第41卷第7期 中国科学技术大学学报 Vol. 41, No. 7
2011年7月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jul. 2011

文章编号:0253-2778(2011)07-0582-07

基于STR非平衡树结构的混合型组密钥管理方案

麻常莎,薛开平,洪佩琳,丁 戎

(中国科学技术大学电子工程与信息科学系信息网络实验室,安徽合肥 230027)

摘要:STR组密钥管理协议因采用非平衡二叉树结构使得密钥更新过程简化,它可以有效降低组密钥管理的通信开销,但这是以高额的计算开销为代价的.提出一种基于STR非平衡密钥树结构的混合分层组密钥管理方案,在继承STR协议低通信开销的特点的同时,该方案显著降低了组密钥管理的计算开销并提高了密钥更新效率.

关键词:组密钥管理;密钥更新效率;计算开销;通信开销;安全

中图分类号:TP309.2 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2011.07.003

Hybrid group key management scheme based on STR unbalanced tree structure

MA Changsha, XUE Kaiping, HONG Peilin, DING Rong

(*Information Network Lab, Department Electronic Engineering and Information Science, University of Science and Technology of China, Hefei* 230027, *China*)

Abstract: STR group key management protocol can simplify key updating for its unbalanced binary-tree structure and decrease communication cost by increasing computing cost in group key management. A hybrid group key management scheme based on STR unbalanced tree was presented. It significantly decreases computation cost, and increases key updating efficiency, while maintaining the low-communication cost feature of STR protocol at the same time.

Key words: group key management; key updating efficiency; computing cost; communication cost; security

0 引言

许多组应用,如远程视频会议、远程协作系统和分布式仿真等,都需要进行安全通信,尤其是要保证组数据的安全性.而保障成员间协作机制的安全性是复杂的,组密钥管理是一种用于创建并维护这种协作机制的方法[1].组密钥管理策略主要分为集中式密钥管理、分布式密钥管理和混合型密钥管理.集中式密钥管理方案需要一个密钥分发中心,可以节省组成员的计算开销和通信开销,但是存在信任和单点失效问题,一旦密钥分发中心不工作了,组通信系统则无法运作;分布式密钥管理没有组管理者,组密钥由成员协作共同生成或者由一个成员生成,它通常涉及高通信开销和计算开销;混合型组密钥管理是二者的结合,通常将一个大的通信系统划分为若干小组,或者把一个中心服务器用若干个子组

收稿日期:2011-04-28;**修回日期**:2011-07-05

基金项目:国家自然基金(60903216),中国高技术研究发展(863)计划(2009AA012002),中央高校基本科研业务费专项资金资助.

作者简介:麻常莎,女,1988年生,硕士生. 研究方向:密钥管理. E-mail: shiyuanseu@126.com

通讯作者:薛开平,博士. E-mail: kpxue@ustc.edu.cn

agent 代替，这在一定程度上避免了单点失效，或者采用分层方案来提高完全分布式密钥管理的效率[2]，可以高效应用在大规模的分布式环境中[3]. 分布式组密钥管理方案基本基于 DH (Diffie-Hellman)密钥交换协议[4]. 其代表协议包括 GDH，STR，TGDH 等. STR，TGDH 协议都可保证较为高效的组成员间通信，而 STR 的非平衡树结构使得密钥的更新过程更为简单、密钥树结构的维护更容易. Kim 在文献[5]中提出了 STR 的改进协议，该协议具有比 TGDH 更小的通信量. STR 协议的这些优点使其高效应用在组密钥管理方案中，然而高计算开销和高延迟限制了 STR 协议的应用范围.

分层方案可以降低 STR 协议的计算开销，也可以减小大规模组场景下组密钥更新的延迟. 文献[6]将分层方案与 STR 协议结合，把 STR 分级树状结构运用在分布式子组管理模式中，降低了密钥管理产生的通信量和计算量，但是存在单点失效问题[7-8]，安全性得不到保障. 文献[9]提出基于双线性配对的三方密钥交换协议，可以显著降低 STR 协议的通信开销和计算开销，但是难以解决密钥树结构的维护问题.

本文提出一种基于 STR 的混合组密钥管理方案——H-STR(hybrid skinny tree). 该密钥树沿用 STR 协议的非平衡结构，分为主树子树，压缩了树高，较之 STR 协议提高了组密钥的更新效率；降低了节点活动对其他节点的影响，从而节省了节点活动造成的计算开销；同时具备 STR 协议通信开销低的优点，适用于节点活动频繁的大规模组通信环境. 在下文中，节 1 对 H-STR 密钥树结构和组建过程进行了描述；节 2 从密钥树组建角度对 H-STR 和 STR 的密钥更新效率进行对比，又从节点加入、节点退出、拆分组播组和合并组播组 4 个方面对比两个协议的通信开销和计算开销；节 3 进一步讨论了 H-STR 密钥树的最优结构并证明了协议的稳定性和安全性；节 4 对文章进行了总结.

1　H-STR 密钥树结构

1.1　协议设计思想

H-STR 协议是 STR 协议的扩展，采用分层结构，所有组成员被分为数个子组. H-STR 密钥树采用 STR 密钥树的非平衡结构，分为主树和子树，子树对应一个子组. 同 STR 协议一样，每个组成员拥有一对私钥和盲密钥，另外，相同子组的组成员共用一对子组私钥和盲密钥. 计算组密钥分 2 步：①组成员用自己的私钥和同子组其他成员的盲密钥通过 DH 密钥协商计算得到子组私钥；②组成员用子组私钥和其他子组的子组盲密钥通过 DH 密钥协商计算得到组密钥. 密钥树所有成员自下向上从左至右对应一个递增的序号，更新密钥树的同时更新序号. 以下是下文需要用到的符号说明.

N：组成员数量

m：子组数量

n：子组成员数量

C：当前组成员集合

BT：密钥树

M_i：第 i 个组成员

G_i：第 i 个子组

R_i：子组 G_i 的私钥

BR_i：子组 G_i 的公钥

r_i：组成员 i 的私钥

br_i：组成员 i 的公钥

K_i：中间节点 i 的私钥

BK_i：中间节点 i 的公钥

k_i：子组中间节点 i 的私钥

bk_i：子组中间节点 i 的公钥

k'：k 更新后的密钥

p：模数，为选定的大素数

α：模 p 的本原元

H-STR 密钥树有两类节点：中间节点(IN)和叶子节点(LN).

中间节点对应一对私钥和公钥，主树中用 K_i，$BK_i(i=1,2,\cdots,m)$表示，其中 $BK_i=\alpha^{K_i} \bmod p$，对应的子树中用 $k_{(i-1)n+j}$，$bk_{(i-1)n+j}(j=1,2,\cdots,n)$表示，其中 $bk_{(i-1)n+j}=\alpha^{k_{(i-1)n+j}} \bmod p$. k_{in}，bk_{in}代表子组私钥盲密钥，K_m 代表组密钥.

叶子节点对应一对私钥和盲密钥，主树叶子节点对应一个子组，用 R_i，$BR_i(i=1,2,\cdots,m)$表示，其中 $BR_i=\alpha^{R_i} \bmod p$，对应的子树中用 $r_{(i-1)n+j}$，$br_{(i-1)n+j}(j=1,2,3,\cdots,n)$表示，其中 $br_{(i-1)n+j}=\alpha^{r_{(i-1)n+j}} \bmod p$.

图 1 所示的是 N 成员的 H-STR 密钥树，分为 m 个子组，每个子组有 n 个成员，满足 $N=m\times n$，框图内是子组 G_i 的内部结构.

1.2　H-STR 密钥树的组建

假设组播组有 N 个成员，H-STR 协议将其划

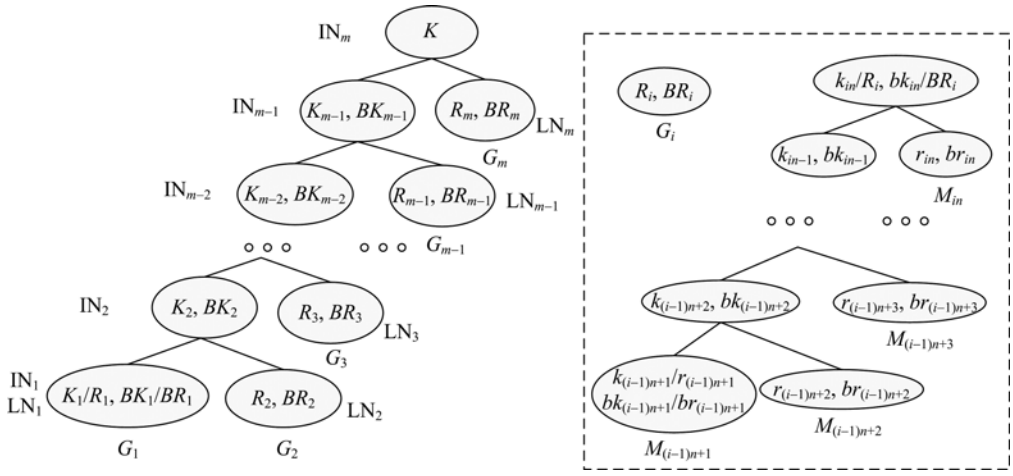

图 1 H-STR 示意图

Fig. 1 H-STR key tree

分为 $m=\lceil\sqrt{N}\rceil$个子组，前 $m-1$ 个子组 $G_1, G_2, \cdots, G_{m-1}$的成员数各为 $n=\lceil N/m\rceil$，G_m 的成员数为$n'=N-(m-1)\lceil N/m\rceil$.

首先构建各子组，以 G_1 为例：

①成员 $M_1, M_2, \cdots, M_n$ 生成随机数 $r_1, r_2, \cdots, r_n$ 作为私钥，计算 $br_1, br_2, \cdots, br_n$ 并在组内广播.

②M_1 计算子组密钥：

$$k_2 = (br_2)^{r_1} \bmod p = \alpha^{r_1 r_2} \bmod p, bk_2 = \alpha^{k_2} \bmod p;$$

$$k_3 = (br_3)^{k_2} \bmod p, bk_3 = \alpha^{k_3} \bmod p;$$

$$\cdots$$

$$k_n = (br_n)^{k_{n-1}} \bmod p, bk_n = \alpha^{k_n} \bmod p.$$

k_n, bk_n 即子组密钥和盲密钥.

③M_1 广播子树中间节点的公钥（包括子组盲密钥）$bk_i(1\leqslant i\leqslant n)$，子组其他成员 $M_j(j=2,3,\cdots,n)$计算子组密钥：

$$k_j = (bk_{j-1})^{r_j} \bmod p;$$

$$\cdots$$

$$k_n = (br_n)^{k_{n-1}} \bmod p.$$

子组创建完成后，创建组播组，过程如下：

①G_1, G_2 选择 M_1, M_{n+1} 作为创建组播组的承担者（以下表示为 sponsor）. M_1, M_{n+1}将子组盲密钥 BR_1, BR_2 发送给对方.

②M_1 计算组密钥：

$$K_2 = (BR_2)^{R_1} \bmod p = \alpha^{R_1 R_2} \bmod p,$$

$$BK_2 = \alpha^{K_2} \bmod p;$$

$$K_3 = (BR_3)^{K_2} \bmod p, BK_3 = \alpha^{K_3} \bmod p;$$

$$\cdots$$

$$K_m = (BR_m)^{K_{m-1}} \bmod p.$$

K_m 即为组密钥.

③M_1 广播主树中间节点的公钥 $BK_i(1\leqslant i\leqslant n-1)$，其他成员 $M_j(j=2,3,\cdots,N)$计算组密钥：

$$K_j = (BK_{j-1})^{R_j} \bmod p;$$

$$\cdots$$

$$K_m = (BR_m)^{K_{m-1}} \bmod p.$$

至此所有成员获知了组密钥，密钥树组建完成.

2 H-STR 协议与 STR 协议对比

2.1 密钥更新效率对比

组成员可同时计算盲密钥，子组可同时组建. 设组大小为 N；成员进行一次加密操作（大数幂模运算）的时间为 T；成员计算能力相当；忽略网络延迟等客观因素和随机数生成时间，组成员数量和密钥树结构是密钥更新效率的决定因素.

根据 STR 密钥树组建的过程[5]，我们首先考虑 STR 协议的密钥更新效率：

①组成员计算盲密钥，所需时间为 T；

②M_1 计算 $N-2$ 个中间节点的私钥和公钥，以及组密钥，计算时间为$(2N-3)T$，广播中间节点公钥；

③其他成员计算到组密钥的路径上的中间节点私钥以及组密钥，各自需要$(N-1)T$，$(N-2)T$，$\cdots$，T 时间，然而这个过程各成员可以同时进行，所以这个过程只需要额外花费$(N-2)T$ 的时间（M_2 计算 IN_2 的私钥的过程可以与 M_1 同时进行）.

因此，STR 协议组建一个新的组播组的时间为

$$T+(2N-3)T+(N-2)T=(3N-4)T.$$

而 H-STR 密钥树组建需要的时间为：

①构建子组，$(3n-4)T+T$；

②构建组播组，$(3m-4)T-T$.

只需$(3m+3n-8)T$.

$$(3m+3n-8)T \leqslant (3N-4)T,$$

“=”当且仅当 $m=1$，或 $n=1$ 时取得，此时 H-STR 协议退化为 STR 协议. 因此，H-STR 协议较之 STR 协议，密钥更新效率有所提高.

2.2　节点加入开销对比

节点加入时的组有 N 个成员，m 个子组，这并不一定会使 H-STR 协议获最优性能(后文会加以讨论). 分两种情况处理加入行为：① $m \geqslant \lceil \sqrt{N} \rceil$，新成员加入成员数最小位置最上的子组 $G_i(1 \leqslant i \leqslant m)$；② $m < \lceil \sqrt{N} \rceil$，主树高度加 1，新成员加入新增子组 G_{m+1}.

新成员 M_{N+1}广播加入请求和盲密钥 br_{N+1}. 收到消息后，成员协作更新密钥树、选定 sponsor.

第 1 种情况，G_i 子树增加一对中间节点 IN_{N+1} 和叶子节点 LN_{N+1}；G_i 的最上最右成员 M_i(sponsor)更新会话密钥，计算 br'_i，k'_i，bk'_i以及新子组密钥 k_{N+1}，bk_{N+1}，并将当前盲密钥树 BT(包括组成员数、分组数、分组大小和所有公钥)发送给 M_{N+1}和老成员；G_i 各成员(包括 M_{N+1})用 IN_{i-1}的公钥 BK_{i-1}和新的子组私钥 $R'_i(k_{N+1})$计算得到新的组密钥 K'_i；其他成员用 IN_{i-1}的私钥 K_{i-1}和 G_i 新的子组盲密钥 $BR'_i(bk_{N+1})$计算得到组密钥.

第 2 种情况下，主树增加一对中间节点 IN_{m+1} 和叶子节点 LN_{m+1}；G_m 最上最右的成员担任 sponsor，计算 br'_N、新的子组密钥 k'_N，bk'_N，和 K'_m，BK_m；G_m 中各成员用 $R'_m(k_{N+1})$和 BK_{m-1}计算并存储 IN_m 的私钥，再用 M_{N+1}的盲密钥 br_{N+1}(即 G_{m+1}的子组盲密钥)计算得到组密钥 K_{m+1}；M_{N+1}用自己的私钥(即 G_{m+1}的子组私钥)和 BK_m 直接计算得到组密钥；其他成员用 K_{m-1}和 $BR'_m(bk_{N+1})$逐层计算得到组密钥.

节点加入协议的步骤可概括表示为：

Step 1　新成员广播加入申请；

$$M_{N+1} \longrightarrow C=\{M_1, M_2, \cdots, M_N\}$$

Step 2　组成员协作更新密钥树，移除 M_i(sponsor)的盲密钥；M_i 更新会话密钥，计算相关中间节点私钥和公钥，广播更新后的树 BT；

$$C \cup \{M_{N+1}\} = \{M_1, M_2, \cdots, M_N, M_{N+1}\} \xleftarrow{BT} M_N$$

Step 3　成员通过新的密钥树 BT 计算新的组密钥(k', bk')，节点加入操作完成.

图 2 表示 M_{17} 加入组，M_{16}更新会话密钥 r_{16}以保证前向安全性.

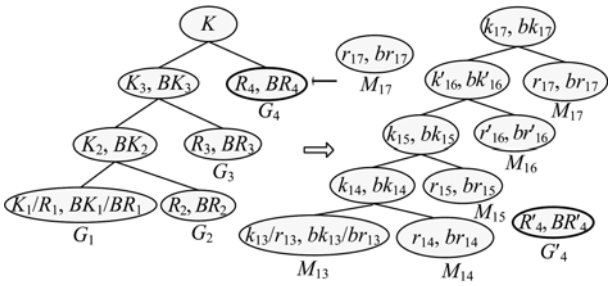

图 2　H-STR 节点加入子协议示意图

Fig. 2　Tree update in JOIN

H-STR 节点加入需两轮通信，与 STR 通信开销一致. 计算开销方面，若节点加入导致主树高度变化，则所需加密次数为 6；否则，加密次数与节点加入位置相关，可表示为 $4+2(m-i)$，其中 m 是子组数，i 是节点加入的子组序号. H-STR 协议中，节点加入协议平均计算开销的期望值为

$$\sum_{i=1}^{m} 4+2(m-i) = 3+m = 3+\sqrt{N}.$$

H-STR 协议相比 STR 协议没有增加通信开销，而计算开销的非显著增加可以在其他子协议中得到补偿.

2.3　节点退出开销对比

假设一个如节 2.2 描述的组播组$\{M_1, \cdots, M_N\}$，G_i 的成员 $M_d((n-1)i+1 \leqslant d \leqslant ni)$在某一时刻离开该组，则该成员所在子树对应的叶子节点和中间节点被删除，M_{d-1}担任 sponsor，表示为 M_s(若 $d=(n-1)i+1$，M_s 为 M_{d+1})；若 M_d 是 G_i 最后一个成员，则子组以及该子组对应的主树中间节点被删除，G_{i-1}最上最右成员担任 sponsor.

M_s 更新会话密钥，计算盲密钥以及子树相关路径上所有中间节点的私钥和公钥；并用新的子组私钥和中间节点 $IN_{i-1}(IN_{i-2})$的公钥逐层向上计算 $IN_i(IN_{i-1})$到根节点主树相关路径上所有中间节点的私钥、公钥以及新组密钥；广播新密钥树 BT 给所有其他组成员. 组成员可逐层向上计算组密钥.

节点退出协议的步骤可概括表示为：

Step 1　成员协作更新密钥树，移除相关子树和主树中间节点的私钥和公钥；M_s 更新会话密钥，计算相关中间节点私钥公钥，广播新的密钥树；

$$C-\{M_d\} \xleftarrow{\text{BT}} M_s$$

Step 2 成员通过新密钥树 BT 计算新组密钥.

图 3 表示 M_{15} 离开组播组,为保证后向安全性,M_{14} 更新会话密钥 r_{14},计算 br'_{14},k'_{14},bk'_{14},k'_{16},bk'_{16},广播 BT 给其他组成员,所有成员(包括 M_{14})计算组密钥 K'. M_{15} 则无法计算组密钥,因为它的会话密钥已经失效.

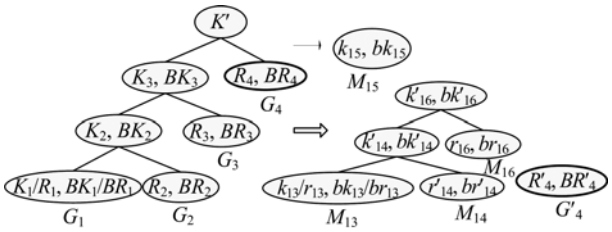

图 3 H-STR 节点退出子协议示意图

Fig. 3 Tree update in LEAVE

与 STR 协议相同,H-STR 节点退出子协议需要进行一次广播通信,通信开销相同. 二者的计算开销皆与离开节点的位置有关.

根据文献[10],STR 节点退出子协议的加密次数为:

①当 $d>2$ 时,$2(N-d)+1+(N-d)+1=3N-3d+2$;

②当 $d=1,2$ 时,$3N-7$.

平均计算开销为 $3(N/2)+2$.

H-STR 节点退出子协议的加密次数为:

①当 $i>1, d>(i-1)n+2$ 时,

$$3m-3i+2+3n-3d+2=3(m+n)-3(i+d)+4;$$

②当 $i>1, d=(i-1)n+1, (i-1)n+2$ 时,

$$3m-3i+2+3n-7=3(m+n)-3i-5;$$

③当 $i=1, d>(i-1)n+2$ 时,

$$3m-4+3n-3d+2=3(m+n)-3d-2;$$

④当 $i=1, d=(i-1)n+1, (i-1)n+2$ 时,

$$3m-4+3n-7=3(m+n)-11.$$

平均计算开销为 $3((m+n)/2)+4$.

而 $m+n\leqslant N$,"="当且仅当 $m=1$,或 $n=1$ 时取得,此时 H-STR 协议退化为 STR 协议.

图 4 表示 STR 和 H-STR($m=n=\sqrt{N}$)节点退出计算开销(用幂模运算次数的自然对数值衡量)与组成员数量的关系,H-STR 的计算开销随组成员数量增加的变化缓于且远远小于 STR 协议.

2.4 拆分组播组开销对比

网络错误或拥堵会引起组播组拆分,相当于多个成员同时退出组. 与节点退出协议唯一不同的是 sponsor 的选择. 拆分情况下 M_s(不止一个)是各子树最下离开成员 $M_{d_x}((n-1)i+1\leqslant d_x\leqslant ni)$ 的相邻下层叶子节点. 若 $d_x>(n-1)i+1$,为 M_{d_x-1};若 $d_x=(n-1)i+1$,为 M_{d_x+1}.

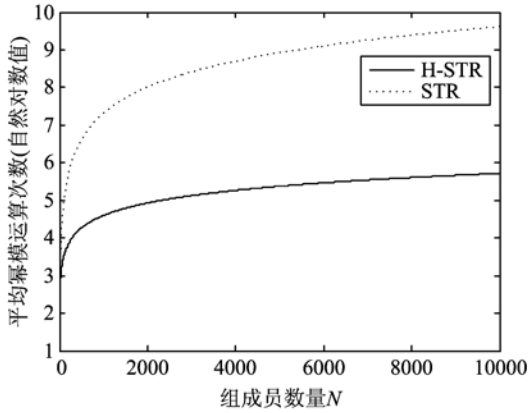

图 4 H-STR 与 STR 节点退出子协议计算开销对比

Fig. 4 Computing cost of LEAVE in H-STR and STR

删除所有离开节点后,各 M_s 更新会话密钥,计算子树相关中间节点的私钥和公钥,广播公钥给其他成员;下标最小的 M_s 计算主树相关中间节点的私钥和公钥,广播公钥给其他成员.

图 5 表示 M_4 和 M_{15} 退出组播组的情况. M_3 和 M_{14} 皆为 M_s,二者更新会话密钥,并各自负责 G_1 和 G_4 的子树更新; M_3 计算 K'_2, BK'_2, K'_3, BK'_3; M_3 广播新的密钥树、主树中间节点公钥以及子组 G_1 中间节点的公钥,M_{14} 广播子组 G_4 中间节点公钥.

计算开销与节点退出子协议相同,低于 STR 协议. 通信开销与拆分牵涉的子组数相关,通信轮数等于相关子组数,与 STR 协议的通信开销相等.

2.5 合并组播组开销对比

网络恢复正常时,多个小型组播组需重新合并成一个大的组播组. 假设 m 个小组($G_1, G_2, \cdots, G_m$,成员数量从大到小排列)需合并. *H-STR* 将它们看作子组,G_1 在最底层,G_m 在最高层. 选择 $G_i(1\leqslant i\leqslant m)$ 最上层成员 M_s^i 为 sponsor. M_s^i 更新自己的会话密钥,计算 G_i 的子组私钥和盲密钥,广播子组盲密钥给所有其他成员. $M_s^1, M_s^2, \cdots, M_s^m$ 按照节 1.2 描述的步骤组建大组播组.

图 6 表示 3 个大小为 4,4,3 的小型组播组合并为一个大组播组,它们分别被看作子组 G_1, G_2, G_3; M_4, M_8, M_{11} 分别为 3 个小组的 sponsor. M_4, M_8, M_{11} 更新会话密钥,计算 G_1, G_2, G_3 子组私钥盲密

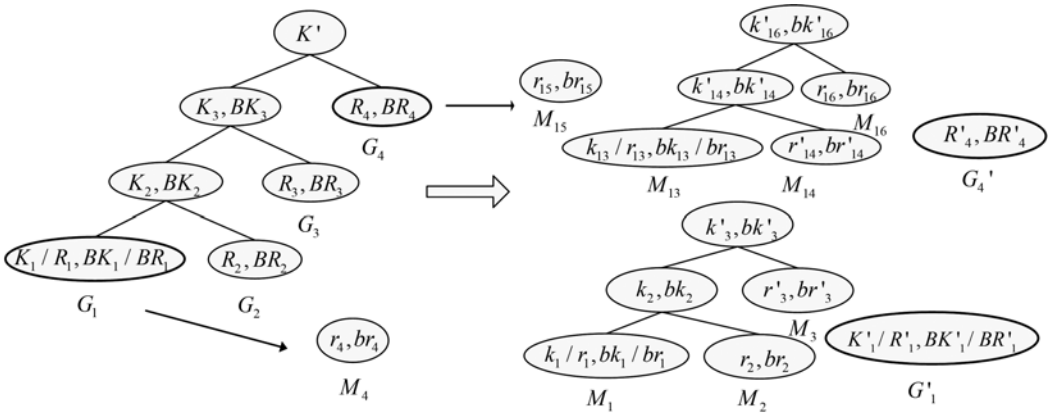

图 5　拆分组播组子协议示意图

Fig. 5　Tree update in PARTITION

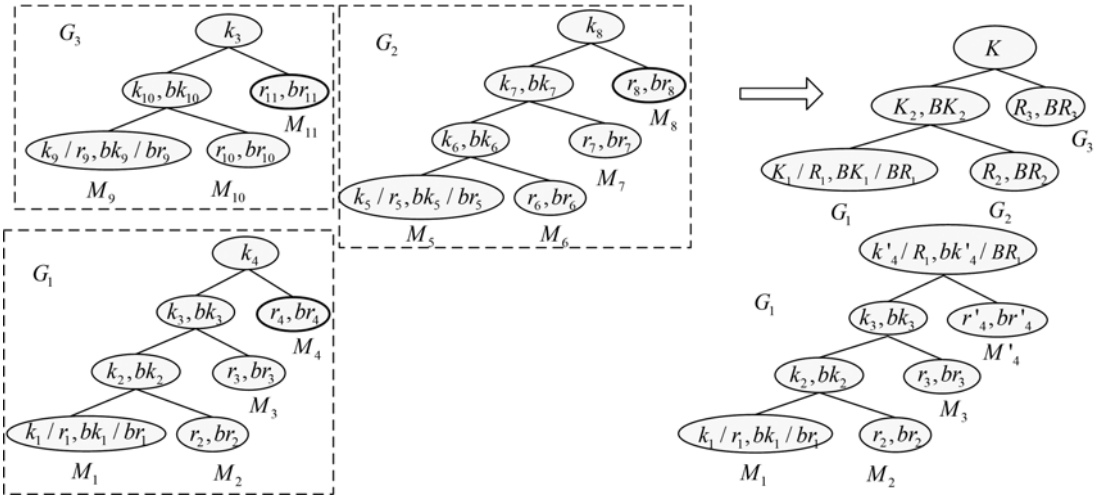

图 6　合并组播组子协议示意图

Fig. 6　Tree update in MERGE

钥，并广播各自的子组盲密钥树；M_4 计算 K_2，BK_2，K，并广播 BK_2 和新的密钥树给其他成员.

H-STR 协议合并 m 个小组需 2 轮通信，发送 $m+1$ 个消息，通信开销与 STR 一致. 而 STR 协议计算开销与合并组成员总数成正比，H-STR 只与合并组数目成正比，大大减少了计算开销.

2.6　小结

表 1 对 STR，H-STR 协议通信开销和计算开销进行了对比总结，其中在节点退出和分割的情况下，计算开销指组播组进行幂模运算次数取平均值. 可以看出，H-STR 协议通信开销与 STR 协议相当，而计算开销大大降低. 在组规模庞大、节点活动频繁的环境中，如 ad hoc 网络，H-STR 协议的优势将非常显著.

表 1　STR 协议与 H-STR 协议开销对比

Tab. 1　Cost in STR and H-STR

协议	子协议	通信开销		计算开销
		通信轮数	通信次数	幂模运算
STR	节点加入	2	2	4
	节点退出	1	1	$(3N)/2+2$
	分割	1	1	$(3N)/2+2$
	合并	2	$m+1$	$3N+1$
H-STR	节点加入	2	2	$3+\sqrt{N}$
	节点退出	1	1	$(3(m+n))/2+4$
	分割	1	1	$(3(m+n))/2+4$
	合并	2	$m+1$	$3m+1$

3　对 H-STR 协议的进一步讨论

3.1　H-STR 密钥树最优结构

根据节 2 的描述，组建一个新的 H-STR 组播

组的时间为 $3(m+n-2)T$，节点退出平均计算开销为 $3((m+n)/2)+4$，即最小的 $m+n$ 值对应最优密钥树结构. 而 $N=m\times n$，$m+n=m+N/m\geqslant\sqrt{N}$，子组数 m 越接近 $\sqrt{N}$，$m+n$ 越小，对应密钥树结构最优. 因此在组建 *H-STR* 密钥树时，取值 m 为 $\lceil\sqrt{N}\rceil$.

图 7 表示 100 成员的组播组节点退出计算开销与子组数量的关系，子组数为 10 时对应最小计算开销，与上述分析一致.

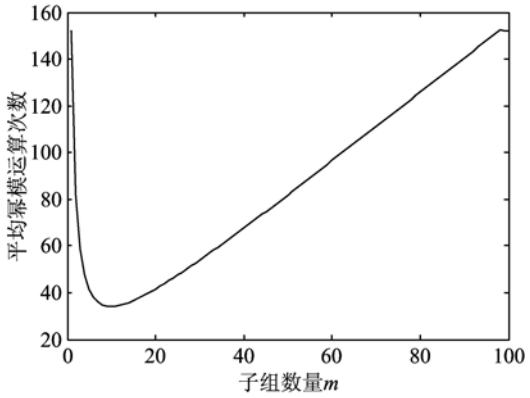

图 7 节点退出的计算开销与子组数量关系曲线

Fig. 7 Relationship between Subgroup numbers and computing cost of LEAVE

3.2 H-STR 密钥树的稳定性

节点频繁活动势必导致密钥树结构变化，但不会导致 H-STR 协议性能下降，我们给出如下证明：

假设一段时间后组播组大小为 N，子组数目为 m，子组大小不再一致，设成员数为 n_i，其中 i 对应子组序号. 节点加入、子组合并子协议的计算通信开销皆不受此变化的影响，而节点退出、拆分组播组子协议的平均计算开销可如下计算：

$$\sum_{i=1}^{m}\frac{3m-3i+2+3(n_i/2)+2}{m}=$$

$$\sum_{i=1}^{m}\frac{3m-3i}{m}+\frac{\frac{3}{2}\sum_{i=1}^{m}n_i}{m}+4=$$

$$\frac{3}{2}\left(m+\frac{N}{m}\right)+4$$

节点加入子协议保证了 m 可取得最优值，因此 H-STR 协议可以稳定保持最优结构.

3.3 H-STR 协议安全性

H-STR 协议继承了 STR 协议的安全性[5]，可以保证组密钥的秘密性[11]、前向安全性和后向安全性. 下面将对此一一进行分析.

（Ⅰ）秘密性

组成员关系变动时，至少一个组成员变更其会话密钥，使子组密钥和组密钥改变，攻击者在获知所有盲密钥的情况下也无法计算得到新的子组密钥和组密钥，因此保证了组密钥的秘密性.

（Ⅱ）前向安全性

前向安全性指限制退出的用户使用旧的密钥解密它退出之后的通信数据. 在节点退出子协议和拆分组播组子协议中，M_s（sponsor）在组成员退出组后更新会话私钥和盲密钥，更新所有相关中间节点的私钥和公钥以及组密钥，使得离开的组成员无法获知密钥路径上的私钥，从而保证了前向安全性.

（Ⅲ）后向安全性

后向安全性指防止新加入的用户解密它加入之前的通信数据. 在节点加入子协议中，新成员加入组播组，M_s 更新会话私钥和盲密钥，同时更新相关中间节点的私钥和盲密钥，广播盲密钥树，新加入成员和被动攻击者知道的信息是一样的，它们都无法获知旧的组密钥，从而实现了后向安全性.

4 结论

本文提出了一种基于 STR 逻辑密钥树的混合分层组密钥管理方案——H-STR. 该协议适用于节点活动频繁的较大规模组播组，较之 STR 协议降低了节点加入退出行为的计算开销、提高了密钥更新效率，是一个高效安全的密钥管理协议.

参考文献（References）

[1] Jiang Bibo, Hu Xiulin. A survey of group key management [C]//Proceedings of 2008 International Conference on Computer Science and Software Engineering. Piscataway, NJ, USA: IEEE, 2008: 994-1 002.

[2] Li Shuquan, Wu Yue, Zhu Dayong, et al. A hierarchical key management scheme for large and dynamic multicast groups [C]//Proceedings of 2009 International Conference on Apperceiving Computing and Intelligence Analysis. Piscataway, NJ, USA: IEEE, 2009: 270-273.

[3] Rong B, Chen H H, Qian Y, et al. A pyramidal security model for large-scale group-oriented computing in mobile ad hoc networks: The key management study [J]. IEEE Transactions on Vehicular Technology, 2009, 58(1): 398-408.

（下转第 635 页）