文章编号:1001 - 9081(2010)01 - 0220 - 04

基于权重加入离开树的贡献型组密钥管理方案

曹震寰,车彦刚
(甘肃省信息中心,兰州 730030)
(matergg@126.com)

摘　要:由于相对较重的计算开销,贡献型组密钥管理方案致力于提高系统的扩展性和组密钥更新的效率。提出一种基于权重加入离开树的贡献型组密钥管理方案 WJT。首先给出了密钥树结构;其次,为了降低成员离开时的时间复杂度,将离开树组织为权重树;最后,根据最优化方法选取了最优的离开树的大小,并确定了离开树的激活条件。理论分析和仿真表明,WJT 在成员加入和离开时密钥更新的时间复杂度为 $O(1)$。

关键词:组播;组密钥;组安全通信;逻辑密钥树
中图分类号: TP391　　**文献标志码**:A

Weighted-join-exit-tree-based contributory group key management scheme for key rekeying

CAO Zhen-huan, CHE Yan-gang
(*Gansu Information Center, Lanzhou Gansu* 730030, *China*)

Abstract: Contributory group key management works on how to make the system more scalable and time efficient due to relatively heavy computation cost. In this paper, a contributory group key management based on Weighted-Join-Exit-Tree, named as WJT was presented to get better time efficiency in key updates. First, a new key tree topology was put forward. Then, exit tree was organized as a weighted tree to reduce the key updating complexity when user leaves. Last, optimal capacity of the exit tree and the condition of activation of the exit tree were selected through optimization method. Theoretical analysis and simulations show that the asymptotic average join time and leave time are reduced to $O(1)$.

Key words: multicast; group key; secure group communication; logical key hierarchy

0　引言

许多面向组播的应用,例如,按次付费电视、秘密聊天和远程会议等,要求提供通信内容的保护。由于数据的私密性和完整性通常和组密钥紧密地联系在一起,建立和管理组密钥就成为组安全机制中最基础的内容。同时由于组播组中成员是动态变化的,需要实时更新组密钥来防止新加入的成员获得过去的通信内容,已经离开的成员获得以后的通信内容。这就需要组成员共同协商一个密钥建立和更新的密钥管理算法。

贡献型组密钥管理算法的核心是每个成员都对密钥的管理和产生贡献自己的份额,减轻了在集中式和分布式组密钥管理算法中的单点失效问题和信任问题。它使用一个包含了所有成员贡献的函数来生成组密钥,可以提供密钥的独立性和完美向前保密。但是,与集中式和分布式密钥算法相比,贡献型密钥管理需要相对较重的模指数运算和额外的组成员之间的通信开销。

文献[1]中提出了基于树形的贡献型组密钥管理方案(Tree-based Group Diffie-Hellman, TGDH)。TGDH 按照逻辑密钥树组织密钥,每一个成员仅需要知道它所在路径上的密钥。这就表明当密钥更新时,工作量被分配给了每个成员。文献[2]中提出了基于加入离开树的组密钥管理方案(Join-Exit-Tree key management, JET),通过采用加入树和离开树将成员加入和离开时间开销的近似上界从 $O(\log n)$ 降为 $O(\log(\log n))$。JET 使用多个加入事件或离开事件的平均时间开销来衡量密钥更新的复杂度。此外,JET 中离开树的引入也是基于一种前提,即成员在组中的停留时间已知。

文献[3]中,假定成员的动态信息不可获得,提出了改进的基于加入树的组密钥管理方案 JDH(join-tree-based contributory group key management),并把成员加入时间开销的近似渐进上界降低为 $O(1)$。

本文在 JDH 的基础上,假定成员的停留信息可知,提出了基于权重加入离开树的组密钥管理方案 WJT(Weighted-Join-exit-Tree)。WJT 由三部分组成:主树、加入树和离开树。加入树位于密钥树的根部,作为成员加入时的一个临时缓存区;离开树是一个根据成员的离开概率组成的权重树。

1　WJT 方案

1.1　树拓扑结构

WJT 的密钥树结构(见图 1)以树型和 Diffie-Hellman(DH)[4]算法为基础,与 JET 的结构相似,包括三部分:加入树(Join tree)、离开树(Exit tree)和主树(Main tree)。加入树和离开树的引入是在保证安全组通信的基础上,借鉴了批量移动的思想,使用多个加入事件或离开事件的平均开销来降低密钥更新的复杂度。以前的研究[1,5]表明,如果成员从离树根近的地方加入组或离开组将会减少组密钥更新的开销。基于此,本文提出了需要最少加入和离开时间的加入算法和离开算法。加入算法与 JDH 中相同,将新成员加入到加入树

收稿日期:2009 - 07 - 16;**修回日期**:2009 - 08 - 31。
作者简介:曹震寰(1976 -),男,甘肃庄浪人,工程师,主要研究方向:网络安全;　车彦刚(1979 -),男,陕西渭南人,助理工程师,主要研究方向:网络安全。

的根部。当加入树达到一定高度时,将加入树中的成员合并到主树中,合并的复杂度与加入树中的成员个数有关,因此一个不平衡的加入树不会带来额外的开销。离开算法的提出是基于与 JET 同样的前提,即成员在组中的停留信息可知。因此离开树是一个根据成员的离开概率组织的权重树,它的高度也与它包含的成员个数相等。当批量移动条件成立时,将即将要离开的成员从主树批量移动到离开树。因为批量移动的开销也与离开树中的成员数目相同,而与树的高度无关,因此离开树也不需要维持为一个平衡树。

新成员加入过程与 JDH 相似,包括两部分,新成员加入加入树的过程和从加入树移动到主树的过程,这里不再赘述。不同的一点是在 WJT 中引入了离开队列,当成员从加入树移动到主树时,将可获得的成员的离开概率加入离开队列。本文重点描述离开算法。

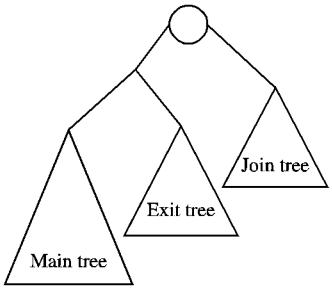

图 1　WJT 的密钥树结构

1.2　离开算法描述

1)离开树结构。

离开树是一个根据成员的离开概率组织的权重树。离开树中成员的位置是根据它们的离开概率确定的。离开概率越大,成员所在的位置越接近于离开树的根部。图 2 是一个由 4 个成员组成的离开树的图例,每个成员有一个离开概率 W_i,W_i 的值满足:

$$W_4 > W_3 > W_2 > W_1 \tag{1}$$

u_4 的离开概率 W_4 最大,因此它位于距离离开树树根最近的位置。当 u_4 离开时,仅需要更新 u_4 知道的密钥 $K_{<0,0>}$,$K_{<1,0>}$ 和 $K_{<2,1>}$,需要 3 次连续指数运算(其中辅助者需要进行 6 次模指数运算,其他成员最多需要进行 3 次盲钥计算)。从图 2 可以看出,离开树也是一个倾斜树。由于成员离开的开销与离开树的结构无关,而且成员批量移动时的开销依赖于移动成员的数目,而不是树的组织结构,因此一个倾斜的离开树的离开性能并不比一个平衡的离开树差。

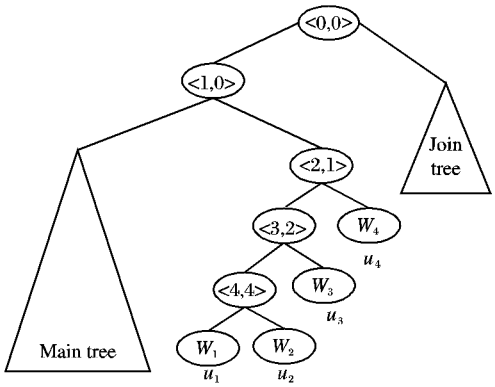

图 2　WJT 的离开树结构

2)成员批量移动。

当批量移动条件满足时,将离开队列中离开概率大的成员从主树移动到离开树,并将移动成员的离开概率从离开队列中删除。假设有 x 个成员被移动到离开树中,移位过程如下:

①首先,在离开树中根据算法 1 在离开树中找到 x 个叶子节点;

②根据离开队列中的信息,将离开概率大的 x 个成员同时移动到离开树中 x 个叶子节点的位置,从离开队列中删除移动成员的信息,标记需要更新的密钥;

③自下而上的更新所有标记的密钥。

在批量移动过程中,组通信并没有中断,可以使用旧的组密钥进行通信。

算法 1　在离开树中寻找插入点。

W_j ←leave probability of New-user
x ←exit-tree-root
While x. *rightchild*. $W_i > W_j$
　$x \leftarrow x$. *leftchild*
End while
insertion-node← x
insertion-node. rightchild←New-user
insertion-node. leftchild← x

3)成员离开。

在正确的成员离开信息下,离开概率大的成员被移动到了离开树中,因此成员大部分会从离开树中离开[2]。当成员离开时,密钥的更新步骤如下:

①每个成员单独更新自己的密钥树,删除离开成员的节点,由离开成员的兄弟节点代替离开成员的父节点,标记需要更新的密钥;

②当成员从主树中离开并且加入树中有成员时,或者成员加入树离开时,移动加入树中的成员到主树,标记需要更新的密钥;

③当成员从离开树离开并且满足批量移动条件时,移动主树中即将要离开的成员到离开树,标记需要更新的密钥;

④自下而上地更新所有标记的密钥;

⑤计算新的加入树和离开树的容量,更新密钥树。当主树的成员数目小于加入树和离开树激活的门限值并且加入树和离开树为空时,取消加入树和离开树。

2　性能分析

2.1　指标说明

1)连续指数运算的次数。

在基于 Diffie-Hellman(DH)算法的贡献型组密钥算法中,一个成员的加入或离开事件包含一些连续且昂贵的模指数运算。DH 算法通常要求在计算得到一个指数运算后才能进行下一个运算。因此连续指数运算的概念是指需要串行进行的一系列指数运算。指数运算的次数依赖于加入或者离开点在密钥树中的位置。在本文中,这个指标均指最坏情况下的指数运算次数,也即我们假定成员总是从树的底部加入或离开。

2)时间效率。

在文献[6,7]中,发送和接收的消息条数,以及耗费的带宽数量都是衡量一个密钥管理方案效率的重要指标。同样,密钥更新过程中轮数的数目[2]是衡量密钥更新效率的另外一个指标。通常来说,基于 DH 算法的模指数运算需要比基于对称加密运算多几个数量级的运算开销,在整个计算开销中占了很大比重,因此本文使用连续指数运算次数来衡量组密钥建立和更新的效率。在连续的加入或者离开事件中,使用平均连续指数运算次数来衡量。

①平均加入时间:定义成员的加入时间为在成员加入过程中所需的连续指数运算次数。用 T_{join} 来表示平均加入时间,定义如下:

$$T_{join} = E_{join}/N_{join} \tag{1}$$

其中 E_{join} 是对于 N_{join} 个加入事件所需的总的连续指数运算次数。

②平均离开时间:E_{leave} 表示对于 N_{leave} 个离开事件所需的总的连续指数运算次数。平均离开时间定义如下:

$$T_{leave} = E_{leave}/N_{leave} \tag{2}$$

2.2 批量移动条件

当成员从离开树离开并且满足批量移动条件时,执行批量移动。用 U_p 表示最后一次批量移动后离开树中的成员数目,U_c 表示当前离开树中的成员数目。当 U_p 和 U_c 满足式(3)时:

$$U_c \leqslant \rho U_p \tag{3}$$

认为批量移动条件满足,其中 $\rho \in [0,1)$ 是离开树剩余率。在批量移动前,总共有 $(1-\rho)U_p$ 个成员从离开树中离开,因此应该重新移动主树中 $(1-\rho)U_p$ 个成员到离开树中。批量移动的个数 b 可以由离开树的容量表示为:

$$b = (1-\rho)C_E \tag{4}$$

其中 C_E 是离开树容量。

2.3 离开树容量

假设 b 个成员被批量移动到离开树中,离开树的容量为 C_E。批量移动带来的开销为 $\max(3h-3,3h'-3)$,其中 $h = \lceil \mathrm{lb}\, N_M \rceil + 3$ 是主树的高度 $\lceil \mathrm{lb}\, N_M \rceil + 1$ 加上额外的两层;$h' = C_E + 2$ 是离开树的高度 C_E 加上离开树之上的两层(假设主树是平衡的)。如前文提到的一样,成员从主树离开带来的开销为 9。因此,根据式(2),一个成员离开安全组所需的平均密钥更新开销为:

$$T_{leave} \leqslant \frac{1}{b}\max(3\lceil \mathrm{lb}\, N_M \rceil + 6, 3C_E + 3) + 9 \tag{5}$$

其中 $\frac{1}{b}\max(3\lceil \mathrm{lb}\, N_M \rceil + 6, 3C_E + 3)$ 是从主树移动到离开树的平均开销,9 是成员从离开树离开所需的密钥更新开销。

为了最小化 $\max(3\lceil \mathrm{lb}\, N_M \rceil + 6, 3C_E + 3)$,$C_E$ 应该满足:

$$C_E \leqslant \lceil \mathrm{lb}\, N_M \rceil + 1 \tag{6}$$

结合式(4)和式(5),得到:

$$T_{leave} \leqslant \frac{1}{(1-\rho)C_E}(3\lceil \mathrm{lb}\, N_M \rceil + 6) + 9 \tag{7}$$

从式(7)可以看出,当 $x \to \infty$,T_{leave} 可以达到最小值 9。但是受到式(6)的限制,最优的离开树的容量为:

$$C_E = \lceil \mathrm{lb}\, N_M \rceil + 1 \tag{8}$$

因此,平均离开开销的上限为:

$$T_{leave} \leqslant \frac{4}{1-\rho} + 9 \tag{9}$$

这表明离开树的容量,同样也是离开树的高度,应该等于主树的高度。当有了正确的成员停留信息后,平均离开时间的耗费为 $O(1)$。

2.4 离开树激活

离开树激活后,采用了离开树的平均离开耗费应该低于没有采用离开树的耗费。如果没有采用离开树,则最大平均离开时间为 $3h-3$,其中 $h = \lceil \mathrm{lb}\,(N_M + C_E) \rceil + 1$,表示密钥树的高度。由于 C_E 远小于 N_M,h 可表示为 $h = \lceil \mathrm{lb}\, N_M \rceil + 2$。当式(10)成立时,采用离开树可以减小离开开销:

$$\frac{1}{(1-\rho)C_E}(3\lceil \mathrm{lb}\, N_M \rceil + 6) + 9 \leqslant 3\lceil \mathrm{lb}\, N_M \rceil + 3 \tag{10}$$

或

$$\lceil \mathrm{lb}\, N_M \rceil \geqslant \frac{2(1-\rho)C_E + 2}{(1-\rho)C_E - 1} \tag{11}$$

其中式(10)的左边是采用离开树的平均离开开销,右边是仅有主树的离开开销。主树的成员个数为 N_M。

根据 ρ 值的不同,式(11)满足的条件也不同。表 1 列出了在不同 ρ 值下离开树激活的门限值;当 $\rho \leqslant 0.4$ 时,离开树应该在 $N_M > 8$ 时被激活;当 $0.4 < \rho \leqslant 0.6$,离开树应该在 $N_M > 16$ 时被激活;当 $\rho = 0.7$ 时,离开树应该在 $N_M > 32$ 时被激活;当 $\rho = 0.8$ 时,离开树应该在 $N_M > 128$ 时被激活;当 $\rho = 0.9$ 时,离开树应该在 $N_M > 4096$ 时被激活。

表 1 不同 ρ 值下的 N_M 值

ρ	满足式(11)的 N_M 值	ρ	满足式(11)的 N_M 值
0	>8	0.5	>16
0.1	>8	0.6	>16
0.2	>8	0.7	>32
0.3	>8	0.8	>128
0.4	>16	0.9	>4096

值得注意的是,ρ 的选择也是一种折中。如果 ρ 接近于 0,那么只有离开树几乎为空时批量移动的条件才能满足;当 ρ 设置的接近于 1,批量移动的次数将会太频繁,因此也会带来较大的开销。根据文献[2]的建议,设置 $\rho = 0.5$。

3 仿真验证

3.1 成员连续加入时组密钥的建立

在这个仿真中,假设成员连续加入安全组。图 3 是三个方案 WJT、JET 和 TGDH 方案在成员连续加入时平均加入时间的比较。

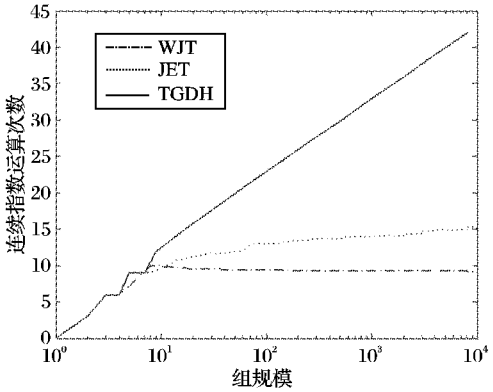

图 3 成员连续加入时所需平均加入时间

从图 3 中可以看出,当组规模小于 8 时,JET 和 TGDH 获得同样的性能;但是随着组的继续增大,JET 的性能要优于 TGDH。而对于 WJT,当组规模小于 4 时,它与 TGDH 和 JET 获得同样的性能,当组规模大于 4 时,由于加入树的激活,它的性能开始优于 TGDH 和 JET。图 3 与文献[2]中图 6 的区别在于图 3 使用连续指数运算次数来衡量平均加入时间,而在文献[2]中的图 6 使用简单轮数来衡量。如果不考虑数量的差异,可以看出,两个图中的趋势是相同的。另一个值得注意的现象是,WJT 的性能并不随着组规模的增大而下降。这验证了 WJT 的平均加入时间的渐进上界为 $O(1)$。

WJT 和 JET 在成员加入时的平均时间优于 TGDH 的原因

是加入树的引入。当成员加入时,加入树充当类似于缓存区的作用。由于加入树较小,成员加入加入树显然需要较小的开销,而批量更新开销平均到每一个成员来说显然是很小的。在 WJT 和 JET 方案中,因为总的移动开销被分摊到每个成员,因此每个成员的平均批量移动开销是恒定的。而 WJT 通过将成员加入加入树根部,进一步缩短了成员加入加入树的时间,而在 JET 中,成员总是被插入加入树中深度最浅的最右边节点,因此所需的加入开销大于 WJT 方案。

3.2　有成员正确离开信息时的组密钥更新

在本节,假设成员连续离开安全组,并可以获得成员在组中的离开概率。当主树的规模大于 16 时,离开树被激活,而在 JET 中,离开树在主树大于 256 时被激活。图 4 比较了三种方案 WJT、JET 和 TGDH 在成员连续离开时所需的平均离开时间。

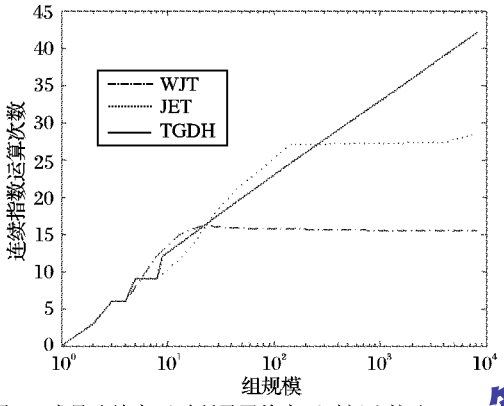

图 4　成员连续离开时所需平均离开时间比较 ($\rho = 0.5$)

从图 4 中可以看出,WJT 与 TGDH 和 JET 在组小于 4 时获得同样的性能。当组规模大于 4 时,WJT 中的加入树被激活时,WJT 的性能先优于 TGDH 和 JET,然后低于 TGDH 和 JET,最后在组规模大于 16 时优于它们。这主要是由于在离开树激活前,加入树的激活首先降低了密钥树的高度,然后由于相对较小的加入树容量增加了密钥树的高度。对于 JET,情况也是相同的。同时也注意到 WJT 的平均离开时间并没有随着组规模的增大而增加,而其他两个方案都随着组规模的增大而逐渐增大。这就证实了在成员动态信息可知时,通过离开树的引入,WJT 在成员离开时密钥更新的渐近性能上界为 $O(1)$。

引入离开树作为一个临时缓存区是 WJT 和 JET 相比 TGDH 方案获得较好性能的一个原因。在 WJT 和 JET 方案中,成员离开的开销包括两部分:从主树移动到离开树的开销和从离开树离开的开销。批量移动的开销在两种方案中都是恒定的,因为批量移动开销被分配给了每个移动的成员。但是通过使用权重离开树,WJT将成员从离开树离开的开销降低为常数,而在 JET 方案中,成员离开的开销为 $O(\log(\log n))$。

4　结语

本节在 JDH 方案的基础上,引入了基于权重离开树的贡献型组密钥管理方案 WJT 来获得较好的成员离开时的密钥更新性能。WJT 利用加入树和离开树充当新加入成员和即将要离开成员的暂时缓存区,主要包括加入算法和离开算法。加入算法与 JDH 方案中的加入算法相同,将新成员加入到加入树根部,当成员从加入树移动到主树中时,将可获得的新成员的离开概率加入离开队列。离开算法是假设成员在组中的离开概率可以预先获得。当批量移动条件满足时,将那些即将要离开的成员从主树移动到离开树。由于批量移动的开销依赖于被移动成员的数目,而不是离开树的结构。因此将离开树根据成员的离开概率组织为一个权重树,它的高度与所包含的成员个数相等。改进的算法可以将成员加入和成员离开时的平均密钥更新时间的上界降低为 $O(1)$。仿真和实验说明 WJT 在密钥建立和成员加入时,优于现有方案,在成员动态信息可知时的离开开销也要优于其他方案。

值得注意的是,WJT 的性能在成员的离开概率不确定时会下降。因此 WJT 的应用应该限于那些可以预先获得成员停留时间的组播应用,例如按次付费,远程会议等。

参考文献:

[1] KIM Y, PERRIG A, TSUDIK G. Tree-based group key agreement [J]. ACM Transactions on Information and System Security, 2004, 7(1): 60 – 96.

[2] MAO YILLIAN, SUN YAN, WU MIN, *et al*. JET: Dynamic join-exit-tree amortization and scheduling for contributory key management[J]. IEEE/ACM Transactions on Networking, 2006, 14(5): 1128 – 1140.

[3] GU XIAOZHUO, YANG JIANZU, YU JING, *et al*. Join-tree-based contributory group key management[C]// 2008 10th IEEE International Conference on High Performance Computing and Communications. New York: IEEE, 2008: 564 – 571.

[4] DIFFIE W, HELLMAN M. New directions in cryptography [J]. IEEE Transactions on Information Theory, 1976, 22(6): 644 – 654.

[5] DONDETI L R, MUKHERJEE S. DISEC: A distributed framework for scalable secure many-to-many communication[C]// Proceedings of 5th IEEE Symposium on Computer and Communications Security. Washington, DC: IEEE Computer Society, 2000: 693 – 698.

[6] STEINER M, TSUDIK G, WAIDNER M. Diffie-Hellman key distribution extended to group communication[C]// Proceedings of 3rd ACM Conference Computer and Communications Security. New York: ACM, 1996: 31 – 37.

[7] BECKER K, WILLE U. Communication complexity of group key distribution[C]// Proceedings of 5th ACM Conference on Computer and Communications Security. New York: ACM, 1998: 1 – 6.

(上接第 209 页)

参考文献:

[1] 康治平, 向宏, 傅鹏. 基于 APIHOOK 技术的特洛伊木马攻防研究[J]. 信息安全与通信保密, 2007(2): 145 – 148.

[2] 李焕洲, 唐彰国, 钟明全, 等. 基于行为监控的木马检测系统研究及实现 [J]. 四川师范大学学报, 2009, 32(3): 386 – 389.

[3] FISKIRAN A M, LEE R B. Runtime execution monitoring (REM) to detect and prevent malicious code execution[C]// Proceedings of the IEEE International Conference on Computer Design. Washington, DC: IEEE Computer Society, 2004: 452 – 457.

[4] 何志, 范明钰, 罗彬杰. 基于远程线程注入的进程隐藏技术研究[J]. 计算机应用, 2008, 28(6): 92 – 94.

[5] 何志, 范明钰. 基于 HSC 的进程隐藏检测技术[J]. 计算机应用, 2008, 28(7): 1772 – 1775.

[6] 陈京浩. 进程检测模型及相关技术的研究与实现[D]. 重庆: 重庆大学, 2007.

[7] 米渊. 使用差异比较法对木马程序进行防杀的技术探讨[D]. 呼和浩特: 内蒙古大学, 2008.