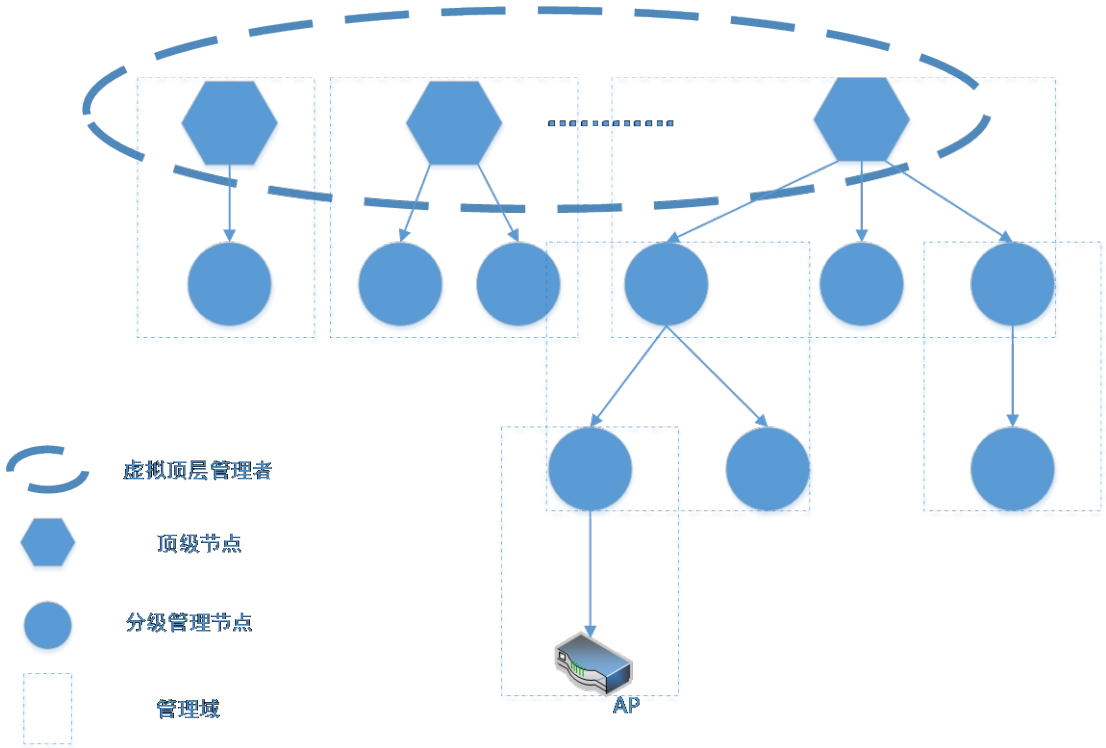
0 综述

0.1 总述



构造一个森林状的管理网络，每个节点均有自己的标识，整个体系的加密模式为 IBE。每个非叶子节点（也包括管理域生成初期的叶子节点）生成一个以自己为管理员节点的管理域（简称为“域”），域内包含其本身与所有子节点。此外，所有树的根节点（称为“顶级节点”，除此之外的非叶子节点称为“分级管理员节点”）构成虚拟顶级管理者。

本方案包含的算法为（1）基于秘密共享的顶级节点参数生成算法、（2）管理员节点参数更新算法、（3）非顶级节点的加入算法、（4）非顶级节点的退出算法、（5）顶级节点的加入算法、（6）顶级节点的退出算法、（7）非顶级节点的域生成算法、（8）域内节点的认证与密钥交换算法和（9）域间节点的认证与密钥交换算法。

算法（1）~（7）涉及管理网络的构建，而网络构建后使用算法（8）（9）在节点通信前建立信任并协商密钥。整个体系初始时仅由若干顶级节点组成，且顶级节点并未形成虚拟顶级管理者，此时使用算法（1），使得所有顶级节点产生虚拟顶级管理者，并且每个顶级节点形成以自己为管理员节点的管理域。算法（5）（6）用来加入或退出新的顶级节点，算法（3）（4）用来加入或退出新的非顶级节点。算法（7）用来使非顶级节点形成以自己为管理员节点的管理域，用于层级管理和扩张。当节点参数需要更新时，使用算法（2）。通信前需要协商会话密钥时，若节点属于同一管理域，则使用算法（8）；若节点不属于同一管理域，则使用算法（9）。

0.2 符号与参数

0.2.1 符号

节点代号	$U_{i,j}$	i代表节点的深度，若$i=0$则为顶级节点 j为节点在该层代号，每一层编号唯一
标识的哈希值	$Q_{i,j}$	$ID_{i,j}$的哈希值，哈希函数参考 SM9
消息的哈希值	Q_M	M的哈希值

临时会话密钥	Key	节点通信时建立的会话密钥，采用某种对称加密（待定）
符号函数	$sig(i,j)$	如果$i>j$则值为 1，如果$i=j$则值为 0，如果$i<j$则值为-1

0.2.2 参数

初始顶级节点数	n	体系初始时顶级节点的个数，对所有节点公开
大素数	p	用于取模的大素数，对所有节点公开
群的生成元	P	选用的生成元，对所有节点公开
请求否认消息	NAK	节点否认请求时发送该消息，具体值待定
强制退出消息	QC	节点强制其子节点退出时发送该消息，具体值待定
退出告知消息	QA	节点退出时发送该消息，具体值待定
更新消息	IU	节点告诉子管理员节点需要更新时发送的消息，具体值待定
密钥协商消息	KN	协商密钥时发送的消息，具体值待定
节点参数{		
//节点初始拥有		
节点标识	$ID_{i,j}$	节点的标识信息，对所有节点公开
节点公钥	$Q_{i,j}$	公钥加密时的公钥，为标识的哈希值，对所有节点公开（主公钥）
//顶级节点加入后获得		
秘密	D_j	初始顶级节点拥有的秘密
秘密扰动	$D_{j,j'}$	与顶级节点$U_{0,j'}$之间协商的扰动，任意两个顶级节点间均有故每个节点有顶级节点数量减 1 个秘密扰动，且$D_{j,j'}=D_{j',j}$
//加入某个管理域后获得		
节点私钥 1	$s'_{i,j}Q_{i,j}$	作为非管理员节点进行域内通信时的私钥，（域内主私钥 1）由节点保留，其中$s'_{i,j}$为其父节点管理域的保密安全参数，对于顶级节点则为s
//生成管理域后获得		
节点私钥 2	$s_{i,j}Q_{i,j}$	作为管理员节点进行域内通信时的私钥，（域内主私钥 2）由节点保留，非管理员节点没有该参数
}		
虚拟顶级管理者参数{		
//生成虚拟顶级管理者后获得		
保密安全参数	s	保密
公开安全参数	sP	对所有节点公开
}		
管理域参数{		
//生成管理域后获得		
保密安全参数	$s_{i,j}$	由管理域的管理员节点保留
公开安全参数	$s_{i,j}P$	对所有节点公开
验证消息	$M_{i,j}$	在生成管理域时获得的验证消息，由管理员节点保留
消息的签名	$s'_{i,j}Q_{M_{i,j}}$	消息的哈希值，$s'_{i,j}$为其父节点管理域的保密安全参数，由管理员节点保留
}		

0.3 对 SM9 的调用

对 SM9 的调用包含:

SM9 的公钥加密算法（以下简称“算法（A）”，参考 SM9 第四部分 5.1）
SM9 的公钥解密算法（以下简称“算法（B）”，参考 SM9 第四部分 5.2）
SM9 的数字签名生成算法（以下简称“算法（C）”，参考 SM9 第二部分 4）
SM9 的数字签名验证算法（以下简称“算法（D）”，参考 SM9 第二部分 5）
SM9 的密钥交换协议（以下简称“算法（E）”，参考 SM9 第三部分 4）

1 基于秘密共享的顶级节点参数生成算法

1.1 算法目的：初始n个顶级节点共同参与，可以协商出一个共同的未知的秘密作为虚拟顶级管理者保密安全参数（主私钥），并获得虚拟顶级管理者公共安全参数（主公钥）和自己在虚拟顶级管理者内通信的节点私钥 1（域内主私钥 1），同时自动生成管理域并获得自己的节点私钥 2（域内主私钥 2）。

1.2 算法参与者及参数：

顶级节点代号	$U_{0,k}(1 \le k \le n)$

1.3 算法生成：

顶级节点的秘密	D_k
虚拟顶级管理者保密安全参数（主私钥）	s
虚拟顶级管理者公共安全参数（主公钥）	sP
顶级节点的秘密扰动	$D_{k,k'}$
顶级节点私钥 1（域内主私钥 1）	$sQ_{0,k}$
顶级节点私钥 2（域内主私钥 2）	$s_{0,k}Q_{0,k}$
保密安全参数	$s_{0,k}$
公开安全参数	$s_{0,k}P$
验证消息	$M_{0,k}$
消息的签名	$s'_{0,k}Q_{M_{0.k}} = sQ_{M_{0.k}}$

1.4 算法步骤：

（1）选择多项式分量：每个顶级节点$U_{0,k}(1 \le k \le n)$随机选择一个$n-1$次的多项式$f_k(x) = a_{k,0} + a_{k,1}x + \cdots + a_{k,n-1}x^{n-1}$，其中多项式应满足$f_k(0) = a_{k,0}, a_{k,i} \in Z_p, 0 \le i \le n-1$。

（2）分发 share：对于所有的$U_{0,k}$，向$U_{0,j}$发送$D_{kj} = f_k(x_j)$，其中x_j为$U_{0,j}$指定的非零参数，满足当$i \ne j$时$x_i \ne x_j$，令$x_j = ID_{0,j}$。该过程可用矩阵表示为$D = \begin{bmatrix} D_{11} & \cdots & D_{1n} \\ \vdots & \ddots & \vdots \\ D_{n1} & \cdots & D_{nn} \end{bmatrix}$。在矩阵$D$中，由于对角线上的元素由节点$U_{0,k}$所私有，因此，除了$U_{0,k}$外其余的$n-1$个点合作也无法得到$U_{0,k}$的多项式$f_k(x)$。

（3）计算 share：顶层节点的秘密共享需要形成一个公共的系统安全参数，这个系统安全参数需要满足（1）没有任何一个用户持有该安全参数具体的值；（2）只有当所有顶级节点共同参与时，才能使用该参数进行某些计算。令$F(x) = \sum_{k=1}^{n} f_k(x)$，则可以形成公共参数$s = F(0) = \sum_{k=1}^{n} f_k(0)$，该公共参数$s$可以满足上述的要求。为了可以使用该参数进行某些计算，所有顶级节点$U_{0,k}(1 \le k \le n)$分别计算其对应于$F(x)$上的点：$D_k = \sum_{j=1}^{n} D_{jk} = \sum_{j=1}^{n} f_j(x_k) = F(x_k)$。同时计算扰动$D_{k,k'} = Q_{D_{kk'}} + Q_{D_{k'k}}$

（4）第二轮参数分发：由于我们现在已知顶级节点共享的秘密s满足：$s = \sum_{k=1}^{n} F(x_k) \prod_{\substack{i \ne k \\ 1 \le i \le n}} \frac{-x_i}{x_k - x_i} = \sum_{k=1}^{n} D_k \prod_{\substack{i \ne k \\ 1 \le i \le n}} \frac{-x_i}{x_k - x_i}$。现在要对所有的顶级节点进行参数的分发，使得顶级节点能获得系统的安全参数sP，自己的私钥$sQ_{0,k}$。对顶级节点$U_{0,k}$，对节点$U_{0,j}(1 \le j \le n, j \ne k)$发送$Q_{0,i}D_k \prod_{\substack{i \ne k \\ 1 \le i \le n}} \frac{-x_i}{x_k - x_i}$以及$PD_k \prod_{\substack{i \ne k \\ 1 \le i \le n}} \frac{-x_i}{x_k - x_i}$，记$l_k = D_k \prod_{\substack{i \ne k \\ 1 \le i \le n}} \frac{-x_i}{x_k - x_i}$。

上述过程可以用矩阵表示为：$SK=\begin{bmatrix} l_1Q_{0,1} & \cdots & l_1Q_{0,n} \\ \vdots & \ddots & \vdots \\ l_nQ_{0,1} & \cdots & l_nQ_{0,n} \end{bmatrix}$，$PP=\begin{bmatrix} l_1P & \cdots & l_1P \\ \vdots & \ddots & \vdots \\ l_nP & \cdots & l_nP \end{bmatrix}$。

（5）计算私钥和公共参数sP：每个节点$U_{0,k}(1\le k\le n)$在得到上述的信息后，可计算出自己的私钥，即计算SK矩阵第k列元素的和：$sQ_{0,k}=\sum_{j=1}^{n}SK_{jk}=\sum_{j=1}^{n}l_jQ_{0,k}$。计算公共参数，即计算$PP$矩阵第k列元素的和：$sQ_{0,k}=\sum_{j=1}^{n}SK_{jk}=\sum_{j=1}^{n}l_jQ_{0,k}$，$sP=\sum_{j=1}^{n}PP_{jk}=\sum_{j=1}^{n}l_jP$。

（6）每个顶级节点选择一个参数$s_{0,k}$，将$s_{0,k}$作为自己管理域的保密安全参数，并计算$s_{0,k}P$作为自己管理域的公开安全参数。将$s_{0,k}P$和$ID_{0,k}$组合成消息$M_{0,k}$，计算出$Q_{M_{0.k}}$后采用算法（A）加密成消息C后发送给所有其他$U_{0,j}$。

（7）$U_{0,j}$采用算法（B）解密后得到$Q_{M_{0.k}}$，计算出$l_jQ_{M_{0.k}}$，交由节点$U_{0,k}$。

（8）$U_{0,k}$收到其他节点的$l_jQ_{M_{0.k}}$后，计算出$l_kQ_{M_{0.k}}$和$s'_{0,k}Q_{M_{0.k}}=sQ_{M_{0.k}}=\sum_{j=1}^{n}l_jQ_{M_{0.k}}$，作为自己消息的签名。至此，管理域的所有参数均生成，故顶级节点的管理域已生成。

2 管理员节点参数更新算法

2.1 算法目的：如果一个管理员节点需要进行更新，则它更新自己的参数，如果其非顶级节点则需要父节点参与，如果其为顶级节点则需要选取n个顶级节点（自己也是其中一个）共同参与。之后把参数告知自己的子节点，要求其更新。如果子节点是管理员节点，则需要迭代更新验证消息和消息的签名。

2.2 算法参与者及参数

待更新的管理员节点代号	$U_{i+1,x}$
其父节点代号（$i\neq -1$时）	$U_{i,j}$
选取的顶级节点代号（$i=-1$时）	$U_{0,j_k}(1\le k\le n)$
其子节点代号	$U_{i+2,l}$

2.3 算法生成：

无

2.4 算法步骤：

（1）$U_{i+1,x}$生成一个更新值w，用$s_{i+1,x}=ws_{i+1,x}$作为新的保密安全参数，$s_{i+1,x}P=ws_{i+1,x}P$作为新的公共安全参数。如果其为顶级节点，则执行（6）。

（2）$U_{i+1,x}$采用算法（A）将$s_{i+1,x}$加密成消息C_1后发送给$U_{i,j}$。

（3）$U_{i,j}$采用算法（B）解密后获得$s_{i+1,x}$，并根据得到的信息判断是否同意该请求。若不同意该节点的请求，则返回NAK，算法结束。

（4）$U_{i,j}$将$s_{i+1,x}P$、$ID_{i+1,x}$、$M_{i,j}$以及$s'_{i,j}Q_{M_{i,j}}$一起作为新的消息$M_{i+1,x}$,同时计算$s_{i,j}Q_{M_{i+1,x}}$，从用算法（A）加密成消息C_2后发送给$U_{i+1,x}$。

（5）$U_{i+1,x}$采用算法（B）解密后，计算$s_{i+1,x}Q_{i+1,x}=ws_{i+1,x}Q_{i+1,x}$作为自己的节点私钥 2，执行（9）。

（6）$U_{i+1,x}$将$s_{i+1,x}P$和$ID_{i+1,x}$以其作为成新的消息$M_{i+1,x}$，计算出$Q_{M_{i+1,x}}$后采用算法（A）加密成消息C_3后发送给所有其他U_{0,j_k}。

（7）U_{0,j_k}采用算法（B）解密后得到$Q_{M_{i+1,x}}$，计算出$D_{j_k}Q_{M_{i+1,x}}$，交由节点$U_{i+1,x}$。

（8）$U_{i+1,x}$收到其他节点的$D_{j_k}Q_{M_{i+1,x}}$后，计算出$D_xQ_{M_{i+1,x}}$（其中D_x为所有D_{j_k}中缺少的那一份，即自己的那一份）和$s'_{0,k}Q_{M_{i+1,x}}=sQ_{M_{i+1,x}}=\sum_{k=1}^{n}D_{j_k}Q_{M_{i+1,x}}\prod_{\substack{m\neq k\\ 1\le m\le n}}\frac{-ID_{j_m}}{ID_{j_k}-ID_{j_m}}$，作为自己消息的签名。

（9）将w发送给所有$U_{i+2,l}$，子节点在收到w后，用$s'_{i+2,l}Q_{i+2,l}=ws'_{i+2,l}Q_{i+2,l}$作为新的节点私钥 1。如果该节点不是管理员节点，则不再进行后续步骤。

（10）$U_{i+2,l}$将$s_{i+2,l}$采用算法（A）加密成消息C_3后发送给$U_{i+1,x}$。

（11）$U_{i+1,x}$采用算法（B）解密后获得$s_{i+2,l}$，并根据得到的信息判断是否同意该请求。若不同意该节点的请求，则返回NAK，算法结束。

（12）$U_{i+1,x}$计算出$s_{i+2,l}P$，和$ID_{i+2,l}$、$M_{i+1,x}$以及$s'_{i+1,x}Q_{M_{i+1,x}}$一起作为消息$M_{i+2,l}$，同时计算$s_{i+1,x}Q_{M_{i+2,l}}$，从用算法（A）加密成消息C_4后发送给$U_{i+2,l}$。

（13）$U_{i+2,l}$采用算法（B）解密后，更新$M_{i+2,l}$和$s'_{i+2,l}Q_{M_{i+2,l}}$，并给所有子管理员节点发送消息IU。

（14）所有子管理员节点接收到消息后迭代执行（11）。

3 非顶级节点的加入算法

3.1 算法目的：如果一个非顶级节点需要加入一个管理域，只需向域的管理员节点提出申请，在经过该节点审核后获取自己所分配的私钥即可，私钥需要通过对称加密进行传输。

3.2 算法参与者及参数：

待加入管理域的节点代号	$U_{i+1,x}$
管理域的管理员节点代号	$U_{i,j}$

3.3 算法生成：

待加入节点的节点私钥 1（域内主私钥 1）	$s'_{i+1,x}Q_{i+1,x} = s_{i,j}Q_{i+1,x}$

3.4 算法步骤：

（1）$U_{i+1,x}$将$Q_{i+1,x}$和自己拟采用的临时会话密钥Key作为消息，采用算法（A）加密成消息C_1后发送给$U_{i,j}$。

（2）$U_{i,j}$采用算法（B）解密后得到$Q_{i+1,x}$和Key，并根据得到的信息判断是否同意节点$U_{i+1,x}$加入域。若不同意该节点的请求，则返回NAK，算法结束。

（3）若$U_{i,j}$同意该请求，则计算出$s_{i,j}Q_{i+1,x}$，并用Key加密成消息C_2后，将其交由节点$U_{i+1,x}$。

（4）节点$U_{i+1,x}$收到C_2后解密得到$s_{i,j}Q_{i+1,x}$，并保存自己的私钥信息。

4 非顶级节点的退出算法

4.1 算法目的：如果一个非顶级节点希望退出当前的管理域或者收到了消息QC后，则将其所有子节点（如果有的话）强制退出后，退出当前的管理域并且告知自己的父节点（管理员节点）。

4.2 算法参与者及参数：

待退出管理域的节点代号	$U_{i+1,x}$
父节点的节点代号	$U_{i,j}$

4.3 算法生成：

无

4.4 算法步骤：

（1）如果$U_{i+1,x}$是叶子节点，则执行（3）。

（2）$U_{i+1,x}$向所有子节点发送QC，然后进入等待，直到所有的子节点返回QA。

（3）向$U_{i,j}$发送消息QA，$U_{i,j}$接收到该消息后执行算法（2）管理员节点参数更新算法。

5 顶级节点的加入算法

5.1 算法目的：如果顶级管理域已经生成，新的顶级节点想要加入时，执行该算法，获得顶级节点的参数，并生成以自己为管理员节点的管理域。

5.2 算法参与者及参数：

待加入的顶级节点代号	$U_{0,t}$
选取的顶级节点代号	$U_{0,j_k}(1 \le k \le n)$
已加入的顶级节点数量	n'
已加入的顶级节点代号	$U_{0,j'_l}(1 \le l \le n')$

5.3 算法生成:

待加入节点的秘密	D_t
待加入节点的节点私钥 1（域内主私钥 1）	$sQ_{0,t}$
待加入节点私钥 2（域内主私钥 2）	$s_{0,t}Q_{0,t}$
待加入节点的秘密扰动	$D_{t,t'}$
保密安全参数	$s_{0,t}$
公开安全参数	$s_{0,t}P$
验证消息	$M_{0,t}$
消息的签名	$s'_{0,t}Q_{M_{0.t}} = sQ_{M_{0.t}}$

5.4 算法步骤:

（1）$U_{0,t}$将$Q_{0,t}$作为消息，采用算法（A）加密成消息C后发送给所有U_{0,j_k}。

（2）U_{0,j_k}采用算法（B）解密后得到$Q_{0,t}$，并根据得到的信息判断是否同意节点$U_{0,t}$加入。若不同意该节点的请求，则返回NAK，算法结束。

（3）若$U_{0,k}$同意该请求，则计算出$D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{ID_t-ID_{j_i}}{ID_{j_k}-ID_{j_i}}+\sum_{\substack{i\neq k\\1\le i\le n}} sig(j_k,j_i)D_{j_k,j_i}$和$Q_{0,j_k}D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{-ID_{j_i}}{ID_{j_k}-ID_{j_i}}$，记$x_{j_k}=ID_{0,j_k}$，$l_{j_k}=D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{-x_{j_i}}{x_{j_k}-x_{j_i}}$，交由节点$U_{0,t}$。

（4）节点$U_{0,t}$收到n个非NAK消息后计算$D_t=\sum_{k=1}^{n}(D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{x_t-x_{j_i}}{x_{j_k}-x_{j_i}}+\sum_{\substack{i\neq k\\1\le i\le n}} sig(j_k,j_i)D_{j_k,j_i})=\sum_{k=1}^{n}D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{x_t-x_{j_i}}{x_{j_k}-x_{j_i}}$和$sQ_{0,t}=\sum_{k=1}^{n}l_{j_k}Q_{0,t}$，并保存自己的秘密和私钥信息。接到消息后，双方均令$D_{t,j'_l}=D_{j'_l,t}=Q_{D_{j_k}\prod_{\substack{i\neq k\\1\le i\le n}}\frac{ID_t-ID_{j_i}}{ID_{j_k}-ID_{j_i}}+\sum_{\substack{i\neq k\\1\le i\le n}} sig(j_k,j_i)D_{j_k,j_i}}$。

（5）$U_{0,t}$选择一个参数$s_{0,t}$，将$s_{0,t}$作为自己管理域的保密安全参数，并计算$s_{0,t}P$作为自己管理域的公开安全参数。将$s_{0,t}P$和$ID_{0,t}$组合成消息$M_{0,t}$，计算出$Q_{M_{0.t}}$后采用算法（A）加密成消息C后发送给所有U_{0,j_k}。

（6）U_{0,j_k}采用算法（B）解密后得到$Q_{M_{0.t}}$，计算出$l_{j_k}Q_{M_{0.t}}$，交由节点$U_{0,t}$。

（7）$U_{0,t}$收到其他节点的$l_{j_k}Q_{M_{0.t}}$后，计算出$s'_{0,t}Q_{M_{0.t}}=sQ_{M_{0.t}}=\sum_{k=1}^{n}l_{j_k}Q_{M_{0.t}}$，作为自己消息的签名。

（8）$U_{0,t}$向其他未参与的前面过程的节点$U_{0,j'_l}(1 \le l \le n', \forall k\ j'_l \neq j_k)$选取$n'-n$个随机数分发给对应的节点，记$U_{0,t}$发送给$U_{0,j'_l}$的随机数为$r_{t,j'_l}$。收到随机数后，$U_{0,j'_l}$也向$U_{0,t}$发送$r_{j'_l,t}$，并令$D_{t,j'_l}=D_{j'_l,t}=r_{t,j'_l}+r_{j'_l,t}$作为两者的秘密扰动。

6 顶级节点的退出算法

6.1 算法目的：如果一个顶级节点希望退出虚拟管理域，则将其所有子节点（如果有的话）强制退出后，退出当前的管理域。

6.2 算法参与者及参数:

待退出的顶级节点代号	$U_{0,t}$
已加入的顶级节点数量	n'
已加入的顶级节点代号	$U_{0,j_k}(1 \le k \le n')$

6.3 算法生成：

无

6.4 算法步骤：

（1）如果$n' \le n$则无法退出，算法结束。$U_{0,t}$向所有子节点发送QC，然后进入等待，直到所有的子节点返回QA。

（2）接收到所有的QA后即可退出，并用算法（A）向所有$U_{0,j_k}(1 \le k \le n', j_k \ne t)$发送自己的序号$t$。

（3）U_{0,j_k}用算法（B）解密后，获得t，删去$D_{j_k,t}$。

（4）j_k最小的节点U_{0,j_1}（假设j_1最小）生成一个不高于$n-1$次、常系数为 0 的非 0 多项式$f(x)$，计算$f(ID_{j_k})(j_k \ne t)$并发给对应的节点（自身仅计算即可，不用发送）。

（5）$U_{0,j_k}(j_k \ne t)$接收到$f(ID_{j_k})$后，令$D_{j_k} = D_{j_k} + f(ID_{j_k})$。

7 非顶级节点的域生成算法

7.1 算法目的：已经加入某个域的非顶级节点希望生成子域时使用该算法。可以获得其安全参数从而形成域。

7.2 算法参与者及参数：

待生成管理域的节点代号	$U_{i+1,x}$
管理域的管理员节点代号	$U_{i,j}$

7.3 算法生成：

该管理域的保密参数	$s_{i+1,x}$
该管理域的验证消息	$M_{i+1,x}$
验证消息的签名	$s'_{i+1,x}Q_{M_{i+1,x}} = s_{i,j}Q_{M_{i+1,x}}$
该节点的节点私钥 2（域内主私钥 2）	$s_{i+1,x}Q_{i+1,x}$

7.4 算法步骤：

（1）$U_{i+1,x}$生成自己待定的管理域保密安全参数$s_{i+1,x}$，采用算法（A）加密成消息C_1后发送给$U_{i,j}$。

（2）$U_{i,j}$采用算法（B）解密后获得$s_{i+1,x}$，并根据得到的信息判断是否同意该请求。若不同意该节点的请求，则返回NAK，算法结束。

（3）$U_{i,j}$计算出$s_{i+1,x}P$，和$ID_{i+1,x}$、$M_{i,j}$以及$s'_{i,j}Q_{M_{i,j}}$一起作为消息$M_{i+1,x}$,同时计算$s_{i,j}Q_{M_{i+1,x}}$，从用算法（A）加密成消息C_2后发送给$U_{i+1,x}$。

（4）$U_{i+1,x}$采用算法（B）解密后，域内公共参数为$s_{i+1,x}P$，成为域管理员节点，并计算$s_{i+1,x}Q_{i+1,x}$作为自己的节点私钥 2。

8 域内节点的认证与密钥交换算法

8.1 算法目的：处于同一个域的节点希望通信前，发送密钥请求消息，然后通过密钥交换算法生成会话密钥。

8.2 算法参与者及参数：

请求方节点代号	U_{i_1,j_1}
目标方节点代号	U_{i_2,j_2}

8.3 算法生成：

临时会话密钥	Key

8.4 算法步骤：

（1）U_{i_1,j_1}将KN作为消息，采用算法（C）向U_{i_2,j_2}生成并发送签名m_1。

（2）U_{i_2,j_2}收到m_1后采用算法（D）验证。如果验证不通过，则返回NAK，算法结束；如果

验证通过，则将KN作为消息，采用算法（C）向节点U_{i_1,j_1}生成并发送签名m_2。

（3）U_{i_1,j_1}收到m_2采用算法（D）验证。如果验证不通过，则返回NAK，算法结束；如果验证通过，与U_{i_2,j_2}采用算法（E）产生临时会话密钥Key。

9 域间节点的认证与密钥交换算法

9.1 算法目的：处于不同域的管理员节点希望通信前，先通过认证建立信任关系，然后通过密钥交换算法生成会话密钥。(如果不是管理员节点，则交由其管理员节点代理，以下为实际处理的节点代号。)

9.2 算法参与者及参数：

请求方节点代号　　U_{i_1,j_1}

目标方节点代号　　U_{i_2,j_2}

9.3 算法生成：

临时会话密钥　　Key

9.4 算法步骤：

（1）U_{i_1,j_1}将KN、M_{i_1,j_1}和$s'_{i_1,j_1}Q_{M_{i_1,j_1}}$作为消息，采用算法（C）向$U_{i_2,j_2}$生成并发送签名$m_1$。

（2）U_{i_2,j_2}收到m_1后采用算法（D）验证。如果验证不通过，则返回NAK，算法结束。如果验证通过且发现其为顶级节点，则验证通过。如果验证通过且发现其非顶级节点，则获得消息M_{i_1,j_1}和$s'_{i_1,j_1}Q_{M_{i_1,j_1}}$，由于$M_{i_1,j_1}$包含其父节点消息$s'_{i_1,j_1}P$，因而可以验证$M_{i_1,j_1}$与$s'_{i_1,j_1}Q_{M_{i_1,j_1}}$是否一致，验证不通过，则返回$NAK$，算法结束；验证通过，则可解析出其父节点对应的消息（假定其父节点为$U_{i\prime_1,j\prime_1}$）$M_{i\prime_1,j\prime_1}$和$s'_{i\prime_1,j\prime_1}Q_{M_{i\prime_1,j\prime_1}}$，再次验证这一步，直到解析到顶级节点的消息（假定其该节点为$U_{i\prime\prime_1,j\prime\prime_1}$）$M_{i\prime\prime_1,j\prime\prime_1}$和$s'_{i\prime\prime_1,j\prime\prime_1}Q_{M_{i\prime\prime_1,j\prime\prime_1}}$，由于$s'_{i\prime\prime_1,j\prime\prime_1}=s$，因而用顶级管理域公共参数$sP$进行验证即可，如果验证不通过，则返回$NAK$，算法结束。

（3）验证通过后U_{i_2,j_2}反过来对U_{i_1,j_1}同（1）发送消息，而U_{i_1,j_1}同（2）进行验证，来完成相互验证。

（4）相互验证通过后U_{i_1,j_1}和U_{i_2,j_2}采用算法（E）产生临时会话密钥Key。