组播密钥管理的研究进展*

徐明伟+, 董晓虎, 徐 恪

(清华大学 计算机科学与技术系,北京 100084)

A Survey of Research on Key Management for Multicast

XU Ming-Wei+, DONG Xiao-Hu, XU Ke

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+Corresponding author: Phn: +86-10-62785822, E-mail: xmw@csnet1.cs.tsinghua.edu.cn, http://netlab.cs.tsinghua.edu.cn

Received 2002-12-23; Accepted 2003-05-27

Xu MW, Dong XH, Xu K. A survey of research on key management for multicast. *Journal of Software*, 2004,15(1):141~150.

http://www.jos.org.cn/1000-9825/15/141.htm

Abstract: The absence of security mechanism has limited the use of multicast. Key management for multicast is used for group members in one multicast session to generate, refresh and transfer keys which are used for encryption and authentication. In addition to the maintenance of keys, issues about scalability, reliability and robustness should be carefully considered. In this paper, the existing problems in key management for multicast are analyzed, and some typical schemes are reviewed.

Key words: IP multicast; key management; logical key tree

摘 要: 缺乏安全机制限制了组播在各种网络业务中的应用.组播密钥管理通过为组播成员生成、发送和更新组密钥来满足加密认证等安全需求.组播密钥管理方案的设计,除了要满足基本的安全需求,还要兼顾可扩展性、健壮性、可靠性等多方面因素.分析了组播密钥管理所面临的问题,并通过对已有的几种组播密钥树方案的介绍,探讨和总结了组播密钥管理的研究现状和发展趋势.

关键词: 组播;密钥管理;逻辑密钥树

中图法分类号: TP309 **文献标识码**: A

组播(multicast)在虚拟会议、网络辅助协同工作、多媒体实时点播、网络游戏等方面有着很广阔的应用前景.这些组播应用对组播的安全性能提出了要求.在虚拟会议中,通常需要确保会议内容的保密性,并且在必要时能够对发言人的身份进行认证;在多媒体实时点播中,要确保只有付费用户才能看到节目内容.但是目前的组

* Supported by the National Natural Science Foundation of China under Grant Nos.90104002, 60203025, 60373010 (国家自然科学基金); the High-Tech Research and Development Plan of China under Grant No.2001AA112132 (国家高技术研究发展计划(863)); the Key Project of Chinese Ministry of Education of China under Grant No.02004 (国家教育部科学技术重点项目)

作者简介: **徐明伟**(1971－),男,辽宁朝阳人,博士,副教授,主要研究领域为计算机网络体系结构,高速路由器体系结构,协议测试;**董晓虎**(1979－),男,硕士,主要研究领域为组播密钥管理,密钥交换协议;**徐恪**(1974－),男,博士,助研,主要研究领域为计算机网络体系结构,计算机系统性能评价.

播协议缺乏安全机制来满足上述要求,采用明文传输的组播报文在网络上很容易被偷听、冒充和篡改.

我们把组播的安全需求归纳为如下几点:

- 保密.只有拥有解密密钥的节点才能解读组播报文的内容.
- 组成员认证.非组成员无法生成有效的认证信息,进而无法冒充组成员发送组播报文.
- 源认证(抗抵赖).组成员无法生成其他组成员的认证信息,进而无法冒充其他组成员发送组播报文.另一方面,组成员也无法否认其发送的信息.
- 匿名性.为组成员提供匿名发言的机制,也就是说,接收方无法从接收到的组播报文推断出发送方的身份.
- 完整性.提供验证收到的组播报文是否被篡改的手段.

对组播报文加密传输是实现组播保密性的一种方法.加密和解密用的密钥只有组成员才知道,这样能够确保被加密的报文只有组成员才能解读.组成员认证也可以利用该密钥来实现,因为只有拥有密钥的组成员才能正确地生成加密的组播报文.利用多方共享密钥来解决安全问题的关键是密钥的生成和分发.这种生成和分发必须是排外的,即非组成员无法获得密钥.

源认证、完整性和匿名服务通常也要利用双方或多方之间信息的排外共享.

在多方通信中,如何实现信息的排外共享是组播密钥管理的研究范畴.本文将要探讨的重点是组播密钥管理如何为组成员生成、发布和更新组密钥(组密钥是所有组员共享的密钥,用来对组播报文进行加密和解密等安全操作),以及由此产生的扩展性、健壮性和可靠性问题.

下面的章节将对组播密钥管理做深入的探讨.第 1 节从总体上介绍组播密钥管理,分析组播密钥管理所要解决的问题和设计上的难点,讨论 3 种拓扑结构的特点.第 2 节对组播密钥树管理方案进行深入的研究,从目前提出的组播密钥管理方案中选择有代表性的几种进行介绍.这几种方案覆盖了组播密钥管理的 3 种拓扑结构和实现组播密钥管理的各类算法.更重要的是,我们试图用发展的眼光,通过这些方案从纵向体现出组播密钥管理算法的发展思路和发展方向.第 3 节总结全文,并指出了进一步研究的方向.

1 组播密钥管理介绍

1.1 组播密钥管理的定义及其面临的问题

组播密钥管理为参与组播的成员生成、分发和更新组密钥(group key).组密钥是所有组成员都知道的密钥,被用来对组播报文进行加密/解密、认证等操作,以满足保密、组成员认证、完整性等需求.

相比单播的密钥管理,前向加密(forward confidentiality)、后向加密(backward confidentiality)和同谋破解是组播密钥管理特有的问题.

前向加密要求主动退出组播的节点或被强制退出的节点(比如恶意节点)无法继续参与组播,即无法利用它们所知道的密钥解密后继组播报文和生成有效的加密报文.重新生成并更新组密钥可以实现前向加密,但要注意的是,密钥更新报文同样可以被前组成员获得,要防止前组成员从密钥更新报文中得到新的密钥.

后向加密要求新加入的组成员无法破解它加入前的组播报文.当新成员加入时更新密钥就可以实现这一点.

组播密钥管理不仅要防止某个节点破解系统,还要防止某几个节点联合起来破解.如果几个恶意节点联合起来,掌握了足够多的密钥信息,使得无论系统如何更新密钥都可以获得更新的密钥,导致组播密钥管理的前向加密和后向加密失败,或者使得恶意节点可以冒充其他节点进行欺骗(破解系统的认证功能),我们把这种情况称为同谋破解.组播密钥管理要杜绝同谋破解或降低同谋破解的概率.

除此之外,组播密钥管理在设计时还要考虑如下因素的影响:

差异性:组播密钥管理涉及到多个通信实体.这些通信实体之间存在着各种差异.这些差异包括是否可信任、是否愿意为其他实体提供服务、是否具有足够的计算能力、是否具有足够的带宽和适当的网络延迟、是否接受组播报文(允许存在只发送不接收的实体)、是否发送组播报文(允许存在只接收不发送的实体)等等.组播密钥管理方案在设计时要考虑这些差异的影响.

可扩展性:可扩展性也是组播密钥管理所要考虑的重点.组播的规模从几个节点到上万个节点甚至更多,随着组播规模的扩大,保存密钥所占用的节点存储空间、密钥生成所需要的计算量、密钥发送所占用的网络带宽、密钥更新的时间延迟和密钥更新的频率都会相应增加.

健壮性:对于单播来说,通信的任一方失败都会使会话终止,而组播中部分节点的失败不应当影响整个组播会话的继续进行.这就对组播密钥管理提出了健壮性的要求.

可靠性:可靠性也是一个确保组播密钥管理正确而有效工作的重要因素.组播密钥管理的控制报文(包括密钥更新报文、组成员关系变动的通知报文等)通常利用不可靠的组播进行传输.这种传输存在丢包、乱序、重复等情况.设想如果缺乏确保可靠性的机制,一个组成员没有收到密钥更新报文,它将无法参与后继的组播通信.

因此,设计一个组播密钥管理方案,需要统筹考虑通信实体间的差异、系统的可扩展性、健壮性和可靠性等诸多因素.与组播的安全需求综合起来,我们把组播密钥管理所要解决的基本问题归纳如下:

(1) 前向加密:确保组成员在退出组后,除非重新加入,否则无法再参与组播,包括获知组播报文的内容和发送加密报文.

(2) 后向加密:确保新加入的组成员无法破解它加入前的组播报文.

(3) 同谋破解:避免多个组员联合起来破解系统(或减少发生的概率).

(4) 密钥生成计算量:通常,协同的密钥生成需要较大的计算量,当节点的计算资源不充足或密钥更新频繁时,要考虑密钥生成给节点带来的负载.

(5) 密钥发布占用带宽:密钥更新报文不应占用过多的网络带宽.

(6) 密钥发布的延迟:密钥更新时要使所有组成员都能及时地获得新的密钥.问题(4)~问题(6)同属可扩展性问题.

(7) 健壮性:当部分组成员失效时,安全组播仍然能够继续工作.

(8) 可靠性:确保密钥分发/更新在不可靠的网络环境中的正确实行.

针对不同的应用,上述问题的必要性和重要性都会有所不同.

对于与组播密钥管理有密切关系的其他问题,比如访问控制、计费等,本文不作进一步的讨论.

需要指出的是:问题(1)和问题(2)的需要与否,直接影响到系统实现的复杂度.如果系统不要求实现前向和后向加密,则问题(3)~问题(7)就变得相对简单,可以用很直接的方式实现(如下文介绍的 GKMP).

1.2 组播密钥管理系统的拓扑结构

根据拓扑结构的不同,可以把组播密钥管理方案分为 3 大类:集中控制式、分布式和分层分组式.上一节提出的 8 个问题,在不同的拓扑中,情况也各有不同.

在集中控制式的组播密钥管理中,存在一个节点负责全组的密钥生成、分发和更新.这个节点通常被称为根(root)或组控制器(group controller,简称 GC).使用集中控制,有利于组播的管理,可以方便地施加身份认证等措施,而很多组播应用在本质上存在着集中控制,适合采用集中控制式的组播密钥管理.但是这类方案对根的依赖性导致了单一失效点问题;root 节点也可能会因为负载过大而成为性能的瓶颈,影响系统的可扩展性;对特定节点的依赖也使得集中控制方式难以应用于 P2P(peer to peer)的模式.

在分布式的组播密钥管理中,参与通信的节点是对等的,通过某种密钥协商算法生成组密钥.这类方案不存在集中控制中单一失效点的问题,并且很适合 peer to peer 的应用模式.但是缺少集中控制给管理带来了困难.

分层分组式的管理方案将参与组播的成员进行分组.每个小组(subgroup)存在一个控制节点.这些控制节点组成了组播密钥管理的层次 I.小组内部的密钥管理属于层次 II.这两个层次可以独立地选择采用集中控制的管理方式或是分布式的管理方式.在每个层次上采用何种方式都会继承这些方式的优缺点.在层次 II 上,通常小组内部成员个数较少,可以采用集中控制式.如果小组规模仍然较大,可以对其进一步分组,生成新的层次.

在这 3 类方案中,前两类是基本形式,分层分组式通过糅合这两种形式,使得在某些方面的性能有所改进,更能适应某种特殊应用,但并没有从根本上解决集中式或分布式所存在的问题.

为了消除或减轻集中控制和分布式控制固有的缺陷,存在一种折衷的思路:对于集中控制来说,通过降低root 节点所承担的责任,将 root 节点的功能分布化,或增加备用节点,以减轻 root 节点的负载,降低单一失效点的风险;对于分布式控制来说,利用目录服务来集中发布信息,缓解因缺乏集中控制而形成的管理难度.

一个实际的系统采用何种拓扑结构取决于系统本身的特性.分布式的结构通常应用于 P2P;具有集中管理特性的应用可以采用集中控制的结构.分层分组的结构适用于具有层次性结构的系统.

2 组播密钥管理的发展

2.1 基本方案

本节介绍 3 种密钥管理方案,这些方案分别属于 3 种拓扑结构.我们将会看到,这 3 种方案初步满足了管理组播密钥的要求,但是都存在着相当大的不足.

GKMP[1,2].

Group Key Management Protocol(GKMP)是一种简单的采用集中控制的组播密钥管理方案,在这个方案中,每个节点都与 root 预先共享一个密钥,用来确保和 root 通信时的安全.root 节点在第 1 个加入成员的帮助下生成组密钥报文(group key packet,简称 GKP).GKP 中包含有用来加密组播报文的密钥 GTEK(group traffic encryption key)和用来加密组密钥的密钥 GKEK(group key encryption key).当新的成员加入的时候,root 用它与该组员的共享密钥加密 GKP 并将加密后的报文发送给该组员.如果需要更新密钥,root 生成新的 GKP,然后用当前 GKEK 加密后通过组播发送给所有组员.由于这样一种密钥更新方式,离开的组员仍然能够解密 GKP,GKMP 实现前向加密的惟一途径就是重新创建一个组.

在不需要前向加密的情况下,GKMP 是个很好的选择.

Clique[3].

Clique 是一种分布式的组播密钥管理算法.它利用 diffie-hellman(DH)[4]密钥协商算法的一种变体来实现组密钥的生成和发布.DH 算法通常被用来在通信双方之间协商密钥.假设节点 A 和 B 要生成共享密钥 k,DH 算法的密钥协商过程如下:

(1) A 和 B 事先选定两个数 q 和 a,q 是一个很大的素数,而 a 具有这样的性质:对从 1 到 $a-1$ 的所有整数 n,存在 k 使得 $n=a^k \bmod q$.q 和 a 不需要保密.

(2) A 选择一个大的随机数 x,计算 $X=a^x \bmod q$.

(3) B 选择一个大的随机数 y,计算 $Y=a^y \bmod q$.

(4) A,B 交换 X,Y.

(5) A 计算 $k=Y^x \bmod q$.

(6) B 计算 $k=X^y \bmod q$.

DH 密钥交换算法的安全性来自于:对于第三方 C 来说,即使它获得 q,a,X 和 Y,推算出 k 的计算量也是非常大的.

在 Clique 中,拥有 n 个组成员的组通过如下步骤协商组密钥 k:所有组员事先选定 q 和 a,然后各自选定一个随机数 xi,并计算幂值 a^{xi}.第 1 个组员将集合 $S_1=\{a^{x1}\}$ 传递给第 2 个组员.第 2 个组员生成新的 $S_2=\{a^{x1},a^{x2},a^{x1x2}\}$,传递给第 3 个组员.$S_k$ 中含有从 a^{x1} 到 a^{xk} 的累乘和从 a^{x1} 到 a^{xk} 中任选 $k-1$ 个幂值的累乘.第 n 个(最后一个成员)收到 S_{n-1} 并计算 S_n,然后将 S_n 用组播发送给所有其他节点.这样,所有节点都可以计算 $k=a^{x1x2\ldots xn} \bmod q$.

Clique 的密钥传输时间延迟的复杂度为 $O(n)$(集合 S_i 的生成必须串行进行),密钥计算的总计算量为 $O(n^2)$,密钥传输所占用的带宽为 $O(n^2)$.因此,Clique 的扩展性很差.

Iolus[5].

Iolus 对组成员进行分组,每个子组有一个组安全代理(group security Agent,简称 GSA),负责管理该子组,所有 GSA 组成一个更高一级的组,由组安全控制器(group security controller)管理.Iolus 的特点是各个子组采用独立的组密钥,组播报文在从子组 A 传播到子组 B 时要被 GSA 翻译,即用 A 的组密钥解密,再用 B 的组密钥加密.

这种设计使得组成员关系变化所导致的密钥更新被限制在所在子组内部,但另一方面,组播报文的传输路径被改变了(穿越子组边界的组播报文必须经过 GSA),而且 GSA 要负责子组的管理和组播报文的翻译,容易成为系统的瓶颈和失效点.

2.2 用于集中控制方式的逻辑密钥树

采用逻辑密钥树的组播密钥管理方案在确保前向加密、后向加密和抗同谋破解的基础上比较好地解决了可扩展性问题.逻辑密钥树又被称为 Hierarchical Tree,首先被应用于集中控制方式,随后被扩展到分布式和分层分组式,并出现了各种改进办法.下面我们首先介绍用于集中控制式的逻辑密钥树.

文献[6]描述了应用于集中控制方式的逻辑密钥树.在逻辑密钥树中,组控制器维护一棵密钥树,树的每个节点都对应于一个密钥,树的叶子节点与组成员(组成员不包括组控制器)一一对应.组控制器知道所有的密钥,每个组成员所知道的密钥是位于从该组成员对应的叶子节点到根节点的路径上的所有节点对应的密钥.之所以称其为逻辑密钥树,是因为它只是组控制器维护的一个数据结构,它的非叶子节点不对应组成员.

图 1 是一个逻辑密钥树的示意图(为了论述方便,我们以一棵平衡二叉树为例来进行讨论,逻辑密钥树对此并不要求).组成员 $u3$ 对应的叶子节点是 $k3$,$u3$ 所知道的密钥是$\{k3,k34,k14,k\}$.所有的节点都知道根节点对应的密钥 k,因此,k 可以被用作组密钥.逻辑密钥树的层次结构还使得中间节点对应的密钥可以被用来为小组的通信加密.比如,$u1$,$u2$,$u3$ 和 $u4$ 可以利用 $k14$ 来进行组内更小范围的安全通信.

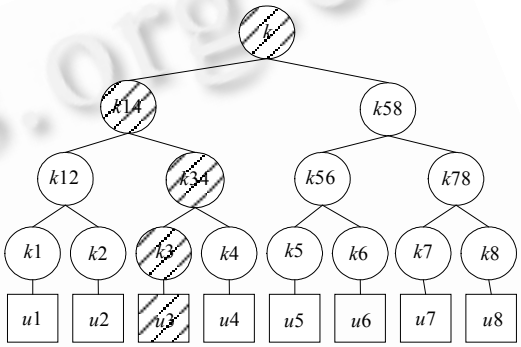

Fig.1 Logical key tree in centralized mode
图 1 集中控制式的逻辑密钥树

加入新组员的操作如下:节点 I 要求加入组播,组控制器通过某种机制对其进行身份认证,并生成共享密钥 ki(这一步可以利用单播的密钥交换协议);然后组控制器在密钥树上为其创建一个叶子节点;最后组控制器将组员 I 所应获知的密钥用 ki 加密传给组员 I.如果需要后向加密,则组控制器应该在发送密钥给组员 I 之前对密钥进行更新.

删除组成员的操作要复杂一些,为了确保前向加密,组控制器要更新所有被删除成员所知道的并被其他成员使用的密钥.为确保离开的组成员无法破解密钥更新报文,密钥更新从叶子节点往根节点向上一步一步进行,并且利用子节点的密钥对父节点的密钥更新报文进行加密.比如在图 1 中,组成员 $u3$ 离开组,则密钥 $k34$,$k14$ 和 k 都要更新.组控制器首先发送$\{k34'\}k4$ 给 $u4$ 以更新 $k34$(\{payload\}key 表示用 key 对 payload 进行加密);然后发送$\{k14'\}k34'$给 $u4$,发送$\{k14'\}k12$ 给 $u1$ 和 $u2$ 来更新 $k14$;最后发送$\{k'\}k14'$给 $u1$,$u2$,$u4$,发送$\{k'\}k58$ 给 $u5$,$u6$,$u7$,$u8$ 以更新 k.密钥更新报文可以使用组播进行分发,以提高更新效率,减少网络负载.

一棵中间节点子节点数目为 k、层数为 $d+1$ 的平衡逻辑密钥树,它的组成员个数为 k^d,每个组成员保存的密钥数量为 $d+1$,组控制器保存的密钥数量为$(k^{d+1}-1)/(k-1)$.删除一个组成员时的组密钥更新所需要的网络流量为 $kd-1$.

对于同样数量的组成员,如果增加 k,可以降低树的深度,减少组成员所需要保存密钥的数量.当 $k=N$ 时(N 为总用户数),层数降为 2,这时逻辑密钥树就退化为 GKMP.

2.3 对逻辑密钥树的几种改进

对逻辑密钥树的改进包括以下几个方面:降低组控制器需要保存的密钥数量;降低密钥更新延迟和网络带宽占用;提高对频繁的组成员关系变化的适应性.

降低根节点保存密钥的数量:对于一个中间节点子节点个数为 k、层数为 $d+1$ 的平衡逻辑密钥树,组控制器保存的密钥个数为$(k^{d+1}-1)/(k-1)$,这将导致组控制器需要很大的密钥存储空间来支持大规模的组播,并将增加密钥操作的运算量.文献[7]提出了一种方案,可以降低组控制器保存的密钥数.在这个方案中,每个组成员都惟一地对应于一个位数为 w 的 ID,组控制器保存一组 KEK(key encryption key)$\{(KEKi.0,KEKi.1)|l=0$ 到 $w-1\}$和组

密钥 TEK(traffic encryption key),对于 ID 为 X 的组员,它保存的密钥为{KEK$i.b$|如果 X 的第 I 位等于 0,b 取 0,否则取 1;I 从 0~w−1}和 TEK.如图 2 和图 3 所示,分别是在 w=4 时,组控制器和组成员(ID=0101)所保存的密钥.

	TEK	
ID Bit0	KEK0.0	KEK0.1
ID Bit1	KEK1.0	KEK1.1
ID Bit2	KEK2.0	KEK2.1
ID Bit3	KEK3.0	KEK3.1

Fig.2 Keys held by root

图 2 根节点保存的密钥

	TEK
ID Bit0=0	KEK0.0
ID Bit1=1	KEK1.1
ID Bit2=0	KEK2.0
ID Bit3=1	KEK3.1

Fig.3 Keys held by member 0101

图 3 组成员 0101 保存的密钥

当组成员离开时,组控制器要更新离开节点知道的所有密钥.密钥更新报文包括两个部分:用所有不需要更新的 KEK 加密的新的 TEK;用新的 TEK 加密的所有需要更新的 KEK.这样,由于离开成员所知道的 KEK 没有用来加密新的 TEK,则它无法得到新的 TEK,进而无法更新 KEK.而所有其他成员,因为 ID 与离开组员不同,必然能够解密用某个不需要更新的 KEK 加密的新的 TEK,进而解密所有更新的 KEK.图 4 给出了在上面例子中,当组成员 0101 离开组时,root 节点生成的密钥更新报文.

	TEK	
ID Bit0=0	(KEK0.0new) TEKnew	(TEKnew) KEK0.1
ID Bit1=1	(TEKnew) KEK1.0	(KEK1.1new) TEKnew
ID Bit2=0	(KEK0.0new) TEKnew	(TEKnew) KEK2.1
ID Bit3=1	(TEKnew) KEK3.0	(KEK3.1new) TEKnew

Fig.4 Message used to exclude member 0101

图 4 组成员 0101 离开时的密钥更新报文

组控制器保存的密钥数量为 W+1,组成员保存的密钥数量为 W,组成员的最大数目为 2^w.当 W=10 时,该方案可以支持最多 1 024 个用户的组播,组控制器仅需要保存 11 个密钥;如果使用二叉逻辑密钥树来支持同样多的用户,则组控制器需要保存的密钥数量是 2 047.

这种方案的缺陷在于,它虽然减少了根节点保存密钥的数量,却导致了同谋破解的可能.一个极端的例子是,如果两个恶意节点的 ID 各位互反,它们联合起来可以使系统的前向加密失效,密钥管理机制将无法从组内删除这两个节点.

降低密钥更新延迟和带宽占用:减少密钥更新所需要发送的信息量,可以降低密钥更新延迟和带宽占用.文献[8,9]分别给出的两种方案都能使二叉密钥树的密钥更新网络流量由 $O(2\log n)$降为 $O(\log n)$.这类方案的思路是:逻辑密钥树在密钥更新时所需发送给每个成员节点的信息非所要更新的密钥,而是一个统一的密钥更新方式所需参数,每个成员只需收到一次更新报文,得到相应的参数,就可以自行计算出它所要更新的所有密钥.

文献[8]的方案要求每个成员要保存其到根节点路径上的所有节点的密钥 KEY 和这些节点的兄弟节点的 BK(blind key).一个节点的 BK 由 KEY 通过一个单向函数 f 得来,即

$$BK=f(KEY).$$

每个非叶子节点的密钥 K 由它的两个子节点的 BK 通过一个混和函数 g 得来,即

$$K=g(BKl\text{-}child,BKr\text{-}child).$$

密钥更新时,假设在成员 I 所知的密钥中,K 是对应节点位置最低的需要更新的密钥.不妨设 I 位于 K 对应节点的左子树上,则 I 只需要得到 K 对应节点的右边子节点的 BK 就可以计算出 K 及 K 以上的所有需要更新的密钥.文献[8]的方案增加了成员的密钥存储量.

文献[9]的方案不需要增加成员的密钥存储量.仍以图 1 为例,所有组员事先商定好一个函数 $K'=f(K,R)$.假设节点 $u1$ 离开组,root 更新密钥的报文如下:$\{R\}k2$,$\{R\}k34$,$\{R\}k58$(R 由 root 节点随机生成).这样,除了 $u1$,每个节点都得到了 R,它们分别利用 f 更新自己的密钥.对于 $u2$ 而言,它的密钥变为 $K2'=f(K2,R)$,$K12=f(K12,R)$,$K14=f(K14,R)$,$K=f(K,R)$.

这些方案对于 K 叉树也是适用的,可以使网络流量由 kd−1 降为$(k-1)d$.

增强对组成员关系变化的适应性:组成员的加入和离开导致密钥更新.如果这种组员关系变化频繁发生,或在某个短的时间内大量发生,会对系统的性能造成很大的影响.这种情况很可能发生在网上会议开始和结束阶段,或视频点播开始和结束阶段.对于这种情况,改进的思路是:批量或定时处理成员的关系变化,不是在每次组成员关系变化时立即生成新密钥并更新,而是在一定的时间后或成员关系改变的数量达到一定值后生成并更新密钥.这种策略牺牲了部分的前向和后向加密,但增强了系统对组员关系变化的承受能力.文献[10]对此作了深入的分析.

2.4 用于分布式模式的逻辑密钥树

O. Rodeh在文献[11]中提出了将二叉逻辑密钥树应用于分布式模式的方案.该方案无须root节点,而是在组成员之间交换和更新密钥.下面我们在一棵已建立起来的分布二叉逻辑密钥树基础上讨论节点加入、删除时,系统密钥更新的操作.如图 5 所示是一棵有 8 个节点的分布二叉逻辑密钥树.树中每个叶子节点对应于一个组成员,每个组成员所知道的密钥仍是从其对应节点到根路径上每个节点对应的密钥.这里引入 leader 的概念:对每一棵以非叶子节点(包括最顶节点)为根节点的子树,其最左边叶子节点是该子树的 leader.图 5 中,$m1$ 是 $\{m1,m2\}$, $\{m1-m4\}$,$\{m1-m8\}$3 个子树的 leader,$m7$ 是$\{m7,m8\}$的 leader.密钥通过 leader 之间的协商来生成,并由 leader 更新给所在子树的其他组成员.Leader 与所在子树其他成员的通信用该子树根节点对应的密钥进行加密保护.

考虑加入的情况,节点 $m9$ 要求加入组播,其加入位置通过某种算法确定.假设由 $m1$ 接纳 $m9$,于是 $m1$ 与 $m9$ 协商,生成密钥 $k19$(协商可以采用 DH 算法或由 $m1$ 单独生成).$k19$ 作为新的组密钥由 $m1$ 用组播报文通知 $m2$ 到 $m8$(用 $k18$ 加密).新的密钥树如图 6 所示.

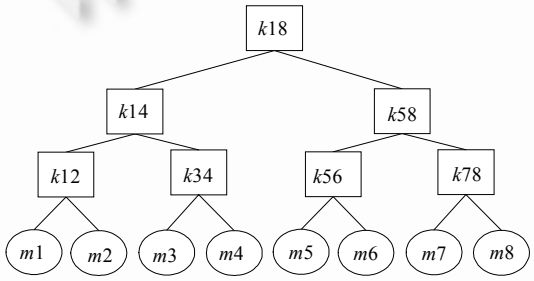

Fig.5 Logical key tree in distributed mode
图 5 分布式的逻辑密钥树

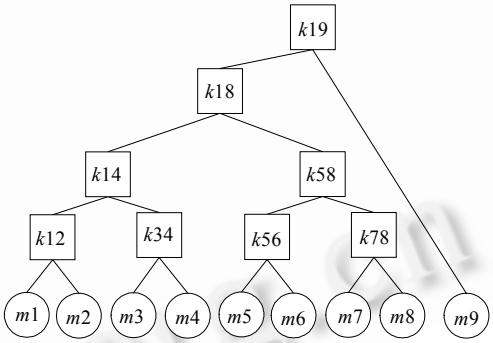

Fig.6 $m9$ joins the logical key tree
图 6 $m9$ 加入后的逻辑密钥树

下面讨论如何在原树中删除节点 $m1$.与集中控制的逻辑密钥树相同,$m1$ 节点所知的所有密钥都要被更新.$m2$ 取代 $m1$ 成为$\{m2-m4\}$,$\{m2-m8\}$的 leader.它分别与 $m3$,$m5$ 协商新的密钥 $k24$ 和 $k28$,$k28$ 成为新的组密钥.$k28$ 通过如下消息在全组内得到更新:$m2$ 通知 $m3$,$m4$(用 $k24$ 加密),$m5$ 通知 $m6$,$m7$ 和 $m8$(用 $k58$ 加密).删除节点 $m1$ 后的组播密钥树如图 7 所示.

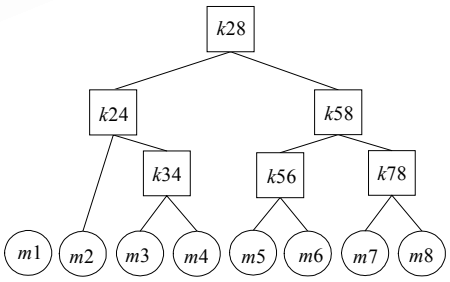

Fig.7 $m1$ leaves the logical key tree
图 7 $m1$ 离开后的逻辑密钥树

文献[12]给出了另一种在分布式环境下应用二叉组播密钥树的方式.它的密钥更新利用了 DH 算法的一种扩展形式——TGDH(tree-based group diffie-Hellman)[13].在使用这种算法的二叉逻辑密钥树中,父节点的密钥由其两个子节点的密钥通过 DH 算法得来,即 $Kp=(a^{Kl} \bmod q)^{Kr} \bmod q=(a^{Kr} \bmod q)^{Kl} \bmod q=a^{Kl\times Kr} \bmod q$,其中 Kp 是父节点的密钥,Kl,Kr 是左右子节点的密钥,a 和 q 是 DH 算法的两个参数,$a^k \bmod q$ 被称为该节点的 BK.叶子节点对应组成员,由于每个节点的 BK 都是公开的,所以每个组成员

可以根据自己的密钥来计算出从它对应的节点到根节点路径上的每个节点对应的密钥.

在成员加入和删除时,一个成员会被选举出来作为辅助成员(sponsor),负责对密钥进行更新.图 8 是新成员加入的示例.*m*7 加入时,*m*4 被选为辅助成员(为新加入成员选择辅助成员可以选择深度最浅的叶子节点对应的成员,如图 8 中的 *m*4 和 *m*3).节点 5 变成中间节点,增加叶子节点 11 和 12 分别对应 *m*4 和 *m*7.*m*7 发布 BK12,*m*4 计算新的 *K*5,*K*2 和 *K*0,并向全组成员公布 BK5,BK2.其他所有成员根据收到的 BK 更新各自需要更新的密钥,*m*1,*m*2 和 *m*3 利用 BK2 计算出新的 *K*0,*m*5,*m*6 利用 BK5 计算出 *K*2,进而计算出 *K*0.

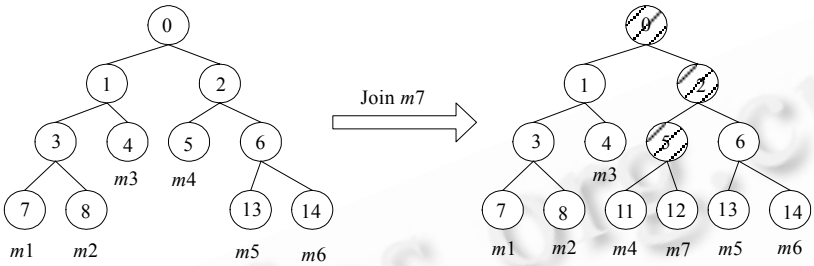

Fig.8 Key refreshment in single join
图 8 单个组员加入时密钥的更新

图 9 是成员离开的示例.*m*5 离开时,*m*4 被选作辅助成员(成员离开时它所对应节点的邻居节点上的一个叶子节点所对应的成员被选为辅助节点).*m*4 发布 BK5 和 BK2.*m*1,*m*2 和 *m*3 利用 BK2 计算出新的 *K*0,*m*6,*m*7 利用 BK5 计算出 *K*2,进而计算出 *K*0.

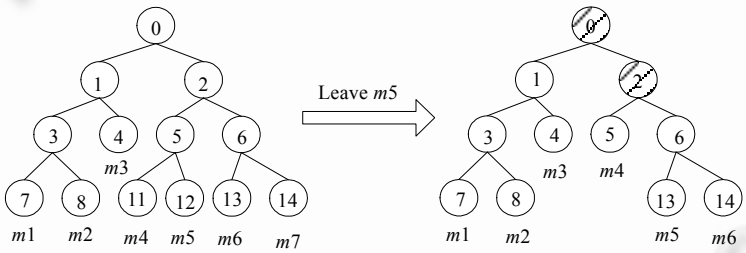

Fig.9 Key refreshment in single leave
图 9 单个组员离开时的密钥更新

文献[12]还给出了两种密钥更新算法,一种是一般的批量更新,另一种是使用队列的批量更新.这两种密钥更新的基本思路是,在若干数量的组成员关系变化发生后或一定的时间间隔后,重新构造出密钥树(密钥树的构造要尽量平衡),并进行必要的密钥更新.与一般的批量更新相比,使用队列的批量更新的不同之处在于,在两次密钥更新间隔中加入的成员首先加入一棵临时的逻辑密钥树 *T*′,当密钥更新时,*T*′被直接合并到原逻辑密钥树中.这减少了一般批量更新中当密钥更新时密钥的计算量和密钥更新的报文数量.

分布式的逻辑密钥树还有很多要考虑的问题.在分布式环境中,没有中央控制节点来控制系统行为,没有一个成员保存有完整的密钥树拓扑结构.在这种情况下,要确保各个成员所保存信息的一致性,才能确保在系统发生变化时行为的一致性.比如在成员加入时,如何判断成员加入的位置;在某个成员失效时,其他节点如何及时获知,尤其是方案 1 中的 leader 成员和方案 2 中的辅助成员,它们的状态必须得到监控,才能确保方案的健壮性.这些问题的解决需要增加完善组播密钥管理的控制信息,增强组成员的功能,使其更“聪明”,能够利用组播控制信息,被动和主动地获得系统状态,并在系统状态变化时作出正确的反应.

2.5 分层分组式的组播密钥管理方案

文献[14]给出了一个采用分层分组方式的组播密钥管理方案.在这个方案里,所有节点都属于最底层 *L*0,通过簇生成协议(clustering protocol),节点分成多个簇(cluser),每个簇都有一个 leader.*L*0 层的所有 leader 组成了 *L*1.与 *L*0 一样,在 *Li* 上执行簇生成协议,生成簇,所有簇的 leader 组成 *Li*+1.当某一层的成员个数为 1 时,不再继续往

上生成新的层.

每一层有一个层密钥(layer key),只有属于该层的成员才能知道该层的层密钥.每个簇都有一个簇密钥,只有属于该簇的成员才能知道该簇的簇密钥.除此之外,每个簇的 leader 都与该簇的其他成员建立点对点的安全通道.密钥服务器(key server)负责层密钥的生成;每个簇的 leader 负责该簇簇密钥的生成.由于所有成员都属于 $L0$,$L0$ 的层密钥被用作组密钥.层 Li 的层密钥更新是由密钥服务器利用 Li+1 层的层密钥加密,将新的 Li 层密钥组播发送给所有 Li+1 层成员,然后由它们利用各自在 Li 层的簇密钥更新给所有其他的 Li 层节点.

首先讨论组成员的加入.组成员 K 加入 $L0$ 中的簇 C.簇 C 的 leader 生成新的簇密钥,然后要求密钥服务器更新 $L0$ 的层密钥.

下面考虑组成员离开时的密钥更新.不考虑顶层节点,当组成员 K 离开时,设它所属的最高层为 L.K 在 L 层中属于簇 C_L,在 Li 层中属于 C_{Li}.K 是簇 C_{L0} 到 $C_{(L-1)}$的 leader 节点.密钥更新的第 1 步是在 C_{L0} 到 $C_{(L-1)}$中选出新的 leader 节点并更新簇密钥.注意,新选出的 C_{Li} 的 leader 节点要加入 L_{i+1};第 2 步是要求密钥服务器更新 $L0$ 到 L 层的层密钥.层密钥的更新是由上到下进行的.顶层节点离开除了可能会导致整个系统层数的减少以外,与其他节点离开没有太大的不同,在此不加以深入讨论.

该方案的特点在于:

(1) 利用簇生成协议动态合并簇或分解簇,保持簇的大小在一个限定区域.这使得系统的分层分组结构保持在一个合理、高效的状态.

(2) 组成员利用成员发现协议生成合理的拓扑结构,确保成员加入最近(跳数最少)的簇,并且成员发现协议还可以用来确认成员是否失败.

该方案可以看做是文献[11]中方案的强化,即通过簇生成协议和成员发现协议来增强密钥管理方案的智能,使其可以适应动态变化的网络环境,具有更强的可扩展性和健壮性.这也说明,通过附加的控制来增强组播密钥管理方案的效能是一种很好的途径.

在这种方案中,节点的逻辑结构、密钥更新路径都非常适合应用层组播[14].

3 总结和展望

组播密钥管理为组播提供安全保障,为组播在新型 Internet 业务中的应用铺平了道路.如前文所述,组播应用因其参与节点在地域分布、性能、网络接入等情况的差异,具有较大的多样性和复杂度,这就导致了难以提供统一的组播密钥管理方案.而逻辑密钥树及类似算法因为较好地解决了可扩展性问题,又兼顾了安全性,得到了广泛的应用.除此之外,在特定应用中,组播密钥管理还有待进一步的研究.

分布式的组播,比如 P2P 的环境下的组播,由于没有中央控制节点,各个节点间必须通过传递控制信息来维持状态的同步,节点必须能在必要的时候主动侦测其他节点和网络的状态,节点还必须能够根据掌握的信息采取正确的行为,要能够处理异常情况.因此,分布式的组播密钥管理要完善密钥管理以外的控制协议,使得系统具有更强的适应性,进而提高系统在动态变化的网络环境中的可扩展性、健壮性和系统行为的可预测性.

应用层组播是在 Internet 上提供组播服务的一种新的策略.在应用层组播中,组成员自组织成应用层的覆盖网络,并自行建立组播转发拓扑结构而不需要网络层提供组播支持.应用层组播的密钥管理方案可以把密钥管理的控制结构与组播转发拓扑结构结合起来,以提高效率[14,15].

References:

[1] Harney H, Muckenhirn C. Group key management protocol (GKMP) specification. RFC2093, 1997.

[2] Harney H, Muckenhirn C. Group key management protocol (GKMP) architecture. RFC2094, 1997.

[3] Setiner M, Taudik G, Waidnet M. Cliques: A new approach to group key agreement. Technical Report, RZ 2984, IBM Research, 1997.

[4] Diffie W, Hellman ME. New directions in cryptography. IEEE Trans. on Information Theory, 1976,IT-22(6):644~654.

[5] Mittra S. Iolus: A framework for scalable secure multicasting. In: ACM SIGCOMM Computer Communication Review, Volume 27, Issue 4. New York: ACM Press, 1997. 277~288.

[6] Wallner D, Harder E, Agee R. Key management for multicast: Issues and architec-tures. RFC 2627, 1999.
[7] Waldvogel M, Garonni G, Sun D, Weiler N, Plattner B. The VersaKey framework: Versatile group key management. IEEE Journal on Selected Areas in Communications (Special Issue on Middleware), 1999,17(9):1614~1631.
[8] Balenson D, McGrew D, Sherman A. Key management for large dynamic groups: One-Way function trees and amortized initialization. IETF Internet Draft (work in progress), 2000.
[9] Canetti R, Caray J, Itkis G, Micciancio D, Naorr M, Pinkas B. Multicast security: A taxonomy and some efficient constructions. In: Proc. of the INFOCOM'99. New York, 1999. 708~716.
[10] Yang L, Li XS, Zhang XB, Lam SS. Reliable group rekeying: A performance analysis. In: ACM SIGCOMM 2001. San Diego, 2001. 27~31.
[11] Rodeh O, Birman K, Dolev D. Optimized group rekey for group communication systeims. Technical Report, Hebrew University, 1999.
[12] Lee PPC, Lui JCS, Yau DKY. Distributed collaborative key agreement protocols for dynamic peer groups. In: Proc. of the ICNP. 2002. 53~62.
[13] Kim Y, Perrig A, Tsudik G. Simple and fault-tolerant key agreement for dynamic collaborative groups. In: Proc. of the 7th ACM Conf. on Computer and Communications Security. 2000. 235~244.
[14] Banerjee S, Bhattacharjee B. Scalable secure group communication over IP multicast. JSAC Special Issue on Network Support for Group Communication, 2002,20(8):156~163.
[15] Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast. In: ACM SIGCOMM 2002. 2002. 43~51.

敬 告 作 者

《软件学报》创刊以来,蒙国内外学术界厚爱,收到许多高质量的稿件,其中不少在发表后读者反映良好,认为本刊保持了较高的学术水平.但也有一些稿件因不符合本刊的要求而未能通过审稿.为了帮助广大作者尽快地把他们的优秀研究成果发表在我刊上,特此列举一些审稿过程中经常遇到的问题,请作者投稿时尽量予以避免,以利大作的发表.

1. 读书偶有所得,即匆忙成文,未曾注意该领域或该研究课题国内外近年来的发展情况,不引用和不比较最近文献中的同类结果,有的甚至完全不列参考文献.

2. 做了一个软件系统,详尽描述该系统的各个方面,如像工作报告,但采用的基本上是成熟技术,未与国内外同类系统比较,没有指出该系统在技术上哪几点比别人先进,为什么先进.一般来说,技术上没有创新的软件系统是没有发表价值的.

3. 提出一个新的算法,认为该算法优越,但既未从数学上证明比现有的其他算法好(例如降低复杂性),也没有用实验数据来进行对比,难以令人信服.

4. 提出一个大型软件系统的总体设想,但很粗糙,而且还没有(哪怕是部分的)实现,很难证明该设想是现实的、可行的、先进的.

5. 介绍一个现有的软件开发方法,或一个现有软件产品的结构(非作者本人开发,往往是引进的,或公司产品),甚至某一软件的使用方法.本刊不登载高级科普文章,不支持在论文中引进广告色彩.

6. 提出对软件开发或软件产业的某种观点,泛泛而论,技术含量少.本刊目前暂不开办软件论坛,只发表学术文章,但也欢迎材料丰富,反映现代软件理论或技术发展,并含有作者精辟见解的某一领域的综述文章.

7. 介绍作者做的把软件技术应用于某个领域的工作,但其中软件技术含量太少,甚至微不足道,大部分内容是其他专业领域的技术细节,这类文章宜改投其他专业刊物.

8. 其主要内容已经在其他正式学术刊物上或在正式出版物中发表过的文章,一稿多投的文章,经退稿后未作本质修改换名重投的文章.

本刊热情欢迎国内外科技界对《软件学报》踊跃投稿.为了和大家一起办好本刊,特提出以上各点敬告作者.并且欢迎广大作者和读者对本刊的各个方面,尤其是对论文的质量多多提出批评建议.