Securing Dynamic Membership Information in Multicast Communications

Yan Sun and K. J. Ray Liu
Department of Electrical and Computer Engineering
and the Institute of Systems Research
University of Maryland, College Park, MD 20742
Email: ysun, kjrliu@glue.umd.edu

Abstract— **In secure multicast communications, key management schemes are employed to prevent unauthorized access to multicast content. Key management, however, can disclose the information about the dynamics of the group membership, such as the group size and the number of join and departure users, to both inside and outside attackers. This is a threat to applications with confidential group membership information. This paper investigates attack/anti-attack strategies for stealing/protecting group dynamic information in the existing key management schemes. We show that attackers can successfully acquire the membership information by exploiting the key updating procedure in popular centralized key management schemes. Particularly, we develop two attack strategies and demonstrate their effectiveness through simulations. Further, we propose an anti-attack technique utilizing batch rekeying and phantom users, and derive performance criteria that describe the security level of the proposed scheme using mutual information. The proposed anti-attack scheme is evaluated based on the data obtained from real MBone sessions.**

I. Introduction

The rapid progress in the technologies underlying multicast networking has led to the development of many multicast services, such as streaming stock quotes, video conferencing and communal gaming [1]. Before these group-oriented multicast applications can be successfully deployed, *access control* mechanism must be developed such that only authorized users can access the group communication [2] [3]. Access control is usually achieved by encrypting the content using an encryption key, known as the session key (SK) that is shared by all legitimate group members. Since the group membership will most likely be dynamic with users joining and leaving the service, it is necessary to update the encryption keys in order to prevent the leaving user from accessing future communication and prevent the joining user from accessing prior communication [2] [3]. The issues of establishing and updating the group keys are addressed by group *Key Management* schemes [2].

Key management schemes can be classified as centralized schemes and contributory schemes [4]. In centralized schemes, such as [3]–[12], group members trust a centralized server, referred to as the key distribution center (KDC), which generates and distributes decryption keys. In contributory schemes, such as [13]–[21], group members are trusted equally and all participate the formation of the group key.

Both centralized and contributory key management schemes address the problem of maintaining access control with dynamic membership and reducing the usage of computation, communication and storage resources [2] [3] [16]. These schemes, however, did not consider the disclosure of information about the dynamics of the group membership to both insiders and outsiders. We collectively refer to *group dynamics information* (GDI) as information describing the dynamic membership of a group application, such as the number of users in the multicast group as a function of time, and the number of users who join or leave the service during a time interval.

In many group communications, group dynamic information is confidential and should not be disclosed to either valid group members or outsiders. For example, in a commercial multicast program, the service provider performs group management and has the knowledge of GDI. Although the service provider may release some audience statistics at his choosing time, it is highly undesirable to disclose instant detailed dynamic membership information to competitors, who would develop effective competition strategies by analyzing the statistical behavior of the audience. Another example is a military group communication scenario, where GDI represents the number of soldiers in the battlefield and the number of soldiers moving into or out of certain areas. In this situation, the valid group members, i.e. regular soldiers, may only be entitled to obtain general information through the secure group communication, but not entitled to acquire GDI. Leaking GDI to outsiders, most likely to the enemies, can be devastating.

The traditional key management schemes are designed to prevent unauthorized access to the multicast content, but unfortunately also provide opportunities for unauthorized parties to obtain group dynamic information. The dynamic group membership information can be revealed unknowingly while performing key management. With the proliferation of access control in many applications, such a new security concern amply arises. Therefore, it is important to investigate this new threat and improve the design of current key management schemes such that both the group dynamic information and the multicast content are protected.

Contributory key management schemes are generally not suitable for the applications with confidential GDI because each group member need to be aware of other group members in order to establish the shared group key in the distributed manner. In this paper, we will focus on centralized schemes.

0-7803-8355-9/04/$20.00 ©2004 IEEE.

We demonstrate that the centralized key management schemes can reveal the GDI easily and propose a framework of protecting GDI from inside and outside attackers. We have developed two effective strategies to attack and *steal* information about the membership dynamics from the tree-based centralized schemes [2]–[7] that employ tree hierarchy for the maintenance of keying material. These strategies involve exploiting the format of rekey messages and estimating GDI directly from the size of the rekey messages. We also developed an anti-attack method that is fully compatible with the existing key management schemes. By utilizing batch rekeying [22] and introducing phantom users, the proposed anti-attack method aims to minimize the mutual information between the rekeying process observed by the attackers and the true group dynamics. Various aspects of the proposed anti-attack scheme, such as the communication overhead and the leakage of GDI, are evaluated based on the data obtained from MBone sessions. The analysis on other non-tree based schemes is also provided.

The rest of the paper is organized as follows. The attack strategies and the anti-attack method for the centralized schemes are presented in Section II and Section III respectively. In Section IV, the performance criteria of the proposed anti-attack method are derived and the optimization problem is formulated. Simulation results based on the user log data from real MBone sessions are shown in Section V, followed by the conclusion in Section VI.

II. GDI Attacks on Centralized Key Management Schemes

In this section, we investigate the attack strategies that aim to attack the centralized key management schemes for obtaining the dynamic group information. In this work, the group dynamic information (GDI) particularly refers to a set of functions as:

- $N(t)$: the number of users in the multicast group at time t
- $J(t_0, t_1)$: the number of users who join the service between time t_0 and t_1.
- $L(t_0, t_1)$: the number of users who leave the service between time t_0 and t_1.

The GDI should be kept confidential in many group-oriented applications, yet to acquire GDI by launching attacks on the key management schemes can be very simple as we will demonstrate. Instead of trying to break the encryption or compromise the key distribution center, the adversaries can subscribe to the service as regular users. In this case, they are referred to as the *inside attackers*. As we will show later in this section, inside attackers can obtain very accurate estimation of GDI by monitoring the messages conveying new key updating information, referred to as the *rekey messages*. Even if the adversaries cannot become valid group members, they still have the opportunities of stealing GDI as *outside attackers* as long as they can observe the traffic and distinguish the rekey messages and other data.

In this section, we consider a popular tree-based centralized key management scheme proposed in [6], then present two attack strategies for inside and outside attackers, and finally discuss the vulnerability of other prevail centralized key management schemes.

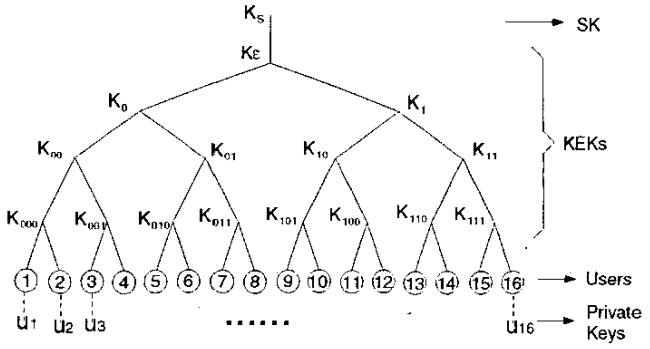

Fig. 1. A typical key management tree

A. Tree-based centralized key management schemes

Similar to other tree-based schemes [2]–[7], the centralized Versakey scheme in [6] employs a key tree to maintain the keying material. As illustrated in Figure 1, each node of the key tree is associated with a key. The root of the key tree is associated with the session key (SK), K_s, which is used to encrypt the multicast content. Each leaf node is associated with a user's private key, u_i, which is only known by this user and the KDC. The intermediate nodes are associated with key-encrypted-keys (KEK), which are auxiliary keys and only for the purpose of protecting the session key and other KEKs. To make concise presentation, we do not distinguish the node and the key associated with this node in the remainder of the paper.

Each user stores his private key, the session key, and a set of KEKs on the path from himself to the root of the key tree. In the example shown in Figure 1, user 16 possesses $\{u_{16}, K_s, K_\epsilon, K_1, K_{11}, K_{111}\}$. The notation x^{old} represents the old version of key x, x^{new} represents the new version of key x, and $\{y\}_x$ represents the key y encrypted by key x.

When a user leaves the service, all his keys need to be updated in order to prevent him from accessing the future communication. According to [6], when user 16 leaves, the KDC generates new keys and conveys new keys to the remaining users through a set of rekey messages as:

- $\{K_{111}^{new}\}_{u_{15}}$: user 15 acquires K_{111}^{new}.
- $\{K_{11}^{new}\}_{K_{111}^{new}}, \{K_{11}^{new}\}_{K_{110}^{old}}$: user 13,14,15 acquire K_{11}^{new}.
- $\{K_1^{new}\}_{K_{11}^{new}}, \{K_1^{new}\}_{K_{10}^{old}}$: user $9, \cdots, 15$ acquire K_1^{new}.
- $\{K_\epsilon^{new}\}_{K_1^{new}}, \{K_\epsilon^{new}\}_{K_0^{old}}$: user $1, \cdots, 15$ acquire K_ϵ^{new}.
- $\{K_s^{new}\}_{K_\epsilon^{new}}$: all remaining users acquire K_s^{new}.

This key updating procedure guarantees that all remaining users obtain the new session key and KEKs, while user 16 is unable to acquire the new keys. Since the rekey messages are transmitted in the multicast channel [5], every user receives

all rekey messages although not all messages are useful for everyone. The session key, KEKs and users' private keys usually have the same length. The communication overhead associated with key updating can be described by *rekey message size*, defined as the amount of rekey messages measured in the unit as the same size as SK or KEKs. In this example, the rekey message size is 8 when user 16 leaves the service. It has been shown that the rekey message size increases linearly with the logarithm of the group size [6].

When a user joins the service, the KDC chooses a leaf position on the key tree to put the joining user. In [6], each key is associated with a revision number. The KDC updates the keys along the path from the new leaf to the root by generating the new keys from the old keys using a one-way function and increasing the revision numbers of the new keys. The joining user obtains the new keys through the unicast channel. Other users in the group will know about the key change when the data packet indicating the increase of the revision numbers first arrives, and compute the new keys using the one-way function. No additional rekey messages are necessary.

The rekeying procedure although has some differences, most tree-based centralized key management schemes [2]–[7] share two common properties. First, group members can distinguish the key updating process due to user join and that due to user departure. Second, rekey message size is closely related with the group size. Due to these properties, the attackers can estimate $J(t_0, t_1)$ and $L(t_0, t_1)$ by examining the rekey processes, and estimate $N(t)$ directly from the rekey messages size. Next, we illustrate these two types of attacks on the key management scheme presented in [6].

B. Attack A1: Estimating the number of join/departure users by inside attackers

An inside attacker, like other regular users, processes K_s, K_ϵ, and a set of KEKs. He receives rekey messages, decrypts the messages that are encrypted by his keys, and observes the rekey message size without having to understand the content of all messages. Since the key updating process for user join and the process for user departure are different, he can estimate $J(t_0, t_1)$ and $L(t_0, t_1)$ using the following strategy:

- When receiving the rekey message containing K_ϵ^{new} encrypted by one of his KEKs, he assumes that one user leaves the service.
- When observing the increase of the revision number of K_ϵ, he assumes that one user joins the service.

This strategy is effective when most users do not join/leave simultaneously and the keys are updated immediately once each user join/departure. Otherwise, more complicated techniques involving examining the rekey message size shall be used. When this attack is successful, $N(t)$ can be calculated from $J(t_0, t_1)$ and $L(t_0, t_1)$ as:

$$N(t_1) = N(t_0) + J(t_0, t_1) - L(t_0, t_1). \quad (1)$$

Even if the attacker do not know the initial value of the group size, he obtains the changing trend of the group size.

C. Attack A11: Estimation of group size from rekey message size

Besides using (1), the group size $N(t)$ can also be estimated directly from the rekey message size. We will derive a Maximum Likelihood estimator for the attackers and then demonstrate the effectiveness of this estimator through simulations.

We assume that $N(t)$ does not change much within a short period of time. In this time period, there are W departure users who do not leave simultaneously. Thus, the attacker makes W observations of the rekey message size due to single user departure, denoted by $Msg = \{m_1, m_2, \cdots, m_w\}$.

Similar to most key management schemes [2]–[6], the key tree investigated in this work is fully loaded and maintained as balanced as possible by putting the joining users on the shortest branches. In the worst-case scenario, the attacker knows this property and the degree of the key tree, denoted by d. Then, the attacker can calculate the depth of the branch where the i^{th} leaving user was located before departure, denoted by L_i. Without losing information, the observed Msg is converted to $\{L_1 = l_1, L_2 = l_2, \cdots, L_W = l_W\}$, where $l_i = \lceil \frac{m_i+1}{d} \rceil$. Then, the Maximum Likelihood (ML) estimator is formulated as:

$$N_{ML} = \arg\max_n \; Prob\{L_i = l_i, \; i = 1, \cdots, W \mid N(t) = n\}. \quad (2)$$

To solve (2), we introduce a set of new variables: $\{S_k\}_{k=L_{min}, L_{min}+1, \cdots, L_{max}}$, where S_k is the number of users who are on the branches with length k, L_{max} is the length of the longest branches, and L_{min} is the length of the shortest branches. It is obvious that

$$\sum_k S_k = n. \quad (3)$$

In addition, the length of the branches of a key tree must satisfy the Kraft inequality [23], i.e. $\sum_j d^{L_{max}-b_j} \leq d^{L_{max}}$, where b_j is the length of the branch on which the user j stays and $j = 1, 2, \cdots, n$. Thus, S_k, which equals to the number of elements in set $\{b_j : b_j = k\}$, must satisfy

$$\sum_k S_k d^{L_{max}-k} \leq d^{L_{max}}, \quad (4)$$

It can be verified that the equality is achieved when all intermediate nodes on the key tree have d children nodes. When the key tree is balanced and fully loaded, it is reasonable to approximate (4) by

$$\sum_k S_k d^{L_{max}-k} = d^{L_{max}}. \quad (5)$$

We assume that the leaving users are uniformly distributed on the key tree, and the number of users in the system is much larger than the number of leaving users, i.e. $N(t) >> W$. Then, the probability mass function (pmf) of L_i is

$$Prob\{L_i = k \mid n, \; S_k\} = \frac{S_k}{n}, \quad k = L_{min}, \cdots, L_{max}.$$

We assume that $L_i, i = 1, \cdots, W$ are i.i.d. random variables. Thus, the probability in (2) is calculated as:

$$Prob\{L_i = l_i,\ i = 1, \cdots, W \mid N(t) = n,\ S_k\} = \prod_k \left(\frac{S_k}{n}\right)^{h(k)} \tag{6}$$

where $h(k)$ denotes the number of elements in set $\{l_i : l_i = k\}$ and obviously, $\sum_k h(k) = W$. Then, the values of n and $\{S_k\}$ that maximize (6) under the constraint (3) and (5) are obtained using Lagrange multiplier as:

$$\{S_k\}_{ML} = \frac{n}{W} h(k) \tag{7}$$

$$N_{ML} = \frac{W}{\sum_k h(k) d^{-k}} \tag{8}$$

This ML estimator was applied to simulated multicast services. As suggested in [24] [25], the user arrival process is modelled as poisson process, and the service duration is modelled as an exponential random variable. In Figure 2(a), 2(b), and 2(c), the estimated group size is obtained by using the estimator in (8), and compared with the true values of $N(t)$. These three plots are for different simulation settings. The entire service period is divided into four sessions. The model parameters, i.e. user arrival rate and average service time, are fixed within each session and vary in different sessions. In the i^{th} session, described by interval $[t_{i-1}, t_i)$, the user arrival rate is λ_i and the average service time is μ_i. In all three cases, $[t_0, t_1, t_2, t_3, t_4]$ is chosen to be $[0, 200, 1600, 3200, 5000]$ minutes, and the initial group size is 0. The parameter λ_i's and μ_i's are given in Figure 2. In addition, Figure 2(d) demonstrates the performance of the ML estimator, when it was applied to a real MBone audio session, CBC Newsworld on-line test, starting on Oct. 29, 1996 and lasted for about 5 days [26].

In all four cases, the changing trend of the group size is well captured by the attacker. It is also observed that the estimated group size tends to be larger than the true $N(t)$, which is due to the approximation that we replace (4) by (5). Although not perfect, this estimator is effective in helping the attackers to achieve many of their goals, such as analyzing audience behavior and monitoring the group size changes.

The inside attackers can launch both attack AI and AII. They obtain $J(t_0, t_1)$ and $L(t_0, t_1)$ using AI, and the initial value $N(t_0)$ using AII. Then, $N(t)$ can be obtained by using either (1) or (8), or jointly.

It has been shown that the rekey messages must be delivered reliably and in a timely manner in order to guarantee the quality of service [27]. Therefore, it is possible that rekey messages are treated differently from the regular data in terms of error control, or even transmitted in a reliable multicast channel separated from the channel used for transmitting multicast content. This provides an opportunity for outsiders to separate the rekey messages and the multicast content. Thus, the outsiders may also launch attack AII directly by monitoring the transmission of the rekey messages.

It should be noted that the performance of the attack AI and AII degrades when many users join/leave simultaneously.

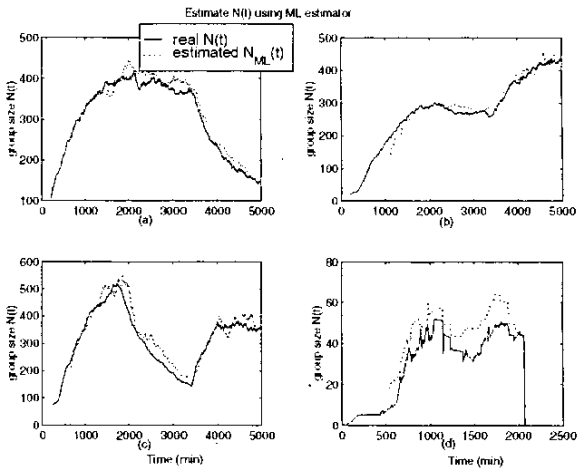

Fig. 2. Performance of the ML estimator.
In plot (a), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4]$= $[0.5, 0.5, 0.5, 0.3]\text{min}^{-1}$ and $[\mu_1, \mu_2, \mu_3, \mu_4]$= $[1400, 800, 600, 400]$min.
In plot (b), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4]$= $[0.1, 0.3, 0.2, 0.5]\text{min}^{-1}$ and $[\mu_1, \mu_2, \mu_3, \mu_4]$= $[1500, 1500, 1000, 800]$min.
In plot (c), $\lambda_1, \lambda_2, \lambda_3, \lambda_4]$= $[0.3, 0.7, 0.1, 0.9]\text{min}^{-1}$ and $[\mu_1, \mu_2, \mu_3, \mu_4]$= $[1400, 800, 600, 400]$min.
Plot(d) is based on the user log file from a real MBone session.

It will be shown in Section III that the rekey message size still reveals a significant amount of information on GDI even when multiple users are removed from or added to the key tree together.

D. Vulnerability of prevail centralized key management schemes

The attack methods described in Section II-B and II-C can be tailored to many other key management schemes. When the inside attacker can separate the rekey messages for user join and those for user departure, they launch *AI type attacks*. When the amount of rekey messages is largely depends on the group size, attackers can launch *AII type attacks*, although the estimator may be slightly different from (8). In this section, we review several key management schemes and discuss their vulnerability to AI and AII type attacks.

Since protecting GDI is not part of the design goal in traditional key management schemes, it is not surprising that some schemes reveal GDI in a very direct way. For example, in the approach proposed in [10], a security lock is implemented based on the Chinese remainder theorem and the length of the lock is proportional to the number of users. Thus, $N(t)$ is obtained by measure the length of the lock, which is the simplest AII type attack.

Tree-based key management schemes have been known for their efficiency in terms of the usage of communication, computation and storage resources. Many tree-based schemes, such as [3], [5]–[7], are similar to that described in Section II-A. In these cases, both AI and AII type attacks can be applied. In [4] [8] [9], another class of tree-based schemes were presented to further reduce the communication overhead

Centralized Key Management Schemes		Is attack AII Effective?	Is Attack AI Effective?
Tree Based	Key Graph [5], Wallner98 [3], Tree-based scheme in VersaKey framework [6] Embedding [7]	Yes	Yes
	One-way function tree [8] Improve Key Revocation [4] ELK [9]	Yes	No
Flat	Security lock [10]	Yes	–
	Flat centralized scheme in VersaKey framework [6]*	No	No
Local security agents	Iolus [11]	Local	Local
	Clustering [12]*	No	No
Others	TMKM [28]	Local	Local

TABLE I

VULNERABILITY OF PREVAIL CENTRALIZED KEY MANAGEMENT SCHEMES

by introducing the dependency among keys, such as using one-way function trees. In these schemes, only AII type attacks are suitable.

Besides the tree-based scheme described in Section II-A, VersaKey framework [6] also includes a centralized flat scheme. When a user joins or leaves the group, the rekey message size equals to the length of the binary representation of the user ID, which can be independent of $N(t)$. Thus, this key management scheme is resistant to both AI and AII type attacks. This scheme, however, is vulnerable to collusion attacks. That is, the KDC cannot update keys without leaking new key information to the leaving user, who has a collusion partner in the group. Although the GDI is protected, this scheme cannot protect the multicast content well when collusion attacks are likely.

In Iolus [11], a large group is decomposed into a number of subgroups, and the trusted local security agents perform admission control and key updating for the subgroups. This architecture reduces the number of users affected by key updating due to membership changes. Since the key updating is localized within each subgroup, the attacker can only obtain the dynamic membership information of the subgroup that he belongs to.

The idea of Clustering was introduced in [12] to achieve the efficiency by localizing the key updating. The group members are organized into a hierarchical clustering structure. The cluster leaders are selected from group members and perform partial key management. Since the cluster leaders establish keys for the cluster members through pair-wise key exchange [12], the cluster members cannot obtain GDI of their clusters. However, the cluster leaders naturally obtain the dynamic membership information of their cluster and all clusters below by participating key management. In [12], the cluster size is chosen from 3 to 15. Therefore, this key management scheme can be applied only when a large potion of group members are trusted to perform key management and obtain GDI.

In [28], a topology-matching key management (TMKM) scheme was presented to reduce the communication overhead associated with key updating by matching the key tree with the network topology and localizing the transmission of the rekey messages. In this scheme, group members receive only the rekey messages that are useful for themselves and their neighbors. Thus, they only obtains the local GDI by using AI or AII type attacks.

As a summary, Table I lists various key management schemes and their vulnerability to AI and AII type attacks. We can see that the AII type attacks are effective for stealing GDI or local GDI from many key management schemes. Two schemes, flat VersaKey [6] and the clustering [12], are resistant to these attacks. Their usage, however, are limited by the fact that they are either not resistant to collusion attacks or must put trust upon a large number of cluster leaders. Therefore, it is very important to investigate the anti-attack techniques to protect group dynamic information that are compatible with a variety of key management schemes.

III. ANTI-ATTACK TECHNIQUES

We have discussed two types of attacks that can steal GDI from centralized key management schemes. This discussion, however, does not cover all aspects of the key management schemes that can reveal group dynamic information. For example, the number of KEKs possessed by the inside attacker equals to the depth of the key tree and reveals at least the order of the group size. We can also show that the IDs of the keys reveal the structure of the key tree. Thus, new attack methods may emerge in the future. Therefore, we propose an anti-attack framework that is robust to various types of attacks and compatible with most centralized key management schemes.

We first introduce the concept of *Batch Rekeying* that plays an important role in our anti-attack technique. As proposed in [22], batch rekeying is to postpone the updates of keys such that several users can be added to or removed from the key tree altogether. Compared with updating keys immediately after each user join or departure, batch rekeying reduces the communication overhead at the expense of allowing the joining/leaving user to access a small amount of information before/after his join/departure.

In this work, batch rekeying is implemented as periodic updating of keys and the time between key updates are fixed and denoted by B_t. Particularly, the users who join or leave the group in the time interval $[(k-1)B_t, kB_t]$, are added to or removed from the key tree together at time kB_t. Then, the

notations of GDI functions are simplified as: $J(k) = J((k-1)B_t, kB_t)$, $L(k) = L((k-1)B_t, kB_t)$, and $N(k) = N(kB_t)$.

Since the AI type attacks are effective only when users are added to or removed from the key tree individually, utilizing batch rekeying can fight against the AI type attacks. However, batch rekeying alone is not enough to fight against the AII type attacks. Figure 3 shows some simulation results for the batch rekeying when B_t is set to be 5 minutes. Simulation setup is similar to that in Section II-C. The solid line in Figure 3(a), 3(b), 3(c), 3(d) represent the $N(k)$, $J(k)$, $L(k)$ and the rekey message size, respectively. One can see that the rekey message size is closely related to $L(k)$ and reflects the trend of $N(k)$. A large amount of information about $N(k)$ and $L(k)$ can be obtained by the attackers from examining the rekey message size.

Besides using batch rekeying, we propose to insert phantom users into the system. These phantom users, as well as their join and departure behavior, are created by the KDC in such a way that the combined effects of the phantom users and the real users lead to a new rekeying process, called *observed rekeying process*, which is observed by the attackers. An important goal is for the system to produce an observed rekeying process that reveals the least amount of information about the GDI.

Let $N_a(k)$ denote the total number of the real and phantom users, and $J_a(k)$ and $L_a(k)$ denote the total number of the real and phantom users who join/leave the service respectively. $N_a(t)$, $J_a(k)$, and $L_a(k)$ are referred to as the *artificial GDI*. From the key management points of view, the phantom users are treated the same as the real users. They occupy leaf nodes on the key tree, and they are associated with a set of KEKs that are updated when they virtually join or leave the group. Thus, the observed rekeying process only depends on the artificial GDI.

We first consider choosing the artificial GDI as a set of constant functions, that is,

$$J_a(k) = L_0, \quad L_a(k) = L_0, \quad N_a(k) = N_0. \tag{9}$$

By doing so, the observed rekeying process does not leak the information about the changing trend of the real GDI. However, the perfect flat artificial GDI functions in (9) may not be achievable. Since the real GDI functions are random processes, it is possible that the predetermined L_0 and Y_0 are not large enough such that the artificial GDI cannot be maintained as straight lines. For example, when $N(k) > N_0$, $N_a(k)$ cannot be the predetermined value N_0 because the number of phantom users must be non-negative. In fact, the artificial GDI functions must satisfies four requirements: (r1) $N_a(k) \geq N(k)$, (r2) $L_a(k) \geq L(k)$, (r3) $J_a(k) \geq J(k)$, and (r4) $N_a(k) = N_a(k-1) + J_a(k) - L_a(k)$. In this work, we choose the artificial GDI functions as:

$$N_a(k) = \max\{N(k), N_0\} \tag{10}$$

$$J_a(k) = \max\{J(k), L(k), L_0\} \tag{11}$$

$$L_a(k) = N_a(k-1) - N_a(k) + J_a(k) \tag{12}$$

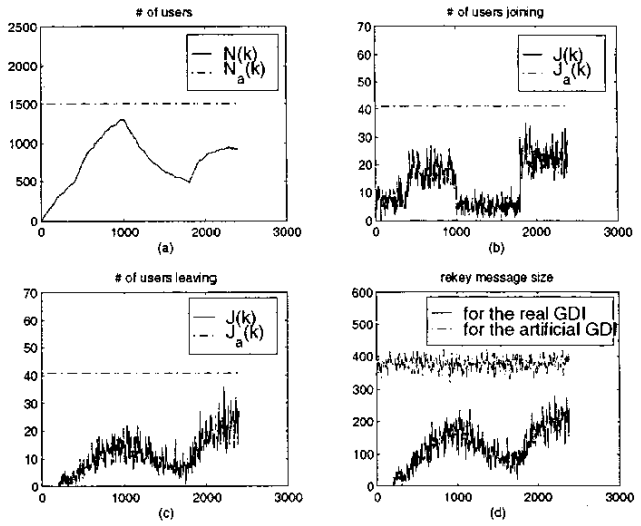

Fig. 3. The anti-attack scheme using phantom users and Batch rekeying

When $N(k) \leq N_0$, $L(k) \leq L_0$, and $J(k) \leq L_0$, equation (10)-(12) are equivalent to (9). We can prove that the artificial GDI functions in (10)-(12) satisfy requirement (r1) (r2) (r3) and (r4).

It shall be noted that there are many other ways to choose the artificial GDI functions. The proposed anti-attack scheme supports any artificial GDI functions that satisfy the requirement (r1)-(r4).

Given the artificial GDI functions, the KDC creates phantom users and performs key management as follows.

(1) Determine N_0 and L_0 based on the system requirements and the users' statistical behavior. The criteria for selecting N_0 and L_0 will be presented in Section IV.

(2) Before the service starts, create N_0 phantom users and establish a key tree to accommodate them. Set index $k = 1$.

(3) While the service is not terminated, execute the following:
   - Record user join and departure requests in the time period $((k-1)B_t, kB_t]$, and obtain $J(k)$ and $L(k)$. During this time, the current session key is sent to the joining users such that they can start receiving the multicast content without delay.
   - At time kB_t, the KDC creates $J_a(k) - J(k)$ phantom users joining the service, and then selects $L_a(k) - L(k)$ phantom users in the current system and makes them leave. Following the key updating procedure presented in any existing key management schemes, the KDC updates corresponding keys for real and phantom users' join and departure. The number of total real and phantom users are maintained to be $N_a(k)$.
   - Set $k = k + 1$.

Figure 3(a), 3(b), and 3(c) illustrate the real GDI ($N(k)$, $L(k)$, $J(k)$) and the artificial GDI ($N_a(k)$, $L_a(k)$, $J_a(k)$)

for a simulated multicast service. The simulation results of communication overhead, i.e. the rekeying message size, is shown in Figure 3(d), where the solid line represents the case without phantom users and the dash line represents case when the proposed anti-attack method is applied. We can see that the observed process reveals very limited information about the real GDI. Not surprisingly, the communication overhead increases, which is a disadvantage of utilizing phantom users.

Utilizing phantom users and batch rekeying is not the only solution to the problem of GDI leakage. There are other techniques that can protect GDI from one or several attacks. For example, embedding rekey messages into the multicast content [7] can prevent outside attackers to launch the AII type attacks. Using the same rekeying procedure for user join and departure is also a good way to prevent the AI type attacks. In addition, the KDC can generate faked rekey messages to prevent the AII type attacks, which is different from the proposed anti-attack scheme where the key tree reserves slots for the phantom users and all rekey messages have meanings.

Compared with other techniques, using phantom users and batch rekeying has two major advantages. First, the proposed anti-attack scheme resists to a variety of attacks. Since the real GDI are concealed *before* the rekey messages are generated, the attackers only see the artificial GDI from the observed rekeying process unless they break the encryption or compromise the KDC. Second, the proposed scheme does not rely on specific rekeying algorithms and is compatible with existing key management schemes.

IV. Performance Measure and Optimization

In this section, we define two performance criteria and evaluate the performance of the proposed anti-attack technique. The criteria are (a) the amount of information leaked to the attackers measured by mutual information, and (b) the communication overhead introduced by the phantom users. We study the tradeoff between these two metrics and provide a framework of choosing proper amount of phantom users, described by the parameter L_0 and Y_0 in (10)-(12).

A. The leakage of GDI

We use mutual information to measure the leakage of the GDI, which is independent of the attack strategies adopted by the attackers and represents the maximum amount of information that the attackers can possibly obtain. Let T be the total number of key updating, that is, the service duration is TB_t. Then, the real GDI is described by a set of random variables as

$$R = \{N(1), \cdots, N(T), J(1), \cdots, J(T), L(1), \cdots, L(T)\}, \quad (13)$$

and the artificial GDI is

$$A = \{N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T), L_a(1), \cdots, L_a(T)\}. \quad (14)$$

The mutual information, $I(R;A)$, describes the reduction in the uncertainty of the real GDI (R) due to the knowledge of the artificial GDI (A) [23]. Therefore, the leakage of the GDI can be measured by

$$I(R;A) = H(A) - H(A|R), \quad (15)$$

where $H(.)$ and $H(.|.)$ denote the entropy and conditional entropy, respectively.

Equation (10) - (12) indicate that the artificial GDI is a set of deterministic functions of the real GDI. Thus, the conditional entropy in (15) equals to zero, i.e. $H(A|R) = 0$. Since $L_a(k)$ is directly computed from $J_a(k)$, $N_a(k)$ and $N_a(k-1)$ in (12), the terms $L_a(1), L_a(2), \cdots, L_a(T)$ can be removed from the expression of the entropy of A, i.e. $H(A) = H(N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T))$. Then, the upper bound of $I(R;A)$ is calculated as:

$$\begin{aligned} I(R;A) &= H(N_a(1), \cdots, N_a(T), J_a(1), \cdots, J_a(T)) \\ &\leq \sum_k H(N_a(k)) + \sum_k H(J_a(k)). \end{aligned} \quad (16)$$

The equality is achieved when $\{N_a(k), J_a(k), k = 1, \cdots, T\}$ are mutually independent. It is noted that the GDI at time kB_t and the GDI at time $(k+1)B_t$ can be approximately independent when B_t is large and the group is high dynamic. In these cases, (16) provides a tight upper bound of $I(R;A)$.

We introduce $p_{N_k}(n)$ and $p_{N_{ak}}(n)$ to denote the pmf of $N(k)$ and $N_a(k)$, respectively. From (10), one can see that

$$p_{N_{ak}}(n) = \begin{cases} \sum_{x=0}^{N_0} p_{N_k}(x), & n = N_0 \\ p_{N_k}(n), & n > N_0 \\ 0, & o.w. \end{cases}$$

Then,

$$\begin{aligned} H(N_a(k)) &= -(1-\epsilon_N^k)\log(1-\epsilon_N^k) \\ &\quad - \sum_{n=N_0+1}^{\infty} p_{N_k}(n)\log p_{N_k}(n), \end{aligned} \quad (17)$$

where $\epsilon_N^k = 1 - \sum_{x=0}^{N_0} p_{N_k}(x)$. Similarly, let $p_{J_k}(x)$, $p_{J_{ak}}(j)$, and $p_{L_k}(y)$ denote the pmf of $J(k)$, $J_a(k)$, and $L(k)$, respectively. We then have,

$$H(J_a(k)) = -\sum_j p_{J_{ak}}(j)\log p_{J_{ak}}(j), \quad (18)$$

and,

$$p_{J_{ak}}(j) = \begin{cases} (1-\epsilon_J^k)(1-\epsilon_L^k), & j = L_0 \\ p_{J_k}(j)\sum_{y=0}^{j-1} p_{L_k}(y) + p_{L_k}(j)\sum_{x=0}^{j-1} p_{J_k}(x) \\ \quad + p_{J_k}(j)p_{L_k}(j), & j > L_0 \\ 0, & o.w. \end{cases} \quad (19)$$

where $\epsilon_J^k = 1 - \sum_{x=0}^{L_0} p_{J_k}(x)$ and $\epsilon_L^k = 1 - \sum_{y=0}^{L_0} p_{L_k}(y)$. Given the pmf of the real GDI functions, the upper bound of $I(R;A)$ is calculated from (16)-(19). Since the observed rekeying process is determined by the artificial GDI, the mutual information between the observed process and the real GDI is bounded by $I(R;A)$ due to the data processing theory [23]. Therefore, $I(R;A)$ is the upper bound of the amount of information that can be possibly obtained by the attackers.

From (10)-(12), one can see that the artificial GDI reveals the real GDI when $N(k) > N_0$, $L(k) > L_0$, or $J(k) > L_0$. We define *overflow probability* as the probability that the artificial GDI cannot be straight lines, i.e. $1 - \min_k(1 - \epsilon_N^k)(1-\epsilon_L^k)(1-\epsilon_J^k)$. Besides the mutual information, overflow probability can be a more visualized complementary measure for the leakage of the GDI. When the overflow probability is zero, the calculation in (16)-(18) leads to the result that $I(R;A) = 0$, which indicates the prefect protection of the real GDI.

B. Communication Overhead

Communication overhead, measured by the rekey message size, is one of the major performance criteria of key management schemes [2] [3]. We introduce the notation $M(L, N, d)$ as the expected value of the rekey message size when removing L users from the key tree that contains total N users and has degree d. We assume that the leaving users are uniformly distributed on a full loaded and balanced key tree. Then, there are d^l KEKs at the l^{th} level of the key tree for $l = 1, \cdots, D-2$ and $D = \lceil \log_d N \rceil$, and the number of the KEKs at the $(D-1)^{th}$ level is $s_1 = \lceil \frac{N - d^{D-1}}{d-1} \rceil$.

Let α^l be the number of KEKs need to be updated at level l when L user leaves the service. Then, $M(L, N, d)$ is expressed as:

$$M(L, N, d) = E\left[\sum_{l=0}^{D-1} \alpha_l\right] = \sum_{l=0}^{D-1} E[\alpha_l] \quad (20)$$

We introduce the notation $B(b, i, a)$, which is equivalent to the expected number of non-empty boxes when putting i items in b boxes with repetition where each box can have at most a items. The detailed calculation of $B(b, i, a)$ is provided in the Appendix. We can show that

$$E[\alpha_l] = d \cdot B(d^l, L, \frac{N}{d^l}), \quad 0 \le l \le D-2, \quad (21)$$

$$E[\alpha_{D-1}] = (d-1) \sum_{\tilde{L}=1}^{L} \frac{\binom{s_1}{\tilde{L}}\binom{N-s_1}{L-\tilde{L}}}{\binom{N}{L}} B(s_1, \tilde{L}, d) \quad (22)$$

Using the fact that $\lceil \frac{i}{a} \rceil \le B(b, i, a) \le \min(b, i)$ (see Appendix), we can derive the upper bound of the $M(L, N, d)$ as:

$$M(L, N, d) \le dL \log_d(N). \quad (23)$$

This upper bound indicates that the communication overhead increases linearly with the number of departure users and with the logarithm of the group size.

Let C_r and C_a be the average communication overhead for rekey process based on real GDI and the artificial GDI, respectively. Then, the extra communication overhead introduced by the proposed anti-attack technique is:

$$C_a - C_r = \frac{1}{T}\sum_{k=1}^{T} M(L_a(k), N_a(k), d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \quad (24)$$

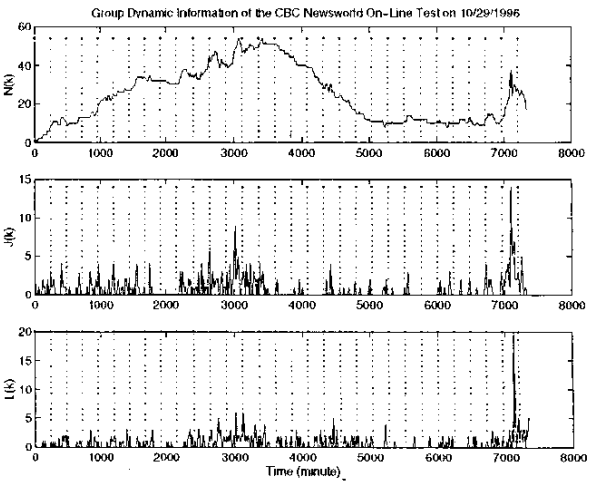

Fig. 4. The GDI of a long audio session in MBone

When the overflow probability is small, (24) can be approximated by:

$$C_a - C_r \approx M(L_0, N_0, d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \quad (25)$$

C. System Optimization

From the system design points of view, parameter L_0 and N_0 should be chosen such that the leakage of the GDI is minimized while the extra communication overhead do not exceed certain requirements. When the overflow probability is small, the optimization problem is formulated as:

$$\min_{N_0, L_0} \sum_k H(N_a(k)) + \sum_k H(J_a(k)) \quad (26)$$

subject to:

$$M(L_0, N_0, d) \le \beta, \quad (27)$$

where β is the maximum allowed communication overhead per key updating. We can show that $H(N_a(k))$ in (18) is monotonous non-increasing with N_0; $H(J_a(k))$ in (17) is monotonous non-increasing with L_0; and the communication overhead $M(L_0, N_0, d)$ in (20) is non-decreasing with L_0 and N_0. Therefore, the optimization problem is simplified as:

$$\min_{L_0} \left(\sum_k H(N_a(k)) + \sum_k H(J_a(k)) \right) \Big|_{N_0 = M^{-1}(\beta)|_{L_0,d}}, \quad (28)$$

where $M^{-1}(\beta)|_{L_0,d}$ is the largest value of N_0 that satieties (27) with given L_0 and d. Fortunately, the number of departure users between two key updates is usually not a large number in practice. Thus, the searching space for parameter L_0 is not large and this optimization problem can be solved by full search.

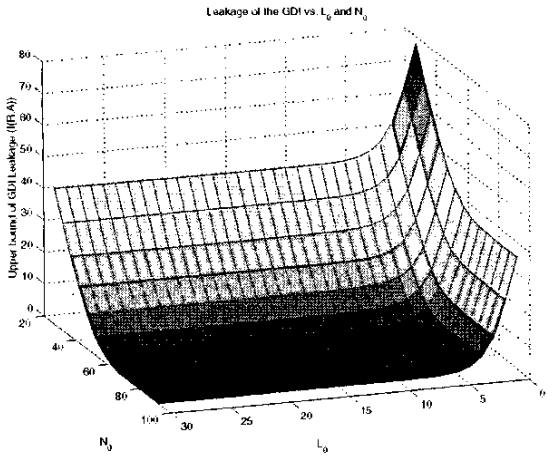

Fig. 5. Upper bound of the GDI leakages

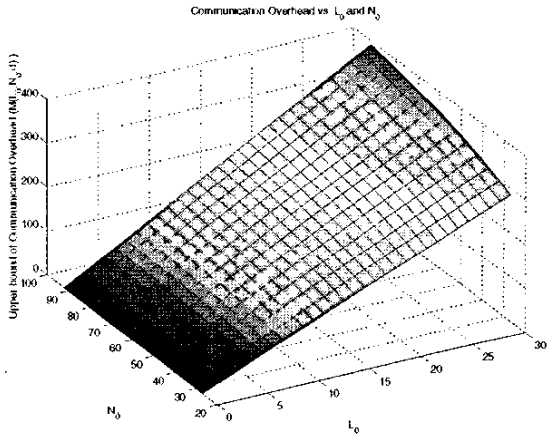

Fig. 6. Communication overhead $M(L_0, N_0, d)$

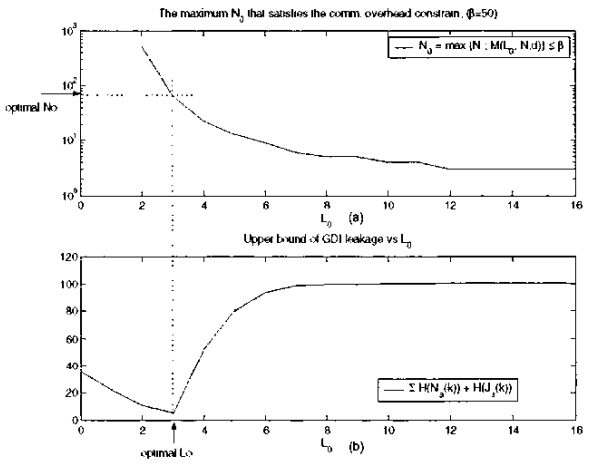

Fig. 7. Illustration of selecting Optimal parameters L_0 and N_0.

V. Simulations of the Anti-Attack Scheme

Mlisten, a tool developed at Georgia Institute of Technology, can collect the join/leave time for the multicast group members in MBone sessions [24]. Using this tool, the characteristics of the membership dynamics of MBone multicast sessions has been studied in [24] [25].

The proposed anti-attack scheme is applied to the data collected in 1996 [26]. Particularly, we selected one audio session that started on Oct. 29th and lasted for about 5 days and 20 hours. Figure 4 shows the $N(k)$, $L(k)$ and $J(k)$ of this session, where the B_t is chosen to be 15 minutes.

It is suggested that the users statistical behavior, such as inter-arrival and membership durations, can be modelled by exponential distribution in a short period of time [24]. In the simulation, the entire service time is divided into non-overlapped sections, as illustrated in Figure 4. The length of these sessions is set to be 4 hours. To simplify the analysis, it is assumed that $N(k)$, $L(k)$ and $J(k)$ are stationary and ergodic Poisson processes in each session. Then, we can calculate the GDI leakage using (16)-(19).

Figure 5 and Figure 6 demonstrate the upper bound of mutual information (see (16)) and the communication overhead $M(L_0, N_0, d)$ for different values of L_0 and N_0, respectively. We can see that communication overhead is a non-decreasing function with L_0 and N_0, while the GDI leakage is a non-increasing function with L_0 and N_0. This verifies the arguments in Section IV.

Figure 7 illustrates the solution of the optimization problem. Figure 7(a) shows the maximum value of N_0 that satisfies the communication overhead constraint in (27) with fixed L_0, i.e. $N_0 = max\{N : M(L_0, N, d) \leq \beta\}$, where β is chosen to be 50 in this example. As discussed in Section IV, the optimal values of L_0 and N_0 must be on this curve. Therefore, the upper bound of the GDI leakage, $\sum_k H(N_a(k)) + \sum_k H(J_a(k))$, is evaluated only at $(L_0, N_0 = max\{N : M(L_0, N, d) \leq \beta\})$, which is shown in Figure 7(b). The optimal values of L_0 and N_0 are also marked.

Figure 8 shows the tradeoff between the communication overhead and the GDI leakage. This figure demonstrates the upper bound of the mutual information as a function of the communication overhead constraint, where the parameters L_0 and N_0 have been optimized. This can help the system designer in determining the proper β for the communication constraint in (27). When not using phantom users, the artificial process is identical to the real process and we have $I(R; A) = I(R; R) = H(R)$. In this case, this particular multicast session require average 3.6 rekey messages to be sent in every 15 minutes ($B_t = 15$) and has $I(R; A) \approx 137$. Figure 8 shows that the proposed anti-attack scheme can reduces $I(R; A)$ to 5.5 by increasing the communication overhead to 23.2 messages every 15 minutes. The communication overhead C_a is significantly larger than C_r because a large amount of activities of the phantom users must be created. However, the absolute value of the C_r is still small compared with the multicast data throughput. On the other hand, the leakage of the group dynamic information is greatly reduced.

It is important to note that this MBone audio session

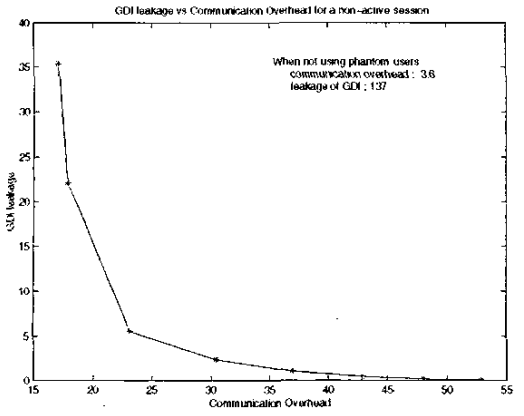

Fig. 8. The GDI leakage versus communication overhead for a real MBone audio session

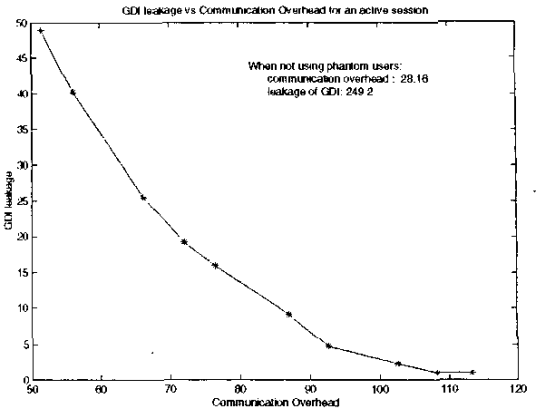

Fig. 9. The GDI leakage versus communication overhead for a simulated multicast session

contains only up to 60 users and represents the scenario where the group size is small and group members are not very active. Due to the lack of the experimental data for large multicast groups, we investigated a simulated multicast session with larger group size and more active group members. The simulation setup is the same as that is used for Figure 2(c) in Section II, where the group size is about 500. When not using phantom users, the KDC sends average 28.16 rekey messages in every 5 minutes ($B_t = 5$), while the amount of information leaked to the attackers, H(R), is 249.2. The performance of the proposed anti-attack methods is shown in Figure 9. We can see that the GDI leakage can be reduced to 5 at the expense of increasing the communication overhead to 93 messages per 5 minutes. The relative communication increase is smaller than that for the less active sessions.

VI. Conclusion

This paper raised the issues of the disclosure of the dynamic group membership information through key management in secure multicast communications. Such a security concern has not been discussed in traditional key management schemes. We demonstrated that the attackers can successfully obtain good estimates of the GDI from a large number of centralized key management schemes, and investigated the techniques of improving the existing key management schemes such that the GDI as well as the multicast content is protected. In particular, we developed two effective attack strategies, which exploit the format and the size of the rekey messages. To protect the GDI, we proposed the anti-attack technique utilizing batch rekeying and phantom users. This anti-attack technique reduces the leakage of the GDI and is fully compatible with the existing centralized key management schemes. We investigated the tradeoff between the communication overhead and the leakage of the GDI, and provided a framework for selecting the proper amount of phantom users. The proposed anti-attack technique was tested on real MBone user log data and simulated multicast sessions.

Acknowledgement

This work was supported in part by the Army Research Office under Award No. DAAD19-01-1-0494.

Appendix

We define $n(b,i,a)$ to be the number of non-empty boxes when randomly placing i identical items into b identical boxes with repetition, where each box can hold at most a items. In this appendix, we calculate $B(b,i,a)$ as the expected value of $n(b,i,a)$, i.e. $B(b,i,a) = E[n(b,i,a)]$. It is obvious that the value of $n(b,i,a)$ is bounded as $B_0 \le n(b,i,a) \le B_1$, where $B_0 = \lceil \frac{i}{a} \rceil$ and $B_1 = \min(i,b)$.

We define an intermediate quantity $w(y,i,a)$ as the number of ways of putting i items into y boxes such that each box contains at least 1 and at most a items. $w(y,i,a)$ can be calculated recursively as:

$$w(B_0,i,a) = \binom{aB_0}{i} \tag{29}$$

$$w(B_0+k,i,a) = \binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1} \binom{B_0+k}{B_0+m} w(B_0+m,i,a), \tag{30}$$

where $0 \le k \le B_1 - B_0$. Then, the pmf of $n(b,i,a)$ can be expressed as:

$$Prob\{n(b,i,a) = B_0+k\} = \frac{1}{N}\binom{b}{B_0+k} w(B_0+k,i,a), \tag{31}$$

where $N = \binom{ab}{i}$ represents the total number of ways of putting i items into b boxes. By substituting (30) into (31), we get:

$$Prob\{n(b,i,a) = B_0+k\} = \frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1} \frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} Prob\{n(b,i,a) = B_0+m\}.$$

It can be shown that:

$$\frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} = \binom{b-B_0-m}{k-m}.$$

Therefore,

$$Prob\{n(b,i,a) = B_0+k\} = \frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1}\binom{b-B_0-m}{k-m} Prob\{n(b,i,a) = B_0+m\}. \quad (32)$$

By substituting (29) into (31), we have:

$$Prob\{n(b,i,a) = B_0\} = \frac{1}{N}\binom{b}{B_0}\binom{aB_0}{i}. \quad (33)$$

Based on (32) and (33), $Prob\{n(b,i,a) = B_0+k\}$ for $k = 0, 1, \cdots, B_1 - B_0$ is calculated recursively. Then, we can calculate $B(b,i,a)$, i.e. $E[n(b,i,a)]$, as:

$$B(b,i,a) = \sum_{k=0}^{B_1-B_0} (B_0+k)\cdot Prob\{n(b,i,a) = B_0+k\}. \quad (34)$$

REFERENCES

[1] S. Paul. *Multicast on the Internet and its applications*. Kluwer Academic Publishers, 1998.

[2] M.J. Moyer, J.R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, Nov.-Dec. 1999.

[3] D.M. Wallner, E.J. Harder, and R.C. Agee, "Key management for multicast: issues and architectures," Internet Draft Report, Sept. 1998, Filename: draft-wallner-key-arch-01.txt.

[4] R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *Proc. IEEE INFOCOM'99*, vol. 2, pp. 708–716, March 1999.

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

[6] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE Journal on selected areas in communications*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.

[7] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Key distribution for secure multimedia multicasts via data embedding," *Proc. IEEE ICASSP'01*, pp. 1449–1452, May 2001.

[8] D. McGrew and A. Sherman, "Key establishment in large dynamic groups using one-way function trees," Technical Report 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[9] A. Perrig, D. Song, and D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. IEEE Symposium on Security and Privacy*, 2001, pp. 247 –262.

[10] G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Software Eng.*, vol. 15, pp. 929–934, Aug 1989.

[11] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGCOMM '97*, 1997, pp. 277–288.

[12] S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over IP multicast," *JSAC Special Issue on Network Support for Group Communication*, vol. 20, no. 8, pp. 1511 –1527, Oct. 2002.

[13] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol. 28, pp. 714–720, Sep. 1982.

[14] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," in *Proceedings on Advances in cryptology*, 1990, pp. 520–528, Springer-Verlag New York, Inc.

[15] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution scheme," *Advances in Cryptology- Eurocrypt*, pp. 275–286, 1994.

[16] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM conference on Computer and communications security*, 1996, pp. 31–37, ACM Press.

[17] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: a new approach to group key agreement," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998, pp. 380 – 387.

[18] M. Steiner, G. Tsudik, , and M. Waidner, "Key agreement in dynamic peer groups," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 11, no. 8, pp. 769–780, Aug 2000.

[19] G. Tsudik Y. Kim, A. Perrig, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM conference on Computer and communications security*, November 2000.

[20] L.R. Dondeti, S. Mukherjee, and A. Samal, "DISEC: a distributed framework for scalable secure many-to-many communication," in *Proceedings of Fifth IEEE Symposium on Computers and Communications*, 2000, pp. 693 –698.

[21] W. Trappe, Y. Wang, and K.J.R. Liu, "Establishment of conference keys in heterogeneous networks," in *proceedings of IEEE International Conference on Communications*, 2002, vol. 4, pp. 2201 –2205.

[22] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: a performance analysis," *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 27 – 38, August 2001.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.

[24] K. Almeroth and M. Ammar, "Collecting and modeling the join/leave behavior of multicast group members in the mbone," in *Proc. High Performance Distributed Computing (HPDC'96), Syracuse, New York*, 1996, pp. 209–216.

[25] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, pp. 224–229, June 1999.

[26] Mbone user activity data, http://ftp.cc.gatech.edu/people/kevin/release-data, March 2003

[27] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. Stanton, and G. Tsudik, "Secure group communication in asynchronous networks with failures: Integration and experiments," in *Proceedings of IEEE ICDCS 2000*, April 2000.

[28] Y. Sun, W. Trappe, and K.J.R. Liu, "An efficient key management scheme for secure wireless multicast," *Proc. of IEEE International Conference on Communication*, vol. 2, pp. 1236–1240, 2002.