一种新的动态批密钥更新算法

李波[1]，潘进[1]，付颖[2]，李国朋[1]，韩明奎[1]
(1.西安通信学院网络安全与对抗研究室，710106，西安；2.解放军66176部队，100076，北京)

摘要：针对传统单次密钥更新方法存在低效、资源浪费、数据和密钥不同步等问题，提出了一种基于密钥树的批密钥更新算法.通过2种方法保持密钥树的平衡：利用加入节点替代离开节点的位置来保持树的结构不变；搜索密钥树中高度最低的节点，然后根据该节点的类型及剩余的可加入节点数，将适量的节点加入到高度最低节点的位置.同时，对服务器的更新开销进行了理论分析，建立了用于计算开销的精确数学模型.仿真实验表明，与单次密钥更新方法相比，所提算法可以将更新开销减少74.6%，显著提高更新效率，并适合于大型动态群组的应用.

关键词：群组通信；密钥树；批密钥更新；更新开销
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2009)12-0065-05

A Novel Dynamic Batch Rekeying Algorithm

LI Bo[1], PAN Jin[1], FU Ying[2], LI Guopeng[1], HAN Mingkui[1]
(1. Laboratory of Network Security and Countermeasure, Xi′an Communications Institute, Xi′an 710106, China;
2. PLA Troop 66176, Beijing 100076, China)

Abstract: Aiming at the problems that the traditional individual rekeying approach is inefficient, waste of resources, and out-of-sync between keys and data, a novel batch algorithm for renewing keys is proposed based on a key tree. In the new algorithm, the key tree is kept balanced by two methods: departed nodes are replaced by joining nodes to keep the structure of the tree unchanged; the node with the smallest height in the tree is searched, and then a proper number of joining nodes are added to the node place based on the type of the node and the number of the remaining nodes to be joined. The cost of renewing servers' keys is analyzed theoretically, and accurate mathematical models are established to calculate the cost. Simulation results and comparison with the individual rekeying approach show that the proposed algorithm decreases the rekeying cost by 74.6% and improves the efficiency of rekeying significantly, and that the algorithm is suitable for large dynamic groups.

Keywords: group communication; key tree; batch rekeying; rekeying cost

随着Internet的迅速发展，基于群组通信的应用不断增加，如视频会议、分布式交互仿真、网络游戏等.群组通信的安全基础是所有组成员共用一个不为组外成员所知的组密钥，用以加密组通信数据[1].组成员具有动态性，为了保证前向和后向安全性，组密钥需要不断更新.

传统的密钥更新是单次密钥更新，这种更新方法主要存在3点不足[2]：效率低、资源浪费、数据和密钥不同步.为了解决这类问题，批密钥更新方法应运而生[2-8].批密钥更新存在多条密钥更新路径，它们将形成一个更新子树.组控制器(GC)在密钥更新之前需要将更新子树标记出来.现有的批密钥更新标记算法存在以下不足：当加入的用户比较多时会产生一个不平衡树[2]；更新开销比较大[3]；算法复

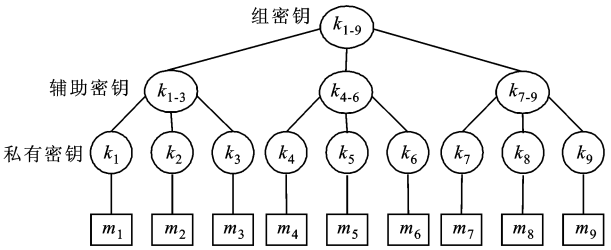
收稿日期：2009-04-27. 作者简介：李波(1985—)，男，硕士生；潘进(联系人)，男，教授. 基金项目：国家高技术研究发展计划资助项目(2007AA01Z472).

杂,且未考虑加入成员与离开成员之间的关系[7].

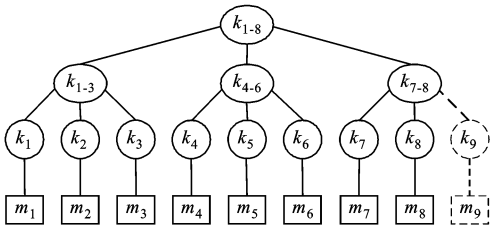
综上,本文设计了一种新的动态批密钥更新算法,可使更新开销更小.

1 基于逻辑密钥分层的单次密钥更新

逻辑密钥分层(LKH)方案是一种集中式密钥管理方案,适合于具有高度安全要求的应用. LKH密钥树[9-11]中存在3种密钥:组密钥、辅助密钥和私有密钥. 用户节点记为 m,其他节点记为 k,假设密钥树中节点数为 N,则 m_j 表示第 j($1\leqslant j\leqslant N$)个用户节点,k_i 表示 m_i($1\leqslant i\leqslant N$)拥有的私有密钥,$k_{l\text{-}q}$ 表示 $m_l\sim m_q$($1\leqslant l<q\leqslant N$)拥有的私有密钥. 图1所示的是度(即节点拥有的子树数)$d=3$、$N=9$,高度(即节点的最大层次数)$h=2$ 的平衡密钥树.



(a)m_9加入



(b)m_9离开

图1 m_9 加入/离开小组的情况

密钥树中每个用户均拥有从该用户节点到根节点的所有密钥,如果一个用户离开,则该用户知道的所有密钥都将被更新. 例如,如果 m_9 离开用户组,则 $k_{1\text{-}9}$ 被更新为 $k_{1\text{-}8}$,$k_{7\text{-}9}$ 被更新为 $k_{7\text{-}8}$,此时GC要发送的更新消息为:$\{k_{7\text{-}8}\}_{k_7}$、$\{k_{7\text{-}8}\}_{k_8}$、$\{k_{1\text{-}8}\}_{k_{1\text{-}3}}$、$\{k_{1\text{-}8}\}_{k_{4\text{-}6}}$、$\{k_{1\text{-}8}\}_{k_{7\text{-}8}}$. 如果 m'_9 想加入用户组,则 $k_{1\text{-}8}$ 被更新为 $k_{1\text{-}9}$,$k_{7\text{-}8}$ 被更新为 $k_{7\text{-}9}$,GC将发送以下更新消息:$\{k_{1\text{-}9}\}_{k_{1\text{-}8}}$、$\{k_{1\text{-}9}\}_{k_9}$、$\{k_{7\text{-}9}\}_{k_9}$、$\{k_{7\text{-}9}\}_{k_{7\text{-}8}}$.

本文将GC发送的更新消息的数量作为衡量更新开销的指标. 对于度为 d,用户数为 N 的平衡密钥树,如果有单个用户离开组,则GC需要发送的更新消息数为 $dh-1$;如果有单个用户加入组,GC需要发送的更新消息数为 $2h$. 这样,采用单次密钥更新,需逐个更新 J 个加入用户和 L 个离开用户,GC的总开销 $C=[2J+dL]h-L$.

2 批密钥更新算法

如果在一次批密钥更新结束后,密钥树不能尽可能达到平衡,同时 h 很大的叶节点对应的用户离开,则在下一次更新时GC的开销将会增加. 所以,密钥在每次更新时保持树的平衡有利于减小总的更新开销.

假设有 J 个用户加入、L 个用户退出群组,并考虑到 J 和 L 大小的变化,本文算法分为下面4种情况,前2种情况与文献[2]类似. 算法的符号说明如表1所示,算法涉及到的4种子操作①~④如图2所示. 在对密钥树中的节点进行搜索时,若存在高度相同的多个节点,则取最左边的叶节点作为高度最低的节点 v_i.

表1 批密钥更新算法符号说明

符号	符号含义
m_i	小组中原有第 i 个成员
M_i	后加入小组的新成员
T'_i	最小单位子树(由一个根节点和 d 个子节点组成)
v_f	非满节点(子节点数小于 d 的节点)
v_d	单叶节点(兄弟节点均为 k 节点的叶节点)
v_i	高度最低的节点
J_s	剩余的加入节点数
T_i	第 i 个最小单位子树

情况1 $J=L$,算法步骤如下.

步骤1:将所有离开用户的节点用加入用户替代.

步骤2:将替代位置到根节点之间的所有节点标记为UPDATE.

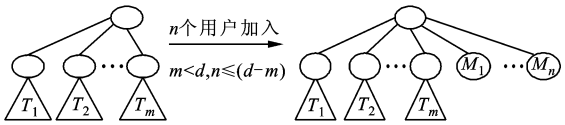
情况2 $J<L$,算法步骤如下.

步骤1:在 L 个离开用户节点的位置中,挑选 J 个高度最小的节点位置,将这 J 个节点位置用加入的 J 个用户替代.

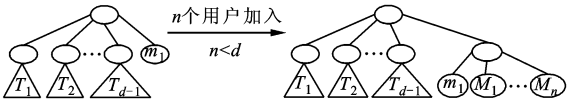
步骤2:将所有从替代位置到根节点的节点标记为UPDATE,将未被加入用户替代的离开节点标记为DELETE,将所有标记为DELETE节点的最邻近兄弟节点标记为UPDATE. 一个 k 节点只有在其所有的孩子节点均被标记为DELETE时才被标记为DELETE.

情况3 $J>L$ 且 $L=0$,算法步骤如下.

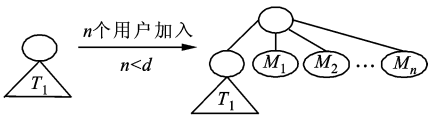
步骤1:在密钥树中搜索 v_i.



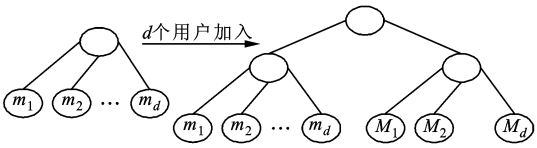
(a)子操作①



(b)子操作②



(c)子操作③



(d)子操作④

图 2　4 种子操作

步骤 2：$v_i=v_f$，执行子操作①；$v_i=v_d$，执行子操作②，同时 $J_s>0$，转步骤 1，否则算法结束；$v_i\neq v_d$ 或 $v_i\neq v_f$，转步骤 3.

步骤 3：$J_s<d$，对 v_i 所在的 T'_i 执行子操作③后结束；$J_s\geqslant d$，对 v_i 所在的 T'_i 执行子操作④，同时 $J_s=0$，结束操作，否则转步骤 1.

步骤 4：将从加入用户节点到根节点的所有节点标记为 UPDATE.

情况 4　$J>L$ 且 $L>0$，算法步骤如下.

步骤 1：将所有离开用户节点的位置用加入用户节点替代，此时相当于加入用户节点数为 $(J-L)$、离开用户节点数 $L=0$ 的情况.

步骤 2：按情况 3 的步骤 1 开始执行.

步骤 3：将从加入用户节点到根节点的所有节点标记为 UPDATE.

密钥树标记完毕之后，密钥服务器将所有标记为 DELETE 的节点移除，所有标记为 UPDATE 的节点将形成密钥更新子树. 密钥服务器通过更新子树产生新的密钥，建立并组播更新消息.

假设一个拥有 5 个用户的组，成员 m_1 离开，$M_1\sim M_4$ 加入用户组，则运用本文算法进行批密钥更新的结果如图 3 所示. 图中的灰色节点均为标记为 UPDATE 的节点，这些节点形成了更新子树，每个被标记的 k 节点的密钥都要通过 2 条更新消息来发送，GC 共需要发送 12 条更新消息.

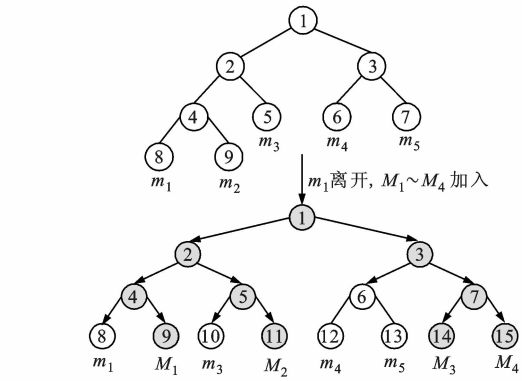


图 3　批密钥更新算法应用实例

3　算法性能分析与仿真

3.1　性能分析

本文对最差情况下的算法开销进行了分析，算法的平均开销分析可参考文献[2]，限于篇幅，在此不做描述.

最差情况下算法开销的分析，主要考虑成员的离开对 GC 开销的影响，当离开的节点均匀地分布于各子树中时，GC 的开销最大.

3.1.1　情况 1　当离开用户数和加入的用户数相等时，GC 的开销等于变动的 k 节点数量 n_k 与 d 的乘积，故要计算 GC 的开销，只需要确定 n_k 即可.

(1) $J=L<d^{h-1}$. 由于离开节点均匀分布于各子树中，因此当 $L\leqslant d$ 时，只有一个根节点被 L 条更新路径共用，当 $L>d$ 时，存在除根节点之外的节点被不同的更新路径共用，可分为以下 2 种情况讨论：①$L\leqslant d$，则有

$$C_{11}=d[hL-(L-1)]=d[(h-1)L+1] \tag{1}$$

②$L>d$，从 $h=0$ 层到 $h=\lfloor \log_d L \rfloor$ 层的所有节点均为变动的 k 节点，从 $h=\lfloor \log_d L \rfloor+1$ 层到 $h=h-1$ 层，每层都有 L 个变动节点，则有

$$C_{12}=d[(1+d+\cdots+d^{\lfloor \log_d L \rfloor})+L(h-1-\lfloor \log_d L \rfloor)]=d\left[\frac{d^{1+\lfloor \log_d L \rfloor}-1}{d-1}+L(h-1-\lfloor \log_d L \rfloor)\right] \tag{2}$$

(2) $J=L\geqslant d^{h-1}$. 此时，无论 L 如何变化，n_k 均为第 0 层至第 $h-1$ 层的所有节点，故有

$$C_2 = d[1+d+\cdots+d^{h-1}] = d\left[\frac{d^h-1}{d-1}\right] \quad (3)$$

3.1.2 情况2 假设 L 个离开节点均被替代，可按 $J=L$ 的情况处理. 实际上由于 $L-J$ 个节点未被替代，所以GC不需要给这些节点发送更新消息，只需在 $J=L$ 的情况基础上减去 $L-J$ 条消息即可.

(1)$L<d^{h-1}$，又分2种：①$L\leqslant d$，有

$$C = C_{11} - (L-J) = d[(h-1)L+1] - (L-J) \quad (4)$$

②$L>d$，有

$$C = C_{12} - (L-J) = d\left[\frac{d^{1+\lfloor \log_d L \rfloor}-1}{d-1} + L(h-1-\lfloor \log_d L \rfloor)\right] - (L-J) \quad (5)$$

(2)$L\geqslant d^{h-1}$，有

$$C = C_2 - (L-J) = d\left[\frac{d^h-1}{d-1}\right] - (L-J) \quad (6)$$

3.1.3 情况3 此时没有用户离开，故所计算的开销等同于平均情况下的开销.

(1)$d>2$. 在 $h-2$ 层，每个节点所在的子树 T_{h-2} 拥有 d 个最小单位子树，每个最小单位子树可以增加 $d(d-1)$ 个节点，故每个 T_{h-2} 可以添加 $d^2(d-1)$ 个节点. 同理，在第1层，每个节点所在的子树 T_1 可以添加 $d^{h-1}(d-1)$ 个节点. 由于节点从第一个 T_{h-2} 开始添加，故 n_{k_1} 的初始值为 $h-1$. 当 J 超过了一个子树所能添加的最大节点数时，就要向相邻子树添加节点，此时 n_{k_1} 的值要加1. 对于第 $h-1$ 层到 h 层，GC要向每个加入节点发送一条消息. 此外，J 每增加 $d-1$，发送的更新消息就要额外增加一条，故有

$$C = d\left[(h-1) + \sum_{j=1}^{h-2} \left\lfloor \frac{(J-1)}{(d-1)d^{j+1}} \right\rfloor\right] + J + \left\lceil \frac{J}{d-1} \right\rceil \quad (7)$$

(2)$d=2$. 当 $d=2$、$J=1$ 时，$n_k=1$，以后每增加一个节点，n_k 将加1，故有

$$C = dn_k = d[h+J-1] \quad (8)$$

3.1.4 情况4 $J>L$，$L>0$. 此时，可以将 J 分为2部分，即 L 和 $J-L$. L 个加入节点替换所有离开的节点，按 $J=L$ 的情况处理. 对于剩下的 $J-L$ 个节点，GC要向每个节点发送一条消息. 此外，J 每增加 $d-1$，发送的更新消息就要额外增加一条.

(1)$L<d^{h-1}$，又分2种：①$L\leqslant d$，有

$$C = C_{11} + (J-L) + \left\lceil \frac{J-L}{d-1} \right\rceil = d[(h-1)L+1] + (J-L) + \left\lceil \frac{J-L}{d-1} \right\rceil \quad (9)$$

②$L>d$，有

$$C = C_{12} + (J-L) + \left\lceil \frac{J-L}{d-1} \right\rceil = d\left[\frac{d^{1+\lfloor \log_d J \rfloor}-1}{d-1} + L(h-1-\lfloor \log_d J \rfloor)\right] + (J-L) + \left\lfloor \frac{J-L}{d-1} \right\rfloor \quad (10)$$

(2)$L\geqslant d^{h-1}$，有

$$C = C_2 + (J-L) + \left\lceil \frac{J-L}{d-1} \right\rceil = d\left[\frac{d^h-1}{d-1}\right] + \left\lceil \frac{J-L}{d-1} \right\rceil + (J-L) \quad (11)$$

可以运用本文算法计算图3中批密钥的更新消息数，以便于验证算式. $J=4$，$L=1$，属于 $J>L$、$L\leqslant d$ 的情况，则有

$$C = d[(h-1)L+1] + (J-L) + \left\lceil \frac{J-L}{d-1} \right\rceil = 2\times[(3-1)\times 1+1] + (4-1) + \left\lceil \frac{(4-1)}{(2-1)} \right\rceil = 12 \quad (12)$$

对于其他情况，本文算法一概成立，限于篇幅，在此不做更多验证.

3.2 性能仿真

首先建立一个用户数为 N、度为 d 的密钥树，通过随机选取 J 个加入用户和 L 个离开用户来计算服务器的更新开销，更新开销的计算以发送的更新消息的条数为指标. 取 (N,d) 值为(4 096,4)，仿真运行100次，取平均值作为服务器的更新开销.

如图4所示，在加入用户数不变的情况下，随着离开用户数的增加，更新密钥带来的开销逐渐增大，加入用户增加到与离开用户数相等，GC的更新开销便开始逐渐减小. 这是因为：当离开用户数开始增加时，发生变动的 k 节点的数量也随之增加，所以更新开销将增大；随着离开用户数的继续增加，供加入用户可插入的节点位置数也随之增加；当加入用户和离开用户相等时，GC不需要创建新的 k 节点，发生变动的 k 节点的数量不再增加，故更新开销开始减小. 离开用户数不变时，随着加入用户数的增加，GC的开销将逐渐增大. 这是因为：当密钥树中可供替换的节点数一定时，随着加入用户数增加，GC的总用户数增加，故GC需要发送的更新消息也相应地增加.

如图5所示，批密钥更新算法的开销明显小于单次密钥更新算法，约为单次密钥更新算法的

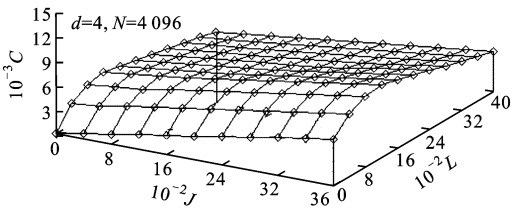


图 4　批密钥的更新开销

25.4%. 随着加入和离开的用户数增加，批密钥更新算法的优势越来越明显. 这是因为：批密钥更新方案集中处理了加入和离开请求，有效地避免了密钥的频繁更新，可以显著地减少密钥服务器更新开销，提高更新效率，节约服务器资源.

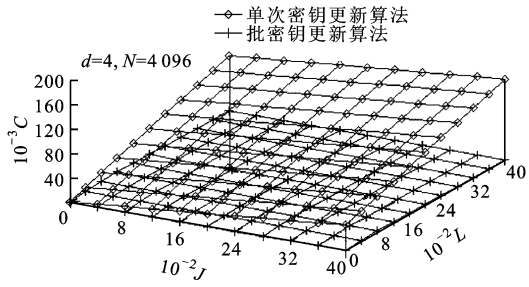


图 5　单次密钥更新和批密钥更新的开销对比

4　结束语

本文针对单次密钥更新算法存在的不足，提出了一种新的批密钥更新算法，建立了更新开销的数学模型，并对算法进行了仿真实验. 理论分析和仿真结果表明，所提算法可以显著地减小更新开销，提高更新效率，节约服务器资源. 下一步将研究周期可变的批密钥更新方案，同时针对各种应用场景，考虑用户加入和离开事件以及用户在群组内的停留时间服从分布函数变化对服务器密钥更新开销的影响.

参考文献：

[1] JUDGE P, AMMAR M. Security issues and solutions in multicast content distribution: a survey [J]. IEEE Network, 2003(1/2): 30-36.

[2] LI Xiaozhou, YANG R, GOUDA M G, et al. Batch rekeying for secure group communications [C]// Proceedings of 10th International World Wide Web Conference. New York, USA: ACM, 2001: 525-534.

[3] ZHANG X B, LAM S S, LEE D Y, et al. Protocol design for scalable and reliable group rekeying [J]. IEEE/ACM Trans on Networking, 2003, 11(12): 908-922.

[4] PEGUEROLES J, RICO-NOVELLA F. Balanced batch LKH: new proposal, implementation and performance evaluation [C]// Proceedings of IEEE Symp Computers and Communacations. Piscataway, NJ, USA: IEEE, 2003: 815-820.

[5] NG W H D, CRUICKSHANK H, SUN Z. Scalable balanced batch rekeying for secure group communication [J]. Elsevier Computers and Security, 2006, 25(2): 265-273.

[6] PHAM T, WATTERS P A. The efficiency of periodic rekeying in dynamic group key management [C]// Proceedings of the 4th European Conference on Universal Multiservice Networks. Piscataway, NJ, USA: IEEE, 2007: 425-432.

[7] NG W H D, HOWARTH M, SUN Z, et al. Dynamic balanced key tree management for secure multicast communications [J]. IEEE Trans on Computers, 2007, 56(5): 590-605.

[8] PRATHAP P M J, VASUDEVAN V. Revised variable length interval batch rekeying with balanced key tree management for secure multicast communications [J]. International Journal of Computer Science and Network Security, 2008, 8(4): 232-241.

[9] JENCHIUN L, CHIENHUA T, FEIPEI L. Optimizing centralized secure group communications with binary key tree recomposition [C]// Proceedings of the 18th International Conference on Advanced Information Networking and Applications. Los Alamitos, CA, USA: IEEE Computer Society, 2004: 202-207.

[10] KWAK D W, LEE J S, KIM J W, et al. An efficient LKH tree balancing algorithm for group key management [J]. IEEE Communication Letters, 2006, 10(3): 222-224.

[11] VARTHINI B P, VALLI S. An efficient group rekeying method using enhanced one way function tree protocol [J]. International Journal of Computer Science and Network Security, 2006, 6(12): 211-218.

（编辑　苗凌）