路径洗牌算法：安全组播中一种高效的组密钥更新算法

刘小虎[①②]　顾乃杰[①②]　陆余良[③]　毕　坤[①②]
[①](中国科学技术大学计算机科学与技术系　合肥　230027)
[②](安徽省计算与通信软件重点实验室　合肥　230027)
[③](解放军电子工程学院网络工程教研室　合肥　230037)

摘　要：安全组播通信使用组内所有成员共享的组密钥来加密通信内容。为了保障安全，密钥服务器需要在组成员关系改变时进行组密钥更新(rekey)。由于组内成员关系的动态性和加解密操作的高代价，组密钥更新性能成为衡量组密钥管理性能的主要指标。基于密钥树(key tree)的组密钥更新方法已经被广泛地使用，并达到了对数级的组密钥更新代价。密钥树的结构需要保证平衡，否则最坏情况下组密钥更新的通信代价会达到 $O(n)$。该文提出了一种新的基于密钥树的路径洗牌算法 PSA(Path Shuffling Algorithm)，该算法能够将密钥树的平衡操作分散到一般的更新密钥操作中，减少了结构调整代价，从而提高了算法的性能。理论分析给出了该算法更新组密钥的平均通信代价，模拟实验也验证了这种算法更新组密钥的平均性能要优于其它同类算法。

关键词：组播；组密钥更新；路径洗牌算法
中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1009-5896(2007)10-2477-05

PSA: An Efficient Group Key Updating Algorithm in Secure Multicast

Liu Xiao-hu[①②]　Gu Nai-jie[①②]　Lu Yu-Liang[③]　Bi Kun[①②]
[①](*Dept. of Computer Science & Technology, University of Science and Technology of China, Hefei* 230027, *China*)
[②](*Anhui Province Key Laboratory of Computing and Communication Software, Hefei* 230027, *China*)
[③](*Teaching and Research Office of Network Engineering, Electronic Engineering Institute of PLA, Hefei* 230037, *China*)

Abstract: Secure multicast uses a group key shared by all group members to encrypt group communication. To ensure security, key server should update the group key (rekey) when an existing member leaves the group or a new member joins the group. Key tree approach is widely used to achieve logarithmic rekeying cost, but the key tree structure has to be kept balanced, otherwise the communication cost in the worst cast will be $O(n)$. In this article a new algorithm named PSA (Path Shuffling Algorithm) is proposed based on key tree. PSA can scatter the balanced operations in normal rekeying operations, so it reduces the restructuring cost and thus improves the performance. Theoretical analysis gives the average updating cost, and experiments show that PSA algorithm has better average-case rekeying performance than other group key updating algorithms.

Key words: Multicast; Rekey; Path shuffling algorithm

1　引言

组播通信在路由器的支持下，能够将一个 IP 数据包有效地发送至多个目的地，节省了网络资源，因此越来越多的网络应用[1]中都利用组播来发送数据。为了保障组播通信的安全性，安全组播需要使用所有组成员共享的传输加密密钥 TEK(Traffic Encrypting Key)来加密组内通信内容。当旧成员离开或者新成员加入时，密钥服务器更新组密钥并发送至合法组成员的过程，称为组密钥更新(rekey)。Canetti 等人[2]认为“通信复杂性是当前应用中最大的瓶颈”，因此本文采用组密钥更新时的通信代价作为组密钥更新代价的衡量标准。

根据密钥服务器组织密钥的方式，可以将组密钥更新方法分为两类：星形的和树形的。树形的组密钥更新算法(即密钥树方法)被广泛地运用在应用和研究中。星形的组密钥更新算法为每个成员分别维护 TEK，组密钥更新代价与组成员规模成正比，这样在组成员规模扩大时，系统的开销非常大，因而不具有可扩展性。为了有效更新组密钥，Wallner 等人[3]和 Wong 等人[4]各自独立地提出密钥树(key tree)方法，使用树形结构的辅助密钥来帮助进行组密钥更新。当密钥树为一棵度为 k 的完全树且组成员的规模为 n 时，组密钥更新(rekey)的代价为 $O(\log_k n)$，这样组密钥更新的通信代价要优于星形结构中的通信代价。

在基于密钥树的组密钥更新方法中，KS 存储一棵度为 k 的密钥树，其中每个结点对应一个密钥。树的叶子上是 KS 和各个组内成员分别协商的 n 个加密密钥，非叶子结点对应

2006-03-06 收到，2006-10-16 改回
国家自然科学基金(60533020)资助课题

辅助密钥，TEK 则位于根结点上。每个成员都存储了辅助密钥集合的一个子集，单个成员存储的密钥集合是从这个成员对应的叶子结点到根结点的路径上所有密钥的集合，因此完全的密钥树中每个成员平均存储的密钥数为 $\lceil \log_k n \rceil$。当新成员加入时，为了保证前向安全性，这个成员所对应密钥树上叶子结点到根结点的路径上所有密钥都必须更新，使得新成员无法解密以前的消息。当旧成员离开组时，为了保证后向安全性，这个成员对应密钥树上叶子结点到根的路径上的所有密钥也都必须更新。

国内外都对密钥树进行了进一步的研究，并提出了很多针对密钥树的改进方法。Chang 等人[5]和 Setia 等人[6]中采用批量更新组密钥(batch rekeying)的方法来降低组密钥更新代价，主要将组内成员关系变化时的实时更新改变为分批更新组密钥，一次更新多个组成员关系。这种方法能够显著减少密钥更新代价，但是降低了安全性。Snoeyink 等人[7]分析了静态密钥树的性能，并证明了使用 2-3 树结构来组织密钥树能够得到最优的组密钥更新性能。

随着组成员关系的持续改变，密钥树可能变得不平衡，在最坏情况下组密钥更新代价可能达到 $O(N)$。为了保证密钥树的在线性能，需要保证密钥树的持续平衡。Moyer[8]采用一种简单的加入删除策略来保证密钥树的平衡：成员加入时选择密钥树上具有最低深度的可加入点作为加入点，而在成员删除时，则采用两种方法来提高性能，一种是删除结点后对密钥树结构进行调整，但是调整的代价比较高；另一种是对一组更新操作暂时保持密钥树的不平衡，然后阶段性地调整密钥树结构，这类似于 Chang 等人[5]和 Setia 等人[6]提出的批量更新密钥方法，也降低了安全性。

Goshi 和 Ladner[9]中提出了 3 种基于平衡密钥树的组密钥更新算法，其中高度平衡的 2-3 树算法(hb23 树算法)总体实验性能最好。hb23 树算法的主要思想是将密钥树组织成 2-3 树的结构，对成员的加入位置选择一个深度最低的可加入点，而在每次成员删除之后对密钥树结构进行调整，以保证密钥树的高度平衡。Goshi 和 Ladner 提出的几种算法都具有良好的密钥树结构，但是都要求比较高的调整代价，因此平均密钥更新性能不如 Wong 等人[4]提出的无平衡策略的密钥树算法。

本文提出一种新的组密钥更新算法：路径洗牌算法 PSA(Path Shuffling Algorithm)来进行组密钥更新，这种方法能够将密钥树的平衡操作分散在一般的组密钥更新操作中，利用 Huffman 结构[10]的平衡性和易于构造的特点来重组密钥树，在保证密钥树平衡的同时只需要少量额外的系统开销，因此具有良好的性能。下面各个部分组织如下：第 2 节介绍基本的密钥树结构与操作要求，第 3 节介绍路径洗牌算法及其性能的理论分析，第 4 节给出了实验的总体设计和比较结果，最后是结束语。

2 算法要求与描述

本文实现的路径洗牌算法 PSA 是一种基于密钥树结构的组密钥更新算法。为了分析和描述 PSA 算法，下面先定义一些基本概念并对基本的密钥树算法进行说明。为了方便比较，这里以 Goshi 和 Ladner[9]所用的定义进行说明。通信代价定义为组成员关系改变时，密钥服务器更新组密钥时所需要发送的加密消息的数目。这也是衡量组密钥更新算法性能的主要标准。密钥树中结点 i 的权值 ω_i 定义为密钥树中从结点 i 到根结点路径上所有结点的度之和，这样子结点的权值要大于其父结点的权值，而根结点的权值为 0。

与文献[1]中定义不同，为了比较平均性能，这里定义密钥树 T 的权值为密钥树上所有叶子结点权值之和：

$$\omega(T) = \sum_i \omega_i \text{，其中 } i \text{ 为 } T \text{ 上的叶子节点} \quad (1)$$

设组成员数目为 n，即密钥树上叶子结点的个数为 n，则密钥树 T 的平均权值为

$$\varpi(T) = \frac{1}{n}\sum_i \omega(i)\text{，} i \text{ 为 } T \text{ 上的叶子节点} \quad (2)$$

因为结点权值是密钥服务器从密钥树中删除这个结点需要的通信代价的一个很好的近似[9]，所以可以用 $\varpi(T)$ 来衡量密钥树的删除操作的平均通信代价。因为密钥树上的删除节点的操作位置固定但加入节点的位置可以自由选择，而且节点删除操作的代价要远高于新节点加入操作的代价，所以本文将 $\varpi(T)$ 作为评价密钥树结构好坏的标准。

3 路径洗牌算法 PSA

本文提出了一种新的密钥树更新算法：路径洗牌算法 PSA。这种算法在基本的密钥树操作上着重考虑密钥树更新的平衡策略，主要目的是一方面保证密钥树具有对数级最坏情况更新代价，另一方面尽量降低密钥树的平均更新代价。

根据前面所述，平衡的密钥树结构是保证密钥树更新性能的关键。本文也采用高度平衡的加入策略(即通过函数 GetBestInsertionPoint 选择密钥树上深度最低的空闲位置)，主要为了尽量减少密钥树中父结点的各个子树高度之间的差异。路径洗牌算法中的加入操作如图 1 所示。

```
Algorithm  Add( TreeNode* toadd)
1   parent ← GetBestInsertionPoint ();
2   if (parent ≠ root) && (parent->IsLeaf() == true)
3         AddChildToLeaf (toadd, parent);
4   else  AddChildNoOverflow (toadd, parent);
```

图 1 路径洗牌算法中的加入操作(算法格式参考文献[11])

离开操作的性能是保证密钥树更新性能的关键，因为密钥树中新成员加入的位置可以有很多种选择，而成员离开的位置是固定的，而且密钥树中相同位置上成员离开的更新代

价要高于加入代价。hb23 树算法中的离开策略中在成员删除后判断密钥树高度是否不平衡，在不平衡时才进行密钥树结构调整，而且调整需要额外的结构调整代价。

路径洗牌算法中的加入策略采用数量平衡的加入策略，离开操作则比较复杂。在基本的密钥树结构中，成员离开时需要更新一部分内部结点，但是有些内部结点是不需要更新的。本文将根结点需要更新而其它内部结点都不需要更新的子树称为独立子树。这些独立子树在密钥更新时只需要用根结点的加密密钥对新密钥进行加密就可以保证整棵子树的密钥更新要求。本文注意到这一点，在离开操作中将这些独立子树组织为一棵 Huffman 结构的密钥树，这样既能保证组密钥更新要求，又能提高密钥树的平衡性能。因为这些独立子树出现在离开结点到根结点的路径上，而把它们重组为 Huffman 结构可以看作将它们重新洗牌一样，所以本文把这种算法称为路径洗牌算法。这种算法能够将密钥树的平衡操作分散到一般的更新密钥操作中，既能保证了密钥树的结构平衡，又减少了结构调整的额外代价，因而具有更好的性能。

```
Algorithm  Delete( TreeNode*  todelete)
1   parent ← todelete->parent;
2   if parent = root
3      then RemoveChildNoShuffling (todelete);
4           return;
5   PriorityQueue *shufflingsubtrees ←new PriorityQueue
    (weightbalanced);
6   while ( parent ≠ NULL )
7      for  i←1  to  parent->GetNumChildren()
8       do shufflingsubtrees->Insert (parent-> GetChild (i));
9      parent ← parent->GetParent ();
10  BuildHuffmanTree(shufflingsubtrees, 3);
```

图 2　路径洗牌算法中的删除操作

在上面给出的算法中，对于成员删除请求，可按图 2 中路程洗牌算法中的删除操作。首先得到成员的父指针，如果待删除结点的父结点是密钥树的根结点，那么直接删除掉这个结点，不需要进行结构调整操作，而如果这个待删除结点的父结点不是密钥树的根结点，那么就需要进行路径洗牌操作。为了进行路径洗牌算法，首先建立一个优先队列 Q，以叶子结点数目作为优先级；然后将待删除结点到根结点的路径上的结点的所有子结点插入这个优先队列 Q；最后根据得到的优先队列 Q，建立一棵度为 3 的 Huffman 树，而密钥树的更新代价则取决于这棵 Huffman 树的性能。

4　算法性能分析

下面分析路径洗牌算法得到的密钥树 T 的平均权值 $\varpi(T)$ 的上界，根据 Goshi 和 Ladner [9]所述，这个上界也可以作为密钥树结构上组密钥平均更新代价的上界。设路径洗牌算法得到的密钥树 T 具有 n 个叶子结点，则有

定理 1　　$\varpi(T_n) \leqslant 3\log_3 n + 4$，

证明　当 n 取 1、2 和 3 时，上式明显成立。

假设上式对于所有正整数 $n<N$ 都成立，则有

$$\varpi(T_n) \leqslant 3\log_3 n + 4,\quad n<N \tag{3}$$

而当 $n=N$ 时，设最终的密钥树由 K 个子树构成，这 K 个子树上叶子结点对应的成员数目分别为 $m_1, m_2, \cdots, m_K$，而构造成密钥树 T_N 后，这 K 个子树的根结点到密钥树 T_N 的路径长度分别为 $l_1, l_2, \cdots, l_K$ 则

$$\begin{aligned} N\bullet\varpi(T_N) &\leqslant 3\sum_{k=1}^{K} m_k l_k + \sum_{k=1}^{K}\varpi(T_{m_k}) \\ &= 3N\sum_{k=1}^{K}\frac{m_k}{N}l_k + \sum_{k=1}^{K}\varpi(T_{m_k}) \end{aligned}$$

又根据文献[12]中最优编码定理得到

$$\sum_{k=1}^{K}\frac{m_k}{N}l_k < \sum_{k=1}^{K}\frac{m_k}{N}\log_3\frac{m_k}{N} + 1$$

$$\begin{aligned} \text{原式} &\leqslant -3N\sum_{k=1}^{K}\frac{m_k}{N}\log_3\frac{m_k}{N} + 3N + \sum_{k=1}^{K}\varpi(T_{m_k}) \\ &\leqslant 3\sum_{k=1}^{K} m_k \log_3 N + \sum_{k=1}^{K} 3\log_3 m_k \\ &\quad - 3\sum_{k=1}^{K} m_k \log_3 m_k + 4K + 3N \end{aligned}$$

因为 $m_k \geqslant 2$，得到上式$= 3N\log_3 N - \sum_{k=1}^{K} 3\log_3 m_k + 4K + 3N$

$\leqslant 3N\log_3 N + 4N$。

这样得到 $\varpi(T_N) \leqslant 3\log_3 N + 4$，所以式(3)对所有正整数 n 均成立。　　证毕

以上证明了定理 1，也就是通过路径洗牌算法来进行密钥更新操作，得到的新密钥树的平均权值 $\varpi(T)$ 为 $O(3\log_3 n)$，也就代表平均情况下这种密钥树更新算法的通信复杂度为 $O(3\log_3 n)$。

5　实验结果

本文用 C++实现了路径洗牌算法，并与 Goshi 和 Ladner[9]提供的高度平衡的 2-3 树算法[1)](简称 hb23 树算法)进行了比较，同时比较的还有 Wong 等人[4]中提供的固定度分别为 3 和 4 的密钥树算法(分别简称 degree3 树和 degree4 树)。

图 3 至图 6 给出了对 degree3 树，degree4 树，hb23 树算法和路径洗牌算法这 4 种组密钥更新算法的实验结果。下面的图形中包含了 4 组实验数据，实验主要是通过构造大数

[1)] 非常感谢 Goshi 和 Ladner 提供源代码

据量的测试序列来比较这几种算法的平均性能。组的初始规模为 100,000 个成员，新加入的成员根据各自组密钥更新算法的加入算法添加进密钥树中，图 3 至图 5 对应前 3 组实验，这 3 组实验中即将被删除的成员通过随机函数选择，图 6 对应第 4 组实验，这一组实验中即将被删除的成员是特别构造的。每一组实验中的操作次数为 100,000 个，其中加入操作和删除操作的数量保证一定的比率，这 3 组实验中加入/删除比率分别取 20%、50%和 80%。第 4 组实验中将被删除的成员尽量选择密钥树上相近的结点所对应的成员，这样的操作序列很容易让密钥树变得不平衡，因此会让缺乏平衡机制的组密钥更新算法的性能大大下降。

实验结果的示意图中，横坐标代表操作数目，以 1000 个操作为一个单位，总共有 100 个单位，共计 100,000 次操作，纵坐标代表平均更新代价。这 4 张图中都包含 4 条曲线图，每条曲线图对应一种算法的实验结果，每条曲线由 100 个点构成，每个点代表这 1000 个操作的平均更新代价。

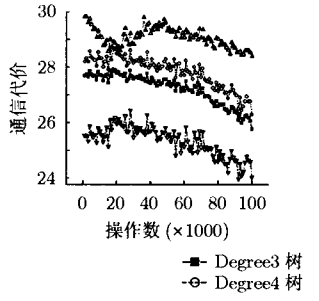

图 3 加入/删除比率为 20%的随机操作

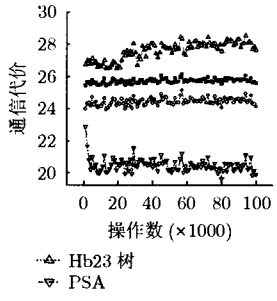

图 4 加入/删除比率为 50%的随机操作

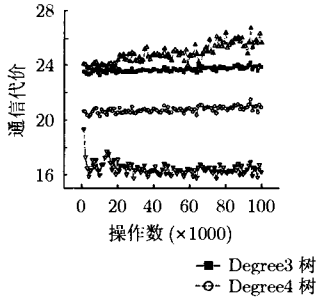

图 5 加入/删除比率为 80%的随机操作

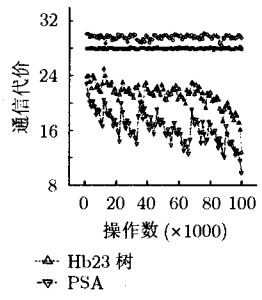

图 6 恶意构造的删除操作

这 4 张图形的区别很明显：第 1 张图的 4 条曲线都呈下降趋势，代表删除的结点要多于加入的结点，因为成员数目越少，通信代价越低，因此曲线呈下降趋势；第 2 张图形曲线比较平稳，代表加入和删除成员比例保持平衡。第 3 张图中这 4 条曲线都有上升的趋势，但是不是很明显，这是因为加入代价比删除代价要小得多，因此这一组操作中主要以加入为主的操作，变化不如删除操作为主的操作序列对应的通信代价变化大。第 4 组操作全都是恶意构造的删除操作，特意选择密钥树上临近的结点进行删除，这对一般的 degree3 和 degree4 树的组密钥更新性能影响很大，随着成员数目的减少，密钥树更新代价仍然没有下降，但是采用了平衡机制的 Hb23 树算法和 PSA 算法就能够随着成员数目的减少而降低密钥树更新代价。

综合这 4 张图形，可以得出以下结论：

(1) Goshi 和 Ladner 提出的 hb23 树算法的平均性能在一般情况下都不如其它 3 种算法。这意味着 hb23 树算法的平衡机制降低了算法的平均更新性能，在前面 3 组实验中的平均性能甚至不如不采用平衡机制的 degree3 树和 degree4 树算法。

(2) 对于恶意构造的删除序列，hb23 树算法要高于 degree3 树算法和 degree4 树算法，但是仍然要低于 PSA 算法。这说明 hb23 树算法的平衡机制在密钥树非常不平衡时，对算法性能的改进作用要大于平衡操作的额外开销，但是不如 PSA 算法的平衡机制。

(3) 总的说来，PSA 算法在实验构造的 4 组操作序列中的平均更新性能都要明显优于其他 3 种算法。PSA 算法能够将密钥树的平衡操作分散到一般的更新密钥操作中，减少了结构调整代价，从而提高了算法的性能。

6 结束语

基于密钥树的组密钥更新方法能够在保持密钥树平衡的情况下，达到对数级的组密钥更新性能。本文提出了一种新的基于密钥树的路径洗牌算法 PSA，这种算法将密钥树的平衡操作分散在一般的更新密钥操作中，在删除操作的时候，将密钥树重组为一棵 Huffman 结构的密钥树，既不需要添加复杂的数据结构，又不需要额外的密钥树结构调整操作，因此提高了算法的实用性与有效性。

目前模拟实验结果表明 PSA 算法的效率很高，但是在实际应用中，组播应用的周期可能更长，成员的改变更具有随机性。为了进一步提高算法性能，需要根据成员关系的不同状态来改变 PSA 算法的相关参数。这是下一步具体的研究工作。

参 考 文 献

[1] Williamson Beau. Developing IP Multicast Networks. Indianapolis: Cisco Press, 2000:15-17.

[2] Canetti R, Malkin T, and Nissim K. Efficient communication- storage tradeoffs for multicast encryption. In Advances in Cryptology, Eurocrypt '99, Prague of Czech Republic, LNCS 1592, May 1999, 459-474.

[3] Wallner D, Harder E, and Agee R. Key management for multicast: issues and architectures. IETF RFC 2627, June 1999.

[4] Wong C K, Gouda M, and Lam S S. Secure group communication using key graphs. *IEEE/ACM Transactions on Networking*, 2000, 8(1): 16-30.

[5] Chang I, Engel R, and Kandlur D, *et al*. Key management for secure internet multicast using boolean function minimization techniques. Proceedings of IEEE Infocom, New York, March 1999(2): 689-698.

[6] Setia S, Koussih S, and Jajodia S, *et al*. Kronos: a scalable group re-keying approach for secure multicast. Proceedings of IEEE Symposium on Security and Privacy, Washington, May 2000: 215-228.

[7] Snoeyink J, Suri S, and Varghese G. A lower bound for multicast key distribution. Proceedings of IEEE Infocom, Alaska, April 2001, 1: 422-431.

[8] Moyer M, Rao J, and Rohatgi P. Maintaining balanced key trees for secure multicast. Internet Draft, http://www.ietf.org/ internet-drafts/draft-irtf-smug-key- tree-balance-00.txt. June 1999.

[9] Goshi J and Ladner R E. Algorithms for dynamic multicast key distribution trees. Proc. ACM Symp. Principles of Distributed Computing (PODC 2003), New York, July 2003: 243-251.

[10] Huffman D A. A method for the construction of minimum redundancy Codes. *Proc. IRE*, 1952, 40: 1098-1101.

[11] Cormen T H, Leiserson C E, and Rivest R L, *et al*. Introduction to Algorithms. Second Edition, Massachusetts: The MIT Press, 2002: 15-20.

[12] 姜丹. 信息论与编码. 第一版, 合肥:中国科学技术大学出版社. 2000: 320-321.
Jiang Dan. Information Theory and Coding. First Edition. Hefei: University of Science and Technology of China Press. 2000: 320-321.

刘小虎：　男，1979 年生，博士生，研究方向为安全组播.

顾乃杰：　男，1961 年生，教授，主要研究方向为并行分布式算法、多级互联网结构、路由算法和组播问题研究.

陆余良：　男，1964 年生，教授，主要研究方向为计算机网络安全.

毕　坤：　男，1981 年生，博士生，研究方向为计算机网络中的算法.