U 中不少于 t 个成员一起执行本步骤，以自己的私钥为输入，揭晓产生该签名的签名人的真实身份，具体方法如下。

（1）验证民主群签名是否有效：t 个成员 $\{\mathrm{ID}_1,\mathrm{ID}_2,\cdots,\mathrm{ID}_t\}$ 以签名验证者的身份验证该签名是否有效，如果该签名无效，则意味着该签名不是群体 U 产生的，所以不需要由群体 U 执行任何追踪计算，否则继续下面的步骤。

（2）重构秘密：群成员 $\mathrm{ID}_i(i=1,\cdots,t)$ 利用自己的私钥为输入计算并公布 $\xi_i=\eta_i^{x_i^{-1}}$；参与运算的群成员根据这些节点广播的数据 ξ_i 执行拉格朗日插值运算，即计算 $\lambda_i=\prod_{j=1,\cdots,t,j\neq i}\frac{j}{j-i},i=1,\cdots,t$，进而重构出由签名人分发的秘密值 $\mu=\prod_{i=1}^{t}\xi_i^{\lambda_i}$。

（3）恢复签名人身份：利用重构的秘密值执行解密运算并恢复出签名人的身份。所述的解密运算即利用恢复出的秘密值的逆元 μ^{-1} 与密文 c 做乘积运算 $y=c\mu^{-1}$，在群体 $U=\{\mathrm{ID}_1,\mathrm{ID}_2,\cdots,\mathrm{ID}_n\}$ 中查找公钥等于 y 的群成员即为产生该签名的真正签名人。

本节介绍的民主群签名体制不需要集权式的群管理员，只由所有群成员一起构成一个群体，群体中所有成员之间的地位都是对称的，从而消除了集权式的实体；群体外的用户无法产生该群体的民主群签名；避免了群体中的任何成员假冒群体中的其他成员产生一个有效的民主群签名；发生争端时，当且仅当不少于适当个数 t 的群成员通过协作计算才能够恢复出真实签名人的身份，少于 t 个群成员欲执行追踪操作在计算上是不可行的，群体外的用户要执行追踪操作在计算上也是不可行的。

以上即为具有门限追踪性的民主群签名体制的完整描述。读者可以将该体制稍加修改，增加一个追踪验证算法 VerifyTrace，任何人都可以利用该算法验证追踪算法 Trace 的输出是否正确。

8.8 多重签名

在一个多重签名（multisignature，多方签名，多签名）体制中，多个签名人（如ℓ个）一起以协作的方式对同一个消息 m 产生数字签名σ。共同产生σ所需的计算代价远小于产

生$\sigma_1,\sigma_2,\ldots,\sigma_\ell$所需的计算量，且$\sigma$的长度远小于$\sigma_1\|\sigma_2\|\cdots\|\sigma_\ell$的长度，其中$\sigma_1,\sigma_2,\ldots,\sigma_\ell$为这些签名人各自独立地对 m 产生的普通数字签名。多重签名可广泛应用于包括蜂窝电话、PDA、RFID、传感器等资源受限环境。

8.8.1　流氓密钥攻击

多重签名的概念最早是由 Itakura 和 Nakamur 提出的。在过去 30 年里，学者对多重数字签名进行了广泛的研究。

多重签名方案通常利用公钥密码学中某些具有同态性的密码计算技巧来实现签名的聚合。由于缺乏可证明安全性分析，早期的多签名方案大多易受攻击，其中最典型的就是流氓密钥攻击（伪造密钥攻击，rogue key attack）。在这种攻击中，敌手按照一定的方式选择自己的公钥，进而伪造一个多重签名。通常敌手所选的公钥是一个关于其他成员的公钥的函数值，且敌手可以不知道所选公钥对应的私钥。

举例来说，如果某方案采用的是 PPK 模型而不是 KOSK 模型（见 8.8.2 节），则可以轻易发起如下的流氓密钥攻击。令第 i 个用户的私钥和公钥分别为 $\mathrm{pk}_i=g^{x_i},\mathrm{sk}=x_i$，消息 m 的多重签名为 $h(m)^{\sum x_i}$，则攻击者允许注册公钥 $\mathrm{pk}=g^s\cdot\prod\mathrm{pk}_i^{-1}=g^{(s-\sum x_i)}$，这里攻击者知道 s 且有 $\mathrm{pk}\prod\mathrm{pk}_i=g^s$。这样攻击者就可以在无须其他签名者参与的情况下，利用 s 伪造一个由他和已有其他用户一起产生的、任意消息 m 的多重签名 $h(m)^{(x+\sum x_i)}=h(m)^s$。

研究表明，在多重签名中采用的抵抗流氓密钥攻击技术一般也可以直接或稍加改动地用到其他多方认证的方案中。

8.8.2　安全模型

人们提出几种模型来克服数学同态性带来的流氓密钥攻击问题。

1. DKG 模型

“专用密钥生成”（Dedicated Key Generatation）协议是 Micali 等在 2001 年的 ACM CCS 会议上提出的。该协议要求所有成员交互式产生用于签名的公、私钥，以确保攻击者不能随意选取其他成员的公钥为输入的函数值作为其公钥，从而避免了流氓密钥攻击。基于这个信任模型的协议有自身的缺点：签名组不能动态更新，新成员的加入都要重新执行“专用密钥生成”协议；协议产生的公钥等参数复杂、巨大，随签名组大小而改变，计算效率不高。

2. KOSK 模型

2003 年 Boldyreva 在 ROM 模型下设计了一个较为高效的多方签名方案：签名和验证算法高效；公钥相对简单；产生的多签名较短。为证明此方案的安全性，需要使用“私钥知识”（KOSK 模型）安全模型。该模型在多方签名的安全性论证上引入了一个可信第三方（如传统公钥基础设施的 CA）的信任模型假设。在此模型中，CA 要求注册公钥的用户在申请公钥数字证书时必须提供“私钥知识”的证明（有的模型中甚至要求直接向