Optimization Key Tree Structures for Batch Rekeying

Bo LI, Jin PAN, Mingkui HAN, Anlin CHEN, Xingqi XU, Zhiguang CHEN

Xi'an Communications Institute, Xi'an, China

Email: *libo_lb*@126.*com*

Abstract: A new algorithm named ITBR (Improved Tree-based Batch Rekeying) is proposed. The algorithm computes new keys of all renewed nodes in the tree, and then it sends all the rekeying messages, which avoids the problem of out-of-sync between keys and data. To reduce the rekeying cost of this algorithm, the structure of key tree is optimized by three cases. When the number of joins equals to the number of leaves, supposing that every user has probability p of being replaced by a new user, if $0<p\leqslant 0.756$, the optimal tree is a tree with degree equals to four; if $0.244<p\leqslant 1$, the optimal tree is key star. If there are only leaves, the degree of optimal tree is two. If there are only joins, then key star is optimal. Simulation results show that the average rekeying cost can be decreased to 89.2% with optimal key tree.

Keywords: key tree; security; batch rekeying; optimization

批密钥更新中密钥树结构的优化

李　波，潘　进，韩明奎，陈安林，徐邢启，陈志广

西安通信学院，西安，中国，710106

Email: libo_lb@126.com

摘　要：提出了一种新的批密钥更新算法——ITBR(Improved Tree-based Batch Rekeying)算法，算法先计算密钥树中所有更新节点的新密钥，然后发送所有的更新消息，从而避免了数据和密钥不同步的问题。为了降低算法的更新开销，分三种情况对密钥树的结构进行最优化。当加入用户和离开用户数相等时，假设每个用户被新用户替代的概率都为 p，若 $0<p\leq 0.756$，则最优密钥树是度数为 4 的密钥树；若 $0.244<p\leq 1$，则最优密钥树为星型密钥树。如果只有离开用户时，最优密钥树的度数为 2。若只有加入用户时，星型密钥树为最优。仿真结果表明，采用优化后的密钥树结构可以将批密钥更新的开销减少至原来的 89.2%。

关键词：密钥树；单向散列函数；批密钥更新；更新开销

1 引言

随着网络技术的不断发展，许多基于群组通信的应用不断涌现。伴随着这些应用的出现，随之而来的是安全性问题。由于组成员具有动态性，所以组密钥需要不断更新，以确保前向安全性和后向安全性。传统的密钥更新方法是采用实时密钥更新，即每当有用户请求时，就进行密钥更新，这种密钥更新方法可以最大程度上确保安全性，但却存在许多不足[1]。为了有效地解决以上问题，批密钥更新方法被提了出来[1-4]。批密钥更新就是在收到加入或离开请求后，并不立即进行密钥更新，而是每隔一段时间进行一次密钥更新，这段时间叫做密钥更新周期。采用批密钥更新方法，可以有效地提高密钥更新效率，节约服务器资源。

基金项目：国家"863"基金资助项目(2007AA01Z472)

基于逻辑密钥树（LKH）的批密钥更新是目前普遍采用的密钥更新方法，而如何根据组成员的动态变化情况来确定密钥树的最优结构，以使组密钥更新的开销最小是一个难点与热点问题。文献[5]证明在 n 个用户离开后紧接着有 n 个用户加入时，密钥更新开销为 $O(n\log n)$；假定在批密钥更新周期内，每个用户被新用户替代的概率为 p，同时假设密钥树是完全平衡的，且第 i 层的节点拥有 2^{k_i} 个孩子节点，则每个中间节点的度数为 4 是最优密钥树[6]；针对任意密钥树结构，当 $p>0.307$ 时，星型密钥树为最优，$p\leq 0.693$ 时，中间节点的度数最多为 4 的密钥树为最优[7]，当 $p\to 0$ 时，文献[8]设计了一个 $O(n)$ 的启发式算法来产生最优密钥树；假设用户一个接一个的离开群组，则中间节点的度数最大为 5 是最优密钥树[9]。

本文首先提出了 ITBR 算法，然后基于该算法对

978-1-935068-10-5

密钥更新中密钥树的结构进行了优化，最后通过仿真验证了优化的有效性。

2 基于逻辑密钥树的密钥更新机制

在逻辑密钥树中[3, 10]，存在三种密钥，分别是组密钥、辅助密钥和私有密钥。本文将所有用户节点的记为 m 节点，其它节点记为 k 节点，假设密钥树中节点数为 N，则 m_j 表示第 $j(1 \le j \le N)$ 个用户节点，k_i 表示 m_i 拥有的私有密钥 $(1 \le i \le N)$，k_{l-q} 表示 m_l 至 m_q 拥有的私有密钥 $(1 \le l < q \le N)$。图 1(a)所示的是度 d 为 3，节点数 N 为 9，高度 h 为 2 的平衡密钥树。密钥树中每个用户都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_1 从组中离开，则 k_{1-9} 将被更新为 k_{2-9}，k_{1-3} 将被更新为 k_{2-3}，此时组控制器要发送的更新消息为：$\{k_{2-3}\}_{k_2}$、$\{k_{2-3}\}_{k_3}$、$\{k_{2-9}\}_{k_{2-3}}$、$\{k_{2-9}\}_{k_{4-6}}$、$\{k_{2-9}\}_{k_{7-9}}$。如果 m_1 想加入刚才的用户组，则 m_1 将作为 m_2、m_3 的兄弟节点插入密钥树中，此时 k_{2-3} 要被更新为 k_{1-3}，k_{2-9} 更新为 k_{1-9}，GC 将发送以下更新消息：$\{k_{1-9}\}_{k_{2-9}}$、$\{k_{1-9}\}_{k_1}$、$\{k_{1-3}\}_{k_9}$、$\{k_{1-3}\}_{k_{2-3}}$。

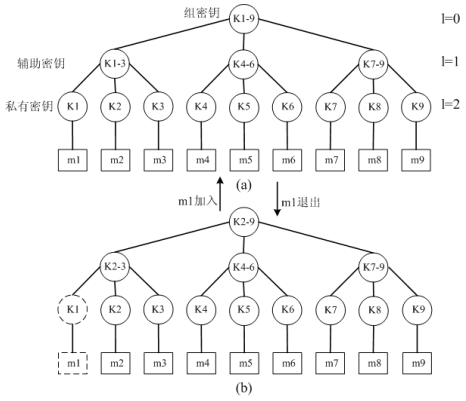

Figure 1. m_1 **joins/leaves the group**
图 1. m_1 **加入/退出小组情况**

3 ITBR 算法

文献[11]提出了一个批密钥更新算法，但是存在以下不足：在更新节点的兄弟节点进行密钥更新之前，发送的消息不可能用更新后的密钥进行加密，如果采用之前的密钥进行加密，那么新成员将无法解密这个密钥。本文提出了一种新的批密钥更新算法——ITBR 算法，算法通过对标记节点搜索并计算更新路径中的更新密钥，在所有更新密钥计算完之后，集中发送更新消息，有效避免了文献[11]中算法的不足，且实现了密钥更新过程中标记更新子树、计算更新密钥和发送更新消息的全自动完成。算法符号说明如下：n_i 为 m_i 的父节点，Q_i 为 R_i 的父节点，R_i 为 r_i 的父节点，k_{Ri} 为 R_i 的密钥，k_{rbj} 为密钥更新之后，t_{bj} 节点的密钥。具体算法如下：

Step 1：GC 与所有加入用户进行相互认证，为每个用户分配私有密钥及随机数 r'；

Step 2：将所有更新节点标记为 UPDATE；

Step 3：建立并发送更新通知消息；

Step 4：在密钥树中搜索标记为 UPDATE 的叶节点，记为 $m = \{m_1, m_2, m_3 \cdots m_J\}$；

Step 5：随机搜索一个 m 中的节点，记为 m_i 节点，令 m_i 为 r_i，n_i 为 R_i；

Step 6：计算 R_i 的密钥，$k'_{Ri} \to k_{Ri}$ 将 r_i、R_i 标记为 updated，记 r_i 的所有兄弟节点为 r_{bi}，$r_i \to q_j$，$r_{bi} \to t_{bj}$，$k_{Ri} \to K_{Rj}$，$Q = \{q_j \mid j = j++\}$，$T = \{t_{bj} \mid j = j++\}$，$K = \{K_{Rj} \mid j = j++\}$；

Step 7：若 R_i 为树根节点或 Q_i 已被标记为 updated，则令 $m' = m - \{m_i\}$，$m' \to m$，转 Step 5；否则，令 R_i 为 r_i，Q_i 为 R_i，转 Step 6；

Step 8：对于 Q 中所有节点 q_j，GC 向其兄弟节点 t_{bj} 发送密钥更新消息，$M_{rbj} = f(k_{rbj} \oplus r') \oplus K_{Rj}$；

4 批密钥更新中最优密钥树结构

与单次密钥更新算法相比，批密钥更新算法可以有效减少密钥更新开销。然而，如果采用批密钥更新算法，则密钥树的度数 d 取值的不同将直接影响密钥更新开销的大小。本文分以下三种情况对密钥树最佳度数的选择进行分析。

4.1 加入用户和离开用户数相等的情况

当群组达到稳定时，在一个批密钥更新周期内，加入用户数和离开用户数是近似相等的。也就是说，当一个用户离开了，一个新用户将会被放置在离开用户所在节点位置。因而，我们可以假设在一个批密钥更新周期内，密钥树中的 n 个叶节点都有相同的概率 p 被新用户替代。那么，对于一个拥有 N_v 个子孙叶节点、d_v 个孩子节点的中间节点，其密钥被更新的概率为 $1-(1-p)^{N_v}$，则密钥更新需要发送的更新消息的数量为 $d_v[1-(1-p)^{N_v}]$。因此，我们可以将密钥树的最优化度数问题定义如下：

定义 1：对于两个给定的参数，$0 \le p \le 1$ 及 $n > 0$，记 $q = 1 - p$，对于一个拥有 n 个叶节点和节点集合 V

（包括中间节点和叶节点）的密钥树，集合 V 中的权重函数 $w(u)=\begin{cases}0, & u=r\\ 1-q^{N_v}, & u\neq r\end{cases}$，其中 v 是 u 的父节点，密钥更新开销 $C(T)=\sum_{u\in V}w(u)$，那么 $C(T)$ 最小的密钥树就是最优密钥树。

定义 2：含有 k 个叶节点且高度为 1，即 $d=k$ 的密钥树又称为 k 阶星型密钥树。

于是我们有以下定理成立：

定理 1：对于高度为 h、度数为 d、用户数为 n 的完全平衡密钥树，$C(T)=\sum_{u\in V}w(u)=d\sum_{l=0}^{h-1}d^l\cdot(1-q^{\frac{n}{d^l}})$

证明 1：考虑密钥树中第 $l(0\le l\le h-1)$ 层的节点 v_l，该层共有 d^l 个 v_l 节点，每个 v_l 节点拥有 $\frac{n}{d^l}$ 个子孙叶节点。所以，对于每个 v_l 节点，其密钥被更新的概率为 $(1-q^{\frac{n}{d^l}})$，由于第 l 层每个 v_l 节点被更新的概率相同且相互独立，故该层可能被更新的节点数为 $d^l\cdot(1-q^{\frac{n}{d^l}})$。每个更新节点需要发送的更新消息数量都为 d，所以整个密钥树的更新开销为 $d\sum_{l=0}^{h-1}d^l\cdot(1-q^{\frac{n}{d^l}})$，得证。

当 $1-3^{-1/3}\le p\le 1$ 时，n 阶星型密钥树为最优，

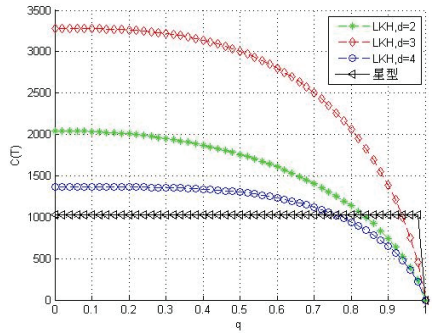

Figure 2. Comparison of rekeying cost

图 2. 密钥更新开销对比情况

对于 $0\le p\le 1-3^{-1/3}$，其中间节点的度数 d 最大值为 4 的密钥树最优[7]。那么，应如何确定 d 的最佳值呢？由于 $C(T)=\sum_{u\in V}w(u)=d\sum_{l=0}^{h-1}d^l\cdot(1-q^{\frac{n}{d^l}})$，那么可以将 $C(T)$ 看成是关于 d 和 q 的函数，则 $C(T)$ 取最小值时对应的 d 即为最优密钥树度数。假设组用户数 n 为 1024，q 在 $[0,1]$ 变化，如图 2 所示，当 $0<q\le 0.756$ 时，n 阶星型密钥树最优，当 $0.244<q\le 1$ 时，d 为 4 的密钥树最优。

4.2 只有用户离开的情况

对于只有用户离开群组的情况，我们首先给出如下定义：

定义 3：对于用户数为 n、高度为 h、度数为 d 的密钥树，如果只有 L 个用户离开，则组密钥更新在最差情况下的开销记为 $C(L,d)$，因此，$\min C(L,d)$ 对应的密钥树记为最优密钥树。

对于用户数为 n、高度为 h、度数为 d 的完全平衡密钥树，如果只有 L 个用户离开，使用 ITBR 算法进行组密钥更新，则密钥更新开销为：

1) $d=n$

$$C(L,d)=n-L \tag{1}$$

2) $2\le d<n$

- $L\le d$

$$C(L,d)=(d-1)[(h-1)L+1]-L \tag{2}$$

- $d<L<d^{h-1}$

$$C(L,d)=(d-1)[\frac{d^{1+\lfloor \log_d L\rfloor}-1}{d-1}+L(h-1-\lfloor \log_d L\rfloor)]-L \tag{3}$$

- $L\ge d^{h-1}$

$$C(L,d)=(d^h-1)-L \tag{4}$$

假设群组中原有用户数 n 为 1024，L 的变化范围为 $[0,1024]$，密钥树的度数 d 的变化范围为 $[2,10]$。为了方便比较，我们将 $d=n$ 时，$C(L,d)$ 的值用图 3 中 d 取 1 时 $C(L,d)$ 的值表示。如图 3 所示，当 L 一定时，d 取 2 时 $C(L,d)$ 的值最小，d 取 5 时 $C(L,d)$ 的值最大，当 $d\ge 6$ 时，随着 d 的增加，$C(L,d)$ 的值也在递增。由此可见，在只有用户离开群组的情况下，最优密钥树的度数 d 为 2。

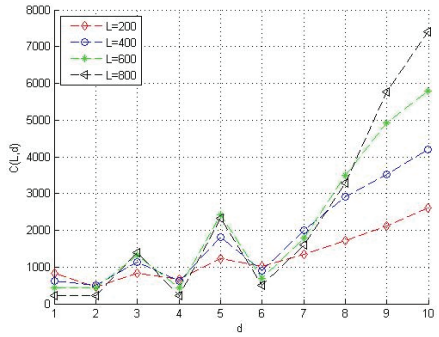

Figure 3. Rekeying cost varying with d

图 3. 密钥更新开销随 d 变化情况

4.3 只有用户加入的情况

当只有加入用户时，不用考虑离开用被加入用户替代的概率，假设有 J 个用户加入群组，则采用 ITBR 算法进行密钥更新时的开销为：

1) $d=n$

978-1-935068-10-5 © 2010 SciRes.

$$C(J,d) = J+1 \tag{5}$$

2) $h \geq 2, 2 < d < n$

$$C(J,d) = (d-1)[(h-1)+\sum_{j=1}^{h-2}\left\lfloor \frac{(J-1)}{(d-1)d^{j+1}} \right\rfloor]+J \tag{6}$$

3) $h \geq 2, d = 2$

$$C(J,d) = (d-1)(h+J-1) \tag{7}$$

假设群组中原有用户数 n 为 1024，J 的变化范围为[0,1024]，密钥树的度数 d 的变化范围为[2,10]。为了方便比较，我们将 $d=n$ 时，$C(J,d)$ 的值用图 4 中 d 取 1 时 $C(J,d)$ 的值表示。如图 4 所示，随着 J 的增加，$C(J,d)$ 的值也在递增；当 J 一定时，d 取 1 时 $C(J,d)$ 的值最小，d 取 3 时 $C(J,d)$ 的值最大，d 取 2 时，$C(J,d)$ 的值略大于 d 取 1 时的值。由此可见，在只有用户加入群组的情况下，最优密钥树的度数 d 为 n，即星型密钥树。

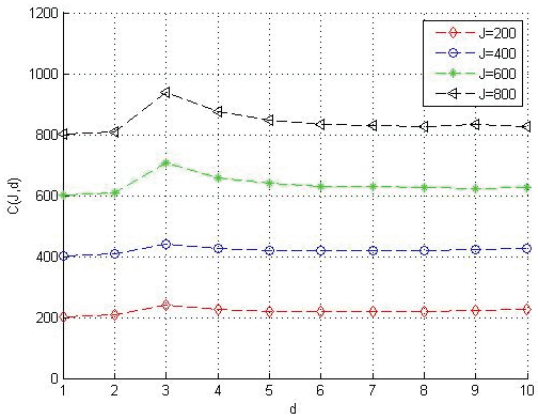

Figure 4. Rekeying cost varying with d

图 4. 密钥更新开销随 d 变化情况

4.4 性能仿真

本文在 1.6GHz 处理器、512M 内存的终端对 ITBR 密钥更新算法进行了仿真。组用户数 N 为 1024，p 取 0.5，仿真分两种情况：一种是采用 d 为 4 的密钥树，另一种采用优化后的密钥树结构。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 GC 发送的更新消息的条数为指标。仿真运行 100 次，取平均值作为服务器的更新开销。

如图 5 所示，采用优化后的密钥树进行密钥更新密钥的更新开销整体上要小于采用 d 为 4 的密钥树，约为 89.2%。由于 p 取 0. 5，故当加入用户数和离开用户数相等时，最优密钥树的 d 也为 4，所以此时两种情况下的更新开销相同。

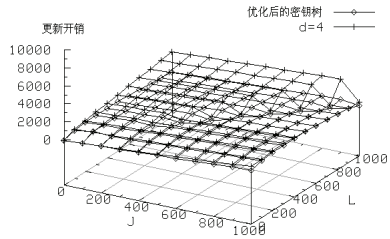

Figure 5. Comparison of rekeying cost

图 5. 密钥更新开销对比

5 结束语

批密钥更新问题是安全群组通信研究领域的热点问题，而密钥树结构的优化问题是批密钥更新问题研究中的重点之一。本文首先提出了一种新的批密钥更新算法——ITBR 算法，然后基于该算法对密钥树的结构进行了优化，仿真结果表明，采用优化后的密钥树进行密钥更新时的开销要小于优化前的更新开销。下一步将考虑如何根据组成员的动态性来确定密钥更新周期的大小，从而进一步降低密钥更新的开销。

References (参考文献)

[1] X. Li, Y. R. Yang, M. G. Gouda, et al. Batch rekeying for secure group communications[C]. In Proc. 10th Int. World Wide Web Conf., Hong Kong, 2001: 525–534.

[2] T. Pham, P. A. Watters. The Efficiency of Periodic Rekeying In Dynamic Group Key Management[C]. Proceedings of the Fourth European Conference on Universal Multiservice Networks, 2007: 425–432.

[3] W. H. D. Ng, M. Howarth, Z. Sun, et al. Dynamic Balanced Key Tree Management for Secure Multicast Communications[J]. IEEE Transactions on Computers, 2007, 56(5): 590–605.

[4] P. M. J. Prathap, V. Vasudevan. 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.

[5] J.Snoeyink, S.Suri, G.Varghese. A lower bound for multicast key distribution [C]. In Proceedings of the Twentieth Annual IEEE Conference on Computer Communications, 2001, 422–431.

[6] F.Zhu, A.Chan, G.Noubir. Optimal tree structure for key management of simultaneous join/leave in secure multicast [C]. In Proceedings of the IEEE Military Communication Conference (MILCOM), Boston, 2003, 773–778.

[7] R. L.Graham, M.Li, F. F.Yao. Optimal Tree Structures for Group Key Management with Batch Updates[J]. SIAM Journal on Discrete Mathematics, 2007, 21(2), 532–547.

[8] M.Li, Z.Feng, N. Zang, et al. Approximately optimal trees for group key management with batch updates [J]. Theoretical Computer Science, 2009, 410:1013–1021.

[9] Z. Z.Chen, Z.Feng, M. Li, et al. Optimizing Deletion Cost for Secure Multicast Key Management[J]. Theoretical Computer Science, 2008, 401: 52–61.

[10] B. P.Varthini, S. Valli. 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.

[11] Baohong Li, Yibin Hou, Yinliang Zhao. Performance Optimization for Rekeying Mechanism in Secure Multicast[J]. Journal of Xi'an Jiaotong University, 2004, 38(10): 1053–1056 (In Chinese).