基于多叉树的多权限群组密钥管理

徐 旸 周 薇 杜秋双 王国军
(中南大学信息科学与工程学院 长沙 410083)

摘 要 在多权限群组通信中,由于用户可根据不同权限获取不同的数据资源,因此其安全问题比传统(单一权限)群组通信更难处理。为此,提出一种新的集中式多权限群组密钥管理方案,即采用多叉树构建密钥图,并为图中节点分配ID,以便用户快速推算出节点间的关系,从而确定需更新的密钥。当群组内的用户关系发生变化时,其他用户可通过单向函数、旧密钥以及密钥更新素材来实现密钥的更新。理论分析与模拟实验显示,相比现有的方案,新方案在保证前/后向安全性的同时,降低了密钥存储和更新的开销,具有更好的性能表现。

关键词 多权限群组通信,密钥管理,多叉树,单向函数,密钥更新素材
中图法分类号 TP393 **文献标识码** A

Multiway Tree-based Group Key Management Scheme for Multi-privileged Group Communications

XU Yang ZHOU Wei DU Qiu-shuang WANG Guo-jun
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract In multi-privileged group communications, since users can access multiple data resources according to their different privileges, security issues become more difficult to solve than that in traditional group communications. Therefore, this paper proposed a novel centralized group key management scheme for multi-privileged environments. The proposed scheme employs multiway tree to construct a key graph and assigns a unique ID for every node in the key graph, so that the relationship between keys can be deduced by an ID which will contribute to locating the affected keys efficiently. As a result, the related users can update the affected keys through previous keys or with a rekeying material by using a one-way function when membership changes dynamically. Theoretical analysis and experimental simulation results show that the proposed scheme can reduce the storage and rekeying overhead efficiently, and it outperforms some previous schemes. Meanwhile, the forward and backward security is also guaranteed.

Keywords Multi-privileged group communications, Key management, Multiway tree, One-way function, Rekeying material

1 引言

随着Internet技术的高速发展,组播技术因其较低的带宽开销而得到了广泛的应用。与此同时,其安全问题也日益突出。安全组播要求只有组内用户可以获取组播信息,而且用户不能获取在其加入之前的组播信息(后向安全性),以及用户在退出组播组后,不能获取后续的组播信息(前向安全性)[1]。

目前,群组密钥管理技术已成为保障组播安全的有效手段之一。

在传统(单一权限)群组通信中,由于只有一种数据资源且所有用户拥有相同的权限,其密钥管理相对简单。组内的数据资源均用同一个会话密钥(Session Key,简称SK)[2]进行加密且为所有合法用户所共享。当有用户加入/离开群组时,就更新该SK,以保证安全性。

然而,在多权限群组通信中,存在着多个数据资源,且用户可根据权限获取不同的数据资源。通常,能访问同一种数据资源的用户集被称为数据组(Data Group,简称DG),而具有相同权限、能访问相同数据资源集的用户集被称为服务组(Service Group,简称SG)。为实现安全组播,各数据资源需使用不同的SK进行加密且用户可根据自己的权限获得相应的SK。当用户关系发生变化时,与该用户共享SK的其他用户都需要更新这些SK。因此,多权限群组密钥管理方案不仅要能够防止用户获取未经授权的数据资源,还要能够有效地更新那些受用户加入/退出/转移操作影响的密钥。

为了解决多权限群组通信中的安全问题,本文提出了一种基于多叉树的多权限群组密钥管理方案(Multiway Tree-based Group Key Management Scheme for Multi-privileged Group Communications,简称MTGKM),即采用多叉树构建密钥图并为图中节点分配ID,以便快速确定需要更新的密

到稿日期:2013-09-17 返修日期:2013-11-21 本文受国家自然科学基金项目(61272151,61073037),高等学校博士学科点专项科研基金(20110162110043)资助。

徐 旸(1988-),男,硕士生,主要研究方向为安全组播通信、透明计算安全问题,E-mail:xuyangcsu@gmail.com;周 薇(1980-),女,博士生,主要研究方向为信息安全、群组通信;杜秋双(1986-),男,硕士生,主要研究方向为多权限组播通信安全问题;王国军(1970-),男,教授,博士生导师,主要研究方向为信息安全、云计算、透明计算等,E-mail:csjwang@csu.edu.cn(通信作者)。

钥;同时,通过单向函数,对旧密钥或结合密钥更新素材的旧密钥进行计算,以实现用户加入/退出/转移过程中的密钥更新,从而减少密钥存储与更新的开销。

2 相关工作

针对传统(单一权限)群组通信,国内外学者已提出了多种有效的群组密钥管理方案。其中,文献[3]提出了经典的逻辑密钥树方案(Logical Key Hierarchy,简称 LKH),它采用树型结构组织密钥图,将密钥分发中心(Key Distribution Center,简称 KDC)更新会话密钥的开销从 $O(n)$ 降至 $O(\log n)$。在此基础上,文献[4]提出了一种基于二叉树的密钥管理模型,使用户可在随机数的帮助下,通过单向函数将旧密钥更新为新密钥。该方案在用户退出时具有较高的效率,但仅适用于采用平衡二叉树结构的密钥图。为降低密钥树的高度,文献[5]提出了基于 m 叉树与 DH 协议的组密钥协商协议,即通过 m 叉树结构降低树的高度来减少密钥节点的数量。但采用 DH 协议更新密钥的代价较大。

在一些新型组播应用中,不仅出现了多个数据资源并存的情况,而且用户可以获得不相同的数据资源,从而形成了不同的组播权限。由于存在多个数据资源且用户可在服务组之间转移,因此传统的群组密钥管理方案无法简单地移植到多权限群组通信中来。

目前,专门针对多权限群组通信的群组密钥管理方案并不是十分常见,其中文献[6]提出的多群组密钥管理方案(Multi-Group Key Management Scheme,简称 MGKMS)利用各个 SG 之间数据资源的冗余关系,将各个 SG 对应的密钥树构建成一个聚合密钥图(Integrated Key Graph,简称 IKG)来进行管理,从而消除了因各 DG 之间用户重叠而造成的冗余。但在用户退出时,由于密钥可能被多个用户共享而加重了 KDC 的密钥更新负担。在 MGKMS 的基础上,我们在文献[7,8]中提出了基于单向函数的群组密钥管理协议(One-way Function-based Group Key Management Protocol,简称 OFGKMP)。该协议根据文献[9]的方式构造密钥图,通过为密钥节点分配 ID 来快速确定需要更新的密钥,同时,利用单向函数使得用户能够自行推算出全部或部分新密钥[10],从而减少密钥更新的开销。但由于 MGKMS 和 OFGKMP 均基于二叉密钥树结构,当用户规模不断扩大时,密钥数量将随着树的层次的增加而迅速增长,进而给用户和 KDC 造成巨大的密钥更新开销。为了进一步降低密钥更新过程中的开销,文献[11]提出了可用于多权限群组的不分裂平衡高阶树方案(Non-Split Balancing Higher Order Tree,简称 NSBHOT),即采用 2-3 树结构来存储密钥,从而减少由于节点分裂所造成的通信开销。为缓解频繁的用户加入/退出操作对 KDC 所造成的压力,文献[12]在平衡树的基础上,提出了一种基于最小(N,T)策略的密钥批量更新方案,当加入/退出的用户数达到 N 值或者离上一次密钥更新的时间达到 T 值时,系统就开始更新密钥。然而,该方案不能严格地保障前向安全性。为了适应多权限群组中数据资源加入/退出的情况,文献[13]所提出的 MM-MSKMS 方案定义了一个由不同密钥树所构成的密钥管理图,从而降低了构建 IKG 的复杂性,实现了 SG 的动态形成以及解体。

3 密钥管理方案

本节以三叉树结构的密钥图为例,描述了本方案中密钥图的组成以及密钥节点 ID 的分配方式,介绍了当用户加入/退出或是在不同 SG 之间转移时的密钥更新方案。本方案的密钥图中采用多叉树结构,可以让同层次树结构容纳更多用户,增强了方案可扩展性;同时可以减少用户持有叶子节点到根节点的密钥数,节省存储开销。在密钥管理方案中引入密钥更新素材,当成员退出或是转出时,服务器用加密的密钥更新素材取代其它方案中加密的新密钥,并将之发送给组中其他成员,以减少密钥更新时的通信开销。

3.1 系统描述

本方案为集中式的群组密钥管理方案,采用聚合密钥图维护所有密钥节点,依赖于统一的 KDC 实现密钥的生成、分发与管理。密钥图由 DG 子图部分与 SG 子树部分组成,其中,DG 子图部分采用二叉树结构,根据各 SG 内 SK 之间的冗余关系构成,是所有 SK 节点以及它们到各个 SG 子树根节点(不含)路径上节点的集合。而 SG 子树部分采用多叉树结构,是所有的 SG 子树的集合。

密钥图中的 SK 节点对应于加密数据资源的 SK,其余非叶子节点均为辅助节点,对应于密钥加密密钥(Key Encrypted Key,简称 KEK),而叶子节点为用户节点,用于存放用户的私人密钥。由图 1 可知,用户拥有从各自用户节点(叶节点)到对应 SK 节点的路径上的所有密钥,且各个 SK 和 KEK 均被多个用户所共享。所以,当群组内的用户关系发生变化时,必须更新该用户所掌握的密钥,以保证安全的群组通信。

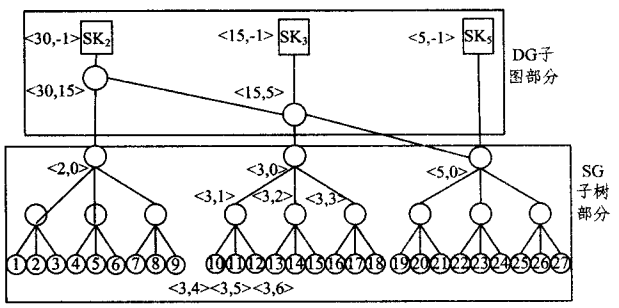

图 1 基于多叉树结构的密钥图($d=3$)

为了快速定位密钥图中的节点,我们对每棵 SG 子树进行编号并为每个节点分配一个 ID。

各 SG 子树采用素数进行编号,即 $SG_2, SG_3, \cdots, SG_i$(i 为素数)。位于 SG 子树部分的密钥节点的 ID 记为$\langle i,m\rangle$,其中 i 表示该节点位于 SG_i 子树,而 m 表示该节点从 SG_i 子树的根节点(ID 为$\langle i,0\rangle$)开始,自上而下、自左而右排列在第 m 个位置($m\geqslant 0$ 且为自然数)。

DG 子图部分的密钥节点的 ID 记为$\langle p,q\rangle$,若某节点有 $k_{\langle p_1,q_1\rangle}, k_{\langle p_2,q_2\rangle}$ 两个孩子节点,则 ID 记为$\langle lcm(p_1,p_2), \max(p_1,p_2)\rangle$。若其仅有唯一子节点 $k_{\langle p_1,q_1\rangle}$,则 ID 为$\langle p_1,-1\rangle$。

根据上述方式,用户可通过某用户节点的 ID$\langle i,m\rangle$,利用 $k_{\langle i,\lfloor (m-1)/d\rfloor\rangle}$($d$ 为多叉树的度)是 $k_{\langle i,m\rangle}$ 的父节点这一关系,推算该用户节点所有祖先节点的 ID,从而找出该用户在其所处的 SG 子树中的全部密钥节点;同时,利用 i 所含的素因子,找出该用户在 DG 子图部分的全部密钥节点,从而推算出自己与其共享的全部密钥,进而在用户动态变化时更新相应的密钥,以保证群组通信过程的安全。

3.2 密钥更新算法

3.2.1 用户加入过程

当用户 u 加入 SG_j 时,从新加入节点到其可达的 SK 节点之间的路径(称为加入路径)上的密钥均需要更新。

首先，KDC 新建一个用户节点并将其插入到 SG_j 子树中。为保持树的平衡，KDC 会在插入前判断 SG_j 子树是否为满树。如果不是，则将新节点作为叶子插入到 SG_j 子树的最短路径处；如果是满树，则 KDC 下移 SG_j 子树中最高层最左端的用户节点，并在该位置生成一个新的 KEK 节点。新的 KEK 节点继承下移节点原来的 ID，并作为下移的用户节点和新插入的 u 节点的父亲节点。然后，KDC 为新节点分配 ID $\langle j,m_1\rangle$，并广播新加入节点 ID$\langle j,m_1\rangle_J$ 和下移节点 ID$\langle j,m_2\rangle_M$。其他用户根据 ID$\langle j,m_1\rangle_J$ 推算自己需要更新的密钥，并通过单向函数更新这些密钥 $k'=f(k)$。而如果用户发现 KDC 同时广播了自己的 ID$\langle j,m_2\rangle_M$，就获知自己的用户节点发生下移，于是将该节点的 ID 更新为$\langle j,m_1-1\rangle$，并计算其父亲节点(新生成的 KEK 节点)对应的密钥 $k'_{\langle j,m_2\rangle}=f(k_{\langle i,0\rangle}\oplus k_{\langle i,m_1-1\rangle})$。最后，KDC 将新用户 u 加入路径上的密钥(更新后)通过安全信道分发给 u。

3.2.2 用户退出过程

当用户 u 退出 SG_l 时，从退出用户节点到其可达的 SK 节点之间的路径(称为退出路径)上的密钥均需要更新。

首先，KDC 广播 u 用户节点的 ID$\langle l,n\rangle_L$，其他用户根据 ID$\langle l,n\rangle_L$ 推算需要更新的密钥，即密钥 $k_{\langle l,\lceil(n-1)/d\rceil\rangle}$，$k_{\langle l,\lceil(\lceil(n-1)/d\rceil-1)/d\rceil\rangle}$，…，$k_{\langle l,0\rangle}$，以及 DG 子图部分中 ID 的第一项可被整除的密钥节点。然后，KDC 生成密钥更新素材 R，并分别使用其它 $SG_{\{a|a\in S且a\neq l\}}$（$S$ 为 SG 编号的集合）子树的根节点密钥以及 SG_l 子树内 u 所在路径上节点的兄弟节点所对应的密钥对 R 进行加密并组播上述密文。其他用户根据退出用户的 ID$\langle l,n\rangle_L$ 进行判断，如果与 u 同属于 SG_l，即拥有 ID 的第一项为 l 的密钥，则用户在自己不需要更新的密钥集中，找出节点 ID 的第一项为 l，而第二项为其中最小值的密钥，解密得到密钥更新素材 R；如果用户不属于 SG_l，则用自己所属子树的根节点密钥解密得到密钥更新素材 R；最后，将需更新的旧密钥 k 与密钥更新素材 R 进行异或运算，再利用单向函数，计算出更新后的新密钥 $k'=f(k\oplus R)$。

3.2.3 用户转移过程

SG_l 中的用户 u 转移至 SG_j 可视为用户 u 先退出 SG_l 再加入 SG_j。其操作过程基本等同于上述加入/退出过程的叠加，KDC 将 u 的用户节点从 SG_l 移至 SG_j 子树末端并广播退出节点 ID$\langle l,n\rangle_{SL}$，转入节点 ID$\langle j,m_1\rangle_{SJ}$，如有节点的位置发生下移，同时广播该节点的 ID。需要注意的是，由于退出路径和加入路径可能会有重叠，因此重叠部分的密钥，也就是 SG_l 和 SG_j 在 DG 子图部分中共享的密钥(即节点 ID 的第一项可整除 $l\times j$ 的节点所对应的密钥)因用户没有改变对相应数据资源的访问权限而不需要更新。

此外，为避免转移用户在转入的 SG_j 中获得密钥更新素材，从而继续获取已退出服务组 SG_l 的后续数据，必须在退出过程完成后再执行加入操作。用户端转移操作算法见图 2。

例如，用户 u_{18} 从 SG_3 转移至 SG_2，可视为用户 u_{18} 先退出 SG_3 再加入 SG_2。KDC 广播$\langle 3,12\rangle_{SL}$，$\langle 2,14\rangle_{SJ}$ 和$\langle 2,4\rangle_M$。

KDC 从 SG_3 中移去用户节点 $k_{\langle 3,12\rangle}$，然后生成密钥更新素材 R，并加密组播给所有需要更新密钥的用户。

KDC→ $u_{16}:E\{k_{\langle 3,10\rangle},R\}$；$u_{17}:E\{k_{\langle 3,11\rangle},R\}$；$u_{10-12}:E\{k_{\langle 3,1\rangle},R\}$；$u_{13-15}:E\{k_{\langle 3,2\rangle},R\}$；$u_{19-27}:E\{k_{\langle 5,0\rangle},R\}$。

```
Input: <l, n>SL, <j, m1>SJ, optional<j, m2>M;
Output: the updated keys;
/*<l, n> is the ID of leaving node, <j, m1> is the ID of newly
 joining node, <j,m2> is the ID of the split node if exists*/
//Deal with the split node;
if (the user holds k<j, m2>) {
// <x , y> is the ID of a node.
    ID.<x , y>=<j, m1-1>;
  k'<j, m2> = f(k<x,y> ⊕ k<j, 0>); } }
//Obtain the rekeying material R
if (the user holds k<x , 0> where x!=l)
//the user doesn't belong to SGl
//Dec(k,*) denotes the decryption of ciphertext * by the key k
 Dec(k<x , 0>, E(k<x , 0>, R));
else{//the user belongs to SGl
 q=n;
 while (q>0&& ID.x==l){
  t=⌊(q-1)/d⌋;
    if (the user holds k<l , q>){
     h=min(ID.y) where ID.y>=t&& ID.y!=q;
     Dec( k<l , h>, E (k<l ,h>, R);break; }
  q= t; } }
//for keys in SG-subtree
while (m1>0 || n>0) {
 n =⌊(n-1)/d⌋;
if (the user holds k<l , n>)
 k'<l, n> = f(k<l, n> ⊕ R);
m1=⌊(m1-1)/d⌋;
  if(the user holds k<j , m1>)
    k'<j, m1> = f(k<j, m1>); }
//for keys in DG subgraph
if(ID.x is a composite number){
 if(ID.x mod l ==0) && (ID.x mod j !=0)
  k'<x, y> =f(k<x, y> ⊕ R);
 if(ID.x mod l !=0) && (ID.x mod j ==0)
k'<x, y> =f(k<x, y>); }
```

图 2 用户端转移操作的密钥更新算法

对于用户 u_{18} 在 SG_3 中拥有而在转入 SG_2 后没有的密钥 $k_{\langle 3,3\rangle}$，$k_{\langle 3,0\rangle}$，$k_{\langle 15,5\rangle}$ 和 $k_{\langle 15,-1\rangle}$，受影响的用户解密得到密钥更新素材 R，将需要更新的旧密钥与密钥更新素材 R 进行异或，再利用单向函数进行一次计算，得出新密钥。

$k'_{\langle 3,3\rangle}=f(k_{\langle 3,3\rangle}\oplus R)$，$k'_{\langle 3,0\rangle}=f(k\langle 3,0\rangle\oplus R)$，

$k'_{\langle 15,5\rangle}=f(k_{\langle 15,5\rangle}\oplus R)$，$k'_{\langle 15,-1\rangle}=f(k_{\langle 15,-1\rangle}\oplus R)$。

在完成退出操作后，KDC 将 u_{18} 用户节点插入 SG_2，因 SG_2 子树是满树，原 u_1 用户节点下移，ID 将变为$\langle 2,13\rangle$，且在原节点 $k_{\langle 2,4\rangle}$ 位置生成一新密钥节点 $k'_{\langle 2,4\rangle}$，u_{18} 在 SG_2 中的 ID 为$\langle 2,14\rangle$。

对于用户 u_{18} 在 SG_3 中没有而在 SG_2 中新获得的密钥 $k_{\langle 2,1\rangle}$，$k_{\langle 2,0\rangle}$ 和 $k'_{\langle 2,4\rangle}$，受影响的用户利用单向函数对旧密钥 $k_{\langle 2,1\rangle}$ 和 $k_{\langle 2,0\rangle}$ 进行一次计算，得到新密钥。

$k'_{\langle 2,1\rangle}=f(k_{\langle 2,1\rangle})$，$k'_{\langle 2,0\rangle}=f(k_{\langle 2,0\rangle})$。

而 $k_{\langle 2,4\rangle}$ 是由于原 u_1 用户节点下移而生成的新密钥节点，其更新操作如下：

$k'_{\langle 2,4\rangle}=f(k_{\langle 2,13\rangle}\oplus k_{\langle 2,0\rangle})$。

最后，KDC 将 u_{18} 所需的密钥通过安全信道发送给它。

4 理论分析

4.1 安全性分析

在本节中，我们从前向安全性与后向安全性两个方面，对所提方案的安全性进行了分析。

4.1.1 后向安全性

当新用户进入（加入或转入）某个 SG 时，其加入路径上的所有旧密钥 k 将通过单向函数 f 更新为新密钥 $k'=f(k)$，且新密钥 k' 是通过安全信道分发新加入的用户。由于单向函数在多项式时间内是不可逆的，因此新用户无法通过新密钥 k' 反向恢复出旧密钥 k，由此实现了后向安全性。

4.1.2 前向安全性

当有用户离开（退出或转出）某个 SG 时，其退出路径上的所有旧密钥 k 将通过单向函数与密钥更新素材 R 进行更新，得到新密钥 $k'=f(k\oplus R)$。虽然离开的用户掌握单向函数 f 的算法与旧密钥 k，但由于 KDC 使用的是离开用户所不具有的密钥，即其他 SG 的根节点密钥以及离开用户所属的 SG 子树内退出路径上节点的兄弟所对应的密钥，因此分别对密钥更新素材 R 进行加密，并发送所得到的多个密文，因而离开的用户无法解密得到 R，从而也就无法计算出新的密钥 k'，由此保证了前向安全性。

而用户在不同 SG 之间转移的过程，可看作其先退出原 SG 再加入新 SG 的过程。对于用户转入的 SG，KDC 先使用该组的旧密钥加密钥更新素材 R 并发送密文，再将新密钥发送给转入用户，因此，虽然转移的用户拥有其转出 SG 的旧密钥 k 并且掌握单向函数 f 的算法，但由于没有转入 SG 的旧密钥，因此无法获得密钥更新素材 R，也就无法计算出其转出 SG 的新密钥。同时，用户退出/加入过程的安全性并无变化，所以仍可保证前向安全性和后向安全性。

综上所述，本文提出的方案有效地保证了多权限群组通信中的安全性。

4.2 性能分析与比较

存储、通信以及计算开销是衡量密钥管理方案的主要性能指标，本节从上述 3 个方面，对所提方案进行分析，并与 MGKMS 与 OFGKMP 方案进行了比较。

4.2.1 存储开销

在 MGKMS、OFGKMP 以及 MTGKM 方案中，密钥图由 DG 子图与 SG 子树两个部分组成，为便于分析，我们将 DG 子图部分的密钥数量记为 N_k^{DG}。由于上述方案 DG 子图的构造相同，且 N_k^{DG} 与用户数量无关，而仅与各 SG 内 SK 之间的冗余有关，因此三者的 N_k^{DG} 相同。而对于 SG 子树部分，由于 OMGM 采用了多叉树结构，相比前二者，其树高从 $1+\lceil\log_2 n_i\rceil$ 下降为 $1+\lceil\log_d n_i\rceil$（n_i 为 SG_i 的用户数量），由此可知 SG_i 子树的密钥总数为 $n_i+\lceil(n_i-1)/(d-1)\rceil$，而 KDC 需要维护的密钥总数为 $N_k^{DG}+\sum_{i\in S}(n_i+\lceil(n_i-1)/(d-1)\rceil)$（$S$ 为各 SG 编号的集合）。由于每个用户需要保存从用户节点到对应 SK 节点的全部密钥，因此，在最坏情况下，SG_i 中的用户需要保存 $1+\lceil\log_d n_i\rceil$ 个 SG_i 子树部分的密钥和 N_{uk}^{DG} 个 DG 子图部分的密钥，所以用户需要存储的密钥总数为 $N_{uk}^{DG}+1+\lceil\log_d n_i\rceil$。

由于密钥存储开销等于密钥数量与密钥长度之积，因此，三者存储开销如表 1 所列，其中，L_k 表示单个密钥长度。

表 1 存储开销的性能比较

	MGKMS	OFGMKP	MTGKM
KDC	$(N_k^{DG}+\sum_{i\in S}(2n_i-1))\times L_k$	$(N_k^{DG}+\sum_{i\in S}(2n_j-1))\times L_k$	$(N_k^{DG}+\sum_{i\in S}(n_i+\lceil(n_i-1)/(d-1)\rceil))\times L_k$
用户	$(N_{uk}^{DG}+\lceil\log_2 n_i\rceil+1)\times L_k$	$(N_{uk}^{DG}+\lceil\log_2 n_i\rceil+1)\times L_k$	$(N_{uk}^{DG}+1+\lceil\log_d n_i\rceil)\times L_k$

由表 1 可知，MTGKM 方案有效地降低了密钥存储开销。

4.2.2 通信开销

通信开销主要由密钥更新时 KDC 传输的消息量决定。

当有用户加入时，3 种方案中的其他用户均能够通过单向函数自行更新密钥，所以组播消息量均为 0。此外，3 种方案均需要由 KDC 将加入路径上经过更新的密钥加密单播给新加入的用户，假设密钥被加密成大小为 L_{Ek} 的密文，因此 MGKMS 和 OFGKMP 单播开销为 $L_{Ek}\times(N_{uk}^{DG}+\lceil\log_2 n_i\rceil)$，其中，$N_{uk}^{DG}$ 代表用户 u 在 DG 子图部分拥有的密钥数。而 MTGKM 由于采用了多叉树结构，其单播开销下降为 $L_{Ek}((N_{uk}^{DG}+\lceil\log_d n_i\rceil)$。当有用户离开时，3 种方案的单播开销均为 0，而组播开销有所差异。MGKMS 需组播 $2(\lceil\log_2 n_i\rceil+N_{uk}^{DG})$ 个经过加密的密钥，而 OFGKMP 通过让部分用户自行更新密钥的方式，将该数量减少了一半。在我们的方案中，KDC 使用其他 SG 子树的根密钥以及 SG 子树内离开用户所在路径上节点的兄弟节点对应的密钥，分别对密钥更新素材 R 进行加密并组播这些密文，因此，其组播开销为 $L_{ER}\times(N_{SG}-1+(d-1)\times\lceil\log_d n_i\rceil)$，其中，$N_{SG}$ 表示 SG 的总数，L_{ER} 表示加密 R 所得的密文大小。由于 R 的长度可小于密钥 k 的长度，因此 $L_{ER}\leqslant L_{Ek}$。而用户转移的过程可看作加入、退出过程的结合，对于 MGKMS 与 OFGKMP 而言，由于在 DG 子图部分存在着 N_{ks}^{DG} 个被共享的、不会引起通信开销的密钥，其单播开销均为 $L_{Ek}\times(N_{uk}^{DG}+\lceil\log_2 n_i\rceil-N_{ks}^{DG})$，而其组播开销分别为 $2L_{Ek}\times(N_{uk}^{DG}+\lceil\log_2 n_i\rceil-N_{ks}^{DG})$ 与 $L_{Ek}\times(N_{uk}^{DG}+\lceil\log_2 n_i\rceil-(N_{ks}^{DG})$。在 MTGKM 方案中，转移过程的通信开销与 N_{ks}^{DG} 无关，而仅与 SG 的总数 N_{SG} 和转移用户原来所处的 SG 子树的高度有关，因此其单播、组播开销分别为 $L_{Ek}\times(N_{uk}^{DG}+\lceil\log_d n_i\rceil)$ 与 $L_{ER}\times(N_{SG}-1+(d-1)\times\lceil\log_d n_i\rceil)$。

4.2.3 计算开销

表 2 KDC 计算开销的性能比较

	MGKMS	OFGMKP	MTGKM
加入	$(N_{jk}^{DG}+\lceil\log_2 n_j\rceil)\times C_f+2C_r$	$(N_{jk}^{DG}+\lceil\log_2 n_j\rceil+1)\times C_f+C_r$	$(N_{jk}^{DG}+\lceil\log_d n_j\rceil+1)\times C_f+C_r$
退出	$(N_{lk}^{DG}+\lceil\log_2 n_l\rceil)\times(C_r+2C_E)$	$(N_{lk}^{DG}+\lceil\log_2 n_l\rceil)\times C_f+C_X+C_E)$	$(N_{lk}^{DG}+\lceil\log_d n_l\rceil)\times(C_f+C_X)+C_E\times(N_{SG}-1+(d-1)\times\lceil\log_d n_j\rceil)$
转移	$2C_r+C_f\times(N_{jk}^{DG}-N_{ks}^{DG}+\lceil\log_2 n_j\rceil)+(C_r+2C_E)\times(N_{lk}^{DG}-N_{ks}^{DG}+\lceil\log_2 n_l\rceil)$	$C_r+C_f\times(N_{jk}^{DG}-N_{ks}^{DG}+\lceil\log_2 n_j\rceil+1)+(C_f+C_E)\times(N_{lk}^{DG}-N_{ks}^{DG}+\lceil\log_2 n_l\rceil)$	$C_r+C_f\times(N_{jk}^{DG}-N_{ks}^{DG}+\lceil\log_d n_j\rceil+1)+(C_f+C_X)\times(N_{lk}^{DG}-N_{ks}^{DG}+\lceil\log_d n_l\rceil)+C_E\times(N_{SG}+(d-1)\times\lceil\log_d n_l\rceil)$

计算开销的分析过程大致与通信开销相似，本节分别用 C_r，C_E，C_X，C_f 表示密钥生成计算开销、加解密计算开销、异或运算开销以及单向函数计算开销，其中 C_f 的开销最大，C_E 次之，而 C_X 最小。此外，N_{jk}^{DG} 与 N_{lk}^{DG} 表示加入路径和退出路径上 DG 子图部分需要更新的密钥，n_j 与 n_l 分别表示 SG_j 与 SG_l 的用户数量。表 2 比较了 MGKMS、OPGKMP 以及 MT-

GKM 3 者在 KDC 端的计算开销。

由表 2 可知，与 MGKMS 和 OFGKMP 相比，MTGKM 方案在用户加入时，降低了 KDC 端密钥更新的计算开销，且当 $N_{SG}-1+(d-1)\times\lceil\log_d n_l\rceil\leqslant N_{lk}^{OG}+\lceil\log_2 n_l\rceil$ 时，KDC 端在用户退出或转移时，也将具有最低的密钥更新计算开销。

5 模拟实验

本节通过模拟实验，比较了在不同用户数量和不同用户转移频率的情况下，MGKMS、OFGKMP 和 MTGKM 3 种方案 KDC 端的密钥更新开销。由于密钥更新开销主要取决于需要更新的密钥数量，因此，本节通过模拟实验统计不同场景下需要更新的密钥数量，来比较不同方案的性能。另外，由于密钥图中 DG 子图并不确定，且 3 种方案 DG 构造方式相同，因此，忽略这部分密钥计算的开销并不影响方案间的性能比较。

5.1 模拟场景

模拟实例包含以下两个场景：

A. 群组中有 3 个数据资源，用户加入/退出/转移的概率均为 0.01，用户数量从 100 增加到 3000，每次增量为 100。

B. 群组中有 3 个数据资源、15000 名用户，用户在不同 SG 间转移的频率从 0.05 增加到 0.6，每次增量为 0.05。

我们在每个实例的初始化阶段，根据所给的数据资源随机构建多个 SG，并将 90% 的用户随机加入到不同的 SG 中，以便执行用户的退出/转移操作。每个模拟实例运行 100 个比特时间，用户以设定概率执行加入/退出/转移操作。

5.2 模拟结果与分析

图 3 是在模拟实例 A 的条件下，3 种方案 KDC 密钥更新开销的对比情况。由图 3 可知，OFGKMP 在密钥更新数量上少于 MGKMS，而 MTGKM 优于 OFGKMP，且 SG 子树为 4 叉树时比 3 叉树更为高效。这是由于当用户数量一定时，SG 子树的度越大，其高度就越低，用户变动时需要更新的密钥数量也就越少。

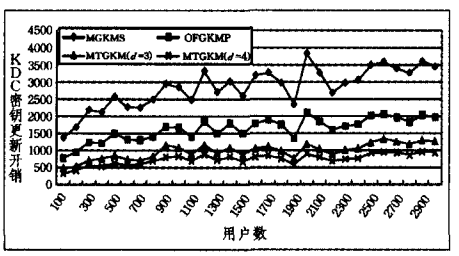

图 3 不同用户数量情况下 KDC 密钥更新的开销

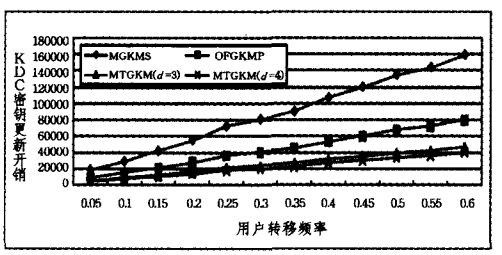

图 4 不同用户转移频率的情况下 KDC 密钥更新的开销

图 4 是在模拟实例 B 的情况下，3 种方案 KDC 密钥更新开销的对比。由于在 3 种方案中，用户的转移操作均可分解为先退出后加入，因此用户的转移频率会显著影响 KDC 的开销。由图 4 可知，三者的性能表现在宏观上与图 3 相似。且由于一次转移操作由一次退出与加入操作构成，因此当转移频率增大时，KDC 密钥更新的开销比图 3 增长得更快，MTGKM 方案也更具优势。

结束语 密钥管理是保障组播安全的核心技术之一。本文针对多权限群组通信，提出了一种基于多叉密钥树结构、结合单向函数和密钥更新素材的群组密钥管理方案。该方案通过降低 SG 子树的高度来减少需要维护的密钥数量，而且使用户可通过来自于 KDC 的密钥更新素材自行更新密钥，从而降低 KDC 更新密钥的开销。理论分析与模拟结果显示，较 MGKMS 及 OFGKMP 而言，本方案在保证安全性的同时，有效地降低了密钥更新的开销。然而，MTGKM 并不支持群组中数据资源的加入或退出，所以，下一步需考虑提出动态构建 DG 密钥子图的有效方法。

参 考 文 献

[1] Rafaeli S, Hutchison D. A Survey of Key Management for Secure Group Communication [J]. ACM Computing Surveys, 2003, 35(3): 309-329

[2] Trappe W, Song J, Poovendran R, et al. Key distribution for secure multimedia multicasts via data embedding[C]// Proceedings of the Acoustics, Speech, and Signal Processing. Salt Lake City: IEEE Signal Processing Society, 2001: 1449-1452

[3] Wong C K, Gouda M, Lam S. Secure Group Communications Using Key Graphs[J]. IEEE/ACM Transactions on Networking, 2000, 8(1): 16-30

[4] 许建真，董永先，梁克会. 一种高效的动态组播密钥管理方案[J]. 计算机应用研究，2010，27(3)：1061-1063

[5] 张志军，郭渊博，刘伟，等. 基于 m 叉树与 DH 协议的组密钥协商协议[J]. 计算机工程，2010，36(1)：161-163

[6] Sun Y, Liu K J R. Scalable hierarchical access control in secure group communications[J]. Proceedings of IEEE INFOCOM'04, 2004, 2(7-11): 1296-1306

[7] 欧阳洁，王国军. 基于单向函数的多权限群组密钥管理协议[J]. 计算机工程，2008，34(8)：156-158

[8] Wang G, Ouyang J, Chen H, et al. Efficient group key management for multi-privileged groups [J]. Computer Communications, Elsevier, 2007, 30(11/12): 2497-2509

[9] Eskicioglu A M, Dexter S, Delp E J. Protection of Multicast Scalable Video by Secret Sharing: Simulation Results[C]// Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents. Santa Clara, USA: the International Society for Optical Engineering, 2003, 7: 505-515

[10] Lin J C, Lai P F, Lee H C. Efficient group key management protocol with one-way key derivation[C] // Proceedings of IEEE conference on Local Computer Networks 30th Anniversary. 2005: 336-343

[11] Muthulakshmi A, Anitha R, Sumathi M. Non-split balancing higher order tree for multi-privileged groups[J]. WSEAS Transactions on Communications, 2011, 10(10): 308-321

[12] Muthulakshmi A, Anitha R. Balanced key tree management for multi-privileged groups using (N, T) policy[J]. Security and Communication Networks, 2012, 5(5): 545-555

[13] Cruz J R P, Hernandez S E P, Gomez G R, et al. Multi-session key management scheme for multimedia group communications [J]. Journal of Internet Technology, 2012, 1(1): 67-78