Attribute-Based On-Demand Multicast Group Setup with Membership Anonymity

Shucheng Yu
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609-2280
yscheng@wpi.edu

Kui Ren
Illinois Institute of Technology
3300 S. Federal Street
Chicago, Illinois 60616
kren@ece.iit.edu

Wenjing Lou
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609-2280
wjlou@wpi.edu

ABSTRACT

In many applications, it is desired to dynamically establish temporary multicast groups for secure message delivery. It is also often the case that the group membership information itself is sensitive and needs to be well protected. However, existing solutions either fail to address the issue of membership anonymity or do not scale well for dynamically established groups. In this paper, we propose a highly scalable solution for dynamical multicast group setup and yet protecting group membership anonymity simultaneously. In the proposed solution, scalability and membership anonymity are achieved via a novel design that integrates both ciphertext-policy attribute-based encryption (CP-ABE) and centralized flat table (CFT) techniques. In our design, multicast groups are specified through group member attributes represented through binary member ID only and thus achieves scalability. Also, high level of membership anonymity is guaranteed such that every group member knows nothing but his own group membership only. The proposed solution is also efficient in communication, that is, the ciphertext size is only O(n), where n is the length of a group member ID and independent to the group size.

1. INTRODUCTION

Multicast is a very important communication function that allows information to be delivered to a group of destinations simultaneously and efficiently. Multicast Backbone (MBone) was built over a decade ago as an experimental backbone for IP Multicast traffic across the Internet. Multicast also found its many wide deployed applications in enterprise networks, such as pay-per-view TV, online game, file distribution, stock information distribution, and so on.

Confidentiality of the information transmitted in a multicast session is a fundamental security service to multicast communication. To encrypt a multicast session, a new group key must be encrypted by some key encryption keys (KEKs) and then distributed to all group members. Many approaches for secure multicast communication have been proposed in recent years, e.g., [15, 9], in which group establishment is usually realized via broadcast encryption. These schemes mostly rely on a central key server to distribute the new group keys and to perform rekeying operations. The central key server is responsible to find the minimal set of KEKs that cover all the group members but are not known to any non-members, and then distribute the new group key encrypted by those KEKs to every group member. When we are dealing with dynamically established groups for large scale networks, even with a very efficient rekeying coding scheme such as SD [11], the number of necessary KEKs (thus the number of encryptions of the group key) may be large and this represents a big communication, computation and storage overhead while keys are distributed in the network.

Privacy is another concern for many multicast applications. In many situations, membership information of the group should be protected. The system may not want anyone, including the intended recipients, to know which other or totally how many members are involved in a task. Ideally, high level of membership anonymity should be achieved even under powerful attacks. Unfortunately, current constructions of broadcast encryption schemes are seldom designed with membership anonymity in mind with the only exception of [1]. However, [1] is not suitable for large-scale networks as the size of ciphertext as well as the computation load grow linearly with the number of users. It is desirable to introduce an efficient scheme which can provide a high level of membership anonymity. In addition, user revocability is often required in most multicast applications.

In this paper, we propose a more efficient group key management and distribution scheme for dynamically formed multicast groups. We observed that in many cases, a dynamically formed multicast group is not simply an arbitrary collection of unrelated nodes. Instead, they are members with certain common attributes. For example, in a battlefield scenario, a commander may need to send secure messages to a group formed by "*Spanish interpreter with rank higher than second lieutenant*". In this case, we may apply a novel cryptographic primitive called ciphertext-policy attribute-based encryption (CP-ABE) [2] such that we encrypt the group key under certain attributes and only those who own the intended attributes are able to decrypt it. To provide a high level of membership anonymity, we enhance the current CP-ABE construction and design a new algorithm. Membership information such as "who is in the group" and "how many members are in the group" are well protected. Unlike [1], our scheme works well in large-scale applications because the ciphertext size is linear to the size of user ID and independent

SecureComm 2008 September 22 - 25, 2008, Istanbul, Turkey
Copyright 2008 ACM 978-1-60558-241-2 ...$5.00.

of the number of users. In addition, single user revocability is possible in our scheme.

1.1 Our Contribution

Main contributions of this paper are as follows:(1) To the best of our knowledge, this is the first work on on-demand multicast group setup with membership anonymity which is guaranteed even when the system is under powerful attacks, e.g. collusion attacks; (2) We introduce the concept of level of membership anonymity in multicast; (3) Our scheme is highly scalable. In our scheme, both the computation complexity and communication complexity are $O(n)$, where n is the length of the GM's ID.

1.2 Related Work

Related work includes two branches: broadcast encryption and attribute-based encyption (ABE). This section provides a brief review of both with respect to the aforementioned requirements.

Broadcast Encryption was first formally explored by Fiat and Naor[8] in 1993. The main effort of previous work on broadcast encryption thereafter, e.g.,[3], is on efficient broadcast and collusion resistance. Receiver anonymity had not been addressed until a private broadcast encryption scheme was proposed by Barth et al.[1] in 2006. In this scheme, a random symmetric key K is generated to encrypt the message M, which is the group key if applied to multicast group setup. K is encrypted once with each receiver's public key. The sender attaches its signature on M to the ciphertext to prevent chosen-ciphertext attack. Each receiver deduces M by decrypting his part of ciphertext using his private key. This scheme can protect receivers' identities effectively. However, the length of ciphertext is linear to the number of receivers. Moreover, this scheme does not protect the privacy information of "how many GMs are in the group". Eavesdroppers can easily deduce the number of receivers from the length of the ciphertext. Although the authors claim that this information can be hidden by padding the recipient set to a given size using dummy recipients, we argue that this will make the scheme inefficient and unscalable.

Ciphertext Policy Attribute-Based Encryption (CP-ABE) was first proposed by Bethencourt, Sahai and Waters [2]. In CP-ABE, each user is associated with a set of attributes. On each attribute, the user is assigned a secret key. When encrypting a message, the encrypter generates an access tree specifying the threshold access structure for his interested attributes and sends it in plaintext. Message is then encrypted based on this access tree such that only those whose attributes satisfy the access structure can decrypt it. CP-ABE makes per message access control possible. Communication and computation complexity is just linear to the number of attributes and independent of the number of recipients. Secrecy of the message is protected even under collusion attacks. However, current schemes of CP-ABE [2, 7] are not designed with membership anonymity in mind. Eavesdroppers can easily derive who is the intended receiver from the access tree which is send in plaintext.

Based on CP-ABE, Cheung et al.[6] proposed a collusion resistant group key management scheme. In this paper, the authors defined each bit of user's ID as an attribute. Match between ciphertext and user's attributes is actually that between user's ID and the sender's intended ID. Our design shares the similar idea on attribute definition in this paper. One significant difference between the two is that the access tree is no longer transmitted in our design. A high level of anonymity is therefore achieved. Moreover, unlike their scheme, the ciphertext size in our design is well controlled as we will discuss in the following sections.

The rest of this paper is organized as follows. Section 2 discusses models and assumptions. Section 3 gives some technique preliminaries. Section 4 presents our solution. Section 5 analyzes our solution in terms of security and performance. We conclude this paper in Section 6.

2. MODELS AND GOALS

In this section, we first discuss models and assumptions used in our design. We then define several levels of anonymity that can be used to evaluate the strength of anonymity protection. Finally, we clearly present our security goals.

2.1 Models and Assumptions

Network Model In this paper, we assume traffic is transmitted via open channels. Therefore, eavesdropping is possible. Also, computation power on each network node is rich, e.g., we can assume each network node has at least the same computation power than a modern PC.

Trust Model In our design, participants are one Group Controller (GC) and many Group Members (GMs). GC is a trusted party. It holds each GM's private key and the system master secret key as we will discuss later. GMs are not trustworthy both to GC and between themselves. Therefore, any GM is not expected to know others' membership information as well as the size of the group.

Adversary Model The adversary could be any party except for GC. He has the ability to control t out of n GMs, where $t \ll n$. However, he has no way to compromise or spoof GC. The adversary's main goal of anonymity violation is to learn the information such as "*who is in the group*" and "*how many GMs are in the group*". Such an attack could be executed either by a single adversary or by a group of cooperative adversaries.

2.2 Level of Anonymity

Pfitzmann [13] defined anonymity by the following statement: "*Anonymity is the state of being not identifiable within a set of subjects, the anonymity set*". This general definition gives the abstract semantic for the term *anonymity.* In practice, however, anonymity may have its concrete semantic. To define the concrete semantic of anonymity in practice, we believe two questions should be answered: "*what subject is not identifiable?*" and "*to whom it is not identifiable?*". The answer to the first question actually defines the subject of *anonymity.* The answer to the other one reflects who is the potential adversary. To measure the strength of anonymity protection, the following question also needs to be answered: "*What type of attacks the adversary may execute?*"

In our interested applications, we believe two kinds of subjects should be unidentifiable: "*the identities of the intended GMs*", and "*the number of the intended GMs*". They should be unidentifiable to *external eavesdroppers*, *unintended GMs*, and *intended GMs.* We distinguish two kinds of attacks: *attacks executed by single adversary* and *attacks executed by colluding adversaries.* Based on these considerations, we define three coarse-grained levels of anonymity:

Level 0 No anonymity. Membership information is transmitted in plaintext and identifiable by anyone.

Level 1 Identities of the group members are protected from the adversaries, including both outsider attackers and colluding intended GMs. An intended GM only knows his/her own membership status but not other intended GMs'.

Level 2 In addition to the requirement to level 1 anonymity, the number of the group members is also protected from the adversaries, including both outsider attackers and colluding intended GMs. GC is the only entity that is aware of the size of the group.

According to this definition, we can see that most of current work is on Level 0 and [1] is on Level 1.

2.3 Security Goals

Our main security goals are as follows:

- *Confidentiality of the group key.* The group key can only be decrypted by intended GMs. Unintended recipients should not have the ability to derive the group key even if they collude.
- *High level of anonymity* The system should achieve the anonymity level 2.
- *Revocability* The system should have the ability to revoke any single GM or a group of GMs sharing common attributes.
- *Backward secrecy* A new GM should not have the ability to access data that were transmitted before he joined.

3. PRELIMINARIES

This section briefly discusses preliminary techniques used by our design.

3.1 Centralized Flat Table

Waldvogel et al.[14] first introduced a flat table (FT) for group key management. This scheme assumes that any potential GM has a n-bit ID, denoted by $X_{n-1}X_{n-2}...X_0$, where $X_i = 0|1$, $i \in \mathbb{Z}_n$. For each bit, GC generates two $KEKs$ corresponding to bit value *0* and *1* respectively. Any GM holds one of the two $KEKs$ for each bit of his ID. KEK for i^{th} bit can be denoted by $KEK_{i,b}$, where $0 \leq i \leq n-1$, $b = 0|1$. For example, GM with ID 1010 holds $KEK_{3,1}$, $KEK_{2,0}$, $KEK_{1,1}$, and $KEK_{0,0}$. Besides the $KEKs$, GC and each GM also hold the group key TEK. We show how CFT deals with member leave events as follows. For space limitation, member join events are not discussed here.

Member Leave Assume $X_{n-1}X_{n-2}...X_0$ is the leaving GM's ID. For forward secrecy, GC needs to update TEK and KEK_{i,X_i} for each $i \in \mathbb{Z}_n$. GC first updates TEK by encrypting the new key TEK' once under each KEK of ID $\bar{X}_{n-1}\bar{X}_{n-2}...\bar{X}_0$. Because any other GM has at least one bit different from the leaving GM, he can decrypt the new TEK for sure. The leaving GM can not decrypt the new TEK because he does not share any KEK with ID $\bar{X}_{n-1}\bar{X}_{n-2}...\bar{X}_0$. GC then updates KEK_{i,X_i} for each $i \in \mathbb{Z}_n$ by encrypting the new key KEK'_{i,X_i} first under the new group key TEK' , then under its corresponding old key, i.e., $\{KEK'_{i,X_i}\}_{TEK',KEK_{i,X_i}}$. The leaving GM does not hold the new TEK and can not decrypt the KEK's.

Multiple Leaves Although multiple leaves can be achieved by repeating the above member leave process, it is not efficient in case of a large number of leaving GMs. A group from IBM [5] proposed an efficient solution for multiple leaves using Boolean function minimization (BFM) techniques. This scheme takes the n bits of ID $X_{n-1}X_{n-2}...X_0$ as input of a Boolean function f. Member leaving event is denoted by the output of f. If GM with ID $X_{n-1}X_{n-2}...X_0$ is leaving, $f(X_{n-1}X_{n-2}...X_0) = 0$, otherwise, $f(X_{n-1}X_{n-2}...X_0) = 1$. In case of multiple leaves, GC can reduce f to a sum of products expression (SOPE) using the Quine-McCluskey algorithm [4]. By this means, GC can achieve "(1) the smallest possible number of ORs, and (2) given the number of ORs, the smallest possible number of literals." [6].

3.2 CP-ABE Algorithm

A typical CP-ABE scheme consists of four algorithms:

Setup This algorithm generates a master key MK and the public parameter PK.

Encrypt This algorithm takes as input the public parameter PK, a message M, and an access structure T. It outputs the cihphertext CT which has the following format:

$$CT = (T, \tilde{C}, C, \forall y \in Y : C_y),$$

where $\tilde{C} = M \cdot X$ and X is a blind factor used to hide M. Y is the intended attribute set. C and all the C_y are credentials for helping decryptors reconstruct X and thus derive M.

KeyGen This algorithm takes as input a set of attributes S associated with the user and outputs a secret key SK that identifies with S.

Decrypt This algorithm takes as input the ciphertext CT and a secret key SK for an attributes set S. If only S satisfies the access structure T, does it return the message M.

Note that, the access structure T is transmitted to the recipients in plaintext. We refer to [2, 7] for more details on CP-ABE.

3.3 Bilinear Maps

Our design is based on some facts about groups with efficiently computable bilinear maps.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$. A bilinear map is is an injective function $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ with the following properties:

1. *Bilinearity*: for $\forall$ $u, v \in \mathbb{G}_0$ and a, b $\in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. *Non-degeneracy*: $e(g, g) \neq 1$.
3. *Computability*: There is an efficient algorithm to compute $e(u, v)$ for $\forall$ $u, v \in \mathbb{G}_0$.

4. OUR SCHEME

Our scheme is composed of four algorithms: *Setup*, *KeyGen*, *Encrypt*, and *Decrypt*. The functionality of each algorithm is similar with that of the CP-ABE algorithm. The main differences are the following: First, our scheme is designed in the way that the encryptor does not need to transmit the access tree T. The recipients' identities are thus protected. Second, our scheme defines attributes on each GM's identity and thus makes single user revocability possible. Third, our current design is not a public-key solution. Therefore, only the GC can setup the multicast group. Such a construction may satisfy the requirements of applications where group setup should be strictly controlled. We leave the public-key construction as a future work. In the following part of this section, we first introduce how attributes and the access structure are defined in our scheme. Then, we present each algorithm of our scheme.

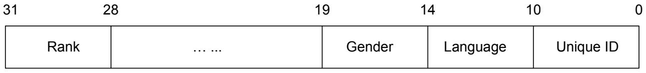

Figure 1: An example of ID

4.1 Attributes and Access Structure

In our scheme, each GM is assigned a n-bit unique ID which is composed of several segments. Each segment represents one attribute. Fig. 1 gives an example for the ID in the aforementioned military scenario.

Access Structure We use 1-level AND and NOT gates to represent the access structure. OR gate can be simulated using concatenation. For each attribute, we support relation predicates such as *equal*, *greater than* and *less than*. Therefore, a valid access structure has the form like "$(gender = male) \wedge (age < 40) \wedge (rank > captain) \wedge (not\ married)$". We believe this kind of access structure is descriptive enough for most applications. Note that, this access structure is only used by GC and not transmitted to the GMs.

Attributes As mentioned previously, we use each segment of ID to represent an attribute. We refer to these attributes as *high level attributes*. Besides these high level attributes, we also define *lower level attributes* for each bit of ID. The string "bit i is b" defines the attribute for bit i which we denote by $Att_{i,b}$, b=0|1. Therefore, each bit is associated with two lower level attributes:$Att_{i,0}$ and $Att_{i,1}$, $i \in \mathbb{Z}_n$. Each GM holds n lower level attributes, each for one bit of his ID. This definition of lower level attributes follows the similar way in which CFT defines the $KEKs$. Any high level attribute can be represented by lower level attributes. In the remaining part of this section, the term *attribute* refers to the lower level attribute.

4.2 Scheme Description

The four algorithms of our scheme are defined as follows.

System Setup This algorithm chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Each attribute is then mapped to an element of group $\mathbb{G}_0$. Let $h_{i,b}$ denote the corresponding element in group $\mathbb{G}_0$ of attribute $Att_{i,b}$. We have

$$h_{i,0} = g^{a_i},\ h_{i,1} = g^{b_i}$$

a_i and b_i are randomly generated from $\mathbb{Z}_p$. Let $\gamma_i = a_i + b_i$. a_i and b_i should be chosen in the way that a_i, b_i, and γ_i are all non-trivial.

This algorithm also chooses other two random numbers $\alpha, \beta \in \mathbb{Z}_p$. The system master secret key (MSK) is output as follows

$$MSK = (\alpha, \beta, \forall i \in \mathbb{Z}_n : h_{i,0}, h_{i,1})$$

MSK is only known to GC.

KeyGen This algorithm takes as input a GM's ID and generates his secret key as follows

$$SK = (D = g^{(\alpha+r+r')/\beta}, D' = g^{r'}, D'' = g^{\beta r}, \forall i \in \mathbb{Z}_n : D_i = g^r h^{r'}_{i,\bar{X}_i})$$

r and r' are two random numbers chosen from $\mathbb{Z}_p$. X_i is the i^{th} bit of the GM's ID and $\bar{X}_i$ is its inverse.

Table 1: Vector for product $\bar{X}_3 X_2 X_0$

	g^{s_i}	$C_{i,0}$	$C_{i,1}$
C_3	g^{s_3}	$h^{s_3+t_3}_{3,0}$	$h^{s_3}_{3,1}$
C_2	g^{s_2}	$h^{s_2}_{2,0}$	$h^{s_2+t_2}_{2,1}$
C_1	g^{s_1}	$h^{s_1}_{1,0}$	$h^{s_1}_{1,1}$
C_0	g^{s_0}	$h^{s_0}_{0,0}$	$h^{s_0+t_0}_{0,1}$

Encryption To deliver the group key (GK) to GMs of his interested attributes, GC first reduces the combination of these attributes to a sum of products expression (SOPE). Then, for each product in the SOPE, GC encrypts GK once and outputs a ciphertext of the following format:

$$CT = \langle C, C', C'', \forall i \in \mathbb{Z}_n : C_i \rangle$$

where $C = (GK||MAC) \cdot X$ and X is a blind factor used to hide $(GK||MAC)$. MAC is the message authentication code for GK. "$||$" means concatenation. C', C'', and all the C_i's are credentials for helping decryptors to reconstruct X and thus derive GK. Each bit of the GM's ID corresponds to a credential C_i, which actually includes a pair $(C_{i,0}, C_{i,1})$ for bit value 0 and 1 respectively.

Now, we present the detailed steps for CT generation. Notation is defined as follows: Each symbol in a SOPE product is called a *literal*, denoted by X'_j if it is the j^{th} bit of ID. If the symbol has the form $\bar{X}$, $X'_j = 0$. Otherwise $X'_j = 1$. We denote a SOPE product by S, a set of literals. The string "the i^{th} bit of a GM's ID X_i is in S" is represented by "$X_i \in S$".

- *Step 1.* Random Number Generation. GC chooses n random numbers $s_0, s_1, ..., s_{n-1} \in \mathbb{Z}_p$, and set $s = \sum_{j=0}^{n-1} s_j$ and $\delta = \sum_{j=0}^{n-1} \gamma_i s_i$.
- *Step 2.* C_i Computation. Here $C_i = (g^{s_i}, C_{i,0}, C_{i,1})$, $i \in \mathbb{Z}_n$. First, GC initializes each C_i as follows: $C_{i,0} = h^{s_i}_{i,0}$ and $C_{i,1} = h^{s_i}_{i,1}$. Next, for each X'_j, GC chooses a non-zero random number $t_j \in \mathbb{Z}_p$ and computes $h^{t_j}_{j,X'_j}$. Then, the item C_{j,X'_j} in the tuple C_j is updated as follows: C_{j,X'_j}=$C_{j,X'_j} \cdot h^{t_j}_{j,X'_j} = h^{s_j+t_j}_{j,X'_j}$. Table 1 illustrates the vector (C_3, C_2, C_1, C_0) for product $\bar{X}_3 X_2 X_0$, where $n = 4$.
- *Step 3.* C' and C'' Computation. GC first computes a value $g^{s'}$ as follows:

$$\begin{aligned} g^{s'} &= \prod_{i=0}^{n-1} (C_{i,0} C_{i,1}) \\ &= \prod_{i=0}^{n-1} (h_{i,0} h_{i,1})^{s_i} \cdot \prod_{\forall j,\ X'_j \in S} h^{t_j}_{j,X'_j} && (1) \\ &= \prod_{i=0}^{n-1} g^{\gamma_i s_i} \cdot \prod_{\forall j,\ X'_j \in S} h^{t_j}_{j,X'_j} && (2) \\ &= g^{\delta} \cdot \prod_{\forall j,\ X'_j \in S} h^{t_j}_{j,X'_j} && (3) \\ &= g^{\delta + x} \end{aligned}$$

Eq. (2) to Eq. (3) holds because $\delta = \sum_{j=0}^{n-1} \gamma_i s_i$. x is some number in $\mathbb{Z}_p$ such that

$$g^x = \prod_{\forall j,\ X'_j \in S} h_{j,X'_j}^{t_j}, \quad x \in \mathbb{Z}_p \tag{4}$$

Then, GC computes another value $g^{s''}$ as follows:

$$g^{s''} = \frac{g^s}{g^{s'}} = g^{s-s'}.$$

Finally, C' and C'' are computed as follows:

$$C' = g^{\beta s'},\ C'' = g^{s''/\beta}.$$

• *Step 4.* Ciphertext Generation. Ciphertext is output as follows:

$$CT = \langle C = (GK||MAC)e(g,g)^{\alpha s'}, C', C'', \forall i \in \mathbb{Z}_n : C_i \rangle \tag{5}$$

where $MAC = hash(GK)$. $hash(\cdot)$ is an one way hash function using algorithms such as SHA-1 [12].

Decryption This algorithm takes as input the ciphertext CT. It returns the group key GK if the GM's attributes satisfy the access structure. Otherwise, it returns an error symbol $\perp$.

• *Step 1.* Credential Pairing. Assume the GM's ID is $X_{n-1}X_{n-2}$...X$_0$. She picks C_{i,X_i} from C_i and computes a value F_i for each bit $i \in \mathbb{Z}_n$ of her ID as follows:

$$\begin{aligned} F_i &= e(D_i, g^{s_i})e(D', C_{i,X_i}) \\ &= e(g^r h_{i,\bar{X}_i}^{r'}, g^{s_i})e(g^{r'}, h_{i,X_i}^{s_i+t_i}),\ (t_i = 0,\ if\ X_i \notin S) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (6) \\ &= e(g,g)^{rs_i} e(g, h_{i,\bar{X}_i} h_{i,X_i})^{r's_i} e(g, h_{i,X_i})^{r't_i} \\ &= e(g,g)^{rs_i} e(g,g)^{r'\gamma_i s_i} e(g, h_{i,X_i})^{r't_i} \end{aligned}$$

In (6), $t_i = 0$ if $X_i \notin S$. Otherwise, $t_i \neq 0$.

• *Step 2.* Pairing Aggregation. GM aggregates the F_i's and computes another value F as follows:

$$\begin{aligned} F &= \prod_{i=0}^{n-1} F_i \\ &= \prod_{i=0}^{n-1} e(g,g)^{rs_i} e(g,g)^{r'\gamma_i s_i} e(g, h_{i,X_i})^{r't_i} && (7) \\ &= e(g,g)^{rs} e(g,g)^{r'\delta} \prod_{i=0}^{n-1} e(g, h_{i,X_i})^{r't_i} && (8) \\ &= e(g,g)^{rs} e(g,g)^{r'\delta} e(g,g)^{r'x'} && (9) \end{aligned}$$

Eq. (7) to Eq. (8) holds because $s = \sum_{j=0}^{n-1} s_j$ and $\delta = \sum_{j=0}^{n-1} \gamma_i s_i$. x' is some number (unknown) in $\mathbb{Z}_p$ such that

$$\begin{aligned} g^{x'} &= \prod_{i=0}^{n-1} h_{i,X_i}^{t_i},\ \ (t_i = 0,\ if\ X_i \notin S) \\ &= \prod_{\forall i,\ X_i \in S} h_{i,X_i}^{t_i},\ \ x' \in \mathbb{Z}_p && (10) \end{aligned}$$

Therefore,

$$e(g,g)^{r'x'} = \prod_{i=0}^{n-1} e(g, h_{i,X_i})^{r't_i},\ \ x' \in \mathbb{Z}_p \tag{11}$$

By Eq. (4) and Eq. (10), we have the following theorem:

THEOREM 1. *$x = x'$ iff the GM's ID contains all the literals of the product, i.e., the GM is in the multicast group.*

• *Step 3.* GK Derivation. GM derives the GK as follows

$$\begin{aligned} M' &= \frac{C}{e(C', D)e(C'', D'')/F} \\ &= \frac{C}{e(g^{\beta s'}, g^{(\alpha+r+r')/\beta})e(g^{s''/\beta}, g^{\beta r})/F} \\ &= \frac{(GK||MAC)e(g,g)^{\alpha s'}}{e(g,g)^{(\alpha s'+rs+r's')}/(e(g,g)^{rs}e(g,g)^{r'(\delta+x')})} \\ &= \frac{(GK||MAC)e(g,g)^{r'(\delta+x')}}{e(g,g)^{r's'}} \\ &= (GK||MAC)e(g,g)^{r'(x'-x)} && (12) \end{aligned}$$

• *Step 4.* GK Verification. We assume GK and MAC have fixed lengths of n_1 and n_2 bits respectively. To verify if she is in the intended multicast group, each GM takes the first n_1 bits and the remaining n_2 bits from M', denoted by M_1 and M_2 respectively, and checks if $M_2 = hash(M_1)$.

From Theorem 1 and Eq. (12), we know that only the intended GMs can recover $(GK||MAC)$ correctly. Therefore, only for the intended GMs, equation $M_2 = hash(M_1)$ holds and M_1 equals GK.

5. SCHEME EVALUATION

For the space limitation, this section just presents the results of our evaluation. Proofs to the theorems as well as other detailed evaluations can be available in the extended version.

5.1 Security Analysis

We analyze security of our scheme in terms of its correctness and fulfillment of our security goals.

Correctness of our design can be shown by the following theorems.

THEOREM 2. *GM can decrypt GK iff she holds all the attributes required by the GC.*

THEOREM 3. *Except for the GC, it is hard for any other parties to generate a valid credential D_i for attribute Att_{i,X_i} even if they have already known credentials of other attributes.*

From above theorems, we can conclude that: One, only the GMs with intended attributes can decrypt GK. Two, Any GM can not generate valid credentials for those attributes which are not assigned to him. Therefore, our design is correct.

Security goals can be shown met as follows.

Confidentiality of the GK As is shown above, only intended GMs can decrypt the GK. Moreover, it can be shown that collusion does not help the unintended GMs decrypt the GK. This is because each GM's SK is blinded by a unique blind factors r and r'.

Anonymity In our design, GM does not know if she is in the multicast group until she has decrypted the ciphertext. The Decryption algorithm takes all the credentials in SK as input. GM can not tell which credentials grant her access to the GK, nor how many credentials contribute to the access grant. Therefore, any GM, no matter intended or unintended, can not tell, even partially, how many bits of her

ID match the intended ID. Moreover, because of the unique blind factors r and r' in each GM's SK, collusion does not help reveal this information.

Backward Secrecy For backward secrecy, any new GM can not decrypt the messages sent before she joined the group. To achieve this goal, we can update the master secret key (MSK) α before any new GM joins. Similar to the process of delivering GK, we can deliver $g^{\alpha'/\beta}$ to all GMs. Upon $g^{\alpha'/\beta}$, each GM updates α as follows: $g^{(\alpha+r+r')/\beta} \cdot g^{\alpha'/\beta} = g^{(\alpha+\alpha'+r+r')/\beta}$. In this way, α is updated as $(\alpha+\alpha')$ securely.

Member Revocation Similar to delivering GK, we can update MSK α to all GMs but those to be revoked. Because each GM has a unique ID, our member revocation algorithm has the ability to revoke any single GM.

5.2 Performance Evaluation

For one product of SOPE, our encryption algorithm needs 1 pairing, $(4n+k+2)$ exponentiations and $(n+k)$ multiplication operations, where n is the number of bits in ID and k means the number of literals in the product $(k < n)$. Our decryption algorithm requires $(n+3)$ pairings, $2n$ exponentiations and one hash operation. Our ciphertext contains $(3n+3)$ group elements in the whole. If GM's ID has 32 bits, our scheme needs 210 ms for decryption and the ciphertext size is 6336 bytes if SS curves are used. If MNT curves are adopted, decryption requires 490 ms and the size of ciphertext is 3506 bytes. This result is based on our testing on the Pairing-Based Crypto (PBC) library[10].

Another factor affecting the performance is the number of product in a SOPE. Because our scheme limits the access structure to the forms such as "$(gender = male) \wedge (age < 40) \wedge (rank > captain) \wedge (not\ married)$" , we can assure that the combination of intended attributes can be reduced to a SOPE with very small number of products.

5.3 Comparison

Membership anonymity in multicast group setup is rarely addressed except for a similar work proposed by A.Barth et al.[1]. Identities of the recipients in this scheme are protected by encrypting the GK using every GM's public key. The computational load as well the ciphertext size are linear to the group size. This makes this scheme applicable to small-scale applications only. Our scheme, on the contrary, is suitable for large-scale applications since the computational load and the ciphertext size are linear to the length of a GM's ID and independent to the group size. Moreover, our scheme also protects the number of the group members while [1] does not.

6. CONCLUSION AND FUTURE WORK

In this paper, we analyzed a novel issue of on-demand multicast group setup with membership anonymity. Based on current techniques such as CP-ABE and CFT, we proposed a scheme in which not only the multicast group members' identities but also the number of members in a multicast group are protected. Anonymity is well protected even under powerful attacks, e.g., colluding attacks. Both attribute-based multicast group setup and single user revocability are supported by our scheme. Analysis shows that our scheme is suitable for large-scale applications since its efficiency is just linear to the length of GM's ID. We believe our scheme presents one point on the privacy-performance curve. One interesting future work could be reducing the computation and communication load while keeping the same level of anonymity.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-0716306 and CNS-0626601.

7. REFERENCES

[1] A. Barth, D. Boneh, and B. Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography '06*, pages 52–64, 2006.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *SP '07*, pages 321–334, 2007.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. *CRYPTO '05*, pages 258–275, 2005.

[4] J. C.H. Roth. *Fundamentals of Logical Design*. Thomson Engineering, 5 edn edition, 2003.

[5] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure lnternet multicast using boolean function minimization techniques. *INFOCOM '99*, pages 689–698 vol. 2, 1999.

[6] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport. Collusion-resistant group key management using attribute-based encryption. Cryptology ePrint Archive, Report 2007/161, 2007.

[7] L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *CCS '07*, pages 456–465, 2007.

[8] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO '93*, pages 480–491, 1994.

[9] C. Grosch. Framework for anonymity in ip-multicast environments. *GLOBECOM '00.*, pages 365–369 vol. 1, 2000.

[10] B. Lynn. Pbc library.

[11] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO '01*, pages 41–62, 2001.

[12] NIST. Secure hash standard. *Federal Information Processing Standard, FIPS-180-1*, April,1995.

[13] A. Pfitzmann and M. K. Anonymity, unobservability, and pseudeonymity – a proposal for terminology. In *International workshop on Designing privacy enhancing technologies*, pages 1–9, 2001.

[14] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The versakey framework: versatile group key management. *Selected Areas in Communications, IEEE Journal on*, 17(9):1614–1631, Sep. 1999.

[15] N. Weiler. Secure anonymous group infrastructure for common and future internet applications. *ACSAC 2001. Proceedings 17th Annual*, pages 401–410, 2001.