Analysis and Protection of Dynamic Membership Information for Group Key Distribution Schemes

Yan Lindsay Sun, *Member, IEEE*, and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—In secure group-oriented applications, key management schemes are employed to distribute and update keys such that unauthorized parties cannot access group communications. Key management, however, can disclose information about the dynamics of group membership, such as the group size and the number of joining and departing users. This is a threat to applications with confidential group membership information. This paper investigates techniques that can stealthily acquire group dynamic information from key management. We show that insiders and outsiders can successfully obtain group membership information by exploiting key establishment and key updating procedures in many popular key management schemes. Particularly, we develop three attack methods targeting tree-based centralized key management schemes. Further, we propose a defense technique utilizing batch rekeying and phantom users, and derive performance criteria that describe security level of the proposed scheme using mutual information. The proposed defense scheme is evaluated based on the data from MBone multicast sessions. We also provide a brief analysis on the disclosure of group dynamic information in contributory key management schemes.**

***Index Terms*—Communication system security, privacy.**

I. Introduction

THE ubiquity of communication networks is facilitating applications that allow communication and collaboration among a large number of diverse users. Group key management, which is concerned with generating and updating secret keys, is one of the fundamental technologies to secure such group communications [1]–[4]. Key management facilitates access control and data confidentiality by ensuring that the keys used to encrypt group communication are shared only among legitimate group members. Thus, only legitimate group members can access group communications. The shared group key can also be used for authentication. When a message is encrypted using the group key, the message must be from a legitimate group member.

There are three types of group key management schemes [2]. In centralized key management, such as [3], [5]–[12], group members trust a centralized server, referred to as the key distribution center (KDC), which generates and distributes encryption keys. In decentralized schemes, such as [13] and [14], the task of KDC is divided among subgroup managers. In contributory key management schemes, such as [15]–[23], group members are trusted equally and all participate in key establishment.

The design of current key management schemes focuses on maintaining key secrecy and reducing overhead associated with key updating [1], [3], [18]. We observe, however, that key management can disclose information about dynamic group membership to both insiders and outsiders. In other words, while the content of group communication is protected by encryption using the secret keys, group dynamic information is disclosed through key management. We collectively refer to group dynamic information (GDI) as information describing the dynamic group membership, including the number of users in a multicast group as a function of time, and the number of joining or departing users in a time interval.

In many secure group applications, group dynamic information should be kept confidential. Key management is a technology that enables key updating in real time as group membership changes. Future commercial multicast services, which could occur in nontraditional broadcast media, such as Internet and 3G/4G wireless networks, will allow a user to subscribe to an arbitrary set of programs and change his or her subscription at any time [10], [24]. The users can choose to pay for exactly what they get, instead of a fixed monthly fee. This new type of services give the most flexibility to users, as well as opportunities to new business models. Over the nontraditional broadcast media, the global media giants as well as small multimedia producers can be the service providers. The service providers perform group management and have the knowledge of GDI (i.e., audience statistics). However, it is highly undesirable to disclose instant and detailed GDI to competitors. Assume a competitor can monitor the audience statistics of the service provider X. Then, the competitor may broadcast its programs at different time slots and see how it affects its own and X's audience statistics. As a consequence, the competitor can develop the best program schedule to compete with X. This example also shows that GDI should also be concealed from insiders. A regular user, who receives the multicast content, should not know the overall audience statistics. Otherwise, the competitor can send one of its employees to register as X's member for a small cost, and collect valuable audience statistics from X. In addition, there are multicast communication scenarios where GDI represents sensitive deployment information about the network. For example, in a sensor network, the base station sends many broadcast messages to sensors. The base station and sensors form a secure multicast

Manuscript received June 14, 2005; revised January 25, 2007. This work was supported by the Army Research Office under Award DAAD19-01-1-0494. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ton Kalker.

Y. L. Sun is with the Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: yansun@ele.uri.edu).

K. J. R. Liu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: kjrliu@umd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2007.897274

group. If some sensors are compromised, the group key should be updated such that the compromised sensors cannot decrypt future multicast messages from the BS. One possible way to update group keys is to use group key management schemes. In such an application scenario, GDI represents the number of sensors deployed in an area, and the number of revoked sensors. In this example, if GDI is not protected, attackers can obtain sensor deployment information by exploiting the key management scheme. From the above two examples, we can see that obtaining GDI through key management is a new dimension of vulnerability.

In this paper, we will analyze GDI leakage problem and propose a framework to protect GDI from insiders and outsiders. In particular, we develop three effective strategies to obtain GDI from the tree-based centralized key management schemes [1], [3], [5]–[8], [10]. These strategies involve exploiting the format of rekeying messages, the size of rekeying messages, and key IDs. To protect GDI, we develop a defense method that is fully compatible with existing key management schemes. By utilizing batch rekeying [25], [26] and phantom users, the proposed 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 defense scheme, such as overhead and GDI secrecy, are evaluated based on the data obtained from MBone [27] sessions. In addition, we provide a brief discussion on GDI protection in the contributory key management schemes.

The rest of this paper is organized as follows. The attack and defense methods for the centralized schemes are presented in Sections II and III, respectively. In Section IV, the performance criteria of the proposed method are derived and the optimization problem is formulated. Simulation results are shown in Section V. The GDI issues in contributory key management schemes is presented in Section VI, followed by a discussion in Section VII. The conclusion is drawn in Section VIII.

II. GDI Disclosure in Centralized Key Management Schemes

In the centralized key management schemes, there exists a key server that generates and distributes the decryption keys [1]. In this section, we investigate the methods that can acquire GDI stealthily from the centralized key management.

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 from key management can be simple and stealthy. Instead of trying to break 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 insiders. As we will show later in this section, insiders can obtain very accurate estimation of GDI by monitoring the rekeying messages, which are the messages conveying new key updating information. Even if the adversaries cannot become valid group members, they can still obtain GDI as the outsiders as long as they can observe the rekeying traffic around a single group member.

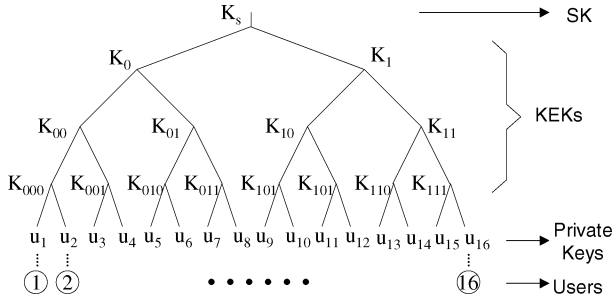


Fig. 1. Typical key management tree.

In this section, we consider a popular tree-based centralized key management scheme proposed in [7], present three methods to obtain GDI, and discuss the vulnerability of other prevalent centralized key management schemes.

A. Tree-Based Centralized Key Management Schemes

Similar to other tree-based schemes [1], [3], [5], [8], [10], the centralized Versakey scheme in [7] employs a key tree to maintain the keying material. As illustrated in Fig. 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-encryption keys (KEK), which are auxiliary keys and used only for the purpose of protecting the session key and other KEKs. To make a concise presentation, we do not distinguish the node and the key associated with this node in the remainder of this paper.

Each user stores his or her private key, the session key, and a set of KEKs on the path from himself or herself to the root of the key tree. In the example shown in Fig. 1, user 16 possesses $\{u_{16}, K_s, K_1, K_{11}, K_{111}\}$. 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 with key x.

When a user leaves the service, all keys known to this user need to be updated in order to prevent him or her from accessing future communications. This is often referred to as forward secrecy [3]. According to [7], when user 16 leaves, the KDC generates new keys and conveys new keys to the remaining users through a set of rekeying messages.

- $\{K_{111}^{\text{new}}\}_{u_{15}}$: user 15 acquires K_{111}^{new}.
- $\{K_{11}^{\text{new}}\}_{K_{111}^{\text{new}}}, \{K_{11}^{\text{new}}\}_{K_{110}^{\text{old}}}$: user 13,14,15 acquire K_{11}^{new}.
- $\{K_{1}^{\text{new}}\}_{K_{11}^{\text{new}}}, \{K_{1}^{\text{new}}\}_{K_{10}^{\text{old}}}$: user $9, \ldots, 15$ acquire K_{1}^{new}.
- $\{K_{s}^{\text{new}}\}_{K_{1}^{\text{new}}}, \{K_{s}^{\text{new}}\}_{K_{0}^{\text{old}}}$: user $1, \ldots, 15$ 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 rekeying messages are usually sent together in one datagram through multicast [5], every user receives all rekeying messages. The session key, the KEKs, and the users' private keys usually have the same length. The

communication overhead associated with key updating can be described by the rekeying-message-size, defined as the amount of rekeying messages measured in the unit as the same size as the SK or the KEKs. In this example, the rekeying message size is 7 when user 16 leaves the service. It has been shown that the rekeying message size increases logarithmically with the group size [7].

When a user joins the group, the KDC chooses a leaf position on the key tree to put the joining user. In [7], 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 a unicast channel. Other users in the group will know about the key change when the data packet indicating the increase of the revision number for K_s first arrives, and compute the new keys using the one-way function. No additional rekeying messages are necessary.

Although having different rekeying procedures, most tree-based centralized key management schemes [1], [3], [5], [7], [8], [10] share several common properties. First, group members can distinguish the key updating process due to user join and that due to user departure. Second, the rekeying-message-size may be related to the group size. Third, the IDs of the keys stay the same even if the key content changes. Because of these properties, we develop several methods that can obtain GDI stealthily from key management. Those methods are presented based on the tree-based key management scheme in [7].

B. Attack 1 : Estimation of $J(t_0, t_1)$ and $L(t_0, t_1)$ From Rekeying-Message Format

An insider receives rekeying messages, decrypts some of the messages, and observes the rekeying 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 is different, he or she can estimate $J(t_0, t_1)$ and $L(t_0, t_1)$ as follows.

- When receiving the rekeying message containing K_s^{new} encrypted by one of his or her KEKs, he or she assumes that one user leaves the group.
- When observing the increase of the revision number of K_s, he or she assumes that one user joins the group.

This strategy is effective when most users do not join/leave simultaneously and the keys are updated immediately after each user joining/departing event. When this method is successful, $N(t)$ can be calculated from $J(t_0, t_1)$ and $L(t_0, t_1)$ as

$$N(t_1) = N(t_0) + J(t_0, t_1) - L(t_0, t_1). \tag{1}$$

Even if the initial group size is unknown, the changing trend of the group size is obtained.

C. Attack 2 : Estimation of the Group Size From the Rekeying Message Size

In some tree-based key management schemes [28], the key tree is fully loaded and maintained as balanced as possible by putting the joining users on the shortest branches. In this case, the group size $N(t)$ can be estimated directly from the rekeying message size. Here, we derive a maximum-likelihood (ML) estimator 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 departing users who do not leave simultaneously. Thus, W observations of the rekeying message size due to single user departure are made. These observations are denoted by $Msg = \{m_1, m_2, \ldots, m_W\}$.

In the worst-case scenario, the insiders and outsiders know the degree of the key tree, denoted by d. Then, they can calculate the length of the branch where the ith leaving user was located before his or her departure, denoted by L_i. Without losing information, the observed Msg is converted to $\{L_1 = l_1, L_2 = l_2, \ldots, L_W = l_W\}$, where $l_i = \lceil (m_i + 1)/d \rceil$. Then, the ML estimator is formulated as

$$N_{\text{ML}} = \arg\max_n \text{Prob}\{\text{L}_1 = \text{l}_1, \text{L}_2 = \text{l}_2, \ldots, \text{L}_\text{W} = \text{l}_\text{W} | \text{N(t)} = \text{n}\} \tag{2}$$

where $\arg\max_n g(n)$ represents the n value that maximizes the function $g(n)$. To solve (2), we introduce a set of new variables $\{S_k\}_{k=L_{\min}, L_{\min}+1, \ldots, L_{\max}}$, 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

$$\sum_k S_k = n. \tag{3}$$

In addition, the length of the branches of a key tree must satisfy the Kraft inequality [29] (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 $j = 1, 2, \ldots, n$. Thus, S_k, which is equal to the number of elements in set $\{b_j : b_j = k\}$, must satisfy

$$\sum_k S_k d^{L_{\max} - k} \leq d^{L_{\max}}. \tag{4}$$

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

$$\sum_k S_k d^{L_{\max} - k} = d^{L_{\max}}. \tag{5}$$

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., $N(t) \gg W$). Then, the probability mass function (pmf) of L_i is

$$\text{Prob}\{L_i = k \,|n,\ S_k\} = \frac{S_k}{n},\ k = L_{\min}, L_{\min} + 1, \ldots, L_{\max}.$$

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

$$\text{Prob}\{\text{L}_1 = \text{l}_1, \text{L}_2 = \text{l}_2, \ldots, \text{L}_\text{W} = \text{l}_\text{W} \,|\text{N(t)} = \text{n},\ \text{S}_\text{k}\} = \prod_k \left(\frac{S_k}{n}\right)^{h(k)} \tag{6}$$

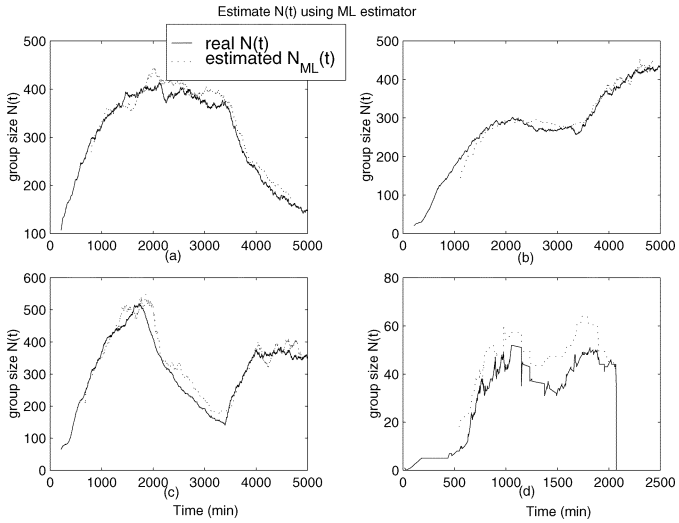


Fig. 2. Performance of the ML estimator. (a)–(c) are for simulated multicast sessions and (d) is for an Mbone session.

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 the Lagrange multiplier as

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

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

This ML estimator is first applied in simulated group communications. As suggested in [30] and [31], the user arrival process is modeled as a poisson process, and the service duration is modeled as an exponential random variable. In Fig. 2(a)–(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., the user arrival rate and the average service time) are fixed within each session and vary in different sessions. In the ith 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 as follows. In plot (a), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4] = [0.5, 0.5, 0.5, 0.3]$ min^{-1} and $[\mu_1, \mu_2, \mu_3, \mu_4] = [1400, 800, 600, 400]$ min. In plot (b), $[\lambda_1, \lambda_2, \lambda_3, \lambda_4] = [0.1, 0.3, 0.2, 0.5]$ min^{-1} and $[\mu_1, \mu_2, \mu_3, \mu_4] = [1500, 1500, 1000, 800]$ min. In plot (c), $\lambda_1, \lambda_2, \lambda_3, \lambda_4] = [0.3, 0.7, 0.1, 0.9]$ min^{-1} and $[\mu_1, \mu_2, \mu_3, \mu_4] = [1400, 800, 600, 400]$ min. Fig. 2(d) demonstrates the performance of the ML estimator, when it was applied to a real MBone audio session, CBC Newsworld online test, began October 29. 1996, and lasted for about five days [27].

In all four cases, the changing trend of the group size is well captured by the estimator. It is also observed that the estimated group size tends to be larger than the true $N(t)$. This is due to the approximation that we replace (5) by (4). Although not perfect, this estimator is effective for analyzing audience behavior and the group size changes.

D. Attack 3: Estimation of Group Size Based on Key IDs

As presented in [7], each key contains the secret material that is the content of the key and a key selector that is used to distinguish the key. The key selector consists of: 1) a unique ID that stays the same even if the key content changes and 2) a version and revision field, reflecting the update of the key. The basic format of the rekeying messages is $\{K_y\}_{K_x}$, representing K_y encrypted by K_x. This message has two parts. The first part is the key selector of K_x, which is not encrypted because otherwise, a user will not be able to understand this message. The second part is K_y and the key selector of K_y, encrypted by K_x. Thus, in the current implementation, everyone who can overhear the rekeying messages can see the IDs of K_x.

One can collect the histogram of these key IDs. Let $P(K_x)$ denote the probability of K_x's ID appears, calculated as the number of rekeying messages containing this ID (as an encryption key ID) divided by the total amount of rekeying messages. Define G_x as the set of users under the node associated with K_x, and $n(G_x)$ as the number of users in G_x. Let K_p denote the parent node of K_x.

Based on the rekey procedure in [7], we observe that $P(K_x)$ is equal to the probability that one or more than one users leaves the subgroup G_p given that there are users leaving the multicast group. In addition, it is reasonable to assume that $P(K_x)$ is proportional to $n(G_p)$. This assumption is valid when users are equally likely to leave and the probability of a user leaving in one round of key updating is small.

Our observation and assumption enable the key ID-based attack. We explain the basic idea of this attack using the example shown in Fig. 3, where the attacker is marked by a triangle.

Step 1) The attacker knows that the keys on the branch from himself or herself to the root are $\{K_s, K_1, K_{11}, K_{111}, K_{1111}\}$. Among these keys, he or she also knows who is whose children node because the parent node keys are always encrypted by the children node keys. The attacker collects $P(K_1)$, $P(K_{11})$, and $P(K_{111})$ by observing a sufficient number of rekeying messages.

Step 2) When there are users leaving G_0, KDC needs to update key K_0 by sending rekeying message $\{K_0\}_{K_{00}}$ and $\{K_0\}_{K_{01}}$, according to the rekeying procedure described in Section II-A. Thus, whenever there are users leaving G_0, the IDs of K_{01} and K_{00} will appear. Therefore, $P(K_{00}) = P(K_{01}) = \Pr(\text{there are users leaving } G_0)$. Similarly, we have $\Pr(\text{there are users leaving } G_1) = P(K_{10}) = P(K_{11})$ and $\Pr(\text{there are users leaving } G_s) = P(K_0) = P(K_1)$. Since $G_s = G_0 \bigcup G1$ and $G_0 \bigcap G_1 = \phi$, it is easy to see that

$$P(K_1) = 1 - (1 - P(K_{11}))(1 - P(K_{00})). \tag{9}$$

In addition, as described earlier, it is reasonable to assume that

$$\frac{n(G_0)}{n(G_1)} = \frac{\Pr(\text{there are users leaving } G_0)}{\Pr(\text{there are users leaving } G_1)} = \frac{P(K_{00})}{P(K_{11})}. \tag{10}$$

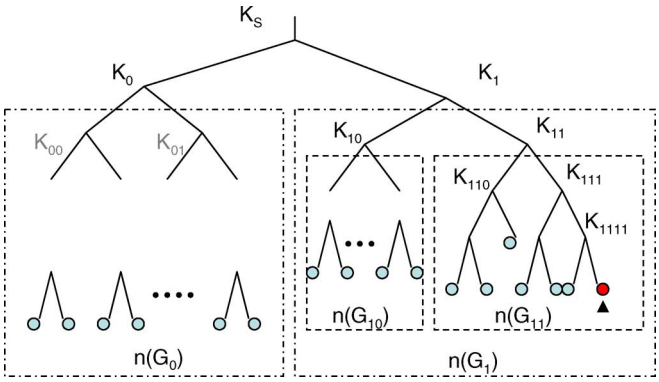


Fig. 3. Key ID-based attack method.

Similarly, the attacker can obtain $n(G_{10})/n(G_{11})$, and $n(G_{110})/n(G_{111})$.

Step 3) The attacker estimates $n(G_{111})$ based on the degree of the key tree. Then, he or she can obtain $n(G_{110})$, $n(G_{10})$, and $n(G_0)$, using the results generated in the previous step. The group size is finally estimated as $n(G_{111}) + n(G_{110}) + n(G_{10}) + n(G_0)$.

The accuracy of this attack depends on the estimation error of $n(K_{111})$. In this example, $n(K_{111})$ can be estimated as either 3 or 4. This results in 25% estimation error in the total group size. The accuracy also depends on the assumption that group members are equally likely to leave and they leave independently. Although it is not a very accurate method, a key ID-based attack can reveal a large amount of GDI information.

More important, (9) and (10) do not rely on specific tree structures. When the key tree is not balanced and/or not fully loaded, those equations are still valid. To see this, we examine an example of an unbalanced key tree, where there are N users under K_1 and $N/5$ users under K_0. When the departuring users are randomly located on the key tree

$$\Pr(\text{there are users leaving } G_1) \approx 5 \cdot \Pr(\text{there are users leaving } G_0).$$

Therefore, the ID of key K_{11} should appears 4 times more frequently than the ID of key K_{00}. Using the procedure in step 2, the attacker can know that the number of users under K_1 are approximately 4 times more than the users under K_0 by examining the key IDs. We can see that Attack 3 can be applied to unbalanced or non-fully loaded key trees. This is the major advantage of Attack 3. Recall that Attack 2 is suitable for balanced and fully loaded key trees.

E. Discussion on Three Attacks

An insider can jointly use all three types of attacks, and an outsider can use AII and AIII under certain conditions. An outsider can apply AII when he or she is able to observe the size of rekeying messages. It has been shown that the rekeying messages must be delivered reliably and in a timely manner in order to guarantee the quality of service [32]. Therefore, it is very likely that rekeying messages are treated differently from the regular data in terms of error control, or even transmitted in a reliable multicast channel that is separated from the channel used for data transmission. This provides an opportunity for the outsiders to differentiate the rekeying messages and the multicast content. As long as an outsider can observe the rekeying traffic sent to one group member, he or she can obtain the rekeying message size and use method AII to estimate the group size. It is noted that error control coding may change the size of the rekeying messages. We assume that the coding rate is not a secret. Thus, the attackers can recover the original rekeying message size without coding. In current key management schemes, the key selector of the encryption key is not encrypted. Thus, an outsider can collect the histogram of key IDs. One straightforward improvement is to use the session key to encrypt the key selector, which will prevent outsiders from using AIII. This requires additional encryption/decryption operations.

TABLE I
COMPARISON AMONG ATTACK METHODS. (* WHEN THE INITIAL GROUP SIZE IS KNOWN)

	applied by insider	applied by outsider	requirement on key trees	accuracy
A1	yes	no	none	high*
A2	yes	possible	balanced, full-loaded	high-moderate
A3	yes	possible	none	moderate-low

In the derivation of the ML estimator in Attack 2, we assume that the key tree is fully loaded. This assumption can be violated in some implementations of key management. For example, the KDC first estimates the maximum group size to be $N_{\max}$. Then, a key tree with $N_{\max}$ leaf nodes is constructed. This key tree will have many empty leaf nodes that are not associated with particular users. A joining user will occupy an empty leaf node after it joins, and a departing user will release a leaf node after it leaves. Since there is no need to split or merge nodes when users join or leave, these types of key trees are easy to maintain. On the other hand, they often require higher overhead to store and update keys than what is necessary. In practice, the type of key trees, referred to as non-fully loaded key trees, are used when $N_{\max}$ is not large or are different between $N_{\max}$ and the average group size is not large. For non-fully loaded key trees, Attack 3 should be applied. Although the accuracy of Attack 3 is not as good as other attacks, it still can provide a large amount of information about GDI. If multiple attackers jointly estimate GDI, the results will be more accurate.

We would also like to point out the difference between the proposed method and the non-fully loaded key trees here. As we will see in later sections, the proposed defense solution also leaves some "phantom" leaf nodes on the key tree. However, the proposed solution is more sophisticated because these phantom leaf nodes are not just empty nodes but have dynamic joining/departing behaviors. Simply leaving empty nodes on the key tree cannot hide GDI because the Attack 3 works for non-fully loaded key trees.

As a summary, the properties of three attacks are listed in Table I.

F. GDI Vulnerability in Other Key Management Schemes

While Attack 3 is only suitable for tree-based schemes, Attack 1 and 2 can be tailored to many other key management

schemes. When the insiders can differentiate the rekeying messages for user join and those for user departure, they use an attack similar to AI, referred to as the AI type method. When the amount of rekeying messages largely depends on the group size, they can use an attack similar to AII, referred to as the AII type method, with an estimator that may be different from (8). Next, we review popular centralized and decentralized key management schemes and discuss their vulnerabilities against AI and AII-type methods.

Since protecting GDI is not a part of the design goal in traditional key management schemes, it is not surprising that some schemes reveal GDI in a very straightforward way. For example, in the approach proposed in [12], 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 method.

Tree-based key management schemes have been known for their efficiency in terms of communication, computation, and storage overhead. Many tree-based schemes, such as [3], [5], and [8] are similar to those described in Section II-A. In these cases, both the AI and AII methods can be applied. In [9]–[11], another class of tree-based schemes was presented to further reduce the communication overhead by introducing the dependency among keys, such as in one-way function trees. In these schemes, the key updating procedures for user join and departure are similar. Thus, AI-type methods are not applicable. Since the size of rekeying messages is closely related with the group size, AII-type methods are suitable.

Besides the tree-based scheme described in Section II-A, the VersaKey framework [7] also includes a centralized flat scheme. When a user joins or leaves the group, the rekeying message size is equal to the length of the binary representation of user IDs, which can be independent of $N(t)$. Thus, this key management scheme is resistant to both the AI- and AII-type methods. 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 does not protect the multicast content when collusion attacks are likely.

In Iolus [13], 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 resulting from membership changes. Since the key updating is localized within each subgroup, the insiders or outsiders can only obtain the dynamic membership information of the subgroups that they belong to or can monitor.

The idea of clustering was introduced in [14] to achieve the efficiency by localizing 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 [14], the cluster members cannot obtain GDI of their clusters. However, the cluster leaders naturally obtain the dynamic membership information of their clusters and all clusters below. In [14], the cluster size is chosen from 3 to 15. Therefore, this key management scheme can be applied only when a large portion of group members is trusted to perform key management and obtain GDI.

TABLE II
VULNERABILITY OF PREVALENT KEY MANAGEMENT SCHEMES

Centralized Key Management Schemes		Is method AII Effective?	Is method AI Effective?
Tree Based	Key Graph [5], Wallner98 [3], Tree-based scheme in VersaKey framework [7] Embedding [8]	Yes	Yes
	One-way function tree [9] Improve Key Revocation [10] ELK [11]	Yes	No
Flat	Security lock [12]	Yes	–
	Flat centralized scheme in VersaKey framework [7]*	No	No
Local security agents	Iolus [13]	Local	Local
	Clustering [14]*	No	No
Others	TMKM [33]	Local	Local

In [33], a topology-matching key management (TMKM) scheme was presented to reduce the communication overhead by matching the key tree with the network topology and localizing the transmission of the rekeying messages. In this scheme, group members receive only the rekeying messages that are useful for themselves and their neighbors. Thus, they only obtain the local GDI by using AI- or AII-type methods.

As a summary, Table II lists various key management schemes we have discussed. We can see that the AII type methods are effective for obtaining GDI or local GDI from many key management schemes. Two schemes–flat VersaKey [7] and the clustering in [14] do not reveal GDI, but their use is limited because they are either not resistant to collusion attacks or must put trust in a large number of cluster leaders. Therefore, the defense techniques that protect GDI should be compatible with a variety of key management schemes.

III. DEFENSE TECHNIQUES

We have discussed several ways to obtain GDI stealthily from the centralized and decentralized key management schemes. This discussion, however, does not cover all aspects of key management schemes that can reveal group dynamic information. New attacks may emerge in the future. Therefore, we design a defense framework that is robust to various threats and compatible with different key management schemes.

The rekeying process reveals GDI in two domains. In the time domain, the insiders/outsiders observe when the rekeying messages are transmitted. In the message domain, the insiders/outsiders observe the size and/or the format of the rekeying messages.

To protect GDI in the time domain, we use batch rekeying [7], [25], which postpones the updates of the keys in order to remove the correlation between the time of key updating and the time when users join/leave the group. In particular, we implement batch rekeying as periodic updates of keys. 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, where k is an positive integer and B_t is the key updating period. By doing so, the time-domain observations do not contain information about when users join/leave the group. It is important to note that batch rekeying was originally proposed to reduce the rekeying overhead. It has been shown in [7], [25], and

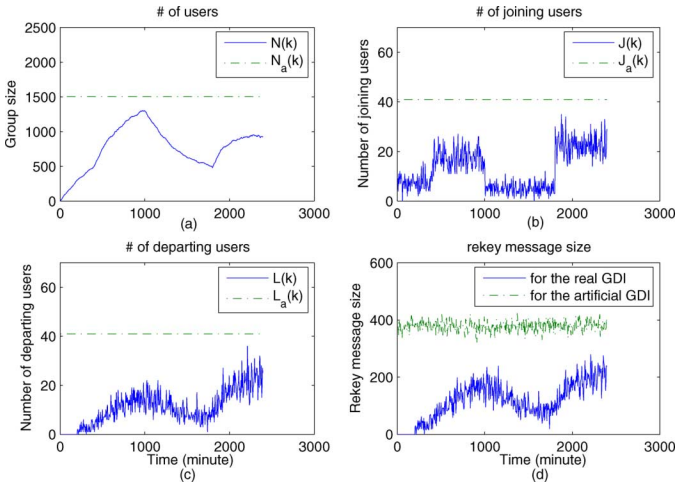


Fig. 4. Defense scheme using phantom users and batch rekeying.

[26] that updating keys for several users together consumes less communication and computation resources than updating keys for the users one by one. The disadvantage of batch rekeying is that the joining/departing users will be able to access a small amount of information before/after their join/departure. Thus, the parameter B_t must be chosen based on the group policies. In particular, B_t should be smaller than the maximum acceptable delay between revoking a user and sending information that should not be accessed by the revoked user. When using batch rekeying, the notations of the GDI functions are simplified as $J(k) = J((k-1)B_t, kB_t)$, $L(k) = L((k-1)B_t, kB_t)$, and $N(k) = N(kB_t)$.

Batch rekeying cannot protect GDI in the message domain. Fig. 4 shows simulation results for the batch rekeying when B_t is set to be 5 min. Simulation setup is similar to that in Section II-C. The solid line in Fig. 4(a)–(d) represents the $N(k)$, $J(k)$, $L(k)$, and the rekeying message size, respectively. One can see that the rekeying message size is closely related to $L(k)$ and reflects the trend of $N(k)$. A large amount of information about $N(k)$ and $L(k)$ is in the message domain.

To reduce the amount of GDI in the message domain, we insert phantom users into the system. These phantom users as well as their join and departure behaviors 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 the observed rekeying process.

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 group, 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 just 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 constant functions, that is

$$J_a(k) = L_0, \;\; L_a(k) = L_0, \;\; N_a(k) = N_0. \tag{11}$$

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 (11) may not be achievable. Since the real GDI functions are random processes, it is possible that the predetermined L_0 and N_0 are not large enough such that the artificial GDI cannot be maintained as the straight lines. For example, when $N(k) > N_0$, $N_a(k)$ cannot be N_0 because the number of phantom users must be non-negative. In fact, the artificial GDI functions must satisfy 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

$$N_a(k) = \max\{N(k), N_0\} \tag{12}$$
$$J_a(k) = \max\{J(k), L(k), L_0\} \tag{13}$$
$$L_a(k) = N_a(k-1) - N_a(k) + J_a(k). \tag{14}$$

When $N(k) \leq N_0$, $L(k) \leq L_0$, and $J(k) \leq L_0$, (12)–(14) are equivalent to (11). The artificial GDI functions in (12)–(14) obviously satisfy the requirement (r1), (r3), and (r4). Next, we prove that the requirement (r2) is satisfied.

- When $N(k) > N_0$, using the fact that $N_a(k-1) \geq N(k-1)$, $N_a(k) = N(k)$, and $J_a(k) \geq J(k)$, one can see that

$$\begin{aligned} L_a(k) &= N_a(k-1) - N_a(k) + J_a(k) \\ L(k) &= N(k-1) - N(k) + J(k) \\ La(k) &\geq L(k). \end{aligned}$$

- When $N(k) \leq N_0$, using the fact that $N_a(k-1) \geq N_0$ and $J_a(k) \geq L(k)$, we obtain $L_a(k) \geq J_a(k) \geq L(k)$.

It should be noted that there are many other ways to choose the artificial GDI functions. Some artificial GDI functions can protect GDI better than others. Artificial GDI functions can also be nondeterministic. In this paper, we use the artificial GDI functions in (12)–(14) to demonstrate our defense mechanism. The search for the best artificial GDI functions will be investigated in future work.

The proposed defense scheme is compatible with 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 group communication starts, create N_0 phantom users and establish a key tree to accommodate them. Set index $k = 1$.

3) While the communication 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 $k = k + 1$.

Fig. 4(a)–(c) illustrates 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 the communication overhead (i.e., the rekeying message size) is shown in Fig. 4(d). Here, the solid line represents the case with batch rekeying but no phantom users. The dashed line represents the case when the proposed defense method is applied. It is important to note that the batch rekeying technique is used for all of the results shown in Fig. 4. We can see that the observed rekeying process reveals very limited information about the real GDI when the proposed defense scheme is used. The rekeying message size resulting from using batch rekeying along is still highly correlated with the group size. In addition, 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 against one or several attack methods. For example, to prevent outsiders from launch—the AII-type attack—the rekeying messages can be embedded into multicast content [8] or transmitted using onion routing [34]. 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 fake rekeying messages to prevent the AII-type methods. The fake rekeying message could have a header indicating it is a rekeying message but the content is random bits. This is different from the proposed defense scheme where the key tree reserves slots for the phantom users and all rekeying messages have meanings. Compared with other techniques, using phantom users and batch rekeying has two major advantages. First, the proposed defense scheme is effective against various attacks. Since the real GDI is concealed before the rekeying messages are generated and even before key selectors are modified, only the artificial GDI can be seen from the observed rekeying process unless the KDC is compromised. Second, the proposed scheme does not rely on specific rekeying algorithms and is compatible with existing key management schemes.

It is important to point out that the idea of employing phantom users is not complicated. The challenge is to determine the amount of phantom users such that the observed rekeying process reveals the least amount of GDI given the resource consumption constraint. This issue will be addressed in the next section.

IV. Performance Measure and Optimization

In this section, we define two performance criteria and evaluate the performance of the proposed defense technique. The criteria are 1) the amount of information that has leaked to the insiders and outsiders measured by mutual information and 2) the communication overhead introduced by the phantom users. We study the tradeoff between these two metrics and provide a framework of choosing the proper amount of phantom users, described by the parameter L_0 and N_0 in (12)–(14).

A. Leakage of GDI

We use mutual information to measure the leakage of the GDI, which represents the maximum amount of information that can possibly be revealed. Let T be the total number of rounds of key updates. The overall service duration is $T \cdot B_t$. Then, the real GDI is described by a set of random variables as

$$R = \{N(1), \ldots, N(T), J(1), \ldots, J(T), L(1), \ldots, L(T)\} \tag{15}$$

and the artificial GDI is

$$A = \{N_a(1), \ldots, N_a(T), J_a(1), \ldots, J_a(T), L_a(1), \ldots, L_a(T)\}. \tag{16}$$

The mutual information $I(R; A)$ describes the reduction in the uncertainty of the real GDI due to the knowledge of the artificial GDI [29]. Therefore, the leakage of the GDI can be measured by

$$I(R; A) = H(A) - H(A|R) \tag{17}$$

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

Equations (12)–(14) indicate that the artificial GDI is a set of deterministic functions of the real GDI. Thus, the conditional entropy in (17) is 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 (14), the terms $L_a(1), L_a(2), \ldots, L_a(T)$ can be removed from the expression of the entropy of A (i.e., $H(A) = H(N_a(1), \ldots, N_a(T), J_a(1), \ldots, J_a(T))$). Then, the upperbound of $I(R; A)$ is calculated as

$$\begin{aligned} I(R; A) &= H(N_a(1), \ldots, N_a(T), J_a(1), \ldots, J_a(T)) \\ &\le \sum_k H(N_a(k)) + \sum_k H(J_a(k)). \end{aligned} \tag{18}$$

The equality is achieved when $\{N_a(k), J_a(k), k = 1, \ldots, T\}$ 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 highly dynamic. In these cases, (18) provides a tight upperbound of $I(R; A)$.

We introduce $p_{N_k}(n)$ and $p_{N_{ak}}(n)$ to denote the probability mass function (pmf) of $N(k)$ and $N_a(k)$, respectively. From (12), one can see that

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

Then

$$H(N_a(k)) = -(1 - \epsilon_N^k) \log(1 - \epsilon_N^k) - \sum_{n=N_0+1}^{\infty} p_{N_k}(n) \log p_{N_k}(n) \tag{19}$$

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

$$H(J_a(k)) = -\sum_j p_{J_{ak}}(j) \log p_{J_{ak}}(j), \tag{20}$$

and, shown in (21), at the bottom of the page, 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 upperbound of $I(R; A)$ is calculated from (18)–(21).

Since the observed rekeying process is determined by the artificial GDI, and the artificial GDI is only related to the real GDI, the following Markov chain can be formed: real GDI $\rightarrow$ artificial GDI $\rightarrow$ observed rekeying process. Thus, the mutual information between the observed process and the real GDI is no more than the mutual information between the real and artificial GDI [29]. Therefore, $I(R; A)$ is the upperbound of the amount of information that can be possibly revealed from the observed rekeying process.

From (12)–(14), 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 complementary measure for the leakage of the GDI. When the overflow probability is zero, the calculation in (18)–(20) leads to the result that $I(R; A) = 0$, which indicates perfect protection of the real GDI.

B. Communication Overhead

Communication overhead, measured by the rekeying message size, is one of the major performance criteria of key management schemes [1], [3]. We introduce the notation $M(L, N, d)$ as the expected value of the rekeying 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 fully loaded and balanced key tree. Then, there are d^l KEKs at the lth level of the key tree for $l = 0, \ldots, D-2$ and $D = \lceil \log_d N \rceil$, and the number of the KEKs at the $(D-l)$th level is $s_1 = \lceil (N - d^{D-l})/(d-1) \rceil$.

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

$$M(L, N, d) = E\left[\sum_{l=0}^{D-1} \alpha_l\right] = \sum_{l=0}^{D-1} E[\alpha_l]. \tag{22}$$

The expectation $E(.)$ is taken over the statistics of user departure behavior and the dynamic tree structure.

We introduce the notation $B(b, i, a)$, which is equivalent to the expected number of nonempty 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

$$E[\alpha_l] = d \cdot B\left(d^l, L, \frac{N}{d^l}\right), \quad 0 \le l \le D-2, \tag{23}$$

$$E[\alpha_{D-1}] = (d-1) \sum_{x=1}^{L} \frac{\binom{s_1}{x}\binom{N-s_1}{L-x}}{\binom{N}{L} B(s_1, x, d)}. \tag{24}$$

Using the fact that $\lceil i/a \rceil \le B(b, i, a) \le \min(b, i)$ (see Appendix), we derive the upper ound of the $M(L, N, d)$ as

$$M(L, N, d) \le dL \log_d(N). \tag{25}$$

This upperbound indicates that the communication overhead increases linearly with the number of departed users and with the logarithm of the group size.

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

$$C_a - C_r = \frac{1}{T}\sum_{k=1}^{T} M(L_a(k), N_a(k), d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \tag{26}$$

When the overflow probability is small, (26) can be approximated by

$$C_a - C_r \approx M(L_0, N_0, d) - \frac{1}{T}\sum_{k=1}^{T} M(L(k), N(k), d). \tag{27}$$

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 does not exceed certain requirements. When the overflow probability is small, the optimization problem is formulated as

$$\min_{N_0, L_0} \sum_k H(N_a(k)) + \sum_k H(J_a(k)) \tag{28}$$

$$p_{J_{ak}}(j) = \begin{cases} (1-\epsilon_J^k)(1-\epsilon_L^k), & j = L_0 \\ p_{J_k}(j) \sum_{y=0}^{j-1} p_{L_k}(y) + p_{L_k}(j) \sum_{x=0}^{j-1} p_{J_k}(x) + p_{J_k}(j) p_{L_k}(j), & j > L_0 \\ 0, & \text{o.w.} \end{cases} \tag{21}$$

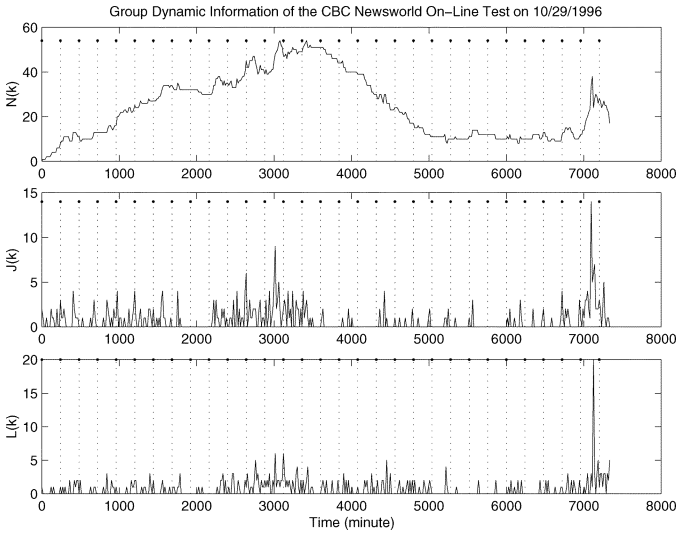


Fig. 5. GDI of a long audio session in MBone.

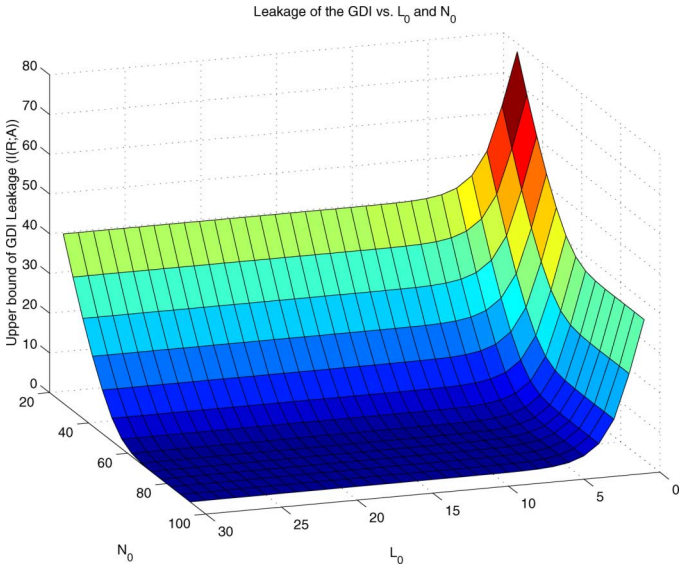


Fig. 6. Upperbound of the GDI leakages. (L_0 and N_0 are parameters in artificial GDI functions.)

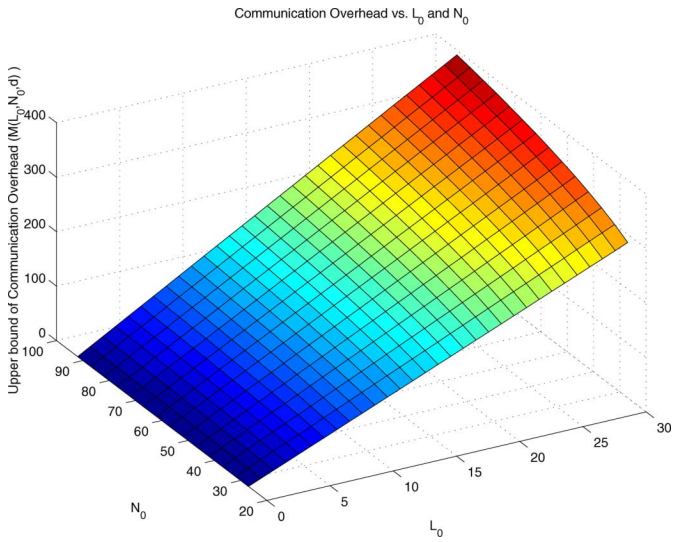


Fig. 7. Communication overhead $M(L_0, N_0, d)$. (L_0 and N_0 are parameters in artificial GDI functions.)

subject to

$$M(L_0, N_0, d) \leq \beta \tag{29}$$

where β is the maximum allowed communication overhead per key updating. We can show that $H(N_a(k))$ in (20) is monotonous nonincreasing with N_0; $H(J_a(k))$ in (19) is monotonous nonincreasing with L_0; and the communication overhead $M(L_0, N_0, d)$ in (22) is nondecreasing with L_0 and N_0. Therefore, the optimization problem is simplified as

$$\min_{L_0} \left(\sum_k H(N_a(k)) + \sum_k H(J_a(k)) \right) \Big|_{N_0 = M^{-1}(\beta)|_{L_0,d}} \tag{30}$$

where $M^{-1}(\beta)|_{L_0,d}$ is the largest value of N_0 that satisfies (29) with given L_0 and d. Fortunately, the number of departed users between two key updates is usually much less than the group size. Thus, the search space for parameter L_0 is not large and this optimization problem can be solved by a full search.

V. Simulations

Mlisten [31], a tool developed at the Georgia Institute of Technology, can collect the join/leave time for the multicast group members in MBone [30] sessions. The proposed defense scheme is applied to the data collected in 1996 [27]. Particularly, we selected one audio session that started on October 29th and lasted for about 5 days and 20 hours. Fig. 5 shows the values of $N(k)$, $L(k)$, and $J(k)$ of this session, where B_t is chosen to be 15 min.

It is suggested that the users' statistical behavior, such as interarrival and membership durations, can be modeled by exponential distribution in a short period of time [30]. In the simulation, the entire service time is divided into nonoverlapped sections, as illustrated in Fig. 5. The length of these sessions is set to be 4 h. 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 (18)–(21).

Figs. 6 and 7 demonstrate the upperbound of mutual information (see (18)) and the communication overhead $M(L_0, N_0, d)$ for different values of L_0 and N_0, respectively. It is noted that these two figures use different axes in order to show the properties of the 3-D curves. We can see that communication overhead is a nondecreasing function with L_0 and N_0, while the GDI leakage is a nonincreasing function with L_0 and N_0. This verifies the statement in Section IV.

Fig. 8 illustrates the solution of the optimization problem. Fig. 8(a) shows the maximum value of N_0 that satisfies the communication overhead constraint in (29) 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 upperbound of the GDI leakage $\sum_k H(N_a(k)) + \sum_k H(J_a(k))$ is evaluated only at $(L_0, N_0 = \max\{N : M(L_0, N, d) \leq \beta\})$,

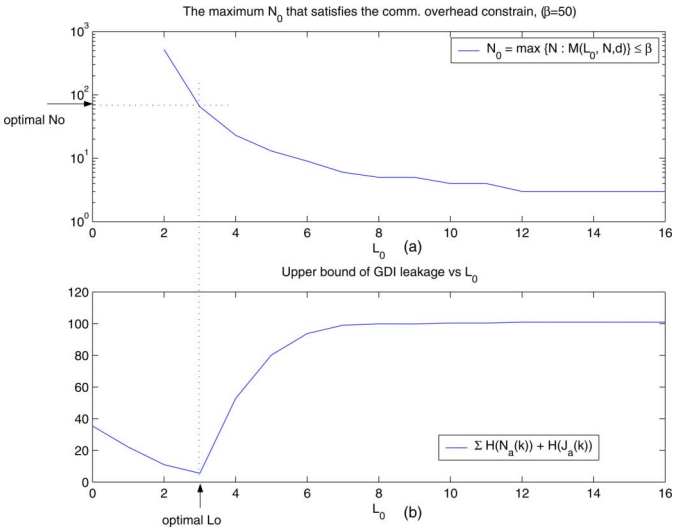


Fig. 8. Illustration of selecting optimal parameters L_0 and N_0.

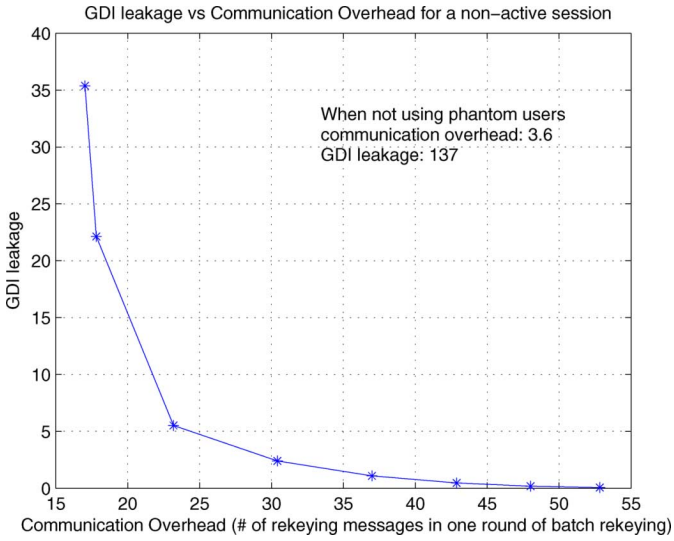


Fig. 9. GDI leakage versus communication overhead for a real MBone audio session, with and without phantom users.

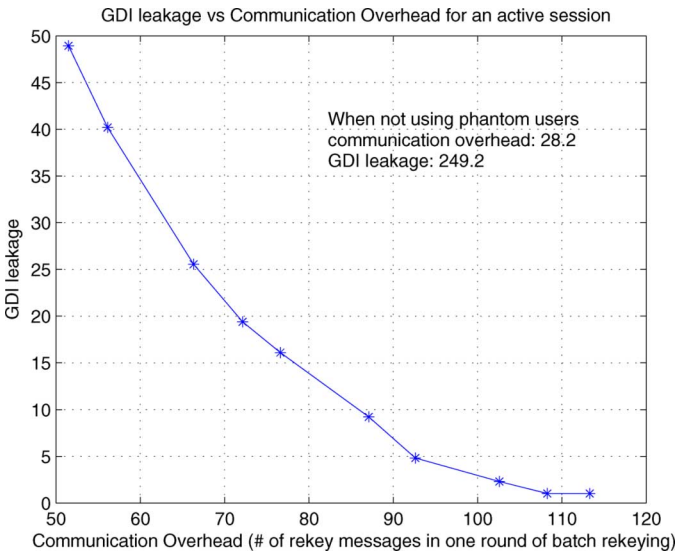


Fig. 10. GDI leakage versus communication overhead for a simulated multicast session, with and without phantom users.

which is shown in Fig. 8(b). The optimal values of L_0 and N_0 are also marked in Fig. 8(b).

Fig. 9 shows the tradeoff between the communication overhead and the GDI leakage. This figure demonstrates the upperbound 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 to determine the proper values of β for the communication constraint in (29). When not using the 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 requires an average of 3.6 rekeying messages to be sent in every 15 min interval ($B_t = 15$) and has $I(R;A) \approx 137$. Fig. 9 shows that the proposed defense scheme can reduce $I(R;A)$ to 5.5 by increasing the communication overhead to 23.2 messages per 15 min. 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 volume. On the other hand, the leakage of the group dynamic information is greatly reduced.

It is important to note that this MBone audio session contains only up to 60 users and represents the scenario where the group size is small and the 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 a large group size and active group members. The simulation setup is the same as that used for Fig. 2(c) in Section II, where the group size is about 500. When not using the phantom users, the KDC sends on average 28.16 rekeying messages every 5 min ($B_t = 5$), while $\mathrm{H(R)}$ is 249.2. The performance of the proposed defense methods is shown in Fig. 10. We can see that the GDI leakage can be reduced to 5 at the expense of increasing the communication overhead to 93 rekeying messages per 5 min. The relative communication increase is smaller than that in less active sessions. It is also noted that rekeying messages can be grouped together into several rekeying packets. when the key size is 128 b and the key selector is 4 B, each rekeying message contributes 24 B to the data payload. When the data payload of a packet is 1000 B long, it can accommodate around 40 rekeying messages. Thus, increasing the rekeying overhead from 28 messages to 93 messages does not significantly increase the number of rekeying packets.

VI. GDI Disclosure and Protection in Contributory Key Management Schemes

In many application scenarios, it is not preferable to rely on a centralized key server or some cluster heads that arbitrate the establishment of the group key. This might occur in applications where group members do not explicitly trust the key server or the cluster heads, or there are no servers or group members who have sufficient resources to maintain, generate, and distribute keying information. Thus, distributed solutions to the key management problem have attracted considerable attention [7], [15]–[23]. In the contributory key management schemes, every

group participates the process of group key establishment. The members' personal keys are not disclosed to any other entities [18]. Compared with the centralized schemes, the contributory schemes have the advantage of not putting full trust in a single entity and, therefore, do not suffer the problem of single-point failure.

In general, the contributory schemes are suitable for small–medium group size applications, where group dynamics are known to group members. In these cases, protecting GDI is not necessary. On the other hand, it is possible that some special applications use contributory key management and require confidential GDI. In this section, we show that there are many ways to obtain GDI and the cost for protecting GDI in contributory schemes is very high.

A. Fully and Partially Contributory Key Management Schemes

There are two types of contributory key agreement schemes: fully contributory and partially contributory. In the fully contributory schemes, all key agreement operations are distributed to every group member [19]. There is no dedicated group manager, and every participant may perform admission control and other administrative functions [19]. Thus, group members are naturally aware of the information about the group membership. Therefore, the fully contributory schemes rely on the members' knowledge of dynamic group membership, and are not suitable for the multicast applications with confidential GDI.

In the partially contributory schemes, one group member takes a special role and performs some administrative operations [18]–[22]. This special member is usually referred to as the group controller. The role of the group controller can be assigned to a fixed member or be handed over to other members when membership changes [19]. The group controller is different from the KDC in the centralized schemes. The group controller does not hold the private keys of other members or generate the complete group key. Instead, it performs admission control and coordinates the process of the key formation. The original purpose of introducing a group controller is to achieve efficient key updating [18]. In the context of protecting GDI, the partially contributory schemes make it possible to confine dynamic membership information to the group controllers while preventing other group members from accessing GDI. In a practical setting, multiple group controllers, who are trusted to keep GDI, must be used to prevent the single-point failure problem. In addition, to protect GDI, regular users cannot replace the group controllers even if all group controllers fail. Thus, the reliability of the group communication may suffer.

As a summary, GDI can only be protected in partially contributory schemes, at the expenses of utilizing trusted group controllers and the risk of communication failure.

B. GDI Disclosure in Contributory Key Management Schemes

Utilizing a group controller is not a complete solution to the GDI protection problem. There are many other opportunities for the insiders to acquire group dynamic information.

The scheme presented in [15] is the earliest attempted to extend the two-party Diffie–Hellman protocol to group applications. This scheme, sometimes referred to as ING [20], arranges members in a logical ring and is executed in $(n-1)$ rounds, where n is the group size. Therefore, every member obtains the group size by simply counting the number of rounds that he or she performed.

Similarly, the schemes presented in [16] and [17], referred to as the STR and BD, respectively, also reveal the group size. Here, each member receives the broadcast messages from all other members and, therefore, must know the existence of other group members.

In [21]–[23], logical tree structures are introduced to manage the formation of the group keys. In these schemes, each member performs L rounds and holds L subgroup keys, where L is the depth of the key tree. Since L is proportional to the logarithm of the group size, the group members know at least the order of the group size.

Another important set of contributory key management schemes is GDH.1, GDH.2, and GDH.3 [18]. These schemes arrange group members in a logical chain and accumulate the keying materials by traversing group members one by one. In GDH.1/2, the kth member receives k or $k+1$ messages from the $(k-1)$th member. Thus, the number of the messages reveals information about the group size. The users who are closer to the end of the chain have more accurate information about the group size. GDH.3 is executed in four stages [18]. In the second and the fourth stage, the last user on the key chain broadcast n messages to the rest of the group, and n is the group size. In all three schemes, the group size information is revealed by the size of keying messages.

C. Cost of Preventing GDI Leakage

It is seen that hiding GDI in contributory schemes is a very difficult task. Therefore, we suggest using the centralized key management schemes for the applications with confidential GDI. However, if the centralized schemes cannot be employed and GDI must be protected, which is a very rare case, a possible solution is to use GDH.3 with two modifications. The first modification is to use the group controller. Among all contributory schemes, GDH.3 has the strongest centrality flavor. The group members are arranged in a logical chain, and the group member at the end of the chain takes more responsibility than other members. If the group member at the end of the logical chain is selected as the group controller, which performs admission control and coordinates the key formation, a regular member only needs to communicate with his or her two neighbors on the key chain and the group controller. The second modification is to replace broadcast messages with multiple unicast messages. This is necessary to prevent GDI leakage through the size of the broadcast messages. In addition, antitraffic-analysis techniques, such as those in [34] and [35], should be used to prevent GDI leakage to the outsiders. This possible solution yields unbalanced load among group members and significantly increases protocol overhead and complexity. The high cost and complexity make the GDI protection impractical in a contributory environment. To summarize, contributory key management is not suitable for applications requiring GDI protection. The centralized key management scheme should be used for applications with confidential GDI.

VII. Discussion

Key management is not the only source, but is a critical source of GDI leakage. Attacks based on key management are effective, stealthy, and easy to launch. An attacker, who registers as a group member or monitors rekeying traffic near a group member, can obtain a large amount of GDI information without being detected.

Besides key management, monitoring multicast data delivery is another dimension for acquiring GDI. For the purpose of debugging, management, and modeling, various tools have been developed to monitoring multicast communications [36]. If the underlying multicast applications are "cooperative" (i.e., not using any preventive methods), one can obtain GDI using these tools. Generally speaking, the attacks based on data delivery monitoring are less attractive than those based on key management for two reasons. First, encryption and antitraffic analysis tools, such as onion routing, can disable or significantly reduce the effectiveness of these monitoring tools. Second, these monitoring tools involve high implementation cost. For example, many require installing agents in multicast-enabled networks in order to collect data delivery or group information [36].

In this paper, we focus on preventing GDI leakage from key management. In the future, for secure multicast applications with confidential GDI, research will be carried out to make the service providers have control over whether multicast monitoring tools can be used.

VIII. Conclusion

This paper raised the issues of the GDI disclosure through key management in secure group communications. Such a security concern has not been addressed in the design of current key management schemes. In particular, this paper has made two main contributions. First, we presented several effective methods that could obtain dynamic group membership information from the current centralized key management schemes. This study showed that GDI could be easily obtained by insiders and outsiders who exploited the rekeying messages in key management protocols. This posed a threat to group communications with confidential GDI. Second, we developed defense techniques that could protect GDI, by utilizing batch rekeying and phantom users. For the proposed defense techniques, the fundamental tradeoff between the communication overhead and the leakage of GDI was studied. In addition, this paper provided a brief discussion on the GDI problem in contributory key management schemes. It was argued that contributory schemes were not suitable for applications in which GDI should be protected.

In this work, the GDI disclosure problem was studied from the key management perspective. In future works, many other aspects, such as traffic analysis, can be jointly investigated with key management such that GDI will be better protected against attacks from other angles.

Appendix

We define $n(b,i,a)$ as the number of nonempty boxes when randomly placing identical i items into identical b boxes with repetition and each box can have, at most, a items. This appendix calculates the expected values 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 $B_0 = \lceil i/a \rceil$ 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 so that each box contains at least 1 and, at most, a items. $w(y,i,a)$ is calculated recursively as

$$w(B_0,i,a) = \binom{aB_0}{i} \tag{31}$$

$$w(B_0{+}1,i,a) = \binom{a(B_0+1)}{i} - \binom{B_0+1}{B_0} w(B_0,i,a) \tag{32}$$

$$\vdots$$

$$w(B_0{+}k,i,a) = \binom{a(B_0{+}k)}{i} - \sum_{m=0}^{k-1} \binom{B_0{+}k}{B_0{+}m} w(B_0{+}m,i,a) \tag{33}$$

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

$$\mathrm{Prob}\{n(b,i,a) = B_0 + m\} = \frac{1}{N}\binom{b}{B_0+k} w(B_0+k,i,a) \tag{34}$$

where $N = \binom{ab}{i}$ represents the total number of ways of placing i items into b boxes. Substituting (33) into (34)

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

It can be shown that $\left(\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}\right)/\binom{b}{B_0+m} = \binom{b-B_0-m}{k-m}$. Therefore

$$\begin{aligned}\mathrm{Prob}\{n(b,i,a) = B_0 + k\} &= \frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} \\ &\quad - \sum_{m=0}^{k-1} \binom{b-B_0-m}{k-m} \mathrm{Prob}\{n(b,i,a) = B_0+m\}.\end{aligned} \tag{35}$$

By substituting (31) into (34), we have

$$\mathrm{Prob}\{n(b,i,a) = B_0 + m\} = \frac{1}{N}\binom{b}{B_0}\binom{aB_0}{i}. \tag{36}$$

Based on (35) and (36), we can calculate $\mathrm{Prob}\{\mathrm{n(b,i,a)} = \mathrm{B_0 + k}\}$ for $k = 0, 1, \ldots, B_1 - B_0$ recursively. Then, we can calculate $B(b,i,a)$ as

$$\begin{aligned}B(b,i,a) &= E[n(b,i,a)] \\ &= \sum_{k=0}^{B_1-B_0} (B_0+k) \cdot \mathrm{Prob}\{n(b,i,a) = B_0+k\}.\end{aligned} \tag{37}$$

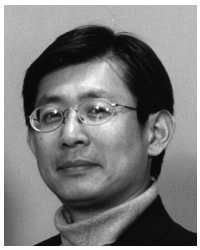
References

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

[2] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surveys*, vol. 35, no. 3, pp. 309–329, 2003.
[3] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures," Internet Draft Rep. 1998 [Online]. Available: draft-wallner-key-arch-01.txt.
[4] O. Rodeh, K. Birman, and D. Dolev, "The architecture and performance of security protocols in the ensemble group communication system," *ACM Trans. Inf. Syst. Security*, vol. 4, no. 3, pp. 289–319, Aug. 2001.
[5] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, Feb. 2000.
[6] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," in *Proc. 7th IEEE Int. Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises*, Jun. 1998, pp. 376–383.
[7] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE J. Selected Areas Commun.*, vol. 17, pp. 1614–1631, Sep. 1999.
[8] W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, "Key distribution for secure multimedia multicasts via data embedding," in *Proc. IEEE ICASSP*, May 2001, pp. 1449–1452.
[9] D. McGrew and A. Sherman, Key Establishment in Large Dynamic Groups Using One-Way Function Trees TIS Labs at Network Associates, Inc., Glenwood, MD, Tech. Rep. 0755, 1998.
[10] 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*, Mar. 1999, vol. 2, pp. 708–716.
[11] A. Perrig, D. Song, and D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. IEEE Symp. Security and Privacy*, 2001, pp. 247–262.
[12] G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Softw. Eng.*, vol. 15, no. 8, pp. 929–934, Aug. 1989.
[13] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGCOMM*, 1997, pp. 277–288.
[14] S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over IP multicast," *IEEE J. Select. Areas Commun. Special Issue Network Support for Group Communication*, vol. 20, no. 8, pp. 1511–1527, Oct. 2002.
[15] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 5, pp. 714–720, Sep. 1982.
[16] D. G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," in *Proc. Advances in Cryptology*. New York: Springer-Verlag, 1990, pp. 520–528.
[17] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution scheme," *Adv. Cryptology–Eurocrypt*, pp. 275–286, 1994.
[18] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," in *Proc. 3rd ACM Conf. Computer and Communications Security*, 1996, pp. 31–37.
[19] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A new approach to group key agreement," in *Proc. 18th Int. Conf. Distributed Computing Systems*, May 1998, pp. 380–387.
[20] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp. 769–780, Aug. 2000.
[21] G. Tsudik, Y. Kim, and A. Perrig, "Simple and fault-tolerant key agreement for dynamic collaborative groups," presented at the 7th ACM Conf. Computer and Communications Security, Nov. 2000.
[22] L. R. Dondeti, S. Mukherjee, and A. Samal, "DISEC: A distributed framework for scalable secure many-to-many communication," in *Proc. 5th IEEE Symp. Comput. Commun.*, 2000, pp. 693–698.
[23] W. Trappe, Y. Wang, and K. J. R. Liu, "Establishment of conference keys in heterogeneous networks," in *Proc. IEEE Int. Conf. Communications*, 2002, vol. 4, pp. 2201–2205.
[24] K. Almeroth and B. Quinn, "Ip Multicast Applications: Challenges and Solutions," IETF Draft 1998 [Online]. Available: draft-quinn-multicast-apps-00.txt.
[25] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying: A performance analysis," in *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2001, pp. 27–38.
[26] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A scalable group re-keying approach for secure multicast," in *Proc. IEEE Symp. Security and Privacy*, Oakland, CA, May 2000, pp. 215–218.
[27] [Online]. Available: http://ftp.cc.gatech.edu/people/kevin/release-data.
[28] W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, "Key management and distribution for secure multimedia multicast," *IEEE Trans. Multimedia*, vol. 5, no. 4, pp. 544–557, Dec. 2003.
[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience, 1991.
[30] 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*, Syracuse, NY, 1996, pp. 209–216.
[31] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Commun. Mag.*, vol. 35, no. 6, pp. 124–129, Jun. 1997.
[32] 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," presented at the IEEE ICDCS, Apr. 2000.
[33] Y. Sun, W. Trappe, and K. J. R. Liu, "An efficient key management scheme for secure wireless multicast," in *Proc. IEEE Int. Conf. Communication*, 2002, vol. 2, pp. 1236–1240.
[34] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Selected Areas Commun.*, vol. 16, no. 4, pp. 482–494, May 1998.
[35] R. E. Newman-Wolfe and B. R. Venkatraman, "High level prevention of traffic analysis," in *Proc. 7th Annu. Computer Security Applications Conf.*, Dec. 1991, pp. 102–109.
[36] K. Saraç and K. C. Almeroth, "Supporting multicast deployment efforts: A survey of tools for multicast monitoring," *J. High Speed Netw.*, vol. 9, no. 3–4, pp. 191–211, 2000.

Yan Lindsay Sun (S'00–M'04) received the B.S. degree (Hons.) from Peking University, Beijing, China, in 1998, and the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park, in 2004.

Currently, she is an Assistant Professor in the Electrical and Computer Engineering Department, University of Rhode Island, Kingston. Her research interests include network security, wireless communications, and networking. She received the Graduate School Fellowship at the University of Maryland from 1998 to 1999, and the Excellent Graduate Award of Peking University, Beijing, China, in 1998. She received the National Science Foundation Career Award in 2007.



K. J. Ray Liu (F'03) is Professor and Associate Chair, Graduate Studies and Research, of Electrical and Computer Engineering Department, University of Maryland, College Park. His research contributions encompass broad aspects of wireless communications and networking, information forensics and security, multimedia communications and signal processing, bioinformatics and biomedical imaging, and signal processing algorithms and architectures. He was the Editor-in-Chief of *IEEE Signal Processing Magazine* and the founding Editor-in-Chief of the *EURASIP Journal on Applied Signal Processing*.

Dr. Liu is Vice President—Publications and on the Board of Governor of IEEE Signal Processing Society. He is the recipient of many honors and awards including best paper awards from the IEEE Signal Processing Society (twice), IEEE Vehicular Technology Society, and EURASIP; IEEE Signal Processing Society Distinguished Lecturer, EURASIP Meritorious Service Award, and National Science Foundation Young Investigator Award. He also received various teaching and research awards from the University of Maryland, including the Distinguished Scholar–Teacher award, Poole and Kent Company Senior Faculty Teaching Award, and the Invention of the Year award.