PROTECTING GROUP DYNAMIC INFORMATION IN LARGE SCALE MULTICAST GROUPS

Yongdong Wu[1], Tieyan Li[1], and Robert H. Deng[2]
[1]Institue for Infocomm Research(I²R), *Singapore, {wydong,litieyan}@i2r.a-star.edu.sg;*
[2]School of information Systems, Singapore Management Univerist, robertdeng@smu.edu.sg

Abstract: Existing key management schemes can secure group communication efficiently, but are failed on protecting the Group Dynamic Information (GDI) that may undermine group privacy. Recently, Sun *et al.*[1] proposed a scheme to hide the GDI with batch updating and phantom members inserting so that an adversary is not able to estimate the number of group members. In this paper, we first point out that their scheme is only applicable in departure-only group communication instead of the common conference groups. Secondly, we introduce our method of estimating the group size at a higher confidence level given a prior departure probability. Further, to enhance GDI protection and extend the application fields, we propose to protect GDI with two new mechanisms: chameleon member identifications and virtual departure events. The proposed scheme is effective to protect both centralized groups and contributory groups. The simulation shows that our scheme is better on protecting the GDI.

1. INTRODUCTION

With the development of network applications such as pay-TV, remote education and videoconference, secure multicast distribution of copyright-protected or confidential material is more and more important. At the heart of the applications, a center broadcasts encrypted data to a large group of receivers so that only a predefined subset is able to decrypt the protected data. Broadcast encryption[2] deals with methods to efficiently broadcast information to a dynamically changing group of end members. It includes three major modules: Registration, Join and Departure[3]. Since the processing for registration and join requests is relatively easy, the main challenge in managing a group is how to exclude some subset of the members efficiently (e.g., Dodis *et al.*[4]). Technically, this *blacklisting Problem* is how to distribute content decryption keys over a shared insecure channel so that only intended receivers can get the key efficiently in terms of key storage[5] and/or communication overhead[6]. Based on the survey[7], the key management methods are categorized into two classes: (1). Centralized Group Key Management: A secure multicast scheme allows one or more group controllers (GCs) to send key updating messages securely over a multicast channel to a dynamically changing group of members[8]. For example, in logical-tree-hierarchy (LKH) scheme[9] and LKH+[10] scheme, the leaf nodes represent the member keys, and the non-leaf nodes are key encryption keys (KEKs) which are used to encrypt the control traffic. Specially, the root represents the group key. (2). Contributory key management: There is no GC but each member has the same task in managing the group key so as to remove the bottleneck of GC and increase the security. For instance, Rodeh *et al.*[11] divided the group members into sub-groups which have leaders. The leaders exchange keys on behalf of the sub-group members to agree a group key.

The state-of-the-art key management schemes are designed to prevent unauthorized access to the multicast content, but unfortunately provide opportunities for unauthorized parties to obtain group dynamic information (GDI). That is to say, the number of join members, departure members and the size of the group are disclosed. As noted in Sun *et al.* scheme[1] (hereafter called as SL scheme): "some applications such as subscription services, and military, 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."

Sun *et al.*[1] not only described the leakage of GDI in the existing key management schemes, but also improved the design of current key management schemes such that both GDI and the multicast content are protected. Technically, Sun *et al.* proposed a protection method by batch re-keying and phantom members. Their upgraded scheme provided some protection on the GDI, but we think it is not sufficient in terms of applications and security strength.

The present paper focuses on the protection of group dynamics, in particular to the large scale group because the adversary is hard to accurately estimate the GDI of a small group. Since non-tree-based schemes (e.g. Safaeli *et al.*[7]) are merely applicable for small groups, we deal with the binary tree-based scheme only in the following sections. We denote a path as the node sequence from leaf to root. The path length is the number of nodes in the path. l(x) represents the number of nodes in the path from node x to root. Particularly, the depth of the root is 0. Each key has a field SN Sequence Number which indicates the position of a key in the tree. Our main contributions include: (1) address the security flaw of the scheme[1]; (2). propose methods to protect GDI with virtual events and chameleon members. The proposed schemes are effective in protecting both centralized group and contributory group.

The remainder of the present paper is organized as follows. Section 2 refreshes the SL scheme. Section 3 addresses our analysis on SL scheme. Section 4 depicts our proposed GDI protection method and its extension to contributory group key management. Section 5 describes the comparison between SL scheme and our scheme. A conclusion is drawn in Section 6.

2. SL HIDING SCHEME

In many group communications, group dynamic information is confidential and should not be disclosed to either inside group members or outsiders. An important goal in SL scheme[1] is to produce an observed re-keying process that reveals the least amount of information about the GDI. The tools in SL scheme to hide GDI include batch re-keying and phantom member insertion.

Periodical batch re-keying is to postpone the updates of keys such that several members can be added to or removed from the key tree altogether. Compared with updating keys immediately after each member joins or departures, batch re-keying reduces the communication overhead at the expense of allowing the joining/leaving member to access a small amount of information before/after his join/departure. But batch updating provides little contribution for the GDI protection if the attacker can intercept and parse the re-keying messages.

Phantom members inserting is a way to hide the number of the real members. These phantom members, as well as their join and departure behavior, are created by GC. As a result, the combined effects of the phantom members and the real members lead to an artificial GDI which is observed by the attackers.

According to notation in SL scheme, N_k, J_k and L_k are the number of the real members, and the real join event, and real departure event at time k respectively, $N_a(k)$, $J_a(k)$ and $L_a(k)$ are the total number of members, total number of join events, and total number of departure events respectively. $N_a(k)$, $J_a(k)$, and $L_a(k)$ are referred to as the artificial GDI whose target values are N_0, J_0 and L_0 respectively. Thus, the artificial GDI functions are:

$$N_a(k) = \max \{N_k, N_0\} \quad (1)$$

$J_a(k) = \max \{J_k, L_k, L_0\}$
$L_a(k) = N_a(k\text{-}1) - N_a(k) + J_a(k)$

Due to Eq.(1), the number of key tree leaves is that of the real members if $N_k \geq N_0$. Therefore, an attacker can estimate the group size if he is permitted to join the group without limitation. For example, an attacker floods N_0 join requests to GC so as to exclude all phantom members. Thanks to the group size technologies described in Section 3, the attacker can obtain the GDI easily. To defeat this attack, N_k must always be less than N_0 by imposing some limitations on the join requests, for example, restrict the number of join requests by checking the identification of the requestor.

3. DISCLOSING GDI PROTECTED WITH SL SCHEME

This section describes an approach to estimate GDI protected with methods in SL scheme. To this end, subsection 3.1 firstly introduces how to estimate the group size with the tree depth since a key-tree tends to be balanced in a long term communication. Secondly, Subsection 3.2 describes how to refine the key-tree with the key SNs based on the re-keying messages. Finally, the number of real members is estimated thanks to some prior knowledge on the member departure.

3.1 Step 1: Grossly estimating the group size

Generally, the key-tree is changed from time to time due to the group dynamics. However, the tree will tend to some stationary status with the updating strategy in SL scheme: (1) The number of tree leaves is invariable, i.e., the umber of departure members is equal to that of the join members at any updating time. (2) Every member (i.e., tree leaf) has the same departure probability, but the join member is always located in the shortest path for the sake of lightest communication overhead.

This subsection discloses the key-tree asymptotic property in SL scheme to estimate the number of tree leaves. Denote the initial tree as T_0, the tree is T_k at time k. $l_{max,k}$ is the length of the longest path and $l_{min,k}$ is the length of the shortest path in the tree T_k.

Theorem: If a join member is always mounted in the shortest path, and a member departures from any leaf at an identical probability, an arbitrary binary tree will tend to be balanced given a departure event is always followed by a join event. Formally, let $P(X)$ to be the probability of random variable X, $\lim_{k\to\infty} P(l_{max,k} - l_{min,k} \leq 1)=1$

Proof: see appendix.

Although the theorem is expressed for two events at each updating time, it holds for any even number of events. Consequentially, the key tree will tend to be a balanced tree if the probability of departure event is identical to that of join event. Thus the size of re-keying messages is almost constant for each departure event[23] in a long term. For example, Fig. 1 illustrates the dynamic property of the key tree. Initially, the tree of depth 10 is very unbalanced ($l_{max,0}$ - $l_{min,0}$ = 8), but $l_{max,k}$ - $l_{min,k}$ is smaller and smaller with more and more events, i.e., the tree is more and more balanced. Because $l_{min,k}$ and/or $l_{max,k}$ are publicly available from the re-keying messages, the adversary estimates the group size N_0 as

$$2^{l_{min,k}} < N_0 < 2^{l_{max,k}}$$

for a sufficiently large k. This method grossly estimates the group size without employing the re-keying message content. In the following, the adversary exploits the key SNs from the re-keying messages so as to refine the number of tree leaves.

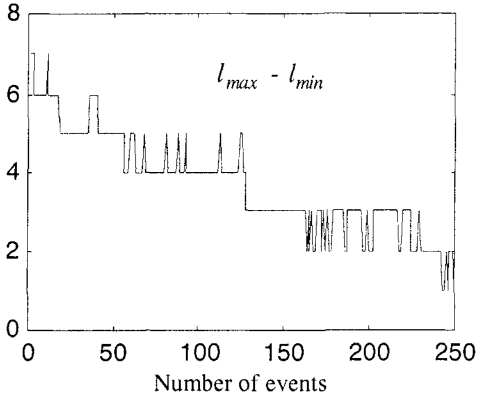

Figure 1. Tree depth difference dynamics. With more and more join/departure events, the difference $l_{max,k}$ - $l_{min,k}$ tends to 1.

3.2 Step 2: Refining the estimation

Assume at time t_0, the attacker starts to monitor the network traffic so as to reconstruct the key tree T_{t_0} at time t_0. To accomplish this, the adversary analyzes the re-keying messages one by one and updates the reconstructing

[23] In the Subsection II-C of SL scheme, the departure position (L_1, L_2, ..., L_W) is assumed to be *i.i.d.* We think this assumption is not true for a long duration communication. Thus, the attack AII addressed SL scheme is impractical.

key tree with the following heuristic knowledge: for each departure event, if the departure member does not join after t_0, the member must be in T_{t_0} ; for each join event, the new leaf position must be empty in T_{t_0} unless it is blank due to the departure events after t_0.

Since phantom members are added into the group such that the group size is invariable in SL scheme, i.e., $N_a(k)= N_0$, what the adversary obtains is the artificial size N_0 which includes the number of the phantom members. Therefore, the adversary has to filter out the number of phantom members with the following method.

3.3 Step 3: Filtering the phantom members

According to the experiments[12,13] on member departure, the member arrival process can be modelled as Poisson; the membership duration of short sessions is accurately modelled using an exponential distribution; and the membership duration of long sessions is accurately modeled using the *Zipf* distribution (`http://www.useit.com/alertbox/zipf.html`). Therefore, the adversary is able to exploit the difference between real member and phantom member. The former departures at its own probability which may be available in advance with the domain knowledge, and the departure probability of the latter is decided by GC. Before elaborate the process of estimating the real members, let's play a game.

Game: With respect to Fig.2, there are 2 boxes called G_1 and G_2. The first box G_1 has a black balls and b white balls, another box G_2 has x black balls and y white balls, where a, b, x+y=N are known, but x and y are unknown. At each time, one ball will be taken from the boxes.

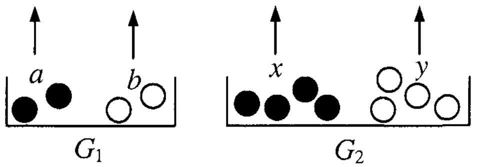

Figure 2. Experiments for simulating group dynamics. The black balls represent real members, while white balls represent the phantom members. The tested members map to the balls in box G_1.

The rules for taking balls with repetition are as follows:

(1) the black balls will be taken with a predefined probability f. Each black ball has the same probability to be taken.

(2) only after no black ball is taken in step (1), the white balls will be selected uniformly. Define the random events

X_1: a black ball taken from the first box G_1

X_2: a white ball taken from the first box G_1
X_3: a black ball taken from the second box G_2
X_4: a white ball taken from the second box G_2
Then the probability of the taken ball is from the first box G_1 as

$$P(X_1 \cup X_2)=P(X_1)+P(X_2) = \frac{a}{a+x} f + \frac{b}{N-a-x}(1-f) \qquad (2)$$

Given that the number of balls in the first box is much smaller than that of the second one (a+b « x+y), and the observer can tell which box the ball is taken from, but he can not tell the color of the taken ball. After many experiments (return the ball to the original box), the occurrence frequency α of balls taken from the first box is

$$\alpha \approx P(X_1 \cup X_2) \qquad (3)$$

That is to say, the observer can estimate the number of white (or black) balls according to Eq.(2) and Eq.(3). Referencing to Table 1, the adversary selects all the joining members as a tested set of members at a updating time, and other members are regarded as non-tested. With the following ways, an adversary can obtain the parameters to play the game on the GDI.

- In an updating period, the attacker can detect all the requests (join and/or departure) by eavesdropping the network. Equally, the attacker knows a in the above experiment. Meanwhile, the attacker can deduce the number b from a re-keying message in the same period. But the attacker can not distinguish the real members from the phantom members due to batch updating. Thus, the adversary can construct a box G_1.
- The number N of the old members (balls in G_2) is known based on Subsections 3.1 and 3.2.
- Since GC does not remove a real member otherwise the victim will protest. Thus, the real member will quit at her own probability f. According to the experiments[13], the probability f may be available.
- A phantom member can be selected and excluded uniformly if no real member quits.

Careful readers may notice some minor difference between the non-tested set and G_2 in that non-tested set may be changed from time to time, while the balls in G_2 are identical all the time. But we think this difference will not incure big estimation error if a+b « N. In addition, the game simplifies the group dynamics with only one ball leaving at each time, However, it can be extended to multiple balls leaving like the case of batch updating.

Strictly speaking, f varies with the number of real members. To increase the estimation of f, we can repeat the above experiment with the estimated number x till the difference between two estimations is smaller than a threshold. As a result, the number x of real members is estimated according to the above game.

Table 1. Mapping the game to GDI. #*X*: the size of *X*

	Game	GDI
G_1	tested balls	tested real members
G_2	non-tested balls	non-tested members
a	# black balls in G_1	# tested real members
b	# white balls in G_1	# tested phantom members
x	# black balls in G_2	# non-tested real members
y	# white balls in G_2	# non-tested phantom members
N	# balls in G_2	# non-tested members
f	probability of black ball leaving	probability of real member leaving

4. PRESENT GDI PROTECTION SCHEME

The present approach employs chameleon ID and virtual member switch for hiding GDI. Chameleon ID means that each user has more than one ID, or multiple leaf positions in the key tree; and virtual member switch is to remove an innocent member and then artificially insert him again. With these technologies, as well as batch updating and phantom members, the phantom member and real member are statistically indistinguishable such that the estimation method in Section 3 is of no use.

4.1 Chameleon Members

In the SL scheme[1], the approach of inserting phantom members provides GDI protection to some degree. But it may be vulnerable in the group communication, in particular to conferences because the phantom members are always silent. If the attacker treats the members who are dumb for a long time as phantom members, the attacker is able to separate the real members from the phantom members. Therefore, the protection methods in SL scheme are merely applicable to the content delivery, i.e., one member acts as a server or speaker, and others are listeners.

In order to hide GDI in all kinds of the group communications, we propose a GDI protection based on the idea of chameleon IDs. Fig.3

illustrates the response process of join event with chameleon IDs. The request message is R_M=(JOIN, M, **E**(ID$||n$, K)), where **E**(.) is an encryption function, ID is a random number for identifying the requestor M, and key K is known to both GC and the requestor and n is the number of chameleon IDs for the requestor M.

With the secret key K, GC obtains ID and n by decrypting R_M and stores them for virtual departure (see Subsection 4.2), then unicasts a response $\{\mathrm{ID}_i, \mathbf{E}(C_i, K)\}^n$ (i=1,2, ..., n) to the requestor, where ID_i=**H**(ID $||$ i), C_i is a set of re-keying messages (SN_j, K'_{SN_j}), SN_j is the key sequence number in the whole key tree. Here, assume that the attacker can intercept all the re-keying messages but can not identify whether the recipients of two messages are identical or not. To be efficient, a weaker assumption is that the attacker can not identify a majority of recipients of the re-keying messages. K'_{SN_j} is derived from the old K_{SN_j} with a one-way function. Hence the requestor joins the group as n members. Because all the ID_i are non-linkable and the number of artificial join requests is unknown to the attacker, the attacker can not distinguish the real join re-keying message from the artificial ones. Furthermore, the chameleon IDs are used at the same probability in the communication sessions such that no dumb members exist.

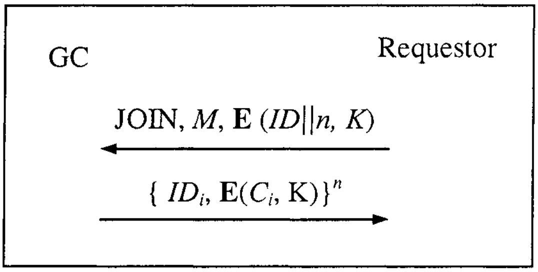

Figure 3. Diagram for member join. After receiving the join request from a new member, GC adds several leaves in the key tree for the new member.

4.2 Virtual Departure

In the attack proposed in Section 3, the adversary exploits the statistics difference between the real member and phantom member. In order to hide *a prior* departure probability of real members, GC will exclude a real member, afterwards let her join. To accomplish this virtual departure, GC will select an innocent member, send the exclusion message as usual. Optionally, not all of her chameleon IDs are removed such that the member can still involve in the group communication and postpone the join process to a later time. When an innocent member is revoked for no reason, she knows that she can obtain a new key from the re-keying message sooner or later. For example, the re-keying messages with ID_i=H(ID || $n+t$) indicates her new join KEKs, where t is the updating time. Therefore, the number of the members is invariable. Since either real member or phantom member has the same departure probability, the attack addressed in Section 3 is foiled.

4.3 Mixed Departure with Join

This section addresses a batch way to reduce the overhead efficiently. To deal with join and departure events simultaneously, whether the join/departure is virtual or real, all the keys from the removed leaf node to the root must be changed. Based on our observation in Section 3.1, most of leaves are located in $D = \lfloor log_2N_0 \rfloor$ and D-1 levels. It is natural to insert the new members into the departure positions. The rules for processing the member departure are as follows:

- Prune the sub-trees which include all and exactly the departure members.
- For each remaining leaf node, if its sibling node departures, its new parent KEK is unicast to him securely.
- For each remaining internal node of level $l < D$-1, its new KEK will be multicast securely to the survival child node if only one child node is removed. Otherwise, if at least one of its non-immediate descendant node departures, its KEK is changed and multicast twice securely to its two children nodes.

4.4 Extension to Contributory Group GDI Protection

In the paper[1], "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." However, this oracle is not true based on our chameleon IDs.

Technically, in the setup stage of a contributory group key, the members broadcast the messages so that all the members share the same key. For the sake of clarity, we adopt the distributed LKH[11] to explain the protection process for contributory group communication. At the stage of forming a group key, each member has several chameleon IDs and each ID corresponds to a leaf of the key tree. After constructing the tree with Rodeh's scheme[11], the root is the shared key. In the conversation stage, the member talks with different IDs from time to time. In this protected contributory GDI, the height of the key tree is public, but the number of chameleon IDs of each member is secret, thus neither the insider nor the outsider can obtain the number of the real members.

In the Distributed LKH, the key tree depth increases $log_2 N_1$ -$log_2 N_0$ if the number of members is increased from N_0 to N_1, thus the communication overhead and computational cost increases $log_2 N_1$-$log_2 N_0$ on average to achieve GDI protection.

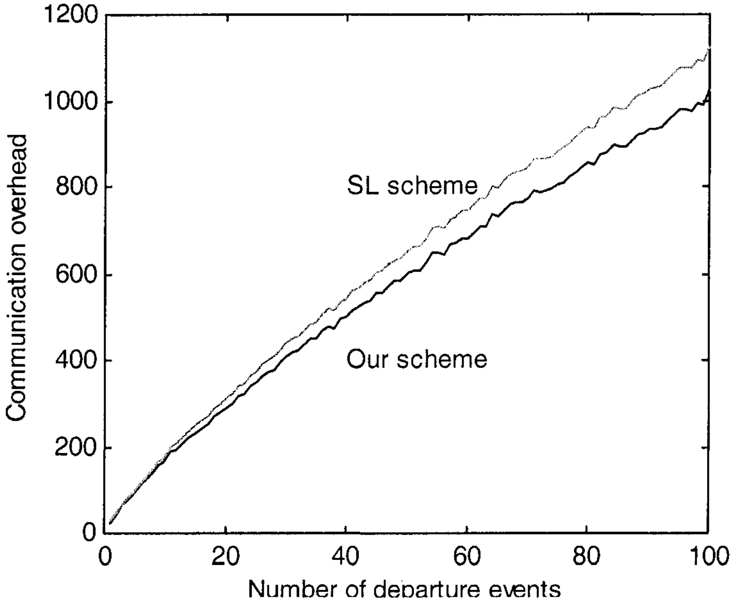

Figure 4. Communication overhead vs. the number of departure events (x-axis). Given a group of 4000 members, the members leave the group uniformly. The overhead is represented with the number of re-keying messages.

5. PERFORMANCE

In the following simulations, suppose that the adversary knows the total number of members, as well as the departure probability of the real members.

5.1 Network overhead

With the proposed approach of batch departure in Subsection 4.3, the communication overhead are shown in Fig.4 and Fig.5. The size of re-keying messages of the proposed batch departure (solid line) is smaller than that in SL scheme (dashed line) since we do not multicast KEKs of pruned subtrees.

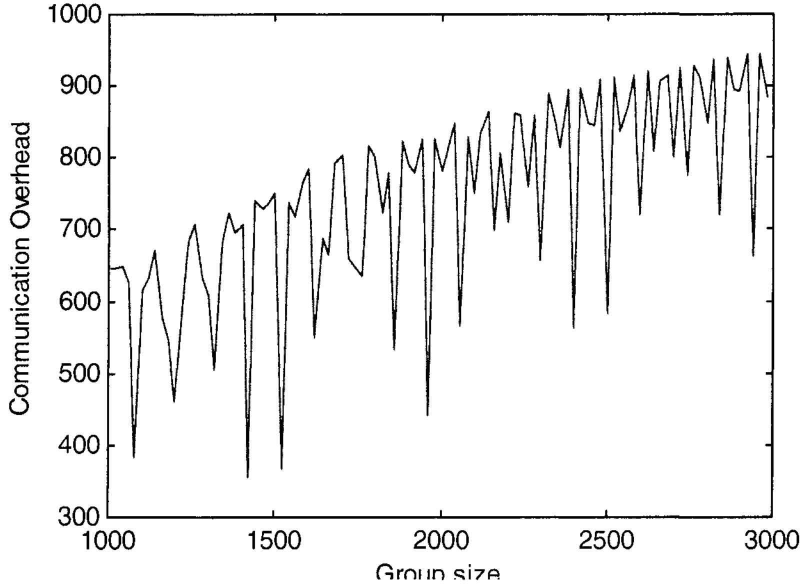

Figure 5. Communication overhead vs. group size. Given 100 members leaves uniformly, the communication overhead increases slowly with the group size.

5.2 GDI Protection Capacity

5.2.1 Estimation on GDI protected with SL scheme

Fig.6 illustrates the estimation result for the real members protected with SL scheme. With reference to Table I, assume G_1 includes a=10 real members and b=10 phantom members, and G_2 has N=4000 members including x=1000 real members and y=3000 phantom members. The experiment is repeated with 10000 times, the departure probability f of the real member is selected as 0.01. Denote $\tilde{N}$ as the estimated number of real members. Define Z=($\tilde{N}$ /N-1) as the normalized estimation error. In Fig.6, mean error $\mu_0 = |$**Mean**$(Z)| = 0.15$ and standard deviation σ_0=**Var**(Z)= 0.1232.From the experiments, we come to a conclusion that the size of the real members can be estimated to some extent. If the probability f is smaller, the estimation result is better. Thus, the attacker may select the time slice for estimation when few real members departure.

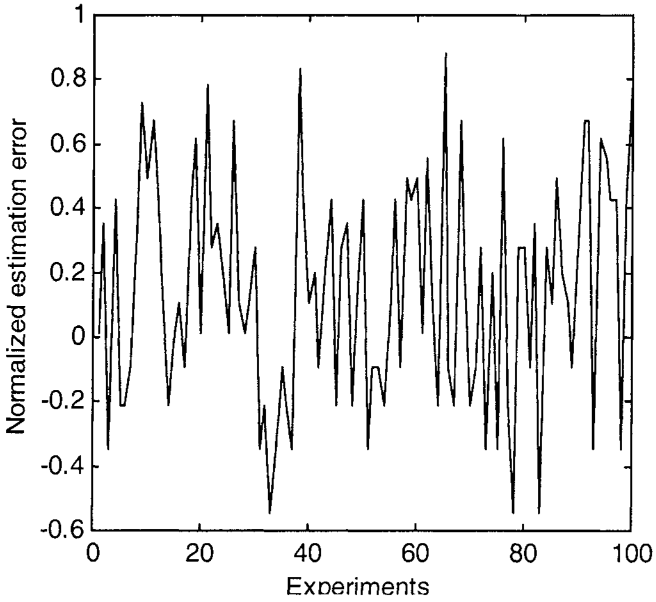

Figure 6. Estimation error of group size when GDI is protected with SL scheme.

5.2.2 Estimation of GDI protected with our protocol

Fig.7 illustrates the estimation result for the real members protected with the virtual departure. Its configuration parameters are the same as those in Subsection 5.2.1. In Fig.8, the mean error $\mu_l = |\mathbf{Mean}(Z)| = 0.3458 > \mu_0$. And the standard deviation $\sigma_l = \mathbf{Var}(Z) = 0.587 > \sigma_0$. It means that the virtual departure results in much greater estimation error, and the estimation results are not stable due to high variance.

With the same parameters as Fig. 6, Fig. 8 illustrates the estimation result for the real members protected with chameleon IDs (here the number of

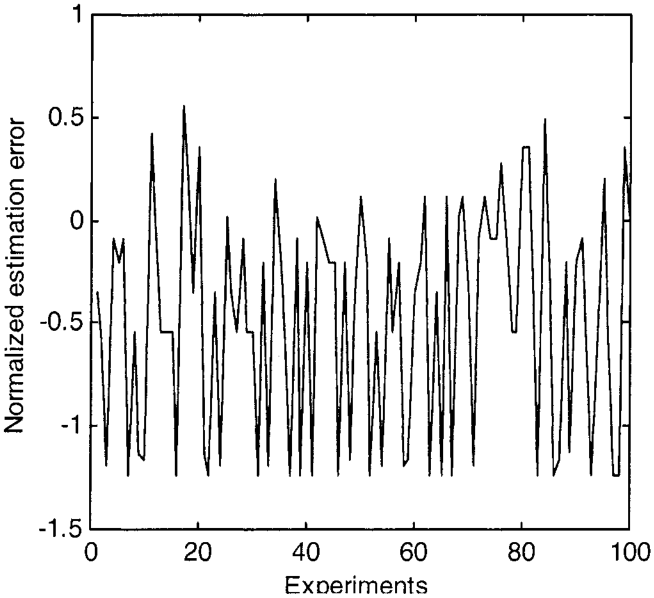

Figure 7. Estimation error of group size when GDI is protected with virtual departure.

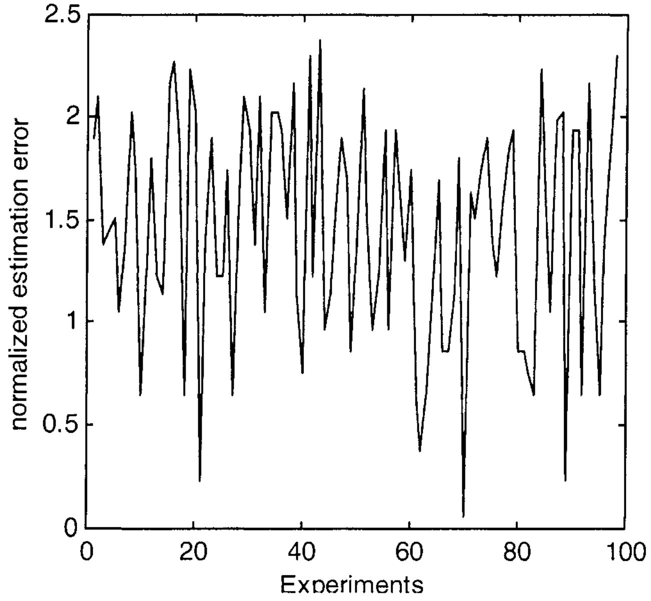

Figure 8. Estimation error of group size when GDI is protected with chameleon IDs.

chameleon ID is equal to the real member). The mean of estimation error $\mu_2 = |\mathbf{Mean}(Z)| = 0.476 >> \mu_0$, and standard deviation $\sigma_2 = \mathbf{Var}(Z) = 0.2956 > \sigma_0$. Therefore, the estimated number of real members is far from true value, and the estimation results are not stable due to high variance.

6. CONCLUSION

This paper investigates the protection schemes of hiding GDI and points out the security flaw in SL scheme. We further propose a method to protect GDI, such as the number of group members, from disclosing. In this paper, we consider the membership dynamic in the whole process including set up, join/leave, and communication session. Although the adversary is assumed to have more knowledge (e.g., knowing the key tree) than that in SL scheme, she is unable to even closely approximate the number of real members due to the technologies of chameleon IDs and virtual departure. In addition, our method is applicable to both centralized group and contributory group.

REFERENCE

1. Yan Sun and K.J. Ray Liu, "Securing Dynamic Membership Information in Multicast Communications, " IEEE Infocom, 2004
2. A. Fiat and M. Naor, "Broadcast Encryption, " Crypto '93, LNCS 773, pp. 480-491, 1993
3. R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor and B. Pinkas, "Multicast Security: a Taxonomy and Some Efficient Constructions, " IEEE Infocom, vol. 2, pp. 708-716, 1999.
4. Yevgeniy Dodis and Nelly Fazio, "Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack, " Public Key Cryptography, LNCS 2567, pp. 100 - 115, 2003
5. Yuh-Min Tseng, "A Scalable Key-management Scheme with Minimizing Key Storage for Secure Group Communications," International Journal of Network Management, pp.419-425, 2003
6. Jack Snoeyink, Subhash Suri and George Varghese, "A Lower Bound for Multicast Key Distribution, " IEEE Infocom, pp.422-431, 2001
7. Sandro Rafaeli and David Hutchison, "A Survey of Key Management for Secure Group Communication, " ACM Computing Surveys, 35(3):309-329, 2003
8. Alan T. Sherman and David A. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," IEEE Trans. Software Eng. 29(5):444-458, 2003.
9. D.M. Wallner, E.J. Harder and R.C. Agee, "Key Management for Multicast: Issues and Architectures," Internet Request for Comments 2627, June, 1999. ftp.ietf.org/rfc/rfc2627.txt

10. 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, 17(9):1614-1631, 1999.
11. O. Rodeh, K. Birman and D. Dolev, "Optimized Group Re-key for Group Communication Systems, " Network and Distributed System Security, pp. 39-48, 2000
12. K. Almeroth and M. Ammar, "Multicast Group Behavior in the Internet's Multicast backbone (MBone), " IEEE Communications Magazine, 35(6):124-129, 1997.
13. S. Acharya, B. Smith and P.Parnes, "Characterizing User Access to Videos on the World Wide Web, " ACM/SPIE Multimedia Computing and Networking, 2000

APPENDIX

Proof of Theorem 1: Let $D=\lfloor log_2 N_0 \rfloor$. The member set $U_k=\{M_i \mid \mathbf{l}(M_i)> \mathrm{D}\}$ is defined after the updating time k, but before the updating time k+1, where M_i is a node in the tree, $\mathbf{l}(x)$ is the depth of a leaf x.

Define two events as Departure event e_k and Join event E_k at time k.

(1) The leaves in U_k will vanish gradually.

Because $N_a(k)=N_0 \le 2^D$, the join member will be inserted into the levels smaller than D+1. Thus, the join event has no impact on U_k. Thus, $U_0 \subseteq U_1 \subseteq \ldots$, the size of U_k is monotonically deceasing, $|U_0| \ge |U_1| \ge \ldots \ge 0$

Denote the departure event of member m as

$$e_k = \begin{cases} 1: & m \in U_{k-1} \\ 0: & otherwise \end{cases}$$

If $|U_{k\text{-}1}|>0$, $P(e_k=1) = |U_{k\text{-}1}|/N_0 \ge 1/N_0$. Considering $e_1, e_2, \ldots$ are independent, for any arbitrary positive ε_1, there is some c such that

$$P(E_1 + E_2 + \ldots + E_c = |U_0) \ge 1\text{-}\varepsilon_1.\ \text{ i.e., } P(|U_c|=0) \ge 1\text{-}\varepsilon_1.$$

(2) The leaves in any level $l< D$-1 will vanish gradually.

Assume the length of the shortest path is $l_{\min,0}<D$-1 initially, and S_k is the set of the leaves in level $l_{\min,0}$. At time k, denote the departure event of member m as

$$E_k = \begin{cases} 1: & m \notin S_k \\ 0: & otherwise \end{cases}$$

Suppose $n_0=|S_0|$ is the number of leaves in the level $l_{\min,0}$ at the initial time. Due to $P(E_k=1)=|S_k|/N_0 \ge 1/N_0$, if $|S_k| \ne 0$, for any arbitrary positive $\varepsilon_2 >0$, there is some r such that

$$P(E_1 + E_2 + \ldots + E_k = n_0) \ge 1\text{-}\ \varepsilon_2.\ \text{ i.e., } P(|S_r| = 0) \ge 1\text{-}\ \varepsilon_2.$$

According to Eq.(1), the leaves in the shortest path will disappear gradually. That is to say, the length of shortest path will increase. Repeat the above steps, the shortest path will be longer and longer till D-1 at the probability of at least 1-ε_2. That is to say, for any level l= $l_{\mathrm{min},0}$, ..., D-2, there is some r_l, no leaves exist at all given arbitrary positive ε_2 after r_l join events.

(3) After sufficient number of events, there is no leaf x where either $\mathbf{l}(x)$>D or $\mathbf{l}(x)$<D-1 at a high probability. Formally, For any arbitrary positive $\varepsilon = \min(\varepsilon_1, \varepsilon_2)$, after updating θ times, the leaves are in levels D and D-1 at the probability at least 1-ε, where

$$\theta = c + r_{l_{\mathrm{min},0}} + r_{l_{\mathrm{min},0}+1} + \ldots + r_{D-2}\,.$$