This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2009 proceedings.

Bandwidth Efficient Key Distribution for Secure Multicast in Dynamic Wireless Mesh Networks

Seungjae Shin, Junbeom Hur, Hanjin Lee, Hyunsoo Yoon
Dept. of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
373-1, Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
Email: {sjshin, jbhur, hjlee, hyoon}@nslab.kaist.ac.kr

***Abstract*—In the near future, various multicast based services will be provided over wireless mesh networks. For secure multicast services, various tree based group key management schemes have been introduced until now. Traditional tree based approaches mainly focus on reducing the number of rekeying messages transmitted by the key distribution center. However, they do not consider the network bandwidth used for transmitting each rekeying message. We propose a bandwidth efficient key tree management scheme for dynamic wireless mesh networks where membership changes occur frequently. Simulation results show that our scheme effectively reduces the bandwidth consumption used for rekeying compared to existing key tree schemes.**

***Index Terms*—key distribution, multicast security, secure group communication, wireless mesh networks**

I. Introduction

With the recent advances in technology underlying WLAN (Wireless Local Area Network), such as IEEE 802.11, wireless mesh networks are envisioned to allow users to access Internet from anywhere at anytime. As wireless connections are provided in many public places, people would desire to receive multicast based services, such as pay-per-view and video conference, via their mobile devices.

In multicast communications, data confidentiality is achieved by encrypting the content with a group key known to privileged group members. For secure group communication, newly joined members should not be able to decrypt messages exchanged among group members before they joined (backward secrecy). And revoked members should not be able to decrypt messages exchanged among group members after they left (forward secrecy). To provide backward and forward secrecy, the group key should be changed when there is a membership change, which is referred to as rekeying. Consider a group of N members. When a member leaves the group, the key distribution center (KDC) has to deliver the new group key to all members except the leaving member. Because this requires $N-1$ transmissions of rekeying messages, the KDC may encounter a large burden when N is large and membership changes are frequent.

To address this scalability issue, tree based group key management schemes were introduced [1], [6], [7], [8], [9], [10]. They reduces the number of rekeying message transmissions to $O(logN)$. However, traditional tree based schemes do not consider the bandwidth consumption used for rekeying. Here, reducing the bandwidth consumption of rekeying can be an important issue to guarantee enough bandwidth for reliable data delivery when large-scale multicast based services (ex. real-time video streaming) are provided over wireless networks.

There are some previous works considering the bandwidth efficiency of tree based group key management schemes [2], [3], [4], [5] for wireless networks. These schemes reduce the bandwidth consumption of rekeying by assigning common keys to members that are close to each other in the network topology. They mainly focus on constructing an initial key tree for given initial group members. But, they do not present enough consideration for the dynamic membership environment in which different members join or leave the group at different time.

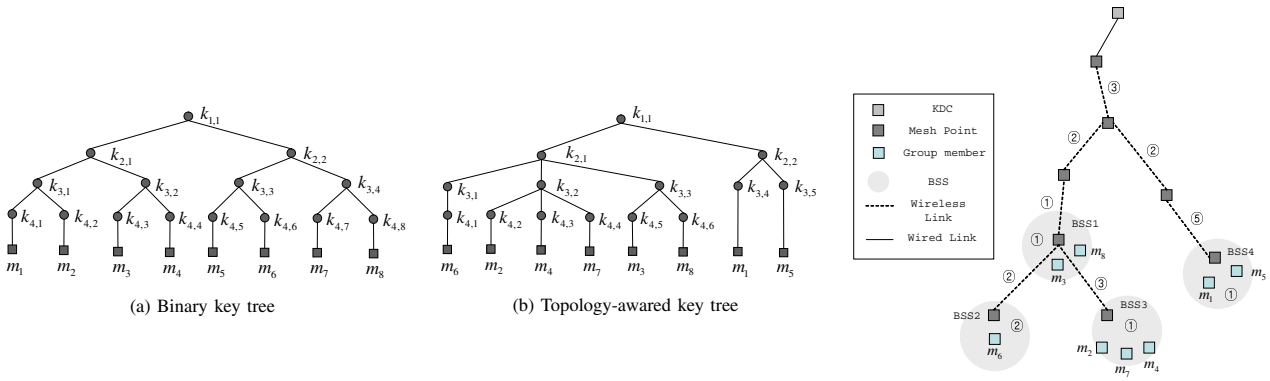
In this paper, we firstly define the metric that represents the expected bandwidth consumption of rekeying for given key tree. Then, we analyze the relationship between the assignment of key encryption keys (KEKs) and the bandwidth consumption of the key tree. Based on our analysis, we propose the ABR (adaptive and bandwidth reducing) tree, a bandwidth-efficient key tree management approach designed for wireless mesh networks when the group membership is dynamically changed. When a new member joins the group, our scheme assigns to the new member the proper KEKs to keep the expected bandwidth consumption of the key tree as low as possible. By this approach, ABR tree effectively reduces the actual bandwidth consumption used for rekeying compared to traditional key tree management schemes.

The rest of this paper is organized as follows: In Section II, we briefly review the tree based group key management scheme and define the metric that represents the expected bandwidth consumption of key tree. In Section III, we introduce the ABR tree as a bandwidth efficient key tree designed for dynamic membership environments. Simulation results are presented in Section IV. Finally, in Section V we conclude.

II. Background and Problem Definition

In this section, we briefly review the rekeying process of tree based key management scheme and define the metric that represents the expected bandwidth consumption of the key tree. We list the notations that used throughout this paper in Table I.

978-1-4244-2948-6/09/$25.00 ©2009 IEEE



(a) Binary key tree

(b) Topology-awared key tree

(c) Example of wireless mesh network

Fig. 1. Example of key trees and wireless mesh network

TABLE I
NOTATIONS

Items	Description
T	key tree
M_T	set of group members associated with key tree T
S_k	sub-group that consists of group members sharing key k
$\{k'\}_k$	key k' encrypted by key k
K_T	set of KEKs except group key (root) in T
$kek(m)$	set of KEKs assigned to the member m except gk_T in T
$anc(k)$	set of ancestor keys of key k except gk_T in T ($k \in K_T$)
$sib(k)$	set of sibling keys of key k in T ($k \in K_T$)
$ch(k)$	set of child keys of key k in T ($k \in K_T$)

A. Background

In tree based group key management schemes, KDC stores KEKs in the hierarchical tree structure as depicted in Fig. 1(a) [1], [8]. The root node of the key tree is the group key and the other nodes of the key tree are KEKs.

Each group member m_i stores its individual key, group key, and all KEKs on the path between group key and individual key. In Fig. 1(a), $k_{1,1}$ is the group key and $k_{4,5}$ is the individual key of m_5. Hence, m_5 stores $k_{1,1}$, $k_{2,2}$, $k_{3,3}$, and $k_{4,5}$.

When m_i joins or leaves the group, KDC changes all the keys associated with m_i and delivers them to all group members except m_i.

In Fig. 1(a), when member m_3 leaves the group, $k_{3,2}$, $k_{2,1}$ and $k_{1,1}$ must be rekeyed. KDC generates $k'_{3,2}$, $k'_{2,1}$ and $k'_{1,1}$ and delivers them as follows:

1) Deliver $\{k'_{3,2}\}_{k_4}$ to m_4.
2) Deliver $\{k'_{2,1}\}_{k'_{3,2}}$ to m_4.
3) Deliver $\{k'_{2,1}\}_{k_{3,1}}$ to m_1, m_2.
4) Deliver $\{k'_{1,1}\}_{k'_{2,1}}$ to m_1, m_2, m_4.
5) Deliver $\{k'_{1,1}\}_{k_{2,2}}$ to m_5, m_6, m_7, m_8.

By above delivering sequence, new keys are delivered to all group members except m_3. Thus, m_3 is successfully revoked. On the other hand, when a new member joins the group, KDC and all group members change the keys by using one-way hash function without message delivery as described in [6], [9]. KDC just needs to deliver the changed KEKs and group key only to the new member by unicasting.

B. Problem Definition

Traditional tree based schemes [1], [6], [7], [8] mainly focus on reducing the number of the rekeying messages multicasted by KDC. To this end, KDC aggregates multiple rekeying messages into one multicast flow, which is referred to as group oriented rekeying [1]. In group oriented rekeying, all rekeying messages are delivered to all group members. This causes the bandwidth waste because rekeying messages are delivered to members who do not need them as well as intended receivers.

If relaying nodes in wireless mesh networks deliver rekeying messages only to group members who are related to the rekeying message, bandwidth waste can be decreased. From now, we refer to this approach as partial multicast. If the KDC knows the basic service set (BSS) [12] of all group members, partial multicast can be easily achieved by marking destination BSS in the header of rekeying messages.

Fig. 1(c) depicts a simple mesh network that consists of 7 backbone links and 4 BSS links. Circled number means the communication cost of each link. The network bandwidth for delivering a rekeying message to subgroup S_k, $c(S_k)$ is the sum of the wireless backbone and BSS link cost of the partial multicast routing tree. ($c(S_k)$ can be defined differently, based on type of application or network administration policy.)

For example, $c(S_{k_{3,3}})$ in Fig. 1(a) is computed as:

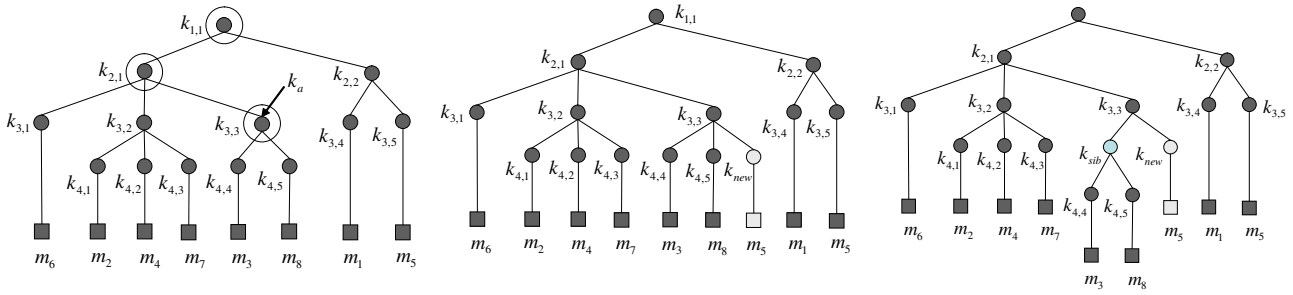
$$c(S_{k_{3,3}}) = c(\{m_5, m_6\}) = 3 + 2 + 5 + 1 + 2 + 1 + 2 + 2 = 18$$

When group member m_5 in Fig. 1(c) leaves the multicast group, the bandwidth consumption for delivering the rekey messages of key tree (a) is:

$$\begin{aligned} C_{m_5} =& c(S_{k_{4,6}}) + c(S_{k_{3,3}} \backslash m_5) + c(S_{k_{2,2}} \backslash m_5) + c(S_{k_{2,1}}) \\ =& 7 + 11 + 10 + 15 + 19 = 62 \end{aligned}$$

where we denote as $S_k \backslash m$, exclusion of m from S_k. The bandwidth consumption of rekeying when member m leaves the group in general form is expressed as:

$$C_m = \sum_{k \in kek(m)} \left(c(S_k \backslash m) + \sum_{k' \in sib(k)} c(S_{k'}) \right) \qquad (1)$$



(a) Before addition (b) Type 1 addition (c) Type 2 addition

Fig. 2. Member addition in ABR tree

By summing (1) for all members associated with the key tree T, we obtain:

$$C_T = \sum_{m \in M_T} C_m \tag{2}$$

The expected bandwidth consumption for rekeying when a member leaves the group is:

$$E[C_m] = \frac{1}{|M_T|} \cdot C_T \text{ (when } m \in M_T) \tag{3}$$

In Fig. 1, $E[C_m]$ of key tree (b) (56.5) is less than that of key tree (a) (73.25). From this, we can observe that tree (b) is more desirable than tree (a) in terms of the bandwidth efficiency. Especially, when the same rekeying message needs to be delivered to members in the same BSS, access point [12] can deliver that rekeying message by only one broadcast. Thus, when the members in the same BSS share common KEKs, we can exploit this broadcast nature of wireless networks. Hence our goal is to find a key tree that minimizes (3). Because $|M_T|$ is a constant, our goal is expressed as:

$$T_{min} = \arg\min_{T} C_T \tag{4}$$

III. ABR Tree

In this section, we introduce ABR (adaptive and bandwidth reducing) tree as a improvement of tree based group key management scheme for dynamic wireless mesh networks.

A. Basic Framework

In a dynamic membership environment, different members join and leave the group at different time. Thus, the key tree is constructed by repeating addition or deletion of members on every membership change. When a member joins or leaves the group, making the optimal key tree T_{min} may need the reconstruction of the key tree. However, reconstruction of the tree may cause significant rekeying overhead because the key assignment structure of the key tree is changed. Thus, ABR tree only performs addition and deletion of members without reconstruction in dynamic environment.

When a new member joins the group, Member addition is performed as follows:

1) KDC chooses the addition point k_a. And all ancestors of k_a are assigned to the new member. In Fig. 2(a), when $k_{3,3}$ is the k_a, $k_{3,3}$, $k_{2,1}$ and $k_{1,1}$ are chosen by KDC.
2) KDC rekeys the chosen keys by using one-way hash function as described in [6], [9].
3) KDC creates the new member's individual key, k_{new}.
4) KDC attaches k_{new} to k_a. There are two addition types.
   a) Type 1 addition: KDC adds k_{new} as a child of k_a as depicted in Fig. 2(b).
   b) Type 2 addition: KDC creates a new sibling key, k_{nsib}. Then it makes k_a have k_{nsib} and k_{new} as its children. And previous children of k_a have k_{nsib} as its new parent. That is depicted in Fig. 2(c). When k_a has no child, only type 2 addition is possible.
5) KDC delivers the chosen keys and k_{new} to the new member by unicast.

Too large breadth of the key tree leads to large number of the rekeying messages transmitted by KDC. Thus, ABR tree only performs the type 2 addition when number of the children of addition point is equal to the pre-determined maximum degree. This prevents the breadth of the key tree is too large.

For the leave events, deletion is performed as follows:

1) KDC deletes the individual key of the leaving member in the key tree as described in [1].
2) KDC rekeys the KEKs associated with the leaving member. Rekeying is done as described in Section II. At this time, the network bandwidth is consumed because of the rekeying message transmission.

In dynamic membership environments, our goal is to keep C_T as low as possible during multicast service time. In case of the leave event, KDC just performs deletion of the member. However, KDC has to determine a proper addition point and type in the join event. Because increase of C_T is determined by the newly assigned KEKs (the addition point and its ancestors) and addition type.

Let $\Delta(k_a, t_a)$ denote the increase of C_T when a new member is attached to k_a in type t_a ($\in$ {type 1, type 2}). Therefore, our final goal is expressed as following minimization problem:

$$(k_{min}, t_{min}) = \arg\min_{k_a, t_a} \Delta(k_a, t_a) = \arg\min_{k_a, t_a} (C_{T^{new}} - C_T) \tag{5}$$

where T^{new} is the new key tree after addition of the new member. By solving (5) for each join event, ABR tree keeps the increase of C_T low in dynamic membership environments.

B. The Increase of C_T in Dynamic Membership Environments

By observing characteristics of the key tree, we obtained the following from (2):

$$\begin{aligned} C_T &= \sum_{m\in M_T}\sum_{k\in kek(m)}\left(c(S_k\backslash m)+\sum_{k'\in sib(k)}c(S_{k'})\right) \\ &= \sum_{m\in M_T}\sum_{k\in kek(m)}c(S_k\backslash m)+\sum_{m\in M_T}\sum_{k\in kek(m)}\sum_{k'\in sib(k)}c(S_{k'}) \\ &= \sum_{k\in K_T}\sum_{m\in S_k}c(S_k\backslash m)+\sum_{k\in K_T}\sum_{k'\in sib(k)}\sum_{m\in S_{k'}}c(S_k) \\ &= \sum_{k\in K_T}\left(\sum_{m\in S_k}c(S_k\backslash m)+\sum_{k'\in sib(k)}|S_{k'}|\cdot c(S_k)\right) \end{aligned} \tag{6}$$

In (6), we define the bandwidth consumption associated with key k as $C_k = \sum_{m\in S_k} c(S_k\backslash m) + \sum_{k'\in sib(k)} |S_{k'}|\cdot c(S_k)$. Therefore, (6) can also be expressed as $C_T = \sum_{k\in K_T} C_k$.

Now, let δ_k denote the increase of the bandwidth consumption associated with k after addition of the new member:

$$\begin{aligned} \delta_k =& {C_k}^{new} - C_k \\ =& \sum_{m\in {S_k}^{new}} c({S_k}^{new}\backslash m) - \sum_{m\in S_k} c(S_k\backslash m) \\ &+ \sum_{k'\in sib(k)} (|{S_{k'}}^{new}|\cdot c({S_k}^{new}) - |S_{k'}|\cdot c(S_k)) \end{aligned} \tag{7}$$

where ${C_k}^{new}$ is the new bandwidth consumption associated with k and ${S_k}^{new}$ is the sub-group that consists of the members sharing k after addition of the new member. When k is not an ancestor of k_a, S_k is not changed because the new member is not a descendant of k. By using this fact, depending on the relation between k and k_a, (7) is expressed differently as follows:

1) When $k \in anc(k_a)\cup\{k_a\}$

$$\begin{aligned} \delta_k =& \sum_{m\in S_k\cup\{m_{new}\}} c(S_k\cup\{m_{new}\}\backslash m) - \sum_{m\in S_k} c(S_k\backslash m) \\ &+ \sum_{k'\in sib(k)} |S_{k'}|\cdot(c(S_k\cup\{m_{new}\}) - c(S_k)) \end{aligned} \tag{8}$$

2) When $k \in sib(k') \wedge k' \in anc(k_a)\cup\{k_a\}$

$$\delta_k = c(S_k) \tag{9}$$

3) When $k \notin sib(k')\cup anc(k_a)\cup\{k_a\} \wedge k' \in anc(k_a)\cup\{k_a\}$

$$\delta_k = 0 \tag{10}$$

As an example, consider the addition of the new member depicted in Fig. 2(b). $\delta_{k_{3,3}}$ and $\delta_{k_{2,1}}$ are calculated by (8). And $\delta_{k_{4,4}}$, $\delta_{k_{4,5}}$, $\delta_{k_{3,1}}$, $\delta_{k_{3,2}}$ and $\delta_{k_{2,2}}$ are calculated by (9). For all other KEKs, δ_k is 0 by (10).

Then, $\Delta(k_a, t_a)$ can be derived as:

$$\begin{aligned} \Delta(k_a,t_a) =& {C_T}^{new} - C_T \\ =& {C_{K_T}}^{new} - C_{K_T} \\ =& \sum_{k\in K_T{}^{new}} C_k - \sum_{k\in K_T} C_k \\ =& \sum_{k\in K_T{}^{new}-K_T} {C_k}^{new} + \sum_{k\in K_T} {C_k}^{new} - \sum_{k\in K_T} C_k \\ =& \sum_{k\in K_T{}^{new}-K_T} {C_k}^{new} + \sum_{k\in K_T}\delta_k \end{aligned} \tag{11}$$

In (11), $\sum_{k\in K_T{}^{new}-K_T} {C_k}^{new}$ is given by:

$$\sum_{k\in K_T{}^{new}-K_T} {C_k}^{new} = \begin{cases} C_{k_{new}} & ,(t = \text{type 1}) \\ C_{k_{new}} + C_{k_{nsib}} & ,(t = \text{type 2}) \end{cases} \tag{12}$$

By applying (7), (8), (9), (10) and (12) to (11), we obtain following equation:

$$\Delta(k_a,t_a) = \begin{cases} \sum_{k\in anc(k_a)\cup\{k_a\}} \Delta_k + \Theta_{k_a}^{T1} & ,(t = \text{type 1}) \\ \sum_{k\in anc(k_a)\cup\{k_a\}} \Delta_k + \Theta_{k_a}^{T2} & ,(t = \text{type 2}) \end{cases} \tag{13}$$

where $\Theta_{k_a}^{T1}$, $\Theta_{k_a}^{T2}$ and Δ_k are described as follows:

1) The partial increase of C_T when a k_{new} is attached to k_a by using type 1:

$$\Theta_{k_a}^{T1} = |S_{k_a}|\cdot c(m_{new}) + \sum_{k\in ch(k_a)} c(S_k) \tag{14}$$

2) The partial increase of C_T when a k_{new} is attached to k_a by using type 2:

$$\Theta_{k_a}^{T2} = |S_{k_a}|\cdot c(m_{new}) + \sum_{m\in S_{k_a}} c(S_k\backslash m) + c(S_{k_a}) \tag{15}$$

3) The partial increase of C_T when k is assigned to the new member:

$$\Delta_k = \delta_k + \sum_{k'\in sib(k)} c(S_{k'}) \tag{16}$$

C. Bandwidth Efficient Key Assignment Algorithm

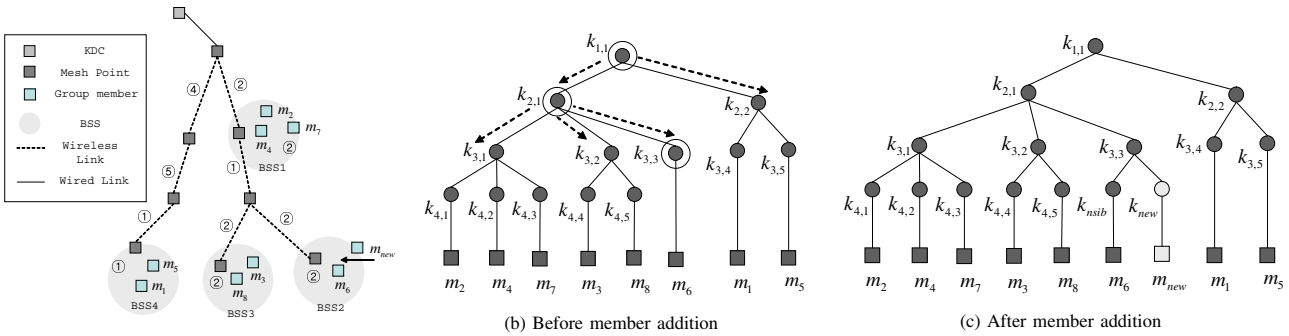
In the previous subsection, we show that the increase of C_T can be separated into terms of Δ_k, the partial increase of the C_T when k is assigned to the new member. We exploit this separability to solve (5) for each member addition procedure.

We construct a greedy algorithm to determine proper k_a and t_a when a new member joins the group. For each join event, our algorithm calculates Δ_k, Θ_k^{T1} and Θ_k^{T1} only $d\cdot O(logN)$ times, where d is the average degree of a key tree.

Our bandwidth efficient key assignment algorithm is described in Algorithm 1. Whenever a new member joins the group, ABR tree firstly determines k_a and t_a, and performs the member addition as described in Section III-A:

As an example, consider a network in Fig. 3(a) and a key tree in Fig. 1(b):

1) A new member m_{new} joins the group at the BSS 2.
2) For the group key $k_{1,1}$,
   a) $\Theta_{k_{1,1}}^{T1} = 80$ and $\Theta_{k_{1,1}}^{T2} = 268$.
   b) $\Delta(k_{1,1}$, type 1$) = \Delta_{sum} + \Delta_{k_{1,1}} + \Theta_{k_{1,1}}^{T1} = 268$.
   c) $\Delta(k_{1,1}$, type 2$) = \Delta_{sum} + \Delta_{k_{1,1}} + \Theta_{k_{1,1}}^{T2} = 80$.
   Δ_{sum} is 0 and next CK is $\{k_{2,1}, k_{2,2}\}$.
3) For $k_{2,1}$ and $k_{2,2}$,
   a) $\Delta_{k_{2,1}} = 28$, $\Theta_{k_{2,1}}^{T1} = 60$ and $\Theta_{k_{2,1}}^{T2} = 146$.
   b) $\Delta(k_{2,1}$, type 1$) = \Delta_{sum} + \Delta_{2,1} + \Theta_{k_{2,1}}^{T1} = 88$.
   c) $\Delta(k_{2,1}$, type 2$) = \Delta_{sum} + \Delta_{2,1} + \Theta_{k_{2,1}}^{T2} = 174$.
   d) $\Delta_{k_{2,2}} = 67$, $\Theta_{k_{2,2}}^{T1} = 36$ and $\Theta_{k_{2,2}}^{T2} = 47$.
   e) $\Delta(k_{2,2}$, type 1$) = \Delta_{sum} + \Delta_{2,2} + \Theta_{k_{2,2}}^{T1} = 103$.
   f) $\Delta(k_{2,2}$, type 2$) = \Delta_{sum} + \Delta_{2,2} + \Theta_{k_{2,2}}^{T2} = 114$.



(a) Example of mesh network

(b) Before member addition

(c) After member addition

Fig. 3. Example of bandwidth efficient key assignment algorithm

Algorithm 1 Bandwidth Efficient Key Assignment Algorithm

T_Δ: table storing $\Delta(k,t)$ for a candidate KEK k.
$MaxDegree$: maximum degree of key node.
$\Delta_{sum} \leftarrow 0$, $\Delta_{min} \leftarrow 0$
$CK \leftarrow$ {group key of the key tree T}.
while $CK \neq \phi$ **do**
  $\Delta_{min} \leftarrow \infty$, $k_{next} \leftarrow NULL$
  for all $k \in CK$ **do**
    $\Delta_k \leftarrow 0$
    if k is not a group key **then**
      compute Δ_k by using (16).
    end if
    if $\Delta_{min} > \Delta_k$ **then**
      $\Delta_{min} \leftarrow \Delta_k$, $k_{next} \leftarrow k$
    end if
    if $|ch(k)| \leq MaxDegree$ **then**
      compute Θ_k^{T1} by using (14).
      $\Delta(k,$ type 1$) \leftarrow \Delta_{sum} + \Delta_k + \Theta_k^{T1}$.
    end if
    compute Θ_k^{T2} by using (15).
    $\Delta(k, type2) \leftarrow \Delta_{sum} + \Delta_k + \Theta_k^{T2}$.
    store $\Delta(k,$ type 1$)$ and $\Delta(k,$ type 2$)$ in T_Δ
  end for
  $\Delta_{sum} \leftarrow \Delta_{sum} + \Delta_{min}$, $CK \leftarrow ch(k_{next})$
end while
$(k_a, t_a) \leftarrow (k,t)$ such that $\Delta(k,t)$ is minimum in T_Δ.

Between Δ_k of $k_{2,1}$ and $k_{2,2}$, $\Delta_{k_{2,1}}$ is less than $\Delta_{k_{2,2}}$. Thus, k_{next} is $k_{2,1}$ and $\Delta_{sum} = \Delta_{sum} + \Delta_{k_{2,1}} = 0 + 28 = 28$. Next CK is $\{k_{3,1}, k_{3,2}, k_{3,3}\}$.

4) Δ_k, Θ_k^{T1}, Θ_k^{T2}, $\Delta(k,$ type 2$)$ and $\Delta(k,$ type 2$)$ are calculated for $\{k_{3,1}, k_{3,2}$ and $k_{3,3}\}$ as in the previous step. Because $\Delta_{k_{3,3}}$ is the smallest, k_{next} is $k_{3,3}$.
5) The algorithm is finished because $k_{3,3}$ is a leaf. $\Delta(k,t)$ of all candidate keys are computed as in Table II. (We also present the Δ_k, Θ_k^{T1} and Θ_k^{T2} in Table II for convenience.) In Table II, $\Delta(k_{3,3},$ type 2$)$ is the smallest. Therefore, $k_{3,3}$ is selected as the addition point. In other words, $k_{3,3}$ and $k_{2,1}$ are assigned to the new member as KEKs. And k_{new} is attached by using type 2 as depicted in Fig. 3(c).

TABLE II
TABLE T_Δ OF FIG. 3

k	Δ_k	Θ_k^{T1}	Θ_k^{T2}	$\Delta(k,$ type 1$)$	$\Delta(k,$ type 2$)$
$k_{1,1}$	0	80	268	80	268
$k_{2,1}$	28	60	146	88	174
$k_{2,2}$	67	36	47	103	114
$k_{3,1}$	48	33	37	109	113
$k_{3,2}$	42	28	35	98	105
$k_{3,3}$	25	-	14	-	67

IV. Simulation Results

We performed discrete time event driven simulation to compare the bandwidth consumption of ABR tree with VP3 (online version) [3] and conventional key tree [1]. Our simulator was developed in C++ and MATLAB.

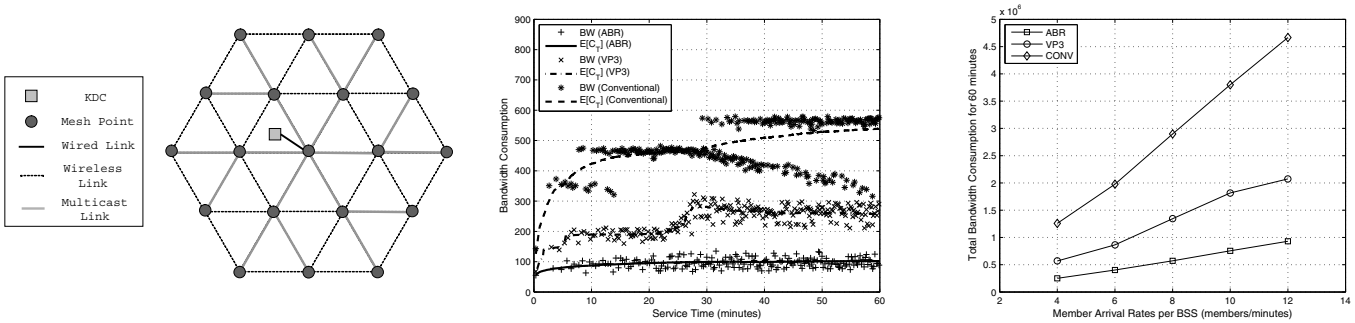
We deployed a mesh network that consists of 19 mesh points (MPs) as depicted in Fig. 4(a). This is a triangular deployment that is proper to the urban area mesh network [11]. KDC is connected to MP at the center of the network through wired link. Except MP at the center, all MPs play a role of the access point that provides the wireless connection to the group members. We assume the following in our simulation:

1) All costs of wireless backbones and BSS links are 1.
2) All MPs support partial multicast.
3) The maximum degree of ABR tree is 4.
4) VP3 and conventional tree use the tree of degree-3 that shows the best performance in terms of the bandwidth efficiency as described in [3].

For simulating dynamic membership environments, we use the Poisson arrival and exponential residence time model to describe the group members' behavior. For each leave event, we measured $c(S_k)$ of all rekeying messages transmitted by KDC. That is the network bandwidth used for delivering rekeying messages. And we also calculated $E[C_m]$ of key trees at each join and leave event.

Fig. 4(b) shows the actual bandwidth consumption of rekeying and E[C_m] for each scheme, when the member join rate (λ) is 8 members per minutes per BSS and mean residence time ($1/\mu$) is 20 minutes. From this, we can observe that actual bandwidth consumptions used for rekeying are closely fit to the $E[C_m]$ ($\propto C_T$). This shows that minimizing C_T is a appropriate strategy to reduce the bandwidth consumption that is actually used for rekeying. The notable thing here is that the deviation of C_m of conventional key tree is higher than other schemes. This is because conventional tree does not do anything to reduce the bandwidth consumption of rekeying. During 60 minutes, $E[C_m]$ of ABR tree is always less than other schemes. Even, it achieves a more than 50% reduction compared to VP3.

Fig. 4(c) shows the total bandwidth consumption used for



(a) Network topology

(b) Actual bandwidth consumption and $E[C_m]$ (λ = 8 and $1/\mu$ = 20)

(c) Comparision of total bandwidth consumption ($1/\mu$ = 20)

Fig. 4. Network topology and simulation results

rekeying according to the member join rate (λ), when the mean residence time is 20 minutes and simulation duration is 60 minutes. All results are averaged over 30 trials. For all join rates, ABR tree achieves around 80% reduction of total bandwidth consumption used for rekeying compared to conventional tree and around 50% reduction compared to VP3.

From above results, we can see that our scheme outperforms VP3 in keeping the increase of $E[C_m]$ low in dynamic membership environments. This is because VP3 does not reflect the expected bandwidth consumption directly to the key tree in the member addition procedure. On the other hand, our scheme computes the increase of $E[C_m]$ accurately based on separability of C_T and use it in the member addition procedure. Therefore, we note that the accurate estimation about the increase of C_T is the main reason of achieving the further bandwidth reduction from VP3.

V. Conclusion

In this paper, we present analytical formulations that describe the bandwidth consumption used for rekeying of the key tree. Based on this analysis, we also presents the ABR tree that effectively reduces the network bandwidth consumption used for rekeying in dynamic wireless mesh networks. The most essential part of our scheme is that KDC determines the proper KEKs that minimize the increase of the expected bandwidth consumption of rekeying and assigns them to the new member for each member join event. By this approach, ABR tree keeps the expected bandwidth consumption as low as possible. Simulation results show that our strategy is effective for reducing the actual bandwidth consumption of rekeying.

Acknowledgement

This work was supported by the the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (2007-C1090-0701-0015) and the IT R&D program of MIC/IITA (2005-S609-02, Development of Core Technologies for Next Generation Mobile Multimedia Platform)

References

[1] C. Wong, M. Gouda, and S. Lam "Secure group communications using key graphs", *IEEE/ACM Trans. Networking*, vol. 8, pp. 16-30, Feb. 2000

[2] Y. Sun, W. Trappe, and K. Liu "A scalable multicast key management scheme for heterogeneous wireless networks", *IEEE/ACM Trans. Networking*, vol. 12, pp. 653-666, Aug. 2004

[3] J. Salido, L. Lazos, and R. Poovendran "Energy and bandwidth-efficient key distribution in wireless ad hoc networks: a cross-layer approach", *IEEE/ACM Trans. Networking*, vol. 15, pp. 1527-1540, Dec. 2007

[4] L. Lazos, and R. Poovendran "Power proximity based key management for secure multicast in ad hoc networks", *ACM/Springer J. Wireless Networks (WINET)*, vol. 13, no. 1, pp. 127-148, Feb. 2007

[5] L. Lazos and R. Poovendran "Cross-layer design for energy-efficient secure multicast communications in ad hoc networks", in proc. *IEEE Int. Conf. Communications, ICC 2004*, Paris, France, May 2004, pp. 3633-3639

[6] D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: One-way function trees and amortized initialization," *Internet Draft Report*, Feb. 2000.

[7] A. Perrig, D. Song, and J. Tygar, "ELK, A new protocol for efficient large-group key distribution," In proc. *IEEE Symposium on Security and Privacy*, Oakland, CA, May. 2001, pp. 247-262.

[8] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architectures," *Internet Draft (RFC 2627)*, Sep. 1998.

[9] R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in proc. *IEEE INFOCOM*, vol. 2, Mar. 1999. pp. 708-716

[10] W. Trappe, J. Song, R. Poovendran, and K. Liu, "Key management and distribution for secure multimedia multicast," *IEEE Trans. Multimedia*, vol. 5, no. 4, pp. 544-557, Dec. 2003.

[11] J. Robinson, and E. Knightyly, "A Performance study of deployment factors in wireless mesh networks," in proc. of *IEEE INFOCOM 2007*, Anchorage, AK, pp. 2054-2062, May, 2007.

[12] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Nov. 1997. P802.11