Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Optimal tree structures for group key tree management considering insertion and deletion cost

Weiwei Wu[a,b,c], Minming Li[b,*], Enhong Chen[a]

[a] Department of Computer Science, University of Science and Technology of China, China
[b] Department of Computer Science, City University of Hong Kong, Hong Kong
[c] USTC-CityU Joint Research Institute, China

ARTICLE INFO

Article history:
Received 22 October 2008
Received in revised form 23 February 2009
Accepted 27 February 2009
Communicated by D.-Z. Du

Keywords:
Key trees
Group keys
Optimality
Updating cost

ABSTRACT

We study the optimal structure for the group broadcast problem where the key tree model is extensively used. The objective is usually to find an optimal key tree to minimize the cost based on certain assumptions. Under the assumption that n members arrive in the initial setup period and only member deletions are allowed after that period, previous works show that when only considering the deletion cost, the optimal tree can be computed in $O(n^2)$ time. In this paper, we first prove a semi-balance property for the optimal tree and use it to reduce the running time from $O(n^2)$ to $O(\log\log n)$ multiplications of $O(\log n)$-*bit* integers. Then we study the optimal tree structure when insertion cost is also considered. We show that the optimal tree is such a tree where any internal node has degree at most 7 and children of nodes with degree not equal to 2 or 3 are all leaves. Based on this result we give a dynamic programming algorithm with $O(n^2)$ time to compute the optimal tree.
© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Many recent works have researched the group broadcast problem due to its cost effectiveness in the applications requiring content security. The applications based on multicast can be divided into two types, one-to-many (e.g., television broadcast and pay per view) and many-to-many (e.g., teleconference, collaborative work and distributed interactive games). All of them require content security which means only authorized users are allowed to access the data broadcasted. Moreover, to deliver messages on insecure channels, we should guarantee confidentiality when users dynamically change in the group with the help of cryptography. Two kinds of confidentiality are usually considered in the literature: future confidentiality (to prevent users deleted from the group from accessing any future keys which will be used to encrypt data) and past confidentiality (to prevent users newly added into the group from accessing any past keys used to encrypt data).

To satisfy these security requirements, the basic strategy is to update the group key whenever a user is deleted from or added into the group. The group controller (GC) will maintain a key structure for the whole group. A recent survey on key management for secure group communications can be found in [2]. However, since encryption is quite time consuming, the critical problem is how to decrease the number of encryptions when group members dynamically change. The key tree model proposed by Wong et al. [7] is widely used for the key management problem. In this model, it is assumed that a leaf node represents a user and stores his individual key and an internal node stores a key shared by all leaf descendants of that node. The above two assumptions imply that every user possesses all the keys along the path from the leaf node to the root. Whenever a new user is added or deleted, the GC will update the keys along the path in a bottom–up fashion and notify the

* Corresponding author. Tel.: +852 27889538; fax: +852 27888614.
E-mail addresses: wweiwei2@cityu.edu.hk (W. Wu), minmli@cs.cityu.edu.hk (M. Li), cheneh@ustc.edu.cn (E. Chen).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.02.039

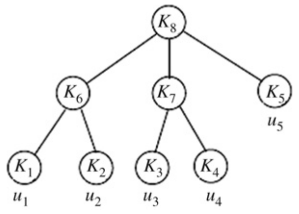

Fig. 1. An example key tree structure for a group with 5 members.

subset of nodes who share the keys. Wong et al. [7] pointed out that the average measured processing time increases linearly with the logarithm of the group size. Soneyink et al. [5] proved any distribution scheme has a worst-case cost of $\Omega(\log n)$ either for adding or for deleting a user. They also proved that the updating cost of a key tree with n insertions followed by n deletions is $\Theta(n \log n)$. Feng et al. [9] studied the structure of the optimal tree when a deletion sequence is performed. They showed that the optimal tree is a tree where any internal node has degree at most 5 and children of nodes with degree not equal to 3 are all leaves. Based on this observation, they designed a dynamic programming algorithm of $O(n^2)$ time to compute an optimal tree. Another scenario where rekeying is done only periodically instead of immediately is studied as a batch rekeying strategy in [4]. This strategy is further investigated in [8,1,3] under the assumption that each user has a fixed probability p of being replaced during the batch period.

We further investigate the scenario proposed in [9]. Firstly, we find a semi-balance property of the optimal tree when only deletion cost is considered and use it to improve the time for computing the optimal tree from $O(n^2)$ to $O(\log \log n)$ multiplications of $O(\log n)$-*bit* integers. Secondly, we study the optimal tree structure when insertion and deletion cost are simultaneously taken into consideration. Suppose in the initial setup period, the group only accepts membership joins. At the end of this period, a certain key tree is established by multicasting certain encryptions to the users. After that period, the group closes new membership and only accepts membership revocations. Notice that the GC update keys only at the end of the initial setup period and whenever a user leaves afterwards. This is different from the scenario of n insertions followed by n deletions considered in [5] where each insertion triggers an update on the key tree. We show that when considering both key tree establishment cost and deletion cost, the optimal tree is such a tree where any internal node has degree at most 7 and children of nodes with degree not equal to 2 or 3 are all leaves.

The rest of this paper is organized as follows. In Section 2 we review the definition of the key tree model and introduce some related results. In Section 3, we prove a semi-balance property of the optimal tree for n deletions and reduce the computation time of the optimal tree from $O(n^2)$ to $O(\log \log n)$ multiplications of $O(\log n)$-*bit* integers. In Section 4 we investigate a more general cost definition where the cost of the key tree establishment is also included. We prove the degree bound for the optimal tree in this scenario and propose an $O(n^2)$ dynamic programming algorithm to compute the optimal tree. In Section 5, we further derive some balance property for the optimal key tree with establishment cost included. Finally, we conclude our work in Section 6.

2. Preliminaries

In this section, we review the key tree model which is referred to in the literature either as key tree [7] or LKH (logical key hierarchy) [6].

In the key tree model, a group controller (GC) maintains the key structure for all the users. A group member holds a key if and only if the key is stored in an ancestor of the member. When a user leaves, GC will update any key that is known by him in a bottom–up fashion, and notify the remaining users who share that key. Take the structure shown in Fig. 1 as an example, the user u_1 holds keys (k_1, k_6, k_8), and u_2 holds keys (k_2, k_6, k_8). When u_1 leaves, keys k_6 and k_8 need to be updated. GC will first encrypt the new k_6 with k_2 and multicast the message to the group. Notice that only u_2 can decrypt the message. Then GC encrypts the new k_8 with k_5, k_7 and with the new k_6 separately and multicast them to the group. Notice that all users except u_1 can obtain the new k_8 by decrypting one of those messages. Hence, the GC needs 4 encryptions to maintain the key tree structure when u_1 leaves. Based on the updating rule, we introduce the definition of deletion cost.

Definition 1. In a key tree T with n leaves, we say a node v has degree d_v if v has d_v children. We denote the set of ancestors of v as $anc(v)$ (not including v itself) and define the ancestor weight of v as $w_v = \sum_{u \in anc(v)} d_u$. The number of leaf descendants of v is denoted as n_v.

Given a leaf v_i in a key tree T, let $v_i u_1 u_2 \ldots u_k$ be the longest path in T where u_j has only one child for $1 \le j \le k$. We define k as the *exclusive length* of v_i. Notice that when the user v_i is deleted from the group, we need not update any key on the path $v_i u_1 u_2 \ldots u_k$. Hence, we have the following deletion cost (defined in terms of the number of encryptions needed to update the keys after deletions). If not specified otherwise, we abbreviate w_{v_i} and n_{v_i} as w_i and n_i respectively.

Definition 2. The deletion cost of v_i is $w_i - k - 1$ where k is the exclusive length of v_i, and we denote this deletion cost as c_i.

Notice that when nodes are deleted, different deletion orders may incur different deletion cost. In the tree shown in Fig. 1, deletion sequence u_1, u_2, u_3, u_4, u_5 has cost $4 + 2 + 3 + 1 + 0 = 10$, while deletion sequence u_1, u_3, u_2, u_4, u_5 has cost $4 + 4 + 2 + 1 + 0 = 11$.

In our work, we further investigate the scenario where a group only accepts membership joins during the initial setup period. After that period, the only dynamic membership changes are deletions. This requires us to focus on the cost of a sequence instead of a single leaf. We first cite the following definitions from [9].

Definition 3. In a key tree T with n leaf nodes, we define $\pi = \langle v_1, v_2, \ldots, v_n\rangle$ as the sequence of all nodes to be deleted in T. Let $\langle c_1, c_2, \ldots, c_n\rangle$ be the resulting sequence of deletion cost when the deletion sequence was performed. Let $C(T,\pi) = \sum_{i=1}^{n} c_i$ denote the deletion cost of the whole tree T under the deletion sequence π. The worst-case deletion cost of the tree T is denoted as $C_{T,\,deletion} = \max_{\pi} C(T,\pi)$. We define the optimal tree $T_{n,opt}$ as a tree (not necessarily unique) which has the minimum worst-case deletion cost over any tree T containing n leaf nodes.

Definition 4. Let T be a tree with n leaves. Given a tree T' with r leaves, we call T' a **skeleton** of T if T can be obtained by replacing r leaf nodes $v_1, v_2, \ldots, v_r$ of T' with r trees $T_1, T_2, \ldots, T_r$, where T_i has root v_i for $1 \le i \le r$.

Under this definition, they proved a recursive formula for the worst-case deletion cost $C_{T,\,deletion}$. Let T' be a skeleton of T as defined above. Given a deletion sequence π' for T' as well as a deletion sequence π_i for each T_i $(1 \le i \le r)$, they derive a deletion sequence π for T as follows. In the first step, π deletes all leaves in subtree T_i in the order specified by π_i, until there is only 1 leaf left. In the second step, π deletes the only remaining leaf of each T_i in the order specified by π'. They denote the deletion sequence for T derived this way by $\pi = \langle \pi_1, \ldots, \pi_r, \pi'\rangle$.

Lemma 1 ([9]). *The sequence $\pi = \langle \pi_1, \pi_2, \ldots, \pi_r, \pi'\rangle$ is a worst-case deletion sequence for T if π_i is a worst-case deletion sequence for T_i and π' is a worst-case deletion sequence for T'. The worst-case deletion cost for T is*

$$C_{T,\,deletion} = C_{T',\,deletion} + \sum_{i=1}^{r}(C_{T_i,\,deletion} + (n_i - 1)w_i). \tag{1}$$

In this formula, $C_{T',\,deletion}$ is the worst-case deletion cost for the skeleton T', and $C_{T_i,\,deletion}$ is the worst-case deletion cost for the subtree T_i. The values n_i and w_i are the abbreviations of n_{v_i} and w_{v_i} respectively.

3. Semi-balance property of key tree structure for *n* deletions

As [9] shows, to minimize the worst-case deletion cost when all n subscribers are deleted from the group one by one, an optimal tree can be found among the trees satisfying the following two conditions: (1) every internal node has degree $d \leqslant 5$ and (2) children of nodes with degree $d \neq 3$ are all leaves. According to this, they gave an algorithm to compute the optimal tree in $O(n^2)$ time. In this section, we prove a semi-balance property of the optimal tree.

According to the result of [9], if a node v has at least one child being an internal node, it must have degree 3. We further prove the following lemma.

Lemma 2. *There is an optimal tree where the children of any degree 3 node are either all leaves or all internal nodes.*

Proof. Suppose on the contrary in the optimal tree there exists an internal node v with degree 3, which has an internal node child v_1 and a leaf child v_0, then v_1 has at most two leaf descendants. Otherwise, if v_1 has $n_1 \geqslant 3$ leaf descendants, we show in the following that the cost of the tree can be decreased, which contradicts the optimality of the tree. Obviously, v_1 can only have degree $2 \leqslant d_1 \leqslant 5$. Firstly, when $d_1 \geqslant 3$, we can decrease the cost by moving a subtree T_2 (rooted at v_2) of v_1 and combine it with the leaf child (v_0) of v, as shown in Fig. 2. We use symbols $\widehat{T}$ to denote the tree after subtree T_2 of v_1 is deleted from T, and $\widetilde{T}$ to denote the tree after T_2 is combined with v_0. The cost reduced by deleting T_2 is

$$C_{T,\,deletion} - C_{\widehat{T},\,deletion} = C_{T_2,deletion} + (n_2 - 1)(d_1 + w_1) + d_1 - 1 + w_1.$$

Here w_1 is the ancestor weight of v_1. Deleting any one of the $(n_2 - 1)$ leaves in T_2 needs extra cost $d_1 + w_1$, and deleting the last leaf in T_2 needs extra cost $d_1 - 1 + w_1$. Similarly, we have the cost increased by adding T_2 into $\widetilde{T}$ (combining with v_0).

$$C_{\widetilde{T},deletion} - C_{\widehat{T},\,deletion} = C_{T_2,deletion} + (n_2 - 1)(2 + w_0) + 2 - 1 + w_0.$$

Here, w_0 is v_0's ancestor weight. Since v_1, v_0 are v's children, we have $w_1 = w_0$ and the deletion cost decreases by $(n_2 - 1)(d_1 - 2) + d_1 - 2 \geqslant 1$ from T to $\widetilde{T}$. On the other hand, when $d_1 = 2$, children of v_1 are all leaves because $d_1 \neq 3$, i.e. v_1 has at most two leaf descendants. Therefore, v has $n_v \leqslant 5$ leaf descendants, and in this case the optimal tree is such a tree where all v's children are leaves. $\square$

As a result, if a node has at least one leaf child, its children are all leaves. Furthermore, to make our proof easy to read, we will give this kind of node a new name in the following.

Definition 5. Given a tree T with L levels, we define the pseudo-leaf nodes to be the nodes whose children are all leaves. In addition, we use l_i to denote the level where the pseudo-leaf node u_i is placed and s_i to denote the number of its children.

Obviously, in an optimal tree all pseudo-leaf nodes can only be placed on level $1 \leqslant l_i \leqslant L - 1$ and have $2 \leqslant s_i \leqslant 5$ children.

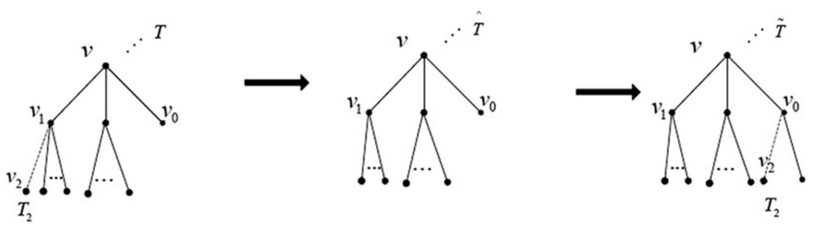

Fig. 2. A degree 3 node with at least one leaf child and one internal node child.

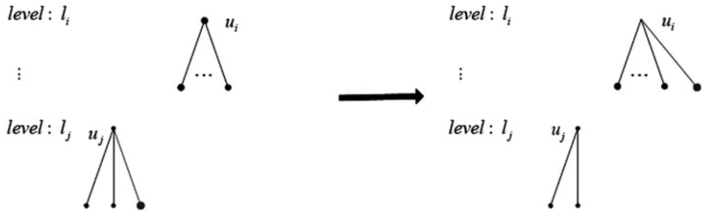

Fig. 3. Transformation of tree T where at least one pseudo-leaf node u_i with $s_i \leqslant 4$ children is on level l_i where $1 \leqslant l_i \leqslant (L-2)$.

Lemma 3. *Given a tree T and a pseudo-leaf u_i in T, if we remove one child of u_i, the cost of the resulting tree $\bar{T}$ decreases by $3l_i + s_i - 1$.*

Proof. The lemma can be proved by choosing all the nodes in T except the children of u_i as the skeleton. By the definition of $C_{T,\,deletion}$, we have

$$C_{T,\,deletion} = C_{T',\,deletion} + w_{u_i}(s_i - 1) + \sum_{1}^{s_i-1} j$$

$$C_{\bar{T},\,deletion} = C_{T',\,deletion} + w_{u_i}(s_i - 1 - 1) + \sum_{1}^{s_i-2} j.$$

Notice that w_{u_i} has the same value in T and $\bar{T}$ because we are choosing the same structure as the skeletons. Since u_i has l_i ancestor nodes and each ancestor node of u_i has 3 children, we have $w_{u_i} = 3l_i$. Therefore, $C_{T,\,deletion}$ is larger than $C_{\bar{T},\,deletion}$ by $3l_i + s_i - 1$. □

Lemma 4. *In an optimal tree $T_{n,opt}$, if a pseudo-leaf node u_i satisfies $1 \leqslant l_i \leqslant L - 2$, then we have $s_i = 5$.*

Proof. Suppose another pseudo-leaf node u_j is on level $l_j = L - 1$. We can get a better tree by moving a child of u_j to be a child of u_i when $s_i \leqslant 4$, as shown by Fig. 3. To make our proof consistent, we use symbols $\widehat{T}$ to denote the tree after a leaf of u_j is deleted from T, and $\widetilde{T}$ to denote the tree after that leaf is added to be a child of u_i. Since u_i and u_j are on levels l_i and l_j respectively and $2 \leqslant s_j \leqslant 5$, the deletion cost is decreased by $3l_j + s_j - 1$ from T to $\widehat{T}$ and increased by $3l_i + s_i$ from $\widehat{T}$ to $\widetilde{T}$ by Lemma 3. Moreover, because $l_j - l_i \geqslant 1$ and $s_j - s_i \geqslant -2$, the deletion cost is decreased by at least 0 from T to $\widetilde{T}$. Therefore, there is an optimal tree where any pseudo-leaf node u_i with $s_i \leqslant 4$ is on level $L - 1$. Hence, the lemma is proved. □

Lemma 5. *In an optimal tree $T_{n,opt}$, for any two pseudo-leaf nodes u_i and u_j satisfying $l_j > l_i$, we have $l_j - l_i \leq 1$, which means the pseudo-leaf nodes should only be on level $L - 1$ or $L - 2$.*

Proof. Suppose on the contrary there are two pseudo-leaf nodes u_i and u_j with $l_j - l_i \geqslant 2$ in the optimal tree. According to Lemma 4, we have $s_i = 5$. We prove that the cost decreases by at least 2 after moving a child of u_j to u_i. We use the symbols $\widehat{T}$ and $\widetilde{T}$ to denote the trees similarly as in the proof of Lemma 4. The deletion cost is decreased by $3l_j + s_j - 1$ from tree T to $\widehat{T}$ and increased by $3l_i + 5$ from tree $\widehat{T}$ to $\widetilde{T}$. Notice that the structure of the subtree whose root is u_i has changed (transformed from root degree 5 to root degree 3) after adding the leaf, as Fig. 4 shows. Given $l_j - l_i \geqslant 2$ and $2 \leqslant s_j \leqslant 5$, the deletion cost is decreased by at least 2 from T to $\widetilde{T}$, which contradicts the optimality of T. Furthermore, we know that there is at least one pseudo-leaf node on level $L - 1$ because nodes on level L are all leaves. Therefore, $l_j - l_i \leq 1$ implies that pseudo-leaf nodes should only be on level $L - 1$ or $L - 2$. Hence, the lemma is proved. □

Lemma 6. *If all pseudo-leaf nodes are on the same level, we have the property that any nodes u_i, u_j on level $L - 1$ satisfy the inequality $|n_i - n_j| \leqslant 1$ where n_i and n_j are the number of leaf descendants of u_i and u_j respectively.*

Proof. Because $n_i = s_i$ for all pseudo-leaf nodes, we only need to prove $|s_i - s_j| \leqslant 1$ for any pseudo-leaf nodes u_i and u_j on the same level. In this case, according to Lemma 5, all of them can only be on level $L - 1$. If $s_i - s_j \geqslant 2$, we can get a better tree by moving a child of u_i to u_j, because the cost decreased is a positive value, which is equal to $s_i - 1 - s_j \geqslant 1$. For the other case when $s_j - s_i \geqslant 2$, we can get a better tree in a similar way. Hence, the lemma is proved. □

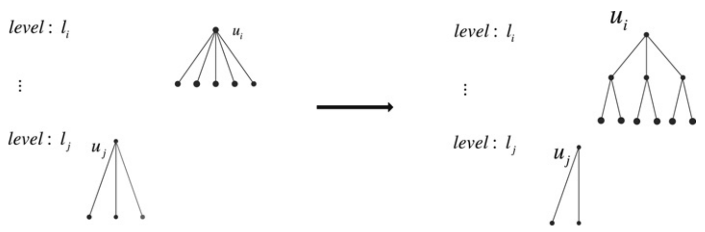

Fig. 4. Transformation of tree T where there are two pseudo-leaf nodes u_i and u_j which were respectively on levels l_i and l_j, and $(l_j - l_i) \geqslant 2$.

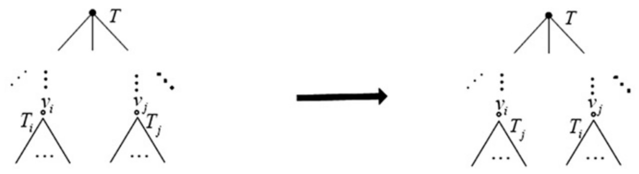

Fig. 5. Exchange of subtrees whose root nodes have the same ancestor weight.

Lemma 7. *If some of the pseudo-leaf nodes are on different levels, then we have the property that for any nodes u_i, u_j on the same level ($L - 2$ or $L - 1$), the inequality $|n_i - n_j| \leqslant 1$ holds.*

Proof. When there are pseudo-leaf nodes on different levels, according to Lemma 5, the possible situation is that some of them are on level $L - 1$ while others are on level $L - 2$. Moreover, pseudo-leaf nodes on level $L - 2$ can only have 5 children. According to this property, we then prove that the pseudo-leaf nodes on level $L - 1$ can only have $s_j = 2$ children. Considering the opposite situation where a pseudo-leaf node u_j on level $L - 1$ has $s_j \geqslant 3$, then there is an optimal tree by moving a child of u_j to a pseudo-leaf node u_k on level $L - 2$. Again we use symbols $\widehat{T}$ and $\widetilde{T}$ to denote the trees similarly as in the previous proofs. Deletion cost is decreased by $3l_j + s_j - 1$ from T to $\widehat{T}$ and increased by $3l_k + 5$ from $\widehat{T}$ to $\widetilde{T}$. Given $l_j - l_k \geqslant 1$ and $3 \leqslant s_j \leqslant 5$, deletion cost is decreased by at least 0 from T to $\widetilde{T}$. Furthermore, the structure of the subtree whose root is u_k has changed from one level tree to two levels (transformed from root degree 5 to degree 3) after adding the leaf. Therefore, if there are some pseudo-leaf nodes on level $L - 2$, all pseudo-leaf nodes on level $L - 1$ can only have 2 children, which means their ancestor on level $L - 2$ has 6 leaf descendants. Hence, the lemma is proved. □

Lemma 8. *Given a tree T, for any subtrees T_i, T_j whose roots v_i, v_j have the same ancestor weight $w_i = w_j$, if we exchange these two subtrees, the cost of the resulting tree does not change.*

Proof. The lemma can be proved by choosing all the nodes in T except subtrees T_i, T_j as the skeleton, as shown in Fig. 5. For any leaf u in subtrees T_i, suppose that its deletion cost in T equals $x + w_i$ where x denotes the cost when u is deleted from T_i. After the whole subtree T_i is moved from v_i to v_j, the cost of deleting u equals $x + w_j$, which equals $x + w_i$. Hence, the cost to delete all leaves in T_i does not change when we exchange subtrees T_i and T_j. Similar results hold for T_j. The lemma is then proved. □

Notice that when all pseudo-leaf nodes are on level $L - 1$, the nodes v_i, v_j on level k ($1 \leqslant k \leqslant L - 1$) have ancestor weight $w_i = w_j = 3k$. Therefore, the total deletion cost will not change when we exchange the subtrees with root nodes on the same level $L - 1$. For the other case when some pseudo-leaf nodes are on level $L - 2$ and others are on level $L - 1$, the nodes v_i, v_j on level $1 \leqslant k \leqslant L - 2$ also have ancestor weight $w_i = w_j = 3k$. According to this observation, for any two nodes v_i, v_j on the upper level, we next prove that inequality $|n_i - n_j| \leqslant 1$ also holds by exchanging the subtrees.

Lemma 9. *There is an optimal tree $T_{n,opt}$ where for any node v_i, v_j on the same level $1 \leqslant l_i = l_j \leqslant (L-2)$, the relation $|n_i - n_j| \leqslant 1$ holds.*

Proof. According to Lemmas 6 and 7 we have the induction foundation that nodes on level $L - 1$ or $L - 2$ satisfy the semi-balance property. Suppose that for nodes on level k, the semi-balance property also holds. Then, level k has 3^k nodes and every node has $t = \lfloor \frac{n}{3^k} \rfloor$ or $t + 1 = \lfloor \frac{n}{3^k} \rfloor + 1$ leaf descendants. If there are m_1 nodes with t leaf descendants and m_2 nodes with $t + 1$ leaf descendants, we have $m_2 = (n \bmod 3^k)$ and $m_1 = 3^k - m_2$. In the following, we prove that nodes on level $k - 1$ also satisfy the semi-balance property by regrouping the subtrees with roots on level k. According to Lemma 8, exchange of the subtrees with roots on the same level will not change the total deletion cost. According to this, we prove the induction in three cases $0 \leqslant m_2 < \frac{1}{2}m_1$, $\frac{1}{2}m_1 \leqslant m_2 < 2m_1$ and $2m_1 \leqslant m_2$. First, when $0 \leqslant m_2 < \frac{1}{2}m_1$, there are m_2 subtrees which have $t + 1$ leaves. For each of these subtrees, we combine it with two subtrees which have t leaves. Hence, there are m_2 subtrees on level $k - 1$ which have $3t + 1$ leaves. Furthermore, for the remaining $m_1 - 2m_2$ subtrees which all have t leaves, we simply combine every three of them. As a result, there are $(m_1 - 2m_2)/3 = 3^{k-1} - m_2$ subtrees on level $k - 1$ which have $3t$ leaves and m_2 subtrees which have $3t + 1$ leaves. The semi-balance property holds for this case. Similarly, we can prove semi-balance also holds when $\frac{1}{2}m_1 \leqslant m_2 < 2m_1$ or $2m_1 \leqslant m_2$. By induction, the theorem holds for any level l ($l \leq L - 2$) of the optimal tree. □

Based on these lemmas, we get the following theorem:

Theorem 1. *When the number of leaves $n \geqslant 6$, there is an optimal tree $T_{n,opt}$ where the sizes of its three subtrees differ by at most* 1.

The correctness of the theorem is evident with the support of Lemmas 6, 7 and 9. Theorem 1 in fact implies that we can get an optimal tree by distributing the leaves into subtrees in a semi-balanced way until every pseudo-leaf node u_i satisfies $2 \leqslant s_i \leqslant 5$. This can be interpreted as the following optimal tree construction rule:
(1) When $n \leqslant 5$, the optimal tree is a one level tree with all leaves on the same level.
(2) When $n \geqslant 6$, the optimal tree is a tree with root degree 3. We distribute n leaves into its three subtrees in a semi-balanced way with $\lceil \frac{n}{3} \rceil$, $\lceil \frac{n-1}{3} \rceil$, $\lceil \frac{n-2}{3} \rceil$ leaves respectively.
(3) For each subtree, recursively construct its optimal structure according to (1) and (2).

The rule above implies that the worst-case deletion cost $T_{n,\max}$ (abbreviated as $T(n)$) can be computed along with the constructing process. For $k = 1$, we have

$$T(n) = T\left(\left\lceil \frac{n}{3} \right\rceil\right) + T\left(\left\lceil \frac{n-1}{3} \right\rceil\right) + T\left(\left\lceil \frac{n-2}{3} \right\rceil\right) + 3n - 6.$$

For $k = 2$, we have

$$\begin{aligned}
T\left(\left\lceil \frac{n}{3} \right\rceil\right) &= T\left(\left\lceil \frac{\lceil \frac{n}{3} \rceil}{3} \right\rceil\right) + T\left(\left\lceil \frac{\lceil \frac{n}{3} \rceil - 1}{3} \right\rceil\right) + T\left(\left\lceil \frac{\lceil \frac{n}{3} \rceil - 2}{3} \right\rceil\right) + 3\lceil \frac{n}{3} \rceil - 6 \\
&= T\left(\left\lceil \frac{\lceil \frac{n}{3} \rceil}{3} \right\rceil\right) + T\left(\left\lceil \frac{\lceil \frac{n-3}{3} \rceil}{3} \right\rceil\right) + T\left(\left\lceil \frac{\lceil \frac{n-6}{3} \rceil}{3} \right\rceil\right) + 3\left\lceil \frac{n}{3} \right\rceil - 6 \\
&= T\left(\left\lceil \frac{n}{9} \right\rceil\right) + T\left(\left\lceil \frac{n-3}{9} \right\rceil\right) + T\left(\left\lceil \frac{n-6}{9} \right\rceil\right) + 3\left\lceil \frac{n}{3} \right\rceil - 6.
\end{aligned}$$

The last equality holds because $\lceil \frac{\lceil \frac{n}{a} \rceil}{b} \rceil = \lceil \frac{n}{ab} \rceil$ where a, b are positive integers. Then $T(n)$ can be computed as

$$T(n) = T\left(\left\lceil \frac{n}{9} \right\rceil\right) + T\left(\left\lceil \frac{n-1}{9} \right\rceil\right) + \cdots + T\left(\left\lceil \frac{n-8}{9} \right\rceil\right) + 3n - 3 \cdot 6 + 3n - 6.$$

In general, for $k = \lceil \log_3 n \rceil - 1$ or $k = \lceil \log_3 n \rceil - 2$, we have

$$T(n) = T\left(\left\lceil \frac{n}{3^k} \right\rceil\right) + T\left(\left\lceil \frac{n-1}{3^k} \right\rceil\right) + \cdots + T\left(\left\lceil \frac{n-3^k+1}{3^k} \right\rceil\right) + 3n \cdot k - 6\sum_{i=0}^{k-1} 3^i.$$

In fact, there are $n \bmod 3^k$ nodes which have $\lceil \frac{n}{3^k} \rceil$ leaf descendants and $3^k - (n \bmod 3^k)$ nodes which have $\lceil \frac{n}{3^k} \rceil - 1$ leaf descendants. Notice that when $2 \leqslant \lceil \frac{n}{3^k} \rceil \leqslant 5$ these nodes are pseudo-leaf nodes, while the optimal structure when $\lceil \frac{n}{3^k} \rceil = 6$ is the tree where root degree is 3 and each child has 2 leaves. Hence, we have the following theorem.

Theorem 2. *The worst-case deletion cost $T(n)$ of the optimal tree $T_{n,opt}$ can be computed with $O(\log\log n)$ multiplications of $O(\log n)$-bit integers according to the equation below.*

$$T_{n,\max} = (n \bmod 3^k) \cdot T\left(\left\lceil \frac{n}{3^k} \right\rceil\right) + (3^k - n \bmod 3^k) \cdot T\left(\left\lceil \frac{n}{3^k} \right\rceil - 1\right) + 3n \cdot k - 3^{k+1} + 3$$

$$\textit{where } k = \begin{cases} \lceil \log_3 n \rceil - 1 & \textit{if } 2 \leqslant \lfloor \frac{n}{3^{\lceil \log_3 n \rceil - 1}} \rfloor \leqslant 5, \\ \lceil \log_3 n \rceil - 2 & \textit{if } 2 \leqslant \lfloor \frac{n}{3^{\lceil \log_3 n \rceil - 2}} \rfloor \leqslant 5. \end{cases}$$

For basic cases $2 \leqslant n \leqslant 5$, we have $C(2) = 1$, $C(3) = 3$, $C(4) = 6$, $C(5) = 10$. Since the fastest method to do $O(n)$-bit integer multiplication has a complexity $O(n \log n 2^{O(\log^* n)})$ [10], we need at most $O(\log n \log^2 \log n 2^{O(\log^* n)})$ time to compute the value of $3^{O(\log n)}$, and hence the optimal tree structure, which is better than the dynamic programming algorithm with $O(n^2)$ time. Furthermore, to construct the optimal tree, we can distribute the users into subtrees in a semi-balanced way.

4. Optimal tree structures for *n* insertions followed by *n* deletions

In this section, we investigate a more general setting where the cost of the initial group setup is also considered. In this new setting, the optimal tree we computed in the previous section is probably no longer optimal. We aim to study the optimal tree structure to minimize the cost for the initial setup followed by n deletions.

Lemma 10. *The number of encryptions needed to build the initial tree equals $N - 1$ where N is the number of nodes in the tree.*

Proof. The tree is built after all the members arrive; we distribute the keys to the users securely in the bottom–up fashion with respect to the tree. Therefore, every key except the key stored in the root will be used once as an encryption key in the whole process, which amounts to $N - 1$ encryptions in total. The lemma is then proved. □

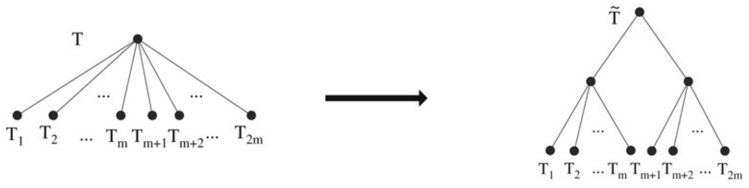

Fig. 6. Transformation of tree T where root v has degree $2m$.

To represent the total cost of insertion and deletion in one formula, we modify the cost of *skeleton* a bit as follows.

Definition 6. Suppose the skeleton T' has t non-root nodes, the worst-case cost for T' is $C_{T',\max} = C_{T',\mathit{deletion}} + t$.

In fact, $C_{T',\mathit{deletion}}$ is the worst-case cost when only deletion is considered, and t is the number of messages encrypted by the keys stored in the t non-root nodes when establishing the initial key tree.

Lemma 11. *If the skeleton T' of T has r leaves, the worst-case cost for T is $C_{T,\max} = C_{T',\max} + \sum_{i=1}^{r}(C_{T_i,\max} + (n_i - 1)w_i)$.*

Proof. When only considering the situation where a deleting sequence is performed, we have the conclusion below, which was already proved in [9].

$$C_{T,\mathit{deletion}} = C_{T',\mathit{deletion}} + \sum_{i=1}^{r}(C_{T_i,\mathit{deletion}} + (n_i - 1)w_i).$$

When we take the insertion cost into consideration, the cost will be $C_{T,\mathit{deletion}} + N - 1$ in total where N denotes the total number of nodes in the tree. Because the number of non-root nodes in the tree is $N - 1$, we can compute the insertion cost each time we compute the deletion cost, which means we should distribute the cost caused by insertion into the deleting process. According to the new definition of the skeleton cost $C_{T,\max}$, each non-root node in the tree T exactly contributes an extra cost of 1 compared to $C_{T,\mathit{deletion}}$, no matter how the skeletons are chosen recursively. The lemma is then proved. □

Definition 7. In a tree T where the skeleton T' has r subtrees, the ancestor weight vector is denoted as $(w_1, w_2, \ldots, w_r)$ where w_i is non-increasing when i increases.

The proof of the following lemma is similar as in [9].

Lemma 12. *There is an optimal tree whose leaf descendant vector $(n_1, n_2, \ldots, n_r)$ is non-decreasing.*

Proof. Because the ancestor weight vector is non-increasing, the subtree cost is minimized when the leaf descendant vector is non-decreasing. □

In the following lemmas, we first prove that the degree of internal nodes cannot exceed 7 and then remove other impossible node degree combinations. Since the possible tree structures with degree $d \leqslant 7$ are still too many to be enumerated, we restrict the possible structures gradually using mathematical methods.

Lemma 13. *There is an optimal tree $T_{n,opt}$ where every internal node v has degree at most* 7.

Proof. Suppose an optimal tree T has root degree $d = 2m$. We can transform the tree to $\widetilde{T}$ as Fig. 6 shows.

$$\begin{aligned}
C_{T,\max} &= C_{T',\max} + \sum_{i=1}^{2m}(C_{T_i,\max} + (n_i - 1)w_i) \\
&= \sum_{i=0}^{2m-1} i + 2m + \sum_{i=1}^{2m} C_{T_i,\max} + 2m(n - 2m) \\
C_{\widetilde{T},\max} &= 1 + 2\sum_{i=3}^{m+1} i + 2 + 2m + \sum_{i=1}^{2m} C_{T_i,\max} + (m + 2)(n - 2m).
\end{aligned}$$

Because $2m(n - 2m) \geqslant (m + 2)(n - 2m)$ for $m \geqslant 2$, we have

$$\begin{aligned}
\Delta &= C_{T,\max} - C_{\widetilde{T},\max} \geqslant \sum_{i=0}^{2m-1} i - \left(2\sum_{i=3}^{m+1} i + 1 + 2\right) \\
&= m^2 - 4m + 1.
\end{aligned}$$

Hence, we have $\Delta \geqslant 0$ when $m \geqslant 4$. In the case where the root has odd degree $d = 2m + 1$ we can get similar results. By transforming the root from degree $2m + 1$ to 2 as Fig. 7 shows, the cost decreases at least by $\Delta = m^2 - 3m - 1$. The condition $m \geqslant 4$ also ensures $\Delta \geqslant 0$. Therefore, the tree after several times of transformation is still optimal but has no internal nodes with degree greater than 7. □

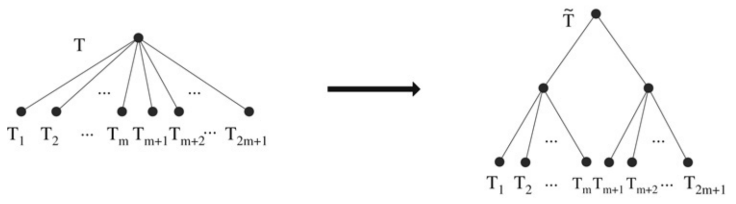
T
T1 T2 ··· Tm Tm+1 Tm+2 ··· T2m+1
T̃
T1 T2 ··· Tm Tm+1 Tm+2 ··· T2m+1

Fig. 7. Transformation of tree T where root v has degree $2m + 1$.

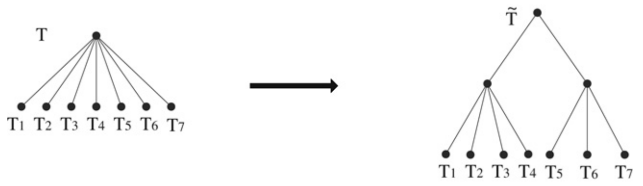

Fig. 8. Transformation of tree T where root v has degree 7 and at least one child of v is an internal node.

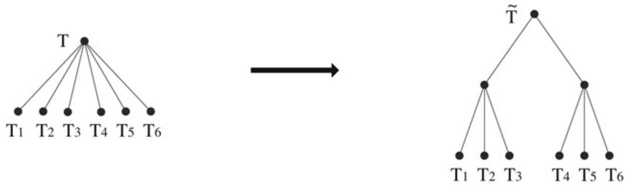

Fig. 9. Transformation of tree T when root v has degree 6 and at least one child of v is an internal node.

Lemma 14. *There is an optimal tree $T_{n,opt}$ where every internal node v has degree at most* 7 *and children of nodes with degree* 7 *are all leaves.*

Proof. In an optimal tree T that satisfies Lemma 13, suppose it has an internal node v with degree 7 which has at least one child being an internal node. In this case, the ancestor weight vector of v's children will be $(w_1, w_2, \ldots, w_7) = (7, 7, \ldots, 7)$. We perform the transformation as Fig. 8 shows. The ancestor vector after the transformation will be (6, 6, 6, 6, 5, 5, 5). The skeleton costs of T and $\widetilde{T}$ are respectively $21 + 7$ and $20 + 9$, increased by 1 after the transformation. When given a non-decreasing leaf descendant vector $(n_1, n_2, \ldots, n_7)$, it is obvious that the cost of the subtrees decreases by at least 2 since $(n_7 - 1)(7 - 5) \geqslant 2$. Hence, we get $\Delta \geqslant -1 + 2 = 1$, which means that in an optimal tree all of the internal nodes with degree 7 are pseudo-leaf nodes. □

Lemma 15. *There is an optimal tree $T_{n,opt}$ where every internal node v has degree at most* 7 *and children of nodes with degree* 7 *and* 6 *are all leaves.*

Proof. In an optimal tree T with n leaves that satisfies Lemma 14, suppose it has an internal node v with degree 6 which has at least one child being an internal node. We have $n \geqslant 5 + 2 = 7$. Then the ancestor vector of v's children will be $(w_1, w_2, \ldots, w_6) = (6, 6, \ldots, 6)$. We perform the transformation as Fig. 9 shows. The ancestor vector after the transformation will be (5, 5, 5, 5, 5, 5). The skeleton costs of T and $\widetilde{T}$ are respectively $15+6$ and $15+8$, increased by 2 after the transformation. When given a non-decreasing leaf descendant vector $(n_1, n_2, \ldots, n_6)$, it is obvious that the subtree cost decreases by $\Delta = -2+6(n-6)-5(n-6) = n-8$. If $n = 7$ then it can be transformed into a one level tree to be optimal. If $n \geqslant 8$, we have $\Delta = n - 8 \geqslant 0$ according to the transformation, which means that in an optimal tree all of the internal nodes with degrees 6 and 7 are pseudo-leaf nodes. □

Lemma 16. *There is an optimal tree $T_{n,opt}$ where every internal node v has degree at most* 7 *and children of nodes with degree* 7, 6 *and* 5 *are all leaves.*

Proof. Given an optimal tree T with n leaves that satisfies Lemma 15, suppose the root v has degree 5. We prove this lemma by considering two cases.
Case 1. At least one child of v is a leaf.

Without loss of generality, we assume that the numbers of leaf descendants of the other four subtrees satisfy $n_a \geqslant n_b \geqslant n_c \geqslant n_d \geqslant 1$. We move the leaf to the subtree T_a which has $n_a \geqslant 2$ leaf descendants and m subtrees, as Fig. 10 shows.

$$C_{T,\max} = 10 + 5 + C_{T_a,\max} + C_{T_b,\max} + C_{T_c,\max} + C_{T_d,\max} + 5(n_a + n_b + n_c + n_d - 4)$$
$$C_{\widetilde{T},\max} = 6 + 4 + C_{\widetilde{T}_a,\max} + C_{T_b,\max} + C_{T_c,\max} + C_{T_d,\max} + 4(n_a + 1 + n_b + n_c + n_d - 4)$$

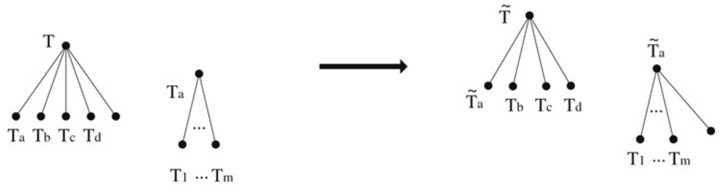

Fig. 10. Transformation of tree T where root v has degree 5 and at least one child of v is a leaf.

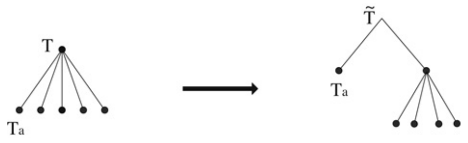

Fig. 11. Transformation of tree T where root v has degree 5 and at least four children of v are leaves.

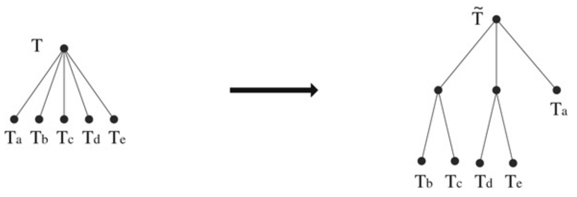

Fig. 12. Transformation of tree T where root v has degree 5 and all children of v are internal nodes.

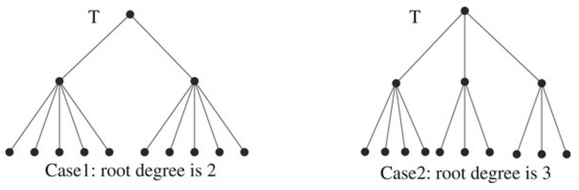

Fig. 13. Two cases of optimal tree T with $n = 10$ leaves: One with cost $37 + 12$, and the other with cost $36 + 13$.

$$\begin{aligned}
C_{T_a,\max} &= \sum_{i=1}^{m-1} i + m + \sum_{i=1}^{m} C_{T_i,\max} + m(n_a - m)\\
C_{\widetilde{T}_a,\max} &= \sum_{i=1}^{m} i + m + 1 + \sum_{i=1}^{m} C_{T_i,\max} + (m+1)(n_a - m)\\
\Delta &= n_b + n_c + n_d - 4.
\end{aligned}$$

If $n_b + n_c + n_d \geqslant 4$, we get $\Delta \geqslant 0$. Otherwise if $n_b + n_c + n_d = 3$, which implies $n_b = n_c = n_d = 1$, we prove that it also can be changed into a better tree with root degree less than 5 when $n \geqslant 7$, as shown by Fig. 11.

$$\begin{aligned}
C_{T,\max} &= 10 + 5 + C_{T_a,\max} + 5(n_a - 1)\\
C_{\widetilde{T},\max} &= 13 + 6 + C_{T_a,\max} + 2(n_a - 1)\\
\Delta &= 3n_a - 7.
\end{aligned}$$

If $n \geqslant 7$, we have $n_a \geqslant 3$ which ensures $\Delta \geqslant 2$. Therefore, the lemma holds in this case.
Case 2. All children of v are internal nodes.

Similarly, we assume $n_a \geqslant n_b \geqslant n_c \geqslant n_d \geqslant n_e \geqslant 2$, which implies $n \geqslant 10$. By transforming it into a tree with root degree $d = 3 < 5$ as Fig. 12 shows, we can also reduce the cost.

$$\begin{aligned}
C_{T,\max} &= 10 + 5 + C_{T_a,\max} + C_{T_b,\max} + C_{T_c,\max} + C_{T_d,\max} + C_{T_e,\max} + 5(n_a + n_b + n_c + n_d + n_e - 5)\\
C_{\widetilde{T},\max} &= 11 + 7 + C_{T_b,\max} + C_{T_c,\max} + C_{T_d,\max} + C_{T_e,\max} + 5(n_b + n_c + n_d + n_e - 4) + C_{T_a,\max} + 3(n_a - 1)\\
\Delta &= 2n_a - 5.
\end{aligned}$$

When $n \geqslant 11$, we have $n_a \geqslant 3$ which implies $\Delta \geqslant 1$. When $n = 10$, we can also get an optimal tree with root degree less than 5. In fact, there are two optimal trees when $n = 10$, both with cost 49, as Fig. 13 shows. Since in both cases, the tree T with root degree 5 can be transformed into a better tree with smaller root degree, we have proved the lemma. □

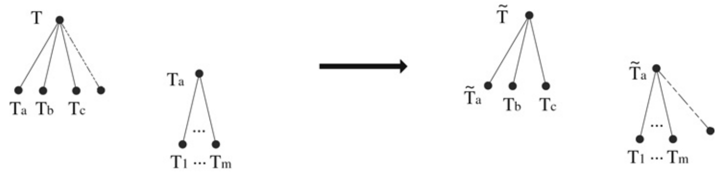

Fig. 14. Transformation of tree T where root v has degree 4 and at least one child of v is a leaf.

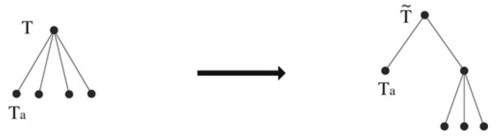

Fig. 15. Transformation of tree T where root v has degree 4 and at least three children of v are leaves.

Lemma 17. *There is an optimal tree $T_{n,opt}$ where every internal node v has degree at most 7 and children of nodes with degree not equal to 2 or 3 are all leaves.*

Proof. Given an optimal tree T that satisfies Lemma 16, suppose on the contrary the root v has degree 4 and at least one child of v is an internal node. We prove this lemma by considering two cases.
Case 1. At least one child of root v is a leaf.

Without loss of generality, we assume that the number of leaf descendants of the other three subtrees T_a, T_b, T_c satisfy $n_a \geqslant n_b \geqslant n_c \geqslant 1$ and $n_a \geqslant 2$. We move the leaf to the subtree T_a as Fig. 14 shows.

$$\begin{aligned}
C_{T,\max} &= 6+4+C_{T_a,\max}+C_{T_b,\max}+C_{T_c,\max}+4(n_a+n_b+n_c-3)\\
C_{\widetilde{T},\max} &= 3+3+C_{\widetilde{T_a},\max}+C_{T_b,\max}+C_{T_c,\max}+3(n_a+n_b+n_c+1-3)\\
C_{T_a,\max} &= \sum_{i=1}^{m-1} i+m+\sum_{i=1}^{m} C_{T_i,\max}+m(n_a-m)\\
C_{\widetilde{T_a},\max} &= \sum_{i=1}^{m} i+m+1+\sum_{i=1}^{m} C_{T_i,\max}+(m+1)(n_a-m)\\
\Delta &= n_b+n_c-3 \geqslant 0.
\end{aligned}$$

If $n_b+n_c \geqslant 3$, we get $\Delta \geqslant 0$. Otherwise we have $n_b+n_c=2$ which implies $n_b=n_c=1$. In this situation, we prove that it can also be changed into a better tree with root degree less than 5 when $n \geqslant 6$, as shown by Fig. 15.

$$\begin{aligned}
C_{T,\max} &= 10+C_{T_a,\max}+4(n_a-1)\\
C_{\widetilde{T},\max} &= 13+C_{T_a,\max}+2(n_a-1)\\
\Delta &= 2n_a-5 \geqslant 0.
\end{aligned}$$

When $n \geqslant 6$, we have $n_a \geqslant 3$ which implies $\Delta \geqslant 1$. Therefore, the lemma holds in this case.
Case 2. All the children of the root v are internal nodes.

We can assume that $n_a \geqslant n_b \geqslant n_c \geqslant n_d \geqslant 2$ without loss of generality. By transforming T as shown in Fig. 16, we can also reduce the cost.

$$\begin{aligned}
C_{T,\max} &= 6+4+C_{T_a,\max}+C_{T_b,\max}+C_{T_c,\max}+C_{T_d,\max}+4(n-4)\\
C_{\widetilde{T},\max} &= 7+5+C_{T_a,\max}+C_{T_b,\max}+C_{T_c,\max}+C_{T_d,\max}+5(n_c+n_d-2)+3(n_a+n_b-2)\\
\Delta &= -2+n_a+n_b-n_c-n_d \geqslant -2.
\end{aligned}$$

If $\Delta \geqslant 0$, the lemma is proved, otherwise we consider two subcases $\Delta=-1$ and $\Delta=-2$. It is obvious that when $\Delta=-1$, the leaf descendant vector $(n_a, n_b, n_c, n_d=n')$ is either $(n'+1, n', n', n')$ or $(n'+1, n'+1, n'+1, n')$, while the leaf descendant vector is (n', n', n', n') when $\Delta=-2$.

Then, we take a second transformation. Notice that optimal trees with the same number of leaves can have the same structures. We prove in the following that by suitably moving a leaf from T_i to T_j $(n_i=n_j)$, the value of Δ will decrease by 1.

In the first subcase where $\Delta=-1$, $(n_a, n_b, n_c, n_d)=(n'+1, n', n', n')$, we move a leaf node u of T_c to its corresponding position in T_b. By Lemma 3, deleting u from T_c decreases the cost by $5+x-1$, and adding u to T_b increases the cost by $3+x$, where x denotes the ancestor weight of u in T_b. As a whole, the total tree cost is reduced by $(4+x)-(3+x)=1$. This means we get $\Delta=-1+1=0$ by this two-step transformation.

In the second subcase where $\Delta=-1$, $(n_a, n_b, n_c, n_d)=(n'+1, n'+1, n'+1, n')$, we can also get $\Delta=-1+1=0$ by moving a leaf node on T_c to T_b. Similarly, in the case where $\Delta=-2$, (n', n', n', n'), we can also get $\Delta=-2+1+1=0$ by

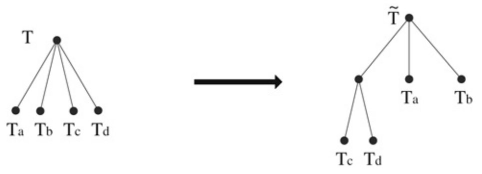

Fig. 16. Transformation of tree T where root v has degree 4 and all children of v are internal nodes.

moving a leaf node from T_c to T_b and moving another leaf node from T_d to T_a. In all the subcases, we can get a tree with less or equal cost.

Since in all the cases, the tree T with root degree 4 can be transformed into a better tree with a smaller root degree, the lemma is finally proved. □

Based on this lemma, we have the following theorem:

Theorem 3. *Algorithm 1 can compute an optimal tree in $O(n^2)$ time.*

Proof. In Algorithm 1, we use R_i to denote the minimum cost of trees with i leaves and root degree restricted to be 3, D_i to denote the minimum cost of trees with i leaves and root degree restricted to be 2, and C_i to denote the minimum cost of all the trees with i leaves. For $n \leqslant 7$, the first three lines initialize the first 7 values for R, D and C. For $n \geqslant 8$, according to Lemma 17, the optimal tree has root degree 2 or 3. For degree 2, the best worst-case cost D_i can be computed by enumerating all possible combinations of its two branches. To be specific, D_i has two branches respectively with $k1$ and $k2$ leaves. Since it needs extra cost $2(k1-1+k2-1)$ to delete these two branches because deleting each leaf will incur extra cost 2 on the skeleton, we have $D_i = C_{k1} + C_{k2} + 2i - 3$. For root degree 3, suppose the smallest subtree has $k1$ leaf descendants, and the remaining structure with $k2$ leaves is treated as a tree with root degree 2 and cost D_{k2}. It needs an extra cost of $3(k1-1)$ for the branch with $k1$ leaves and $(k2-2)$ for the remaining structure when they are deleted from a larger tree with i leaves. By further considering the difference in the skeleton cost, the worst-case cost when the root degree is 3 can be computed as $R_i = C_{k1} + 3(k1-1) + D_{k2} + (k2-2) + 3$. The inner loop enumerates all possible combinations of these two parts. Furthermore, the minimum cost C_i is the smaller one of D_i and R_i. Finally, the running time is $O(n^2)$ because there are two nested loops in this algorithm. The optimal tree structure can be obtained through keeping the branching information in the algorithm. □

Algorithm 1 *Sequence_OPT*

```
1.  $R_1 = 1; R_2 = 3; R_3 = 6; R_4 = 10; R_5 = 15; R_6 = 21; R_7 = 28;$
2.  $D_1 = 1; D_2 = 3; D_3 = 8; D_4 = 13; D_5 = 18; D_6 = 23; D_7 = 29;$
4.  $C_1 = 1; C_2 = 3; C_3 = 6; C_4 = 10; C_5 = 15; C_6 = 21; C_7 = 28;$
5.  for $i = 8$ to $n$
6.     $D_i = i^2; R_i = i^2; C_i = i^2;$
7.     for $k1 = 1$ to $i/2$
8.           $k2 = i - k1;$
9.           if $D_i > C_{k1} + C_{k2} + 2 \cdot i - 1$ then
10.              $D_i = C_{k1} + C_{k2} + 2 \cdot i - 1;$
11.           if $R_i > C_{k1} + D_{k2} + i + 2 \cdot k1 - 2$ then
12.              $R_i = C_{k1} + D_{k2} + i + 2 \cdot k1 - 2;$
13.     end for
14.     $C_i = min(D_i, R_i);$
15.end for
```

5. Discussion

One open problem is whether there is a certain semi-balance property for the optimal tree when the initial setup cost is also considered. We roughly discuss this issue in this section by proving two lemmas.

Lemma 18. *In the optimal tree, if an internal node v (not a pseudo-leaf node) has degree 2, then its child is either a pseudo-leaf node or an internal node with degree 3.*

Proof. Firstly, it is easy to remove the possibility of a leaf directly attached to v (Please refer to the transformation in Fig. 17(c) and take v_2 as a leaf.) We then prove the lemma by showing the correctness of the following two rules:

(1) if neither of its two children is a pseudo-leaf node, then both of these two children have degree 3;

(2) if only one of the two children is a pseudo-leaf node, then the other one has degree 3.

We suppose that v_1, v_2 are the two children of root v. For rule (1), v_1, v_2 can have degree 2 or 3 according to Lemma 17. We will then remove the possibility that $d_{v_1} = 2$, $d_{v_2} = 2$ and $d_{v_1} = 2$, $d_{v_2} = 3$ by doing transformations on the tree. In the

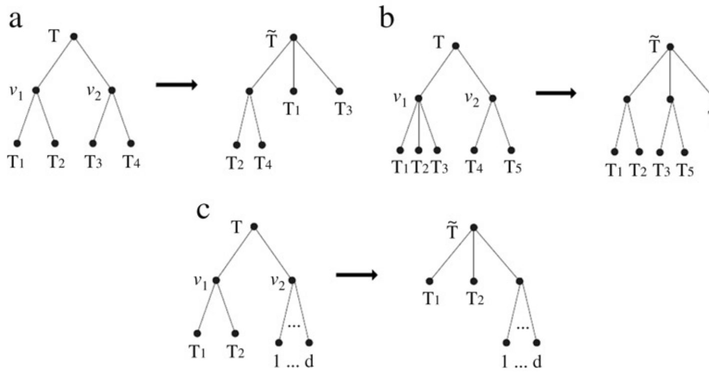

Fig. 17. Three transformations for the tree T.

first case, the four subtrees are transformed as shown in Fig. 17(a). W.l.o.g, we assume that $n_1 \geq n_2, n_3 \geq n_4$. Note that the two trees have skeleton costs $7+6$ and $7+5$ respectively. The cost for T is $C_{T,\max} = \sum_{i=1}^{4} C_{T_i,\max} + 13 + \sum_{i=1}^{4} 4(n_i - 1)$, while the cost for $\widetilde{T}$ is $C_{\widetilde{T},\max} = \sum_{i=1}^{4} C_{T_i,\max} + 12 + 3(n_1 + n_3 - 2) + 5(n_2 + n_4 - 2)$. Thus the transformation decreases the cost by $\Delta = n_1 - n_2 + n_3 - n_4 + 1 > 0$. In this way, we remove the possibility of $d_{v_1} = 2, d_{v_2} = 2$. For the second case, we transform it as shown in Fig. 17(b) where w.l.o.g we assume $n_4 \geq n_5$. The cost for T and $\widetilde{T}$ are respectively $C_{T,\max} = \sum_{i=1}^{5} C_{T_i,\max} + 11 + 7 + 5(n_1 + n_2 + n_3 - 3) + 4(n_4 + n_5 - 2)$ and $C_{\widetilde{T},\max} = \sum_{i=1}^{5} C_{T_i,\max} + 11 + 7 + 5(n_1 + n_2 + n_3 + n_5 - 4) + 3(n_4 - 1)$. The cost is decreased by $\Delta = n_4 - n_5 \geq 0$. By transforming the violating degree 2 nodes in the top–down order, we can also exclude the possibility of the second case in the optimal tree.

For rule (2), suppose that the pseudo-leaf node v_2 has degree d and on the contrary v_1 has two subtrees, we transform it as shown in Fig. 17(c). The cost for T and $\widetilde{T}$ are respectively $C_{T,\max} = C_{T_1,\max} + C_{T_2,\max} + \sum_{i=1}^{d-1}(i+2) + 4 + d + 4 + 4(n_1 + n_2 - 2)$ and $C_{\widetilde{T},\max} = C_{T_1,\max} + C_{T_2,\max} + \sum_{i=1}^{d-1}(i+3) + 3 + d + 3 + 3(n_1 + n_2 - 2)$. Let the number of leaf descendants of v be n_v. If $n_v \geq 14$, the cost is decreased by $\Delta = n_1 + n_2 - d + 1 = n_v - d - d + 1 > 0$ because $d \leq 7$, a contradiction. If $7 < n_v < 14$, by enumeration, we can guarantee that there is always a better structure where the degree of v is 3. If $n_v \leq 7$, then node v is a pseudo-leaf node itself. Therefore, the lemma is true. □

By Lemma 18, we note that roughly at least half of the ancestors of the pseudo-leaf node have degree 3 instead of 2. This leads us to study the maximum difference of positions of the leaves in the optimal tree. The following lemma shows that the leaves in the optimal tree will not differ too much in the levels.

Lemma 19. *In the optimal tree, all the leaves can only be on levels in* $[\lceil \frac{5}{6}L \rceil - 1, L]$.

Proof. In the optimal tree, consider two leaves v_i, v_j whose parents u_i, u_j are respectively on the levels l_i and $l_j = L - 1$, where L is the height of the tree (there must be one leaf on the level L). Suppose that $s_i(s_j)$ is the number of children of $u_i(u_j)$, then $2 \leq s_i, s_j \leq 7$ according to Lemma 13. Furthermore, u_i's ancestor has degree at most 3 and thus any leaf child of u_i has deletion cost at most $3l_i + 7$ after an extra child is added to u_i. Lemma 18 implies that at least half of u_j's ancestors have degree 3. Thus any leaf child of u_j has deletion cost at least $3 \cdot \lfloor \frac{l_j}{2} \rfloor + 2(l_j - \lfloor \frac{l_j}{2} \rfloor) + 1$. Then by moving one leaf child of u_j to u_i (similarly to the proof in Lemma 5), the cost decreases by $\Delta \geq 2l_j + \lfloor \frac{l_j}{2} \rfloor - 3l_i - 6$. Thus we have $2(L-1) + \lfloor \frac{(L-1)}{2} \rfloor - 3l_i - 6 \leq 0$, because of the optimality of the original tree. Therefore, all leaves can only be on levels $[\lceil \frac{5}{6}L \rceil - 1, L]$. □

6. Conclusion

While many works focus on fixing the cost bound under some multicast protocol, we try to find the optimal structure to minimize the cost. We investigate the scenario where the members all arrive in the initial setup time and then leave one by one. This can be applied in teleconferencing or applications where the member list can be fixed beforehand. Feng et al. [9] found the optimal tree structure when only deletion cost is considered. We prove a semi-balance property of the optimal key tree based on their work. We show that the members can be distributed in a semi-balance way in the optimal tree. Using this property we improve the running time from $O(n^2)$ to $O(\log \log n)$ multiplications of $O(\log n)$-*bit* integers. We then focus on the optimal tree structure when insertion cost for the initial period is simultaneously considered. We obtain a recursive formula and use it to eliminate the impossible degrees in the optimal tree. Based on this observation, we give an algorithm to compute the optimal tree with $O(n^2)$ time. Finally, we derive some balance structure for the optimal tree in this scenario.

One possible direction of future work is to investigate a more balanced structure for the optimal tree when insertion cost of the initial setup period is considered together with the deletion cost.

Acknowledgements

The third author was supported by Program for New Century Excellent Talents in University (No. NCET-05-0549) and the National Natural Science Foundation of China (No. 60573077 and No. 60775037). The second author was supported by National Basic Research Program of China Grant 2007CB807900, 2007CB807901 and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 116907].

References

[1] R.L. Graham, M. Li, F.F. Yao, Optimal tree structures for group key management with batch updates, SIAM Journal on Discrete Mathematics 21 (2) (2007) 532–547.
[2] M.T. Goodrich, J.Z. Sun, R. Tamassia, Efficient tree-based revocation in groups of low-state devices, in: Proceedings of the Twenty-Fourth Annual International Cryptology Conference, CRYPTO, 2004, pp. 511–527.
[3] M. Li, Z. Feng, R.L. Graham, F.F. Yao, Approximately optimal trees for group key management with batch updates, in: Proceedings of the Fourth Annual Conference on Theory and Applications of Models of Computation, 2007, pp. 284–295.
[4] X.S. Li, Y.R. Yang, M.G. Gouda, S.S. Lam, Batch re-keying for secure group communications, in: Proceedings of the Tenth International Conference on World Wide Web, 2001, pp. 525–534.
[5] J. Snoeyink, S. Suri, G. Varghese, A lower bound for multicast key distribution, in:Proceedings of the Twentieth Annual IEEE Conference on Computer Communications, 2001, pp. 422-431.
[6] D. Wallner, E. Harder, R.C. Agee, Key Management for Multicast: Issues and Architectures, RFC 2627, June 1999.
[7] C.K. Wong, M.G. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Transactions on Networking 8 (1) (2000) 16–30.
[8] F. Zhu, A. Chan, G. Noubir, Optimal tree structure for key management of simultaneous join/leave in secure multicast, in: Proceedings of Military Communications Conference, 2003, pp. 773–778.
[9] Z.Z. Chen, Z. Feng, M. Li, F.F. Yao, Optimizing deletion cost for secure multicast key management, Theoretical Computer Science 401 (2008) 52–61.
[10] M. Furer, Faster integer multiplication, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, 2007, pp. 57–66.