ID-GC: An Efficient Identity-based Group Key Management Scheme

Zhibin Zhou and Dijiang Huang
Arizona State University

Abstract

We present an efficient identity-based key management scheme for group communications (ID-GC). ID-GC outperforms all existing group key management schemes in terms of reducing the communication overhead for group management, e.g., adding or deleting group members. In average cases, ID-GC requires $O(\log N)$ messages for removing multiple members and 1 message for adding a new group member, where N is the group size. ID-GC scheme provides group forward/backward secrecy, and it is resilience to colluding attackers.

I. Introduction

Media broadcast over the un-trusted networks, e.g., Internet, poses security issues. One problem is the access control to a large number of clients in a public broadcast group. Typical applications include but not limited to: digital-TV, content distributions, on-line multi-player games, video conferencing, and so on, where only registered (or legitimated) users can reveal broadcasted data content. In these application environments where the group size is large, a common solution is to encrypt the broadcast data and to disclose the symmetric group key to groups users only. The main research challenge is how to reduce the communication overhead invoked by group membership management. Particularly, we notice that the communication overhead is highly related to the number of sub-groups each group member can directly participate based on its predistributed secrets. Here, we illustrate their relations using two special-case examples. Firstly, in a group G of size N, each user u_i processes keys for all possible 2^{N-1} sub-groups that include u_i; in other words, each user belongs to 2^{N-1} sub-groups. When deleting a set of users, noted by L, the group controller is required to communicate with $G \setminus L$ group members to update the group key. One updating message is required, since each user $u \in G \setminus L$ stores the shared key of sub-group $G \setminus L$. On the other hand, if each user has two shared keys, one is the pairwise shared key with the group controller, the other is overall group key. Thus, each user belongs to 1 sub-group, which only contains one group member. To transmit an encrypted message to a sub-group $G \setminus L$, it requires $|G \setminus L|$ messages. From these two examples, which represents two extreme cases, we can see that the communication overhead of membership management can be reduced by increasing the number of sub-groups that each group member can directly participate.

The number of participating sub-groups for each user is determined by their available storagespace. Many existing solutions are designed to balance the storage and communication overhead [1], e.g., OFT [2], LKH [3], and ELK [4]. These solutions utilize a tree structure to organize keys for each user, hence reducing the storage overhead to the order of $O(\log N)$. Based on the predistributed keys, each group member can process or derive $\log N$ sub-group keys. However, these solutions cannot effectively minimize the communication overhead due to the fact that the number of sub-groups each user belongs to is not maximized.

The storage complexity of $O(logN)$ is generally considered as an acceptable overhead for most modern communication devices. Thus, based on this storage condition, the research challenge is how to maximize the number of sub-groups each user belongs to. In this paper, we propose a novel identity-based key management scheme for group communications (ID-GC). In ID-GC, the number of sub-groups each user belongs to is maximized to N, given the storage overhead $\log N$. The construction of ID-GC takes a new approach by using threshold secret sharing scheme [**?**] and attribute-based encryption [5]. A group controller (GC) is in charge of the group key distribution. To form a group, the GC first assigns each group member (GM) a unique n-bit ID. Then, a set of secret shares based on the member's ID is distributed to each GM, i.e., $n = \log N$ private keys corresponding to each bit value in the ID are distributed to each GM. Subgroups are defined in terms of common bits in the ID and all GMs in a sub-group share some common bits in their IDs. Note that, private keys mapped to the same bit in the IDs for GMs are different. ID-GC can effectively, in the complexity of $O(\log N)$, handle the group key distribution for all possible sub-groups given the storage overhead $\log N$ for each group member.

To handle extremely large groups, ID-GC can be decentralized by dividing the communication group into multiple clusters and each cluster is controlled by the Cluster Controller (CC). In summary, the main contributions of ID-GC are presented as follows:

- The communication overhead for revoking a GM or multiple GMs is $O(\log N)$, as contrast to existing schemes, e.g., OFT [2], LKH [3], and ELK [4], which is bounded by $O(L \cdot \log M)$, where L is the number of GMs to be revoked and M is the number of GMs in the group. To the best of our knowledge, this is the most efficient group management scheme proposed so far in the state of art.
- The communication overhead of adding GMs is $\Theta(1)$, i.e., only one broadcast message is required.
- The storage overhead for a GM is $\Theta(logN)$.

Paper Organization The rest of this paper is organized as follows. Section II presents notations models used in this paper. We present detailed ID-GC in Section III. In section IV, we analyze the security of ID-GC. In Section V, we discuss the performance of ID-GC scheme and compared with several existing works. We review the related works in VI. Finally, we conclude our work in Section VII.

II. System Models

Attack Models The attackers' goal is to reveal broadcasted group data encrypted by the data encrypting key (DEK). We assume that an attacker 1) can be a GM or a non-GM (i.e., one to be revoked or one to join a group); 2) can receive and stores all transmitted messages; 3) can be passive eavesdropper or active attacker; 4) can collude with other GMs. We assume that attackers cannot compromise cryptographic algorithms. The GC is assumed to be trustable. In addition, our security analysis will focus on forward secrecy, backward secrecy, and colluding attacks.

Bilinear Pairing: is a bilinear map function $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, where $\mathbb{G}_0$ and $\mathbb{G}_2$ are groups with large prime order p. Pairing has the following properties:

Bilinearity:

$$e(aP, bQ) = e(P,Q)^{ab}, \quad \forall P,Q \in \mathbb{G}_1, \forall a,b \in \mathbb{Z}_p^*.$$

Nondegeneracy: If $e(P,Q) = 1$ for all the $Q \in \mathbb{G}_1$, then $P = \mathcal{O}$. Alternatively, for each $P \neq Q$ there exists $Q \in \mathbb{G}_1$ such that $\widehat{e}(P,Q) \neq 1$.

Computability: There exist a efficient algorithm to compute the pairing.

Secret Sharing and Polynomial Interpolation $t-n$ secret sharing is used to divide a secret in n shares and any t shares can collude to re-construct the secret, while combining less than t shares will not disclose any information about the secret. As introduced by Shamir at el. in [6], in a $t-1$ degree polynomial, any t points on the polynomial be used to reconstruct the secret. We define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$: $\Delta_{i,S(x)} = \prod_{(j \in S, j \neq i)} \frac{x-j}{i-j}$.

Notations the notations used in this paper is listed below:

Symbols	Descriptions
U	the ID space
G	the broadcasting group includes all GMs
J	the set includes all joining GMs
L	the set includes all leaving GMs
u	one GM
N	Size of ID space
n	$n = \log N$
M	number of GMs in G
B	bit-assignment
SB	set of bit-assignment in ID
U	universe of bit-assignment

III. Constructions of ID-GC

A. ID and Bit-assignment

Before presenting ID-GC, we first define the ID and bit-assignment for a GM.

ID: ID-GC requires that each GM is identified by a unique binary string **ID**: $b_0 b_1 ... b_{n-2} b_{n-1}$, where $b \in \{0,1\}$ and $n = \log N$. The ID is issued by the GC when a GM joins the group and the ID should be unique in the group. Once the GM leaves the group, his/her ID can be re-assigned to other joining GMs.

Bit-Assignment: We define the **bit-assignment** B_i^b as the bit-assignment at position i in the ID, where $b \in \{0,1\}$. Intuitively, bit-assignment means the i'th bit of an ID is b. For a group with at most $N = 2^n$ GMs, the total number of bit-assignments is $2n = 2\log N$ when the length of an ID is n; that is, one bit position maps two values (one for value 0 and one for value 1). We denote the set of all the possible bit-assignments as the universe U. Each GM u is uniquely identified by a set of bit-assignments: $SB_u = \{B_0^{b_0}, B_1^{b_1}, \ldots, B_{n1}^{b_{n-1}}\}$, and $|SB_u| = n$, $SB_u \subset U$ is, as illustrated in Figure 1. Also, multiple GMs may share some sub-set of bits (however, their mapped secrets are different).

Mapping Bit-Assignment to Integers: For simplicity of the presentation, we use the following map $m(B_j^b) = j * 2 + b + 1$ to map the universe $U = \{B_i = b | \ \forall j \in \{0, 1, \ldots, n-1\}, b \in 0, 1\}$ to $\{1, 2, \ldots, 2n\}$, i.e., each bit-assignment represents one positive integer less or equal to $2\log N$. For example, in a system with 3 bit ID illustrated in Fig. 1, B_0^0 can be mapped to 1; $B_0 = 1$ can be mapped to 2, $B_2 = 1$ can be mapped to 6, as in Fig. 1. For example, a GM u_1's ID is 000 and a GM u_2's ID is 001, $SB_{u_1} = \{1,3,5\}$ and $SB_{u_2} = \{1,3,6\}$ and $SB_{u_1} \bigcap SB_{u_2} = \{1,3\}$.

In Fig. 1, the GMs can be organized in a tree structure with each non-leaf node marked with a bit-assignment. Note that there are only $2n$ different nodes in the tree and each level have 2 different nodes. This is different from existing tree-based schemes, where there are $2N-2$ different internal nodes and the level d contains 2^d different nodes. The bit assignments for a GM can be represented by links from the root down to the leave. Thus, each GM will have at least one bit-assignment, which is different from other GMs.

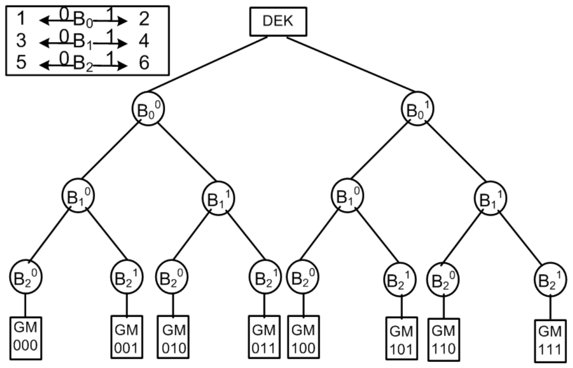

Fig. 1. An illustration of bit-assignments for a 3-bit ID space.

B. Group Setup

In this section, we describe the procedures that GC performs before setting up the broadcast group:

1) chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g.
2) chooses a publicly known one-way function H.
3) chooses two random numbers $\alpha, \beta \in \mathbb{Z}_p$.
4) For each $i \in U$, GC chooses a random number $y_i \in \mathbb{Z}_p$. We denote the set of $2 \log n$ random numbers by Y_B. If we map the $2 \log n$ bit-assignment to $\{1, \ldots, 2 \log N\}$, we can write $Y_B = \{y_1, y_2, \ldots, y_{2 \log N}\}$. In Fig. 2, we depicted a example of distribution of Y_B, supposing ID length is 3.

The group public parameter is published as $GP = \{\mathbb{G}_0, e, g, H\}$. The group master key: $MK = \{\beta, g^\alpha g^\beta, e(g,g)^\alpha\}$ is well guarded by GC.

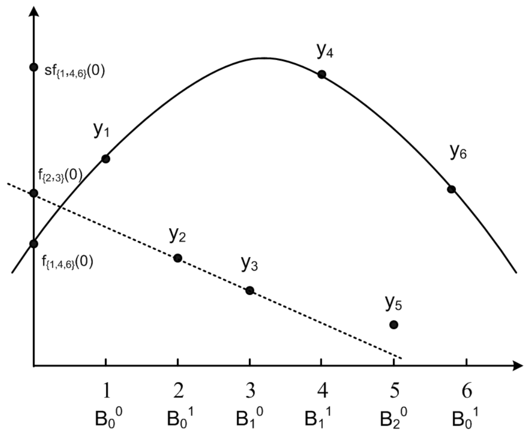

Fig. 2. Random distribution of Y_B; a 2-degree polynomial passing $\{y_1, y_4, y_6\}$; a 1-degree polynomial (doted line) passing $\{y_2, y_3\}$.

Algorithm 1 $KeyGen(MK, SB_u)$

```
Randomly select $r \in \mathbb{Z}_p$;
Compute $g^{\frac{\alpha+r}{\beta}}$;
for each integer $i \in SB_u$ do
    Compute $g^{ry_i}$;
end for
return
$SK_u : \{D = g^{\frac{\alpha+r}{\beta}}, \forall i \in SA_u : D_i = g^{ry_i}\}$;
```

C. Join the Broadcast Group and Key Generation

When GM u joins the group, they need to setup a secure channel with the GC, who checks whether each GM is authorized to join. GC assigns a unique ID $b^u_{n-1}b^u_{n-2}...b^u_0$ and a set of bit assignments $SB_u = \{B_i^{b_i^u} \mid i \in \{0, 1, \ldots, n-1\}\}$ to the new GM. As mentioned above, SB_u is mapped to a set of integers.

To preserve group backward secrecy, i.e., the joining u should not have access to data that were transmitted before he/she joined the group, GC renews the DEK to another random key DEK' and broadcast $\{DEK'\}_{DEK}$.

Then the GC runs $SK_a = KeyGen(MK, SB_u)$ in Algorithm 1, where MK is the group master key and SB_u is the set of bit-assignments in u' ID. The $KeyGen$ algorithm generates a private key components,i.e., blinded secret shares, for each bit-assignment $\in SB_u$. All the private key components are blinded by the same random number r.

Finally, GC sends the private key SK_u and DEK' to the new GM u through a secure channel. In the *join* operation, GC only needs to broadcast one message, i.e., $\{DEK'\}_DEK$, for any number of simultaneously joining GMs.

D. Broadcasting Encryption for a Sub-Group users

In this section, we introduce a novel mechanism that GC can encrypt a message and broadcast it, such that only a sub-group of GMs with certain set of common bit-assignments can decrypt the message. We note that:

- Only GC can perform encryption, and GM can only perform decryption;
- The decryption is based on pre-distributed private keys, which is presented in section III-C.
- the ciphertext can be decrypted by multiple GMs, given that their IDs satisfy the bit-assignments associated with the ciphertext.

First, we describe a broadcasting encryption algorithm $CT = ENC(GP, MK, SB, M)$ in Algorithm 2, which takes inputs of MK - the system master key, GP - the group parameter, SB - a set of bit-assignments, M - the message, and returns the ciphertext CT. Let $d = |SB|$ denote the size of SB, it is easy to find a $d-1$ degree polynomial q that goes through each $y_{B_i^b}$, where $B_i^b \in$

Algorithm 2 $Enc(MK, SB, M)$

```
Let $d = |SB|$, i.e., the size of the SB;
Find the d-1 degree polynomial $f_{SB}$ pass all $\{y_i|\forall i \in SB\}$;
Compute $f_{SB}(0)$;
Randomly select $s \in \mathbb{Z}_p$;
Compute $sf_{SB}(0)$;
Compute $C_0 = Me(g,g)^{\alpha s f_{SB}(0)}$;
Compute $C_1 = g^{\beta s f_{SB}(0)}$;
Compute $C_2 = g^s$;
return
$CT : \{SB, C_0 = Me(g,g)^{\alpha s f_{SB}(0)}, C_1 = g^{\beta s f_{SB}(0)}, C_2 = g^s\}$;
```

SB. For example, suppose $SB = \{1, 4, 6\}$, then we can find a 2 degree polynomial f going through $\{y_1, y_4, y_6\}$ as illustrated in Fig. 2.

On receiving the CT, GMs whose ID satisfied the bit-assignments associated with the ciphertext, can decrypt the CT with their private keys SK using the decryption algorithm $M = DEC(GP, SK, CT)$ presented in Algorithm 3.

Algorithm 3 $DEC(GP, SK, CT)$

```
if $SB_u! \subseteq CT.SB$ then
    return $\perp$;
end if
for each $i \in CT.SB$ do
    Compute $g^{ry_i\Delta_{i,SB}(0)}$;
end for
Compute $\prod_{i\in SB} g^{ry_i\Delta_{i,SB}(0)} = g^{rf_{SB}(0)}$;
Compute $e(C_1, D) = e(g,g)^{(\alpha+r)sf_{SB}(0)} = A_1$
Compute $e(C_2, g^{rf_{SB}(0)}, D) = e(g,g)^{rsf_{SB}(0)} = A_2$
Compute $A_1/A_2 = A_3 = e(g,g)^{\alpha s f_{SB}(0)}$
Compute $C_0/A_3 = M$
return $M$;
```

E. Leave the Multicast Group

We present a key update scheme that updates all the remaining GMs' private keys. Then we present ID-GC *leave* scheme and show how it works with Boolean Function Minimization (BFM) [7].

1) Key Update: For a set of leaving GMs, denoted by L, GC needs to update the $\{MK, DEK\}$ as well as the private keys for each remaining GM $u \in G \setminus L$. However, in our design, these operations can be done efficiently. To perform key updates, GC first changes $MK' = \{\beta, g^{\alpha'}, e(g,g)^{\alpha'}\}$, where α' is randomly selected in $\mathbb{Z}_p$. Then, GC broadcasts an encrypted key updating factor $kuf = g^{\frac{\alpha'-\alpha}{\beta}}$. Note that kuf is encrypted and should NOT be decrypted by any $u \in L$. How to encrypt the key updating factor is presented in the following Sections.

Each $u \in G \setminus L$ updates its private key SK_u based on the key updating factor $g^{\frac{\alpha'-\alpha}{\beta}}$. The original private key is $SK = (D = g^{(\alpha+r)/\beta}; \forall i \in SA_u : D_i = g^{ry_i}$. The update factor only affects D. The new D can be updated by the following method:

$$D \cdot g^{\frac{\alpha'-\alpha}{\beta}} = g^{\frac{\alpha+r}{\beta}} \cdot g^{\frac{\alpha'-\alpha}{\beta}} = g^{\frac{\alpha+r+\alpha'-\alpha}{\beta}} = g^{\frac{\alpha'+r}{\beta}}$$

In this way, each $u \in G \setminus L$ update their DEK K' simply by compute $K' = H(g^{\frac{\alpha'-\alpha}{\beta}})$.

2) Single Leave: We first consider that only one GM leaves the group. The key updating factor is encrypted with the bit-assignments that are complementary to the ones of the departing member. We assume that the leaving GM u's ID is $b^u_{n-1}b^u_{n-2}...b^u_1b^u_0$. We can define the complementary set of u to be those GMs who have at least one different bit assignments with u. GC can encrypt the key updating factor with each one $SB \in \{\{i\}|\forall i \in U \setminus SB_u\}$ and broadcast $\log N$ encrypted key update factors. If GC knows all the IDs that are not currently assigned, the number of encrypted key update factors may be reduced.

For example, we assume that the departing GM u's ID is 101. Therefore, it possesses a set of bit-assignments $SB_u = \{2, 3, 6\}$. The key updating message is encrypted with each one of the $\{1\}$ $\{4\}$ $\{5\}$, i.e., $\{kuf\}_{\{1\}}$, $\{kuf\}_{\{4\}}$, $\{kuf\}_{\{5\}}$ and is broadcasted to the entire group. If ID 100 is not assigned $\{kuf\}_{\{1\}}$ is not necessary. Although the departing member receives all the messages, it cannot decrypt them, since every message is encrypted with a bit-assignment that the departing member does not possess. It also guarantees that all other GMs of the group can decrypt at least one message.

3) Multiple Leave: In this section, we focus on how GC securely broadcasts the encrypted key update message to sets of remaining GMs when multiple GMs leave the group. First, we define some of the terms used in the following presentations.

- *Literal*: A variable or its complement, e.g., B_1, $\overline{B_1}$, B_2, $\overline{B_2}$, etc.
- *Product Term*: Series of literals related by AND gates, e.g., $\overline{B_2}B_1\overline{B_0}$.
- *Sum Term*: Series of literals related by OR gates, e.g., $B_2 + \overline{B_1} + B_0$.
- *Sum-of-Product Expression (SOPE)*: Series of product terms related by OR gates, e.g., $\overline{B_2}B_1B_0 + B_{2\overline{1}}$.

We define the boolean membership functions $F(B_0, B_1, \ldots, B_{n-2}, B_{n-1})$, which is in the form of SOPE and each of the $n = \log N$ variables is a bit-assignment. For example, if the set of leaving GMs $L = \{001, 010, 101\}$ and $G \setminus L = \{011, 111\}$, the $F(B_0, B_1, B_2) = \overline{B_0}B_1B_2 + B_0B_1B_2$. The following properties of membership functions hold:

$$F(b^u_0, b^u_1, \ldots, b^u_{n-2}, b^u_{n-1}) \quad = \quad \begin{cases} 0 & \text{iff } \ u \in C_{G/L} \\ 1 & \text{iff } \ u \in G/L \end{cases}$$

The GC runs the Quine-McCluskey algorithm [7] to reduce F to minimal SOPE $F_{min} = E_0 + \ldots + E_L$,

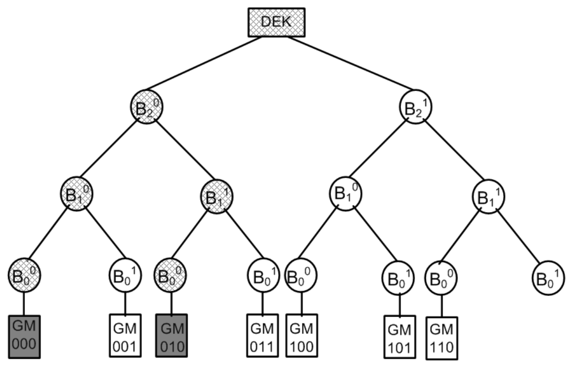

Fig. 3. Leave of Multiple GMs.

in which the term E uses product term. In each of the computations, we can allow *do not care* values on the U/G, which further reduces the size of F_{min}. For example, in Fig. 3, $\{000, 010\}$ are leaving GMs, and $\{001, 011, 100, 101, 110\}$ are remaining GMs, and $\{111\}$ is not assigned (i.e., *do not care*). With the considerations of *do not care* values, F can be reduced to $F(B_0, B_1, B_2)_{min} = X_0 + X_2$. We can find that F_{min} contains 2 literals and 2 product term.

Since F_{min} is in the form of SOPE, GC can encrypt the key updating factor for each product term and the total number of encrypted key updating factor equals to the number of product terms in F_{min}. For each product term, a SB contains all the bit-assignments in the product term is used to encrypt the key updating factor. For example, following form $F(X_0, X_1, X_2)_{min} = \overline{X}_0 X_1 X_2 + X_0 \overline{X}_1$, GC can broadcast $kuf_{\{0,4,6\}}$ and $kuf_{\{2,3\}}$.

F. Clustering for Scalability

In the previous sections, we introduce the basic construction of ID-GC, where the size of max group members is limited by the size of ID space. Also, since the problem of boolean function minimization is NP-hard, the number of bits in ID cannot be very large. If the ID space is restricted by 10 bits, the size of the overall group is limited to 1024. To accommodate small ID spaces, we can divide the broadcasting group into several clusters and each cluster has a cluster controller (CC). The dynamic membership of CC is managed by GC; GC and CCs will setup a shared key K_1, which is only known to GC and CCs. On the other hand, each cluster is managed by its CC and dynamic membership is handled locally. Beside the DEK shared by the entire group, GMs and CC in the same cluster will share a cluster key K_c. In Fig. 4, we depict the clustering of ID-GC and the allocation of shared keys.

Add/Delete GMs: When a GM joins the group, it will be assigned to a cluster and its CC will perform the group member addition operations described in III-C to issue the GM a unique ID and private keys, and update the DEK. Note that ID-GC requires to have a unique cluster ID in addition to the GM ID. In this way, the GMs' IDs are local effective and GMs in different clusters may possess same GM IDs. The deletion of GMs is identical to the section III-E, except that the new DEK needs to be updated by each CC.

Add/Delete Clusters: Dynamically adding and deleting a cluster is performed identical to the addition and deletion of GMs in previous section. When adding a cluster, a cluster controller (CC) is selected and assigned a unique cluster ID, a cluster private key and the K_1 shared with the GC. When deleting a cluster, GC communicate will all remaining CCs to update their private keys and the DEK for all group members. Note that the deletion of CC means all the GMs in the cluster is excluded from the broadcasting group.

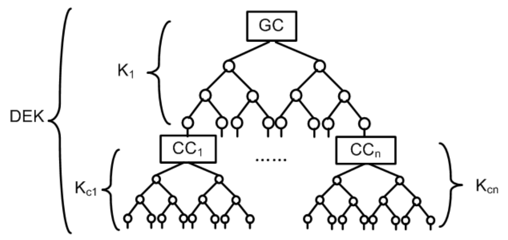

Fig. 4. Illustration of clustering

IV. Security Analysis of ID-GC

ID-GC scheme provides the following security properties:

Proposition 1(Backward Secrecy): ID-GC provides group backward secrecy.

When new GMs *join* the group, a new random DEK K' is encrypted ($\{K'\}_K$), and then it is distributed through broadcasting. Also, suppose the private keys of joining GMs are generated under α'. All the previous key updating messages are encrypted using $e(g, g)^{\alpha}\}$, i.e. α' is another random number other than α. Given 1) randomness of K' and α'; 2) security of symmetric encryption and Discrete Logarithm (DL) [8] problem on $\mathbb{G}_0$ and $\mathbb{G}_1$, the new joining group member cannot derive previous DEK. Thus, the group backward secrecy is satisfied.

Proposition 2:(Forward Secrecy) ID-GC provides group forward secrecy.

When GMs *leave* the group, the GC updates the system parameters to MK' using a new random α' and broadcast the encrypted $g^{\frac{\alpha'-\alpha}{\beta}}$ to all the GMs in $G \backslash L$. All remaining GMs will update their private keys and DEK using the key updating factor $g^{\frac{\alpha'-\alpha}{\beta}}$. The leaving GMs cannot decrypt future encrypted messages since DEK has been changed. Even if a leaving GM stores all encrypted key-updating messages and *join* the group again, he or she cannot decrypt any previous key updating message, since these messages are encrypted under different master keys, which is unknown to the GM. Moreover, using the

TABLE I. Comparison of communication overhead and computation overhead in different group key management schemes.

Scheme	Communication Overhead			Computation Overhead	
	join	single leave	multiple leave	encryption	decryption
ID-GC	$\Theta(1)$	$O(\log N)$	$\approx O(\log N)$	$O(\log N)$	$O(\log N)$
FT	$\Theta(\log N)$	$O(\log N)$	$\approx O(\log N)$	–	–
Subset-Diff	–	$O(t \cdot log^2(t) \cdot \log m)$	$O(t \cdot log^2(t) \cdot \log M)$	–	–
BGW	–	$O(M^{\frac{1}{2}})$	$O(M^{\frac{1}{2}})$	$O(M)$	$O(M)$
ACP	$\Theta(M)$	$\Theta(M)$	$\Theta(M)$	$\Theta(M^2)$	$\Theta(1)$
Tree	$\Theta(1)$	$\Theta(\log M)$	$O(L \cdot \log M)$	–	–

N: the ID space; M: the number of group members; L: the number of leaving members; t: maximum number of colluding users to compromise the ciphertext.

key updating factor $g^{\frac{\alpha'-\alpha}{\beta}}$ to derive g^{α} and β is hard due to the DL problem.

Proposition 3:(Collusion Resistance) Leaving GMs cannot collude to decrypt broadcasted messages targeted to other GMs.

We refer to the collusion attack as any combinations of GMs attempting to derive other GM's private key by combining their private keys. We first show that any two GMs cannot collude using their private keys. Given the private keys of two attackers a_1 and a_2, $SK_{a_1} = \{D = g^{\frac{\alpha+r_{a_1}}{\beta}}, \forall i \in SA_{a_1} : D_A = g^{r_{a_1} y_i}\}$, $SK_{a_2} = \{D = g^{\frac{\alpha+r_{a_2}}{\beta}}, \forall j \in SA_{a_2} : D_A = g^{r_{a_2} y_j}\}$, the problem of deriving $SK_v = \{D = g^{\frac{\alpha+r_v}{\beta}}, \forall k \in SA_v : D_A = g^{r_v y_k}\}$, where $SA_v \subseteq SA_{a_1} \bigcup SA_{a_2}$ and $SA_v \neq SA_{a_1}$ $SA_v \neq SA_{a_2}$, can be reduced to the DL problem on $\mathbb{G}_0$. Furthermore, adding more colluding attackers will not help due to the hardness of DL problem.

V. Performance Assessments

We analyze the performance of ID-GC scheme and compare it with several previous solutions: flat table scheme (FT) [9], [10], subset-difference broadcast encryption scheme (Subset-Diff) [11], BGW broadcasting encryption [12], access control polynomial (ACP) scheme [13], and tree-based schemes (e.g., OFT [2], LKH [3], and ELK [4]). The performance assessments are assessed in terms of storage overhead (group data to be stored on the GC and GM), communication overhead (number of messages to be broadcasted for join and leave operations), and computation overhead (number of cryptographic operations needed in encryption and decryption operations). We denote the group size be N, the number of current GMs in each cluster to be M, the number of clusters to be C, the number of leaving GMs to be L. Also, for the Subset-Diff scheme, t denotes the maximum number of colluding users to compromise the ciphertext. The summary of comparative results is presented in Table I.

A. Storage Overhead

In ID-GC, the storage overhead for GC is $\Theta(C)$ (GC stores the IDs of all current CCs) and for CC is $\Theta(M)$ (CC stores the IDs of all GMs in its cluster). The storage overhead is $\Theta(\log N)$ for GC, since GM stores a private key component for each bit in its ID.

B. Communication Overhead

The communication overhead for the dynamic membership is the most important evaluating factor. In ID-GC, the communication overhead is $\Theta(1)$ for *join* operation since only $\{DEK'\}_{DEK}$ is broadcasted when new GMs join the group.

Here, our discussion focuses on the complexity of *leave* operation. In Subset-Diff scheme, the communication overhead grows linearly with the maximum number of colluding users to compromise the ciphertext. For BGW scheme, the message size is $O(M^{\frac{1}{2}})$ as reported in [12]. In ACP scheme, the size of message depends on the degree of access control polynomial, which equals to the number of current GMs plus the number of joining GMs or the number of current GMs minus the number leaving GMs. Thus, the message size is $\Theta(M)$. For tree-based schemes, the communication overhead for a GM leaving depends on the number of keys in the tree that need to be updated [14], [4]. Some tree-based schemes tried to optimize the number of messages to update all the affected keys in the case of multiple leaves. In ELK [4], which is known to be one of the most efficient tree-based schemes, the communication overhead for multiple leaves is $\Theta(a - L)$, where a is the number of affected keys and L is the number of leaving GMs. Since there are $\log M$ (or $\log N$ in the case of full tree) nodes on the path from root to leaf in the tree structure, the total number of affected keys when L GMs leave the group is $O(L \cdot \log M)$.

We first show that, in the worst cases, ID-GC outperforms all the tree-based schemes. Then, we analyze the average case. Since ID-GC shares some properties with the Flat Table scheme, we utilize some of the performance results from [9].

Lemma 1 (Worst case of removal of 2 GMs [9]): When removing 2 GMs from a group with $N = 2^n$ GMs, the number of key updating messages is at most n. The worst case is achieved when the Hamming distance between 2 GMs is n.
In this case, the number of keys to be updated is $2n-1$, thus ELK requires $2n-3$ messages while ID-GC requires n messages.

Lemma 2 (worst case of removal multiple GMs [9]): The worst case of removal multiple GMs happens when

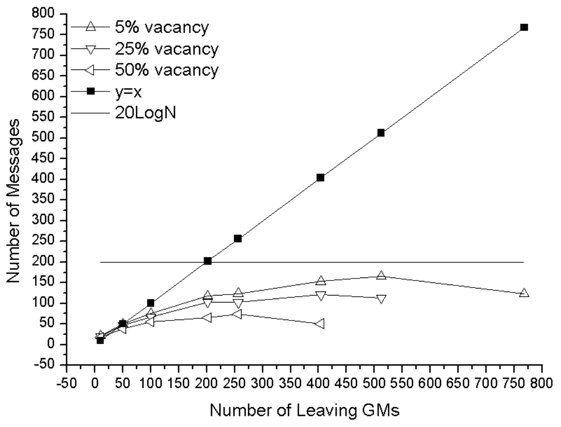

Fig. 5. Simulation of multiple GMs leave for a group with 1024 GMs

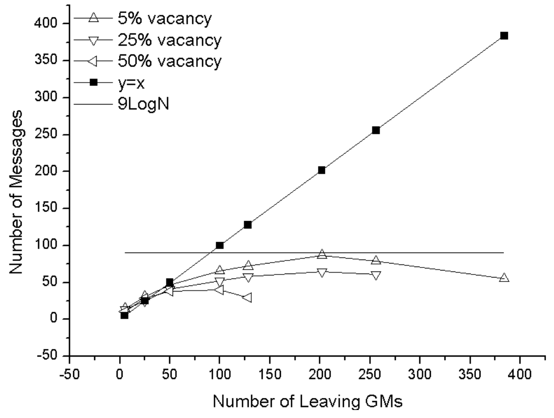

Fig. 6. Simulation of multiple GMs leave for a group with 512 GMs

both of following conditions hold: 1) there are $N/2$ GMs to be removed; 2) the Hamming distance between IDs of any two remaining GMs is at least 2. In the worst case, the number of key updating message is $N/2$.

In this case, the number of keys to be updated is $N - N/2 = N/2$, since there are N non-leaf keys to be updated and the number of leaving GMs is $N/2$. We can see that, in the worst case, ID-GC's performance is same as ELK approach.

Now, we move on to show are in the average cases, ID-GC greatly out-performs all existing solutions. Using ID-GC, the number of messages for leaving operations depends on the number of product terms in the SOPE. In [15], the authors derived an upper bound and lower bound on the average number of products to minimize SOPE, which can be directly used to model the average performance of ID-GC. We first show an average case example in Fig. 3, where $\{000, 010\}$ are leaving GMs, and $\{001, 011, 100, 101, 110\}$ are remaining GMs, and $\{111\}$ is not assigned (i.e., *do not care*). In this example, ID-GC requires 2 messages while tree-based schemes needs at least 3 messages. Then, we derive an approximate bound of the average case using simulation. In the simulation, leaving GMs are randomly selected from a group with 1024 GMs and a group with 512 GMs, where the simulation results are shown in Fig. 5 and Fig. 6, respectively. In each case, we consider 5%, 25%, 50% IDs are not assigned (i.e., *do not care* value). We repeat 100 times for each sub-group construction to average the results. As a comparison, we plotted one line for the complexity of number of messages equals number of leaving GMs and one line for $O(\log N)$. From the result, we can see that ID-GC performs better than all the existing tree-based group key management schemes and achieves $O(\log N)$ complexity, where the constant factor is about 20 for the 1024-member group and 9 for the 512-member group.

C. Computation Overhead

There are two main factors to affect the computation performance. First, the asymmetric cryptographic operations are considered to be more expensive than symmetric-key-based solutions. Second, bigger group may require more computation powers. In ACP scheme, the author reports that the encryption needs $O(M^2)$ finite field operations when the sub-group size if M; in the BGW scheme, the encryption and decryption require $O(M)$ operations on the bilinear group, e.g., a group of Elliptic curve points, which is heavier than finite field operations [16], [17]. In ID-GC, the encryption produce $O(\log N)$ messages and each requires 3 operations on the bilinear groups, and the decryption requires $O(\log N)$ operations on the bilinear groups and 2 pairings. Although the problem of minimizing SOPE is NP-hard, efficient approximations are widely known. Thus, ID-GC is much more efficient than ACP and BGW when the size of group goes larger.

VI. Related Work

Group key management (GKM) has been investigated intensively in centralized group key distribution schemes and decentralized (contributory) key agreement schemes. Due to the richness of related research, we cannot list all the related work in this area. We refer to [18], [19] as two excellent surveys.

Particularly, the construction of our key management scheme is similar to the following described work. Tree-based rekey algorithms have gained popularity, including, notably, Logical Key Hierarchy (LKH) [3], One- Way Function Tree (OFT) [2], One-way Function Chain Tree [20], Hierarchical α-ary Tree with Clustering [1] and Efficient Large-Group Key (ELK) [4]. These algorithms provide different tradeoffs among storage, computation and communication overheads. Compared to these schemes, flat table (FT) scheme achieves high efficiency in terms of storage, computation and communication overheads. In FT schemes [9], [10], there are total $2 \log N$ KEKs:

$\{k_{i,b} | i \in \{0, 1, \ldots, 2\log N - 1\}, b \in \{1, 0\}\}$ and $\log N$ symmetric KEKs are distributed to each GM, with each KEK corresponding to one bit in ID. Despite its efficiency, FT scheme is vulnerable to collusion attacks. In [11], the authors first formally explored the broadcasting encryption. They presented a solution for M users, which is secure against collusion attacks of t users. The communication overhead of this scheme is $O(t \log^2 t \log M)$. In [12], Boneh et al. proposed a collusion-resistant broadcasting encryption scheme. In this approach, the storage, communication and computation overhead grows linearly with the increase of the number of users.

The strategy of ID-GC reducing communication overhead for GM leaving is similar to that of the Flat Table (FT) scheme, which is very efficient but vulnerable to collusion attacks. Flat Table (FT) key management schemes [9], [10] map the set of pre-distributed secret keys for each GM to the bit positions in the GM's ID, in order to reduce communication overheads. However, FT solutions simply adopt the shared key solutions and thus are subject to collusion attacks. For example, GMs 001 and 010 can decrypt ciphertexts destined to other GMs, e.g., 011, 000, by combining their secret keys that are mapped to their bit positions. To prevent the collusion attacks, Cheung et al. [21] proposed CP-ABE-FT to implement the FT using CP-ABE. CP-ABE-FT utilizes a periodic refreshment mechanism to ensure forward secrecy. However, it has several drawbacks: 1) if the ID of a revoked GM is re-assigned to another GM before the refreshment, the revoked GM can regain the access to group data and then the group forward secrecy is compromised; 2) outsiders can impersonate GC to disturb the rekey process by sending CP-ABE [5] ciphertexts. However, the drawbacks of [21] is that the size of message is too large and communication overhead is $\log N \log M$. In our construction, ID-GC is resistant to collusion attacks and communication overhead is further reduced to $\log N$.

VII. Conclusion and Future Work

In this paper, we proposed a novel ID-GC scheme. By utilizing the basic construction of ciphertext policy attribute-based encryption and flat table identity management, ID-GC greatly improved performance in communication ($O(\log N)$ for bulk deletions), storage ($O(\log N)$ for each group member), and computation ($O((\log N)(\log M))$ for both encryption and decryption), where N is the ID space size and M is the number of GMs in the group. Moreover, ID-GC scheme provides group forward/backward secrecy, and it is resilience to colluding attacks.

Based on the proposed solution, future work would focus on extending ID-GC scheme to support dynamic conference without losing efficiency. Particularly, we will investigate distributed group management infrastructure to improve the robustness of the ID-GC.

References

[1] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption, Advances in Cryptology-Eurocrypt99," *Lecture Notes in Computer Science*, vol. 1592, pp. 459–474, 1999.

[2] A. Sherman and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pp. 444–458, 2003.

[3] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *Networking, IEEE/ACM Transactions on*, vol. 8, no. 1, pp. 16–30, 2000.

[4] A. Perrig, D. Song, and J. Tygar, "ELK, A New Protocol for Efficient Large-Group Key Distribution," *IEEE SYMPOSIUM ON SECURITY AND PRIVACY*, pp. 247–262, 2001.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, 2007.

[6] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[7] E. McCluskey, "Minimization of Boolean functions," *Bell System Technical Journal*, vol. 35, no. 5, pp. 1417–1444, 1956.

[8] A. Menezes, *Handbook of Applied Cryptography*. CRC Press, 1997.

[9] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, I. Center, and Y. Heights, "Key management for secure lnternet multicast using Boolean functionminimization techniques," *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 1999.

[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*, vol. 17, no. 9, 1999.

[11] A. Fiat and M. Naor, "Broadcast Encryption, Advances in Cryptology-Crypto93," *Lecture Notes in Computer Science*, vol. 773, pp. 480–491, 1994.

[12] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," pp. 573–592, 2006.

[13] X. Zou, Y. Dai, and E. Bertino, "A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 538–546, 2008.

[14] J. Snoeyink, S. Suri, and G. Varghese, "A lower bound for multicast key distribution," *Computer Networks*, vol. 47, no. 3, pp. 429–441, 2005.

[15] T. Sasao, "Bounds on the average number of products in the minimum sum-of-products expressions for multiple-value input two-valued output functions," *Computers, IEEE Transactions on*, vol. 40, no. 5, pp. 645–651, May 1991.

[16] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[17] A. Ramachandran, Z. Zhou, and D. Huang, "Computing Cryptographic Algorithms in Portable and Embedded Devices," *Portable Information Devices, 2007. PORTABLE07. IEEE International Conference on*, vol. 25-29, pp. 1–7, 2007.

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

[19] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 309–329, 2003.

[20] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, I. Center, and Y. Heights, "Multicast security: a taxonomy and some efficient constructions," *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 1999.

[21] L. Cheung, J. Cooley, R. Khazan, and C. Newport, "Collusion-Resistant Group Key Management Using Attribute-Based Encryption," Cryptology ePrint Archive Report 2007/161, 2007. http://eprint. iacr. org, Tech. Rep.