Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Attribute-based on-demand multicast group setup with membership anonymity [☆]

Shucheng Yu [a,*], Kui Ren [b], Wenjing Lou [a]

[a] Department of ECE, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[b] Department of ECE, Illinois Institute of Technology, Chicago, IL 60616, USA

ARTICLE INFO

Article history:
Received 23 April 2009
Received in revised form 5 August 2009
Accepted 19 September 2009
Available online 27 September 2009
Responsible Editor: W. Wang

Keywords:
Attribute-based encryption
Multicast
Anonymity
Group key

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 with group membership anonymity. In the proposed solution, scalability and membership anonymity are achieved via a novel design that integrates techniques such as ciphertext-policy attribute-based encryption (CP-ABE). In our design, multicast groups are specified through group member attributes. As these attributes are potentially able to be shared by unlimited number of group members, our proposed scheme scales well. Also, high level of membership anonymity is guaranteed such that every group member knows nothing but his own group membership only. The complexity of our proposed scheme in terms of computational overhead and ciphertext size is $O(n)$, where n is the number of attributes 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. Nowadays multicast services have been widely deployed for applications such as video conference and distance learning. We can imagine that in the near future, various multicast-based applications will be increasingly deployed [7,15]. Besides commercial applications, we believe that multicast services will be definitely applied in military and emergency tasks.

In mission-critical application scenarios, 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., [17,9], 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 [12], the complexity in terms of computation and communication overhead is linear to the number of group members. In large-scale application scenarios, such a complexity represents a heavy computation and communication overhead and thus makes these schemes unpractical.

[☆] The preliminary version of this paper appears in SecureComm 2008 [18].
[*] Corresponding author.
E-mail addresses: yscheng@ece.wpi.edu (S. Yu), kren@ece.iit.edu (K. Ren), wjlou@ece.wpi.edu (W. Lou).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.09.009

Privacy is another concern for mission-critical 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 [14] 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 number of the attributes only and independent of the number of users. In addition, single user revocability is supported in our proposed scheme.

1.1. Our contribution

The 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) our scheme is highly scalable; (3) single user revocability is supported. In our scheme, both the computation complexity and communication complexity are $O(n)$, where n is the number of the attributes and independent of the number of users.

1.2. Related work

Traditionally, multicast group key delivery relies on broadcast encryption techniques. In recent years, a novel cryptographic primitive is proposed, namely ciphertext-policy attribute-based encyption (CP-ABE), which is able to provide more flexible yet secure data encryption for one-to-many communications. This section provides a brief review of the two techniques that may be suitable for group key distribution.

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 secret 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 group members (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 et al. [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 encryptor 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.

Based on CP-ABE, Cheung et al. [5] proposed a collusion resistant group key management scheme. This paper enhances the flat table (FT) group key management schemes [16,4], which are vulnerable to collusion attacks, by defining each bit of user's ID as an attribute. With these attributes defined, the proposed scheme is able to employ CP-ABE to encrypt the group key and thus realizes collusion resistance. Following the similar attribute definition method, Cheung et al. proposed a provably secure CP-ABE scheme in [6]. In this paper, each attribute has three occurrences: *positive*, *negative*, and *don't care*. The access structure in their basic scheme is assumed to be 1-level AND gate only. Our design shares the similar idea on attribute definition by assuming that each attribute just has two possible occurrences, i.e., *positive* and *negative*. One significant difference between [6] and our proposed scheme is that the access policy is no longer transmitted in the latter. Consequently, our construction is able to conceal the information of "who is in the multicast group" and "how many GMs are in the group". A high level of anonymity is therefore achieved. Actually, none of current CP-ABE schemes [2,6] is designed with membership anonymity in mind. Eavesdroppers can easily derive who is the intended receiver from the access policy which is send in plaintext.

The rest of this paper is organized as follows: Section 2 discusses models and assumptions as well as some technique preliminaries. Section 3 presents our solution. Section 4 evaluates the performance of our proposed scheme. We conclude this paper in Section 5.

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 our scheme, we assume there is a single group controller (GC) in the network that is the initiator of every multicast group. All the potential group members (GMs) are preloaded with secret keys generated by GC. Later on GC could broadcast the encrypted group key to all these potential GMs such that only the intended GMs are able to decrypt. In this way, GC is able to pick out any subset of the GMs to form a multicast group. In this scheme, we assume the wireless network is able to deliver broadcast packets efficiently to GMs using existing routing protocols. We also assume the wireless device that each GM carries has the capability to support some expensive cryptographic primitives. In wireless networks traffic is transmitted via open channels. Therefore, transmissions could be overheard by eavesdroppers.

Trust model: in our design, we assume GC to be a trusted party. It holds each GM's secret key and the system master key as we will discuss later. GMs are neither trustworthy to GC nor between themselves. Therefore, any GM is not expected to know others' membership information as well as the multicast group size.

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 i 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 small number (less than t) of cooperative adversaries.

2.2. 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 capability to derive the group key even if they collude.
- *High level of anonymity* We want to protect the information of "*who is in the group*" and "*how many GMs are in the group*" from the adversary. To achieve this goal, we need to prevent the adversary from knowing the underlying access structure associated with the ciphertext and keep the ciphertext size unaffected by the selection of the access structure.
- *Revocability* The system should have the ability to revoke any single GM or a group of GMs sharing common attributes, which also means *forward secrecy* of the scheme.
- *Backward secrecy* A new GM should not have the ability to access data that were transmitted before he joined.

2.3. Preliminaries

This section briefly discusses preliminary techniques that will be used in our design.

2.3.1. CP-ABE scheme

A typical CP-ABE scheme consists of four algorithms:

Setup: this algorithm takes as input a security parameter κ. It outputs 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 ciphertext *CT* which has the following format:

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

where $\widetilde{C} = M \cdot X$ and X is a blind factor used to hide M. Y is the intended attribute set. C and all the C_y's are ciphertext components that help decryptors reconstruct the blind factor X and thus derive the message M.

KeyGen: this algorithm takes as input the master key *MK* and a set of attributes *S* associated with the user. It 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* here is transmitted to the recipients in plaintext. We refer to [2,6] for more details on CP-ABE.

2.3.2. Bilinear maps

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

Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$. A bilinear map is an injective function e: $\mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$ 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 $\forall u, v \in \mathbb{G}_0$.

3. The proposed scheme

This section describes our proposed scheme. In the following parts, we first introduce how attributes and the access structure are defined in our scheme. Then, we give an overview of our scheme. The detailed description for each algorithm of our proposed scheme is presented afterwards. In the last part of this section, we discuss the security result of our scheme.

3.1. Attributes and access structure

Attributes definition: in this work, we differentiate two kinds of attributes: *application level attributes* and *algorithm level attributes*. Application level attributes refer to those meaningful to human being, e.g., skill, occupation, rank, etc. Algorithm level attributes refer to those suitable for computer to interpret. Application level attributes can be mapped to algorithm level attributes. The mapping method is another interesting topic which is out of the scope of this work. In this paper, an attribute refers to an algorithm level attribute which is defined in such a way that it has two possible occurrences: *positive* and *negative*. We assume each algorithm level attribute is meaningful to every recipient. Therefore, the *don't care* case for any attribute is not considered in this paper. To formally represent each attribute, we use symbols $Att_{i,1}$ and $Att_{i,0}$ to denote the positive and the negative occurrences of attribute i respectively, where $i \in \mathbb{Z}_n$ is the index of the attributes and n is the total number of attributes. The set of attributes that each *GM* possesses is $\{Att_{i,b} | \forall i \in \mathbb{Z}_n, b = 0|1\}$. In our scheme description, we also use the binary string $X_{n-1}X_{n-2}\cdots X_0$ to denote the set of attributes that the *GM* possesses, where bit value 1 and 0 represent positive and negative occurrences of the attribute, respectively.

Access structure: we use 1-level *AND* logic over algorithm level attributes to represent the access structure. For example, in the case of $n = 4$, an access structure may have the form $(attribute3ispositive) \wedge (attribute1isnegative)$. If we use a product term to represent this access structure, it would be $X_3\bar{X}_1$. *OR* logic can be simulated using concatenation. Complex access structures over application level attributes can be easily realized using these logics. For example, we can realize the access structure "*rank > second lieutenant*" as follows: first, we map the application level attribute *rank* to a set of algorithm level attributes by enumerating all the ranks: $\{\cdots$, *lieutenant*, *captain*, $\cdots\}$. Then, the logic "*rank > second lieutenant*" can be implemented by *AND* all the negative algorithm level attributes for ranks lower than lieutenant. In this way, we can realize access structures over application level attributes such as "$(rank > second\ lieutenant) \wedge (service\ year < 5\ years) \wedge (gender = female)$". In the remaining part of this paper, we just consider the access structure over algorithm level attributes which can be represented via one product term, i.e., *AND* logic only. The term *attribute* in the remaining part of this paper refers to algorithm level attribute.

3.2. Scheme overview

Our scheme is composed of four algorithms: *Setup*, *KeyGen*, *Encryption*, and *Decryption*. The functionality of each algorithm here is similar to that of the CP-ABE scheme. The main difference is that our proposed scheme, if applied to multicast group setup, protects the membership information well while current CP-ABE schemes do not. To achieve membership anonymity, our scheme omits the access structure *T* from the ciphertext *CT*. To enable decrypting without the access structure while protecting the membership information, our scheme is designed as follows: first, our ciphertext comprises components for both positive and negative occurrence of all the attributes. Positive and/or negative attributes interested by the access structure are secretly marked so that it is hard to distinguish them from unmarked attributes. This effectively prevents eavesdroppers from deducing the membership information from the ciphertext. Second, our *Decryption* algorithm requires the recipient to take as input secret key components of all his attributes. After decryption, the recipient know nothing about which or how many attributes grant or decline him the access. This prevents the intended group members from knowing others GMs' membership information. As our current design is not a public-key solution, 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.

3.3. Scheme description

The four algorithms of our scheme are defined as follows.

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 $h_{i,0} = g^{a_i}$ and $h_{i,1} = g^{b_i}$, where 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 key (*MK*) is output as follows:

$$MK = (\alpha, \beta, \{a_i, b_i\}_{\forall i \in \mathbb{Z}_n}).$$

MK is only known to GC. Note that $h_{i,0}$ and $h_{i,1}$ are also only known to GC.

KeyGen: this algorithm takes as input a GM's attribute set $X_{n-1}X_{n-2}\cdots X_0$ and generates his secret key as follows:

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

where r is a random numbers chosen from $\mathbb{Z}_p$. X_i is the value of the ith attribute and $\overline{X}_i$ is its inverse.

Encryption: this algorithm takes as input the group key (*GK*), the access structure which is a product term, and the master key (*MK*). It outputs the ciphertext with the following format:

$$CT = (\widetilde{C}, \check{C}, \{\widehat{C}_j\}_{j=0,1}, \{C_i\}_{\forall i \in \mathbb{Z}_n}),$$

where $\widetilde{C} = (GK\|MAC) \cdot X$ and X is a blind factor used to hide $(GK\|MAC)$. *MAC* is the message authentication code for *GK*. "$\|$" means concatenation. $\check{C}$, $\widehat{C}_j$'s, and C_i's are ciphertext components to help decryptors reconstruct X and thus derive *GK*. Each bit of the GM's attribute set $X_{n-1}X_{n-2}\cdots X_0$ corresponds to a ciphertext component C_i which is a triple as we will describe later in this section. Before presenting the details of *CT* construction, we define notation as follows: Each symbol in a *product term* is called a *literal*, denoted by X'_i if it is for the *i*th bit of GM attribute set. If the symbol has the form $\overline{X}, X'_i = 0$; otherwise $X'_i = 1$. We denote the product term by *S*. The string "there is a literal in *S* for the *i*th bit of a GM's attribute set X_i" is represented by "$X_i \in S$". *CT* is constructed by the following steps:

- *Step 1.* Random Number Generation. GC chooses random numbers $s_0, s_1, \ldots, s_{n-1}, k_0, k_1 \in \mathbb{Z}_p$, and set $\delta = \sum_{j=0}^{n-1} \gamma_i s_i$.
- *Step 2.* C_i Computation. C_i is a triple of the form $C_i = (g^{s_i}, C_{i,0}, C_{i,1})$ and s_i is a random number generated in *step 1*. Both $C_{i,0}$ and $C_{i,1}$ are elements of group $\mathbb{G}_0$. If $X_i \in S$, GC chooses a random number $t_i \in \mathbb{Z}_p$ and calculates $C_{i,X'_i} = h_{i,X'_i}^{s_i+t_i}$ and $C_{i,1-X'_i} = h_{i,1-X'_i}^{s_i}$. Otherwise, it outputs $C_{i,0} = h_{i,0}^{s_i}$ and $C_{i,1} = h_{i,1}^{s_i}$.
- *Step 3.* $\check{C}$ Computation and C_i update. GC first computes a value $g^{s'}$ as follows:

$$g^{s'} = \prod_{i=0}^{n-1} (C_{i,0} C_{i,1}) = g^{\delta+x},$$

where *x* is some number in $\mathbb{Z}_p$ such that

$$g^x = \prod_{\forall j, X'_j \in S} h_{j,X'_j}^{t_j}. \quad (1)$$

Then, GC computes $\check{C} = g^{\beta s'}$. Finally, it updates $C_{i,0}$ and $C_{i,1}$ for all $i \in \mathbb{Z}_p$ as follows:

$$C_{i,0} = g^{k_0} C_{i,0}, \quad C_{i,1} = g^{k_1} C_{i,1}.$$

Table 1 illustrates an example vector (C_3, C_2, C_1, C_0) for product term $\overline{X}_3 X_2 X_0$, where $n = 4$.
- *Step 4.* Ciphertext Generation. Ciphertext is output as follows:

$$CT = (\widetilde{C} = (GK \| MAC) e(g,g)^{\alpha s'},$$
$$\check{C} = g^{\beta s'}, \{\widehat{C}_j = g^{\frac{k_j}{\beta}}\}_{j=0,1}, \{C_i\}_{\forall i \in \mathbb{Z}_n}), \quad (2)$$

where *MAC* = *hash*(*GK*). *hash*(·) is an one way hash function using algorithms such as SHA-1 [13].
Decryption: this algorithm takes as input the ciphertext *CT* and the GM's attribute set. It returns the group key *GK* if the GM's attributes satisfy the access structure. Otherwise, it returns an error symbol ⊥.
- *Step 1.* Credential Pairing. Assume the GM's attribute set is $X_{n-1}X_{n-2} \ldots X_0$. It first calculates

$$B_j = e(\widehat{C}_j, D'') = e\left(g^{\frac{k_j}{\beta}}, g^{\beta r}\right) = e(g,g)^{rk_j}, \quad j = 0, 1.$$

Table 1
Vector for product $\overline{X}_3 X_2 X_0$.

	g^{s_i}	$C_{i,0}$	$C_{i,1}$
C_3	g^{s_3}	$g^{k_0} h_{3,0}^{s_3+t_3}$	$g^{k_1} h_{3,1}^{s_3}$
C_2	g^{s_2}	$g^{k_0} h_{2,0}^{s_2}$	$g^{k_1} h_{2,1}^{s_2+t_2}$
C_1	g^{s_1}	$g^{k_0} h_{1,0}^{s_1}$	$g^{k_1} h_{1,1}^{s_1}$
C_0	g^{s_0}	$g^{k_0} h_{0,0}^{s_0}$	$g^{k_1} h_{0,1}^{s_0+t_0}$

Then, for each $i \in \mathbb{Z}_n$, it picks C_{i,X_i} from C_i and computes a value F_i for bit *i* of his attribute set as follows:

$$\begin{aligned} F_i &= e(D_i, g^{s_i}) e(C_{i,X_i}, D')/B_{X_i} \\ &= e(h_{i,\overline{X}_i}^r, g^{s_i}) e(g^{k_{X_i}} h_{i,X_i}^{s_i+t_i}, g^r)/B_{X_i} \\ &= e(g,g)^{r\gamma_i s_i} e(g, h_{i,X_i})^{rt_i}. \end{aligned} \quad (3)$$

In (3), $t_i = 0$ if $X_i \notin S$. Otherwise, $t_i \neq 0$. In the last step of derivation, we cancel B_{X_i} and $e(g,g)^{rk_{X_i}}$ as they are equal.
- *Step 2.* Pairing Aggregation. GM aggregates the F_i's and computes another value *F* as follows:

$$F = \prod_{i=0}^{n-1} F_i = \prod_{i=0}^{n-1} e(g,g)^{r\gamma_i s_i} e(g, h_{i,X_i})^{rt_i} = e(g,g)^{r\delta} e(g,g)^{rx'}$$

x' is some number (unknown) in $\mathbb{Z}_p$ such that

$$g^{x'} = \prod_{i=0}^{n-1} h_{i,X_i}^{t_i} = \prod_{\forall i, X_i \in S} h_{i,X_i}^{t_i}, \quad (t_i = 0, if X_i \notin S). \quad (4)$$

Therefore,

$$e(g,g)^{rx'} = \prod_{i=0}^{n-1} e(g, h_{i,X_i})^{rt_i}, \quad x' \in \mathbb{Z}_p. \quad (5)$$

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.*

Proof. By Eqs. (1) and (4), it is easy to see that if the GM's ID contains all the literals of the product, $x = x'$. On the other hand, if the ID does not contain all the literals of the product, the GM has negligible probability to output *F* in which $x' = x$ since t_i's are all random numbers. □

- *Step 3. GK* Derivation. GM derives the *GK* as follows:

$$\begin{aligned} M' &= \frac{\widetilde{C}}{e(\check{C}, D)/F} = \frac{(GK \| MAC) e(g,g)^{\alpha s'}}{e(g,g)^{(\alpha s' + rs')}/e(g,g)^{r(\delta + x')}} \\ &= (GK \| MAC) e(g,g)^{r(x'-x)}. \end{aligned} \quad (6)$$

- *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. (6), 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*.

3.4. 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 he holds all the attributes required by the GC.*

Proof. To decrypt GK, blind factor $e(g,g)^{\alpha s'}$ should be removed from $\widetilde{C}$ as illustrated by Eq. (6). The only way to construct $e(g,g)^{\alpha s'}$ is to perform a bilinear mapping between $g^{\beta s'}$ and $g^{(\alpha+r)/\beta}$, i.e., $e(g^{\beta s'}, g^{(\alpha+r)/\beta})$, which introduces another blind factor $e(g,g)^{rs'}$. As shown in Eq. (6), cancelling this blind factor requires $x = x'$ holds. By Theorem 1, $x = x'$ holds iff the GM 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 secret key component D_i for attribute Att_{i,X_i} even if they have already known secret key components of other attributes.*

Proof. As defined in Section 3, $h_{i,\overline{X}_i} = g^{a_i}$ or g^{b_i}, where $a_i, b_i \in \mathbb{Z}_p$ are two independent random numbers. Without lose of generality, we assume $h_{i,\overline{X}_i} = g^{a_i}$. Therefore, the secret key component for attribute Att_{i,X_i} is $D_i = h^r_{i,\overline{X}_i} = g^{ra_i}$, where r and a_i are not known to any GM. Any GM not assigned the attribute Att_{i,X_i} only knows $g^r, g^{\beta r}$, and g^{rb_i}. Without knowing a_i and b_i, it is hard to manipulate g^{ra_i} given $g^r, g^{\beta r}$ and g^{rb_i} since a_i and b_i are independent. Therefore, this theorem holds. □

From above theorems, we can conclude that: (1) only the GMs with intended attributes can decrypt GK; (2) 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 blind factor *r* unique to each *GM*.

Anonymity: first, we show that the eavesdroppers are not able to derive the access structure information (and hence the recipient information) from the ciphertext as follows. In our ciphertext, the intended attributes are secretly marked with a random number $t_j \in \mathbb{Z}_p, j \in \mathbb{Z}_n$. Assume $C_{i,0}$ and $C_{i,1}$ of attribute i have the following form: $C_{i,0} = g^{k_0} h^{s_i+t_i}_{i,0}$ and $C_{i,1} = g^{k_1} h^{s_i}_{i,1}$. Since $h_{i,0}$ and $h_{i,1}$ are not publicly known, $C_{i,0}$ and $C_{i,1}$ appear as the form of $C_{i,0} = g^{k_0} g^{a_i(s_i+t_i)}$ and $C_{i,1} = g^{k_1} g^{b_i s_i}$ from the eavesdroppers' viewpoint. As a_i and b_i are randomly and independently chosen for any attribute i, $C_{i,0}$ and $C_{i,1}$ appear to be independent and random for the eavesdroppers. Therefore, they are not able to tell which one is marked, without which they can not tell which and how many attributes are actually used in the access structure. Next, we show that the intended recipients are not able to derive the access structure information. This can be shown by observing the steps in the *Decryption* algorithm. As is shown in the steps, the GM does not know if he is in the multicast group until he has aggregated the secret key components of all his attributes and decrypted the ciphertext in *Step* 4. Since his attributes take effort if only they are aggregated, the GM can not tell which attributes grant or decline his access to the GK, nor how many attributes contribute to the access grant or declination. Therefore, any GM, no matter intended or unintended, can not tell, even partially, which or how many attributes are actually used in the access structure. Collusion does not help reveal this information because of the unique blind factor *r* in each GM's *SK*. In addition, any GM is not able to derive the number of associated attributes from the ciphertext size because it is constant in our proposed scheme.

Backward secrecy: for backward secrecy, any new GM cannot decrypt the messages sent before he joined the group. To achieve this goal, we can update the master key (MK) 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: $g^{(\alpha+r)/\beta} \cdot g^{\alpha'/\beta} = g^{(\alpha+\alpha'+r)/\beta}$. In this way, α is updated as $(\alpha + \alpha')$ securely. Member revocation can be realized in the same way except that we now update MK α to all GMs but those to be revoked.

4. Scheme evaluation

This section presents our evaluation results over the proposed scheme in terms of computation and communication overhead as well as storage overhead. We will present both numerical results and the experimental results. Finally, we give a brief discussion as well as the comparison between our proposed scheme and existing work. In the following part of this section, we assume the total number of attributes is *n*. We denote one scalar multiplication on the elliptic curve by an *EXP*, and one point addition operation by a *MUL*.

4.1. Numerical results

4.1.1. Computation load on GC

GC is responsible for execution of three algorithms: *Setup*, *KeyGen*, and *Encryption*. The main computation load of the algorithm *Setup* is caused by the calculation of $h_{i,0}$ and $h_{i,1}$, which involves *2n EXP* operations in total. The algorithm *KeyGen* is responsible for computing the secret key *SK*. The main computation overhead is caused by the calculation of $\{D_i = h^r_{i,\overline{X}_i}\}_{\forall i \in \mathbb{Z}_n}$. It accounts for *n EXP* operations. *KeyGen* consumes $(n+3)$*EXP* operations in total as the secret key components D, D', and D'' each accounts for one *EXP*. The main computation overhead of the algorithm *Encryption* comes from items $\{C_i = (g^{s_i}, C_{i,0}, C_{i,1})\}_{\forall i \in \mathbb{Z}_n}$ which represent $3n$ *EXP* operations.[1] In total the number of *EXP* operations required by *Encryption* is $(3n+3)$ since $\check{C}$ and $\widehat{C}$ consume 1 and 2 *EXP* operations respectively. In addition, *Encryption* requires one one-way hash operation. The bilinear pairing operation required by the item $\widetilde{C}$ can be ignored since we can pre-compute $e(g,g)^{\alpha}$. We do not count the integer field operations into our computation load because they

[1] Note that since *GC* knows the master secret key *MK*, he can first calculate the exponents and then compute $C_{i,j}, j = 0|1$, by one *EXP* operation. We present the calculation of $C_{i,j}$ by two *EXP*'s and one *MUL* in the algorithm just for clear description of our scheme. It is the similar case for $g^{s'}$.

account for a trivial part as compared to operations over elliptic curves.

4.1.2. Computation load on GM

Computation load on the GM side mainly comes from *Decryption*. According to our scheme description, following operations are required to decrypt one ciphertext: $(2n+3)$ pairings and one hash. In most cases, a pairing operation is more expensive than an *EXP* operation or a *MUL* operation. Therefore, it is desirable to minimize the number of pairing operations. For this purpose, we can modify our decryption algorithm a little bit. We show this by expanding the calculation of F in *Step* 2 of *Decryption*:

$$F = \prod_{i=0}^{n-1} F_i = \prod_{i=0}^{n-1} e(D_i, g^{s_i}) e(C_{i,X_i}, D')/B_{X_i}$$
$$= e\left(\prod_{i=0}^{n-1} C_{i,X_i}, D'\right) \cdot \prod_{i=0}^{n-1} (e(D_i, g^{s_i})/B_{X_i}). \quad (7)$$

Instead of computing $e(C_{i,X_i}, D')$ for each attribute as shown in *Step* 1, GM can first multiply all C_{i,X_i} for all i in $\mathbb{Z}_n$, and then apply one pairing between D' and the product of the multiplication as is shown in Eq. (7). This revision saves $(n-1)$ pairings and causes n *MUL* operations on the GM side. The computation load of *Decryption* algorithm now becomes $(n+4)$ pairings, n *MUL* operations and one hash. We do not take into count the computation of dividing B_{X_i}'s as they are trivial as compared to operations on elliptic curves. Table 2 concludes the main cryptographic operations required by our design.[2]

4.1.3. Communication load

Our ciphertext is composed of four parts: $\widetilde{C}, \check{C}, \{\widehat{C}_j\}_{j=0,1}, \{C_i\}_{\forall i \in \mathbb{Z}_n}$. Each C_i has three parts: $g^{s_i}, C_{i,0}, C_{i,1}$. Therefore, the ciphertext contains $(3n+3)$ G_0 and 1 G_T group elements in total.

4.1.4. Storage load on GM

The main storage load for each *GM* comes from the secret key *SK* which represents $(n+3)G_0$ group elements in total.

4.2. Experimental results

In our experiment, we implement our algorithms based on the Pairing-Based Crypto (PBC) library [11]. We test our program on Ubuntu 8.04 with Intel Pentium D 3.40GHz CPU.

4.2.1. The choice of elliptic curves

In our experiment, we test on three types of elliptic curves: supersingular curves (type A), MNT curves (type D), and type F curves as is named in [11]. Type A curves enable fast pairing, while type D and type F curves require short group element. In particular, type F curves provides

[2] Hash operations are not counted in the table since they just cause a negligible computational overhead.

Table 2
Cryptographic operations.

	MUL	EXP	Pairing
Setup[a]	0	$2n$	0
KeyGen[a]	0	$n+3$	0
Encryption[a]	0	$3n+3$	0
Decryption[b]	n	0	$n+4$

[a] Operations performed by the GC.
[b] Operations performed by GMs.

1920-bit RSA security with only 160-bit group element. For type D curves, we test on two kinds of curves, namely $d159$ and $d201$. The detailed description of these curves can be found in [11].

To understand how the choice of elliptic curves affects our ciphertext size, it is necessary to discuss two kinds of groups characterized by bilinear maps: symmetric bilinear groups and asymmetric bilinear groups. The bilinear map in the former case has the form: $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$, while the latter has the form: $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T, \mathbb{G}_1 \neq \mathbb{G}_2$. Type A curves are characterized by having symmetric bilinear groups. Type D and type F curves are characterized by having asymmetric bilinear groups. For fast pairing, we usually choose elliptic curves from symmetric bilinear groups with small embedding degrees. For example, type A curves have embedding degree of 2 and base field size of $\mathbb{G}_0$ is 512 bits. On the other hand, for short group size, we usually choose elliptic curves from asymmetric bilinear groups with high embedding degrees. For example, type F curves have embedding degree of 12. This turns out that the base field size of $\mathbb{G}_1$ is just 160 bits. The embedding degree of type D curves is 6. The base field size of $\mathbb{G}_1$ is just 170 bits for 1020-bit RSA security.

4.2.2. Ciphertext size

Our design uses symmetric bilinear groups by default. Therefore, the ciphertext size is $512(3n+4)$ bits in the case of type A curves as both $\mathbb{G}_0$ and $\mathbb{G}_T$ can be represented by 512 bits. To achieve short ciphertext size, we can easily modify our design by using asymmetric bilinear groups. After this revision the ciphertext would just contain the following group elements: $(2n+3)\mathbb{G}_1, n\mathbb{G}_2$, and 1 $\mathbb{G}_T$. Table 3 gives a summary of our ciphertext size under the selected curves. As shown in Table 3, type A curves (supersingular) exhibits the longest ciphertext while type F curves provides the shortest ciphertext.

4.2.3. Computation load

In our experiment, we run our algorithms over elliptic curves of type A (SS), $d159$, $d201$, and type F, respectively. The number of attributes is chosen to be 4, 8, 16, 32, 64,

Table 3
Ciphertext size (bits).

Type A	Type d159	Type d201	Type F
$512(3n+4)$	$159(5n+6)$	$201(5n+6)$	$160(4n+9)$

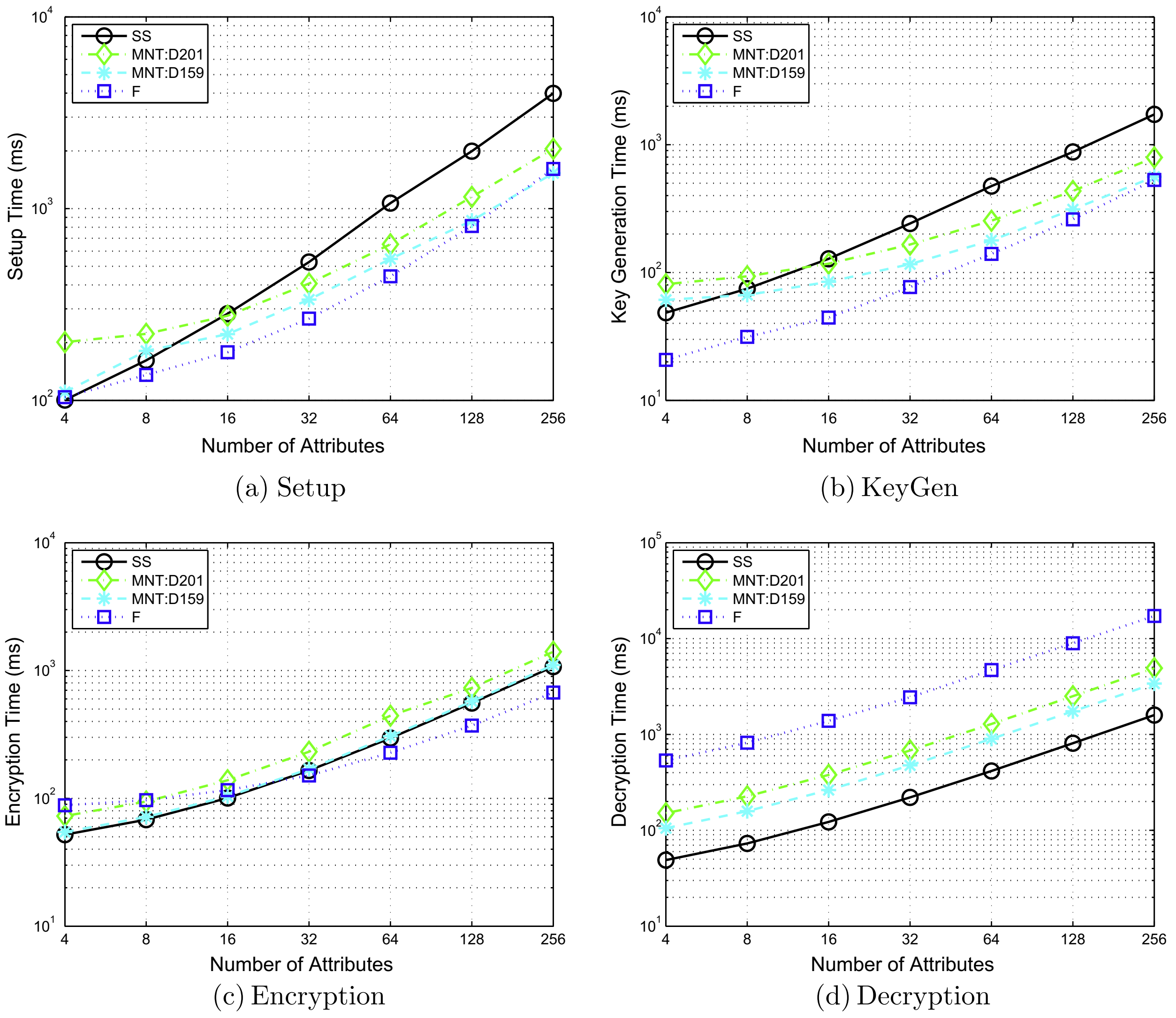


Fig. 1. Experiment result on computation load. As is shown in (a), (b), (c) and (d), computation load is linear to the number of attributes.

128, and 256. The experiment results are shown in Fig. 1a–d. From these figures we can see that the computation load of our scheme is linear to the number of attributes, which verifies our numerical analysis results in Table 2. As is analyzed in Table 2, computation overhead of algorithm *Setup*, *KeyGen*, and *Encryption* is dominated by scalar multiplication operations on elliptic curves. Therefore, the elliptic curves with small base field size are more efficient than others. This is verified by Fig. 1a–c which show that type F curves are the most efficient. Type A (SS) curves are the least efficient as is shown in Fig. 1a and b. In Fig. 1c, however, type A (SS) curves outperform type $d201$ curves and exhibit comparable encryption efficiency with $d159$ curves. This is because *Encryption* algorithm also involves n scalar multiplication operations on group $\mathbb{G}_2$. The base field size of $\mathbb{G}_2$ in the case of type F, $d159$, type A, and $d201$ are 320 bits, 477 bits, 512 bits, and 603 bits, respectively. As the base field size of $\mathbb{G}_2$ is much larger than that of $\mathbb{G}_1$, these n scalar multiplication operations dominate the computation load of *Encryption* algorithm. Fig. 1d shows that type A (SS) curves have the best decryption performance while type F curves are the worst. This coincides with our previous analysis.

Specifically, from these figures we can see that, in the case of 256 attributes our *Encrytion* algorithm takes about 0.7 s under type *F* curves and 1 s under type *D* and *F* curves. Our *Decryption* algorithm takes less than 2 s for type A curves in the case of 256 attributes. In the case of 64 attributes, *Decryption* takes about 0.3 s for type A curves and 1 s for type D curves.

4.3. Discussion and comparison

From above numerical and experimental results, we can see that our scheme exhibits an acceptable computation load if the number of attributes are carefully chosen. Actually, the computation overhead for both ECC and bilinear pairing operations can be further reduced to the magnitude of μs under hardware implementations [10]. Moreover, as the computing power of processors is increasing rapidly, computation overload should not be a problem. What actually matters is the communication overload as bandwidth is a limited resource under environments such as wireless networks. Fortunately, the communication overload of our scheme grows linearly to the number of attributes only. Since attributes are shared by unlimited number of users, communication overload of our scheme can be well controlled even in the case of large-scale application scenarios. As a matter of fact, even in large-scale systems, the number of attributes required could be relatively small. To evaluate the performance of our scheme, we can compare it with current work. Currently, the only similar work, to the best of our knowledge, is proposed by Barth et al. [1], in that scheme, identities of the recipients are protected by encrypting the GK using every GM's public-key. The computational load as well as the ciphertext size grow linear to the group size. Assume we are using 1024 bit RSA, the ciphertext size would be 1024N bits, where N is the total number of users. This represent a huge communication load in large-scale applications and not applicable for large-scale application scenarios. Moreover, our scheme also protects the number of the group members while [1] does not.

5. Conclusion and future work

In this paper, we analyzed an important problem of on-demand multicast group setup with membership anonymity. Based on current techniques such as CP-ABE, we proposed a scheme in which not only the multicast group members' identities but also the number of members in a multicast group are concealed. 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. Numerical and experimental results show that our scheme is suitable for large-scale applications since its overhead is just linear to the number of attributes, rather than the number of members in the group. One interesting future work could be reducing the computation and communication load while keeping the same level of anonymity.

Acknowledgment

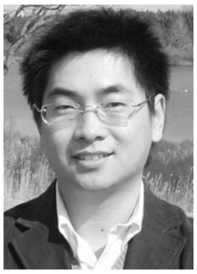
This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0746977, CNS-0831628, and CNS-0831963.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2009.09.009.

References

[1] A. Barth, D. Boneh, B. Waters, Privacy in encrypted content distribution using private broadcast encryption, in: Financial Cryptography '06, 2006, pp. 52–64.
[2] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: SP '07, 2007, pp. 321–334.
[3] D. Boneh, C. Gentry, B. Waters, Collusion resistant broadcast encryption with short ciphertexts and private keys, in: CRYPTO '05, 2005, pp. 258–275.
[4] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, Key management for secure internet multicast using boolean function minimization techniques, in: INFOCOM '99, vol. 2, 1999, pp. 689–698.
[5] L. Cheung, J.A. Cooley, R. Khazan, C. Newport, Collusion-resistant group key management using attribute-based encryption, in: Cryptology ePrint Archive, Report 2007/161, 2007.
[6] L. Cheung, C. Newport, Provably secure ciphertext policy abe, in: CCS '07, 2007, pp. 456–465.
[7] C.D.M. Cordeiro, H. Gossain, D.P. Agrawal, Multicast over wireless mobile ad hoc networks: present and future directions, IEEE Network 17 (2003) 52–59.
[8] A. Fiat, M. Naor, Broadcast encryption, in: CRYPTO '93, 1993, pp. 480–491.
[9] C. Grosch, Framework for anonymity in ip-multicast environments, in: GLOBECOM '00, 2000, pp. 365-369.
[10] M. Keller, T. Kerins, W. Marnane, FPGA implementation of a GF (2^{4m}) multiplier for use in pairing based cryptosystems, Field Program. Logic Appl. (2005) 594–597.
[11] B. Lynn, Pbc library, <http://crypto.stanford.edu/pbc>.
[12] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: CRYPTO '01, 2001, pp. 41–62.
[13] NIST, Secure hash standard, in: Federal Information Processing Standard, FIPS-180-1, April 1995.
[14] A. Pfitzmann, M.K. Anonymity, unobservability, and pseudeonymity – a proposal for terminology, in: International Workshop on Designing Privacy Enhancing Technologies, 2001, pp. 1–9.
[15] V. Upkar, Multicast over wireless networks, Commun. ACM 45 (12) (2002) 31–37.
[16] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner, The versakey framework: versatile group key management, IEEE J. Selected Areas Commun. 17 (9) (1999) 1614–1631.
[17] N. Weiler, Secure anonymous group infrastructure for common and future internet applications, in: ACSAC 2001, Proceedings 17th Annual, 2001, pp. 401–410.
[18] S. Yu, K. Ren, W. Lou, Attribute-based on-demand multicast group setup with membership anonymity, in: Secure Comm '08, 2008.



Shucheng Yu is currently a third year Ph.D student in the Electrical and Computer Engineering department at Worcester Polytechnic Institute. He received the M.E degree in Computer Science and Engineering from Tsinghua University, China, in 2004, the B.E degree in Computer Science and Engineering from Nanjing University of Post&Telecommunication, China, in 1999. He worked as a senior software engineer at Beijing R&D center of Cadence Design Systems from July 2006 to December 2006, at Chinese National Source Coding Center, from July 2004 to June 2006. From July 1999 to August 2001, he worked as a software engineer at Bright Oceans Corporation, Beijing, China. His research interests include network security and applied cryptography. His current research area focuses on privacy preserving access control of data on semi-trustable storage.

Kui Ren is an assistant professor in the Electrical and Computer Engineering department at Illinois Institute of Technology. He obtained his Ph.D degree in Electrical and Computer Engineering from Worcester Polytechnic Institute in 2007. He received his B.E and M.E both from Zhejiang University, China, in 1998 and 2001, respectively. He worked as a research assistant at Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, from March 2001 to January 2003, at Institute for Infocomm Research, Singapore, from January 2003 to August 2003, and at Information and Communications University, South Korea from September 2003 to June 2004. His research interests include security and privacy in networks and systems, cloud computing security, and wireless physical layer security.

Wenjing Lou obtained her Ph.D degree in Electrical and Computer Engineering from University of Florida in 2003. She received the M.A.Sc degree from Nanyang Technological University, Singapore, in 1998, the M.E degree and the B.E degree in Computer Science and Engineering from Xi'an Jiaotong University, China, in 1996 and 1993, respectively. From December 1997 to July 1999, she worked as a Research Engineer at Network Technology Research Center, Nanyang Technological University. She joined the Electrical and Computer Engineering department at Worcester Polytechnic Institute in 2003 where she is now an associate professor. Her current research interests are in the areas of ad hoc, sensor, and mesh networks, with emphases on network security and routing issues. She has been an editor for IEEE Transactions on Wireless Communications since 2007. She was named Joseph Samuel Satin Distinguished fellow in 2006 by WPI. She is a recipient of the US National Science Foundation Faculty Early Career Development (CAREER) award in 2008.