Improving Security and Efficiency in Attribute-Based Data Sharing

Junbeom Hur

Abstract—With the recent adoption and diffusion of the data sharing paradigm in distributed systems such as online social networks or cloud computing, there have been increasing demands and concerns for distributed data security. One of the most challenging issues in data sharing systems is the enforcement of access policies and the support of policies updates. Ciphertext policy attribute-based encryption (CP-ABE) is becoming a promising cryptographic solution to this issue. It enables data owners to define their own access policies over user attributes and enforce the policies on the data to be distributed. However, the advantage comes with a major drawback which is known as a key escrow problem. The key generation center could decrypt any messages addressed to specific users by generating their private keys. This is not suitable for data sharing scenarios where the data owner would like to make their private data only accessible to designated users. In addition, applying CP-ABE in the data sharing system introduces another challenge with regard to the user revocation since the access policies are defined only over the attribute universe. Therefore, in this study, we propose a novel CP-ABE scheme for a data sharing system by exploiting the characteristic of the system architecture. The proposed scheme features the following achievements: 1) the key escrow problem could be solved by escrow-free key issuing protocol, which is constructed using the secure two-party computation between the key generation center and the data-storing center, and 2) fine-grained user revocation per each attribute could be done by proxy encryption which takes advantage of the selective attribute group key distribution on top of the ABE. The performance and security analyses indicate that the proposed scheme is efficient to securely manage the data distributed in the data sharing system.

Index Terms—Data sharing, attribute-based encryption, revocation, access control, removing escrow

1 Introduction

Recent development of the network and computing technology enables many people to easily share their data with others using online external storages. People can share their lives with friends by uploading their private photos or messages into the online social networks such as Facebook and MySpace; or upload highly sensitive personal health records (PHRs) into online data servers such as Microsoft HealthVault, Google Health for ease of sharing with their primary doctors or for cost saving. As people enjoy the advantages of these new technologies and services, their concerns about data security and access control also arise. Improper use of the data by the storage server or unauthorized access by outside users could be potential threats to their data. People would like to make their sensitive or private data only accessible to the authorized people with credentials they specified.

Attribute-based encryption (ABE) is a promising cryptographic approach that achieves a fine-grained data access control [3], [4], [5], [6]. It provides a way of defining access policies based on different attributes of the requester, environment, or the data object. Especially, ciphertext-policy attribute-based encryption (CP-ABE) enables an encryptor to define the attribute set over a universe of attributes that a decryptor needs to possess in order to decrypt the ciphertext, and enforce it on the contents [5]. Thus, each user with a different set of attributes is allowed to decrypt different pieces of data per the security policy. This effectively eliminates the need to rely on the data storage server for preventing unauthorized data access, which is the traditional access control approach of such as the reference monitor [1].

Nevertheless, applying CP-ABE in the data sharing system has several challenges. In CP-ABE, the key generation center (KGC) generates private keys of users by applying the KGC's master secret keys to users' associated set of attributes. Thus, the major benefit of this approach is to largely reduce the need for processing and storing public key certificates under traditional public key infrastructure (PKI). However, the advantage of the CP-ABE comes with a major drawback which is known as a key escrow problem. The KGC can decrypt every ciphertext addressed to specific users by generating their attribute keys. This could be a potential threat to the data confidentiality or privacy in the data sharing systems. Another challenge is the key revocation. Since some users may change their associate attributes at some time, or some private keys might be compromised, key revocation or update for each attribute is necessary in order to make systems secure. This issue is even more difficult especially in ABE, since each attribute is conceivably shared by multiple users (henceforth, we refer to such a set of users as an attribute group). This implies that revocation of any attribute or any single user in an attribute group would affect all users in the group. It may result in bottleneck during rekeying procedure or security degradation due to the windows of vulnerability.

• *The author is with the School of Computer Science and Engineering, Chuang-Ang University, 221 Heukseok-dong, Dongjak-gu, Seoul, Republic of Korea. E-mail: jbhur@cau.ac.kr.*

Manuscript received 25 Oct. 2010; revised 4 Jan. 2011; accepted 6 Mar. 2011; published online 18 Mar. 2011.
Recommended for acceptance by K.-L. Tan.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-10-0564.
Digital Object Identifier no. 10.1109/TKDE.2011.78.

1.1 Related Work

ABE comes in two flavors called key-policy ABE (KP-ABE) and ciphertext-policy ABE. In KP-ABE, attributes are used to describe the encrypted data and policies are built into users' keys; while in CP-ABE, the attributes are used to describe users' credentials, and an encryptor determines a policy on who can decrypt the data. Between the two approaches, CP-ABE is more appropriate to the data sharing system because it puts the access policy decisions in the hands of the data owners [2], [13].

1.1.1 Removing Escrow

Most of the existing ABE schemes are constructed on the architecture where a single trusted authority, or KGC has the power to generate the whole private keys of users with its master secret information [3], [5], [6], [17], [18], [19]. Thus, the key escrow problem is inherent such that the KGC can decrypt every ciphertext addressed to users in the system by generating their secret keys at any time.

Chase and Chow [22] presented a distributed KP-ABE scheme that solves the key escrow problem in a multi-authority system. In this approach, all (disjoint) attribute authorities are participating in the key generation protocol in a distributed way such that they cannot pool their data and link multiple attribute sets belonging to the same user. One disadvantage of this kind of fully distributed approach is the performance degradation. Since there is no centralized authority with master secret information, all attribute authorities should communicate with the other authorities in the system to generate a user's secret key. This results in $O(N^2)$ communication overhead on the system setup phase and on any rekeying phase, and requires each user to store $O(N^2)$ additional auxiliary key components besides the attributes keys, where N is the number of authorities in the system.

Recently, Chow [23] proposed an anonymous private key generation protocol in identity-based literature such that the KGC can issue a private key to an authenticated user without knowing the list of users' identities. It seems that this anonymous private key generation protocol works properly in ABE systems when we treat an attribute as an identity in this construction. However, we found that this cannot be adapted to ABE systems due to mainly two reasons. First, in Chow's protocol, identities of users are not public anymore, at least to the KGC, because the KGC can generate users' secret keys otherwise. Since public keys (attributes in the ABE setting) are no longer "public," it needs additional secure protocols for users to obtain the attribute information from attribute authorities. Second, since the collusion attack between users is the main security threat in ABE, the KGC issues different personalized key components to users by blinding them with a random secret even if they are associated with the same set of attributes. The random secret is unique and should be consistent with the same user for any possible attribute change (such as adding some attributes) of the user. However, it is impossible for the KGC to issue a personalized key component with the same random secret as that of attribute key components to a user, since the KGC can by no means know which random secrets (used to issue a set of attributes key components) are assigned to which users in the Chow's key issuing protocol.

1.1.2 Revocation

Bethencourt et al. [5] and Boldyreva et al. [8] proposed first key revocation mechanisms in CP-ABE and KP-ABE settings, respectively. These schemes enable an attribute key revocation by encrypting the message to the attribute set with its validation time. These attribute-revocable ABE schemes [5], [8], [10] have the security degradation problem in terms of the backward and forward secrecy [11]. They revoke attribute itself using timed rekeying mechanism, which is realized by setting expiration time on each attribute. In ABE systems, it is a considerable scenario that membership may change frequently in the attribute group. Then, a new user might be able to access the previous data encrypted before his joining until the data are reencrypted with the newly updated attribute keys by periodic rekeying (backward secrecy). On the other hand, a revoked user would still be able to access the encrypted data even if he does not hold the attribute any more until the next expiration time (forward secrecy). Such an uncontrolled period is called the window of vulnerability.

Recently, the importance of immediate user revocation (rather than attribute revocation) has been taken notice of in many practical ABE-based systems [6], [7], [12]. The user revocation can be done by using ABE that supports negative clauses, proposed by Ostrovsky et al. [6]. To do so, one just adds conjunctively the AND of negation of revoked user identities (where each is considered as an attribute here). One drawback in this scheme is that the private key size increases by a multiplicative factor of $\log n$, where n is the maximum number of attributes. Lewko et al. [7] proposed more efficient instantiations of Ostrovsky et al.'s framework [6] for nonmonotonic ABE, where public parameters is only $O(1)$ group elements, and private keys for access structures involving t leaf attributes is of size $O(t)$. However, these user-revocable schemes also have a limitation with regard to the availability. When a user is revoked even from a single attribute group, he loses all the access rights to the system, which is not desirable in many pragmatic scenarios since the other attributes may be still valid.

Attrapadung and Imai [9] suggested another user-revocable ABE schemes addressing this problem by combining broadcast encryption schemes with ABE schemes. However, in this scheme, the data owner should take full charge of maintaining all the membership lists for each attribute group to enable the direct user revocation. This scheme is not applicable to the data sharing system, because the data owners will no longer be directly in control of data after storing their data to the external storage server. Yu et al. [13] also recently addressed the user revocation in the ABE-based data sharing system. In this scheme, the user revocation is realized using proxy reencryption by the data server. However, in order to revoke users, the KGC should generate all secret keys including the proxy key on behalf of the data server. Then, the server would reencrypt the ciphertext under the proxy key received from the KGC to prevent revoked users from decrypting the ciphertext. Thus, the key escrow problem is also inherent in this scheme, since the KGC manages all secret keys of users as well as the proxy keys of the data server.

1.2 Contribution

In this paper, we propose a novel CP-ABE scheme for a secure data sharing system, which features the following achievements.

First, the key escrow problem is resolved by a key issuing protocol that exploits the characteristic of the data sharing system architecture. The key issuing protocol generates and issues user secret keys by performing a secure two-party computation (2PC) protocol between the KGC and the data-storing center with their own master secrets. The 2PC protocol deters them from obtaining any master secret information of each other such that none of them could generate the whole set of user keys alone. Thus, users are not required to fully trust the KGC and the data-storing center in order to protect their data to be shared. The data confidentiality and privacy can be cryptographically enforced against any curious KGC or data-storing center in the proposed scheme.

Second, the immediate user revocation can be done via the proxy encryption mechanism together with the CP-ABE algorithm. Attribute group keys are selectively distributed to the valid users in each attribute group, which then are used to reencrypt the ciphertext encrypted under the CP-ABE algorithm. The immediate user revocation enhances the backward/forward secrecy of the data on any membership changes. In addition, as the user revocation can be done on each attribute level rather than on system level, more fine-grained user access control can be possible. Even if a user is revoked from some attribute groups, he would still be able to decrypt the shared data as long as the other attributes that he holds satisfy the access policy of the ciphertext.

Data owners need not be concerned about defining any access policy for users, but just need to define only the access policy for attributes as in the previous ABE schemes. The proposed scheme delegates most laborious tasks of membership management and user revocation to the data-storing center while the KGC is responsible for the attribute key management as in the previous CP-ABE schemes without leaking any confidential information to the other parties. Therefore, the proposed scheme is the most suitable for the data sharing scenarios where users encrypt the data only once and upload it to the data-storing centers, and leave the rest of the tasks to the data-storing centers such as reencryption and revocation.

1.3 Organization

The rest of the paper is organized as follows. Section 2 describes the system architecture and security requirements. Section 3 reviews cryptographic background and defines the general framework of the escrow-free CP-ABE with revocation capability. In Section 4, we propose our construction. We analyze the efficiency and security of the proposed scheme in Sections 5 and 6, respectively. In Section 7, we conclude the paper.

2 Data Sharing Architecture

In this section, we describe the data sharing architecture and define the security model.

2.1 System Description and Key Management

Fig. 1 shows the architecture of the data sharing system, which consists of the following system entities:

1. Key generation center. It is a key authority that generates public and secret parameters for CP-ABE. It is in charge of issuing, revoking, and updating attribute keys for users. It grants differential access rights to individual users based on their attributes. It is assumed to be honest-but-curious. That is, it will honestly execute the assigned tasks in the system; however, it would like to learn information of encrypted contents as much as possible. Thus, it should be prevented from accessing the plaintext of the encrypted data even if it is honest.
2. Data-storing center. It is an entity that provides a data sharing service. It is in charge of controlling the accesses from outside users to the storing data and providing corresponding contents services. The data-storing center is another key authority that generates personalized user key with the KGC, and issues and revokes attribute group keys to valid users per each attribute, which are used to enforce a fine-grained user access control. Similar to the previous schemes [2], [13], [14], we assume the data-storing center is also semitrusted (that is, honest-but-curious) like the KGC.
3. Data owner. It is a client who owns data, and wishes to upload it into the external data-storing center for ease of sharing or for cost saving. A data owner is responsible for defining (attribute-based) access policy, and enforcing it on its own data by encrypting the data under the policy before distributing it.
4. User. It is an entity who wants to access the data. If a user possesses a set of attributes satisfying the access policy of the encrypted data, and is not revoked in any of the valid attribute groups, then he will be able to decrypt the ciphertext and obtain the data.

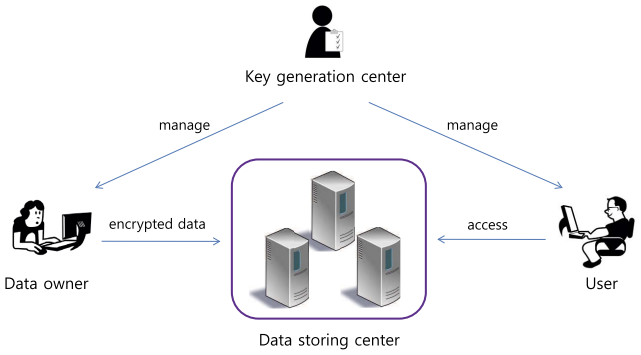


Fig. 1. Architecture of a data sharing system.

Since both of the key managers, the KGC and the data-storing center, are semitrusted, they should be deterred from accessing plaintext of the data to be shared; meanwhile, they should be still able to issue secret keys to users. In order to realize this somewhat contradictory requirement, the two parties engage in the arithmetic 2PC protocol with master secret keys of their own, and issue independent key components to users during the key issuing phase. The 2PC protocol deters them from knowing each other's master secrets so that none of them can generate the whole set of secret keys of users individually. Thus, we take an assumption that the KGC does not collude with the data-storing center since they are honest as in [2], [13], [14] (otherwise, they can guess the secret keys of every user by sharing their master secrets).

2.2 Threat Model and Security Requirements

1. Data confidentiality. Unauthorized users who do not have enough attributes satisfying the access policy should be prevented from accessing the plaintext of the data. Additionally, the KGC is no longer fully trusted in the data sharing system. Thus, unauthorized access from the KGC as well as the data-storing center to the plaintext of the encrypted data should be prevented.
2. Collusion resistance. Collusion resistance is one of the most important security property required in ABE systems [3], [4], [5]. If multiple users collude, they may be able to decrypt a ciphertext by combining their attributes even if each of the users cannot decrypt the ciphertext alone. We do not want these colluders to be able to decrypt the private data in the server by combining their attributes. Since we assume the KGC and data-storing center are honest, we do not consider any active attacks from them by colluding with revoked users as in [2], [13].
3. Backward and forward secrecy. In the context of attribute-based encryption, backward secrecy means that any user who comes to hold an attribute (that satisfies the access policy) should be prevented from accessing the plaintext of the previous data distributed before he holds the attribute. On the other hand, forward secrecy means that any user who drops an attribute should be prevented from accessing the plaintext of the subsequent data distributed after he drops the attribute, unless the other valid attributes that he is holding satisfy the access policy.

3 Preliminaries and Definition

3.1 Cryptographic Background

We first provide a formal definition for access structure by recapitulating the definitions in [4], [5]. Then, we will briefly review the cryptographic background about the bilinear map and its security assumption.

3.1.1 Notations

In this paper, $x \in_R S$ denotes the operation of picking an element x at random and uniformly from a finite set S. For a probabilistic algorithm $\mathcal{A}, x \xleftarrow{\$} \mathcal{A}$ assigns the output of $\mathcal{A}$ to the variable x. 1^λ denotes a string of λ ones, if $\lambda \in \mathbb{N}$. A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible ($\mathsf{negl}(k)$) if for every constant $c \geq 0$ there exists k_c such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

3.1.2 Access Structure

Definition 1 (Access structure). *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of nonempty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In CP-ABE schemes, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. From now on, by an access structure we mean a monotone access structure.

3.1.3 Bilinear Pairings

Definition 2 (Bilinear map). *Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be a multiplicative cyclic group of prime order p. Let g be a generator of $\mathbb{G}_0$. A map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ is said to be bilinear if $e(P^a, Q^b) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_0$ and all $a, b \in \mathbb{Z}_p^*$, and nondegenerate if $e(g, g) \neq 1$ for the generator g of $\mathbb{G}_0$.*

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ can be computed efficiently and there exists $\mathbb{G}_1$ for which the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ is efficiently computable.

3.1.4 Bilinear Diffie-Hellman (BDH) Assumption

Using the above notations, the Bilinear Diffie-Hellman problem is to compute $e(g, g)^{abc} \in \mathbb{G}_1$ given a generator g of $\mathbb{G}_0$ and elements g^a, g^b, g^c for $a, b, c \in \mathbb{Z}_p^*$. An equivalent formulation of the BDH problem is to compute $e(A, B)^c$ given a generator g of $\mathbb{G}_0$, and elements A, B and g^c in $\mathbb{G}_0$.

An algorithm $\mathcal{A}$ has advantage $\epsilon(\kappa)$ in solving the BDH problem for a bilinear map group $\langle p, \mathbb{G}_0, \mathbb{G}_1, e \rangle$, where κ is the security parameter (the bit length of p), if $\Pr[\mathcal{A}(p, \mathbb{G}_0, \mathbb{G}_1, A, B, g^c) = e(A, B)^c] \geq \epsilon(\kappa)$. If for every polynomial-time algorithm (in the security parameter κ) to solve the BDH problem on $\langle p, \mathbb{G}_0, \mathbb{G}_1, e \rangle$, the advantage $\epsilon(\kappa)$ is a negligible function, then $\langle p, \mathbb{G}_0, \mathbb{G}_1, e \rangle$ is said to satisfy the BDH assumption.

3.1.5 One-Way Anonymous Key Agreement

In a Boneh-Franklin identity-based encryption setup [15], a trusted key authority called private key generator (PKG) generates private keys d_i for users with identities ID_i using a master secret s. A user with identity ID_i receives the private key $d_i = H(ID_i)^s \in \mathbb{G}_0$, where $H : \{0, 1\}^* \to \mathbb{G}_0$ is a cryptographic hash function.

On the basis of this setup, Kate et al. [16] proposed a one-way anonymous key agreement scheme by replacing the identity hashes with pseudonyms generated by users. One-way anonymous key agreement is to guarantee anonymity for just one of the participants; the other participant works as a nonanonymous service provider and the anonymous participant needs to confirm the service provider's identity. In this setting, two participants can agree on a session key in a noninteractive manner.

Suppose Alice and Bob are clients of the same key authority. Alice has identity ID_A and private key $d_A = Q_A^s = H(ID_A)^s$. Alice wishes to remain anonymous to Bob whose identity is ID_B. Then, the key agreement protocol progresses as follows:

1. Alice computes $Q_B = H(ID_B)$. She chooses a random integer $r_A \in \mathbb{Z}_p^*$, generates the corresponding pseudonym $P_A = Q_A^{r_A}$, and computes the session key $K_{A,B} = e(d_A, Q_B)^{r_A} = e(Q_A, Q_B)^{sr_A}$. She sends her pseudonym P_A to Bob.

2. Bob computes the session key $K_{A,B} = e(P_A, d_B) = e(Q_A, Q_B)^{sr_A}$ using his private key d_B.

Kate et al. proved that this protocol is secure in the random oracle model assuming the BDH problem in $\langle p, \mathbb{G}_0, \mathbb{G}_1, e\rangle$ is hard in terms of the unconditional anonymity, session key secrecy, and no impersonation. The proof can be found in [16].

3.2 Definitions

In this section, we define a general key issuing and encryption/decryption framework for escrow-free CP-ABE with user revocation capability.

Let $\mathcal{U} = \{u_1, \ldots, u_n\}$ be the universe of users. Let ID_t be the identity of a user u_t. Let $\mathcal{L} = \{\lambda_1, \ldots, \lambda_p\}$ be the universe of descriptive attributes in the system. Let $G_y \subset \mathcal{U}$ be a set of users that hold the attribute λ_y, which is referred to as an attribute group. G_y will be used as a user access (or revocation) list to λ_y. Let $\mathcal{G} = \{G_1, \ldots, G_p\}$ be the universe of such attribute groups. Let K_{λ_y} be the attribute group key that is shared among the nonrevoked users in $G_y \in \mathcal{G}$.

3.2.1 Escrow-Free Key Issuing Protocol for CP-ABE

The KGC and the data-storing center are involved in the user key issuing protocol. In the protocol, a user is required to contact the two parties before getting a set of keys. The KGC is responsible for authenticating a user and issuing attribute keys to him if the user is entitled to the attributes. The secret key is generated through the secure 2PC protocol between the KGC and the data-storing center. They engage in the arithmetic secure 2PC protocol with master secret keys of their own, and issue independent key components to a user. Then, the user is able to generate the whole secret keys with the key components separately received from the two authorities. The secure 2PC protocol deters them from knowing each other's master secrets so that none of them can generate the whole secret keys of a user alone.

The escrow-free key issuing protocol consists of the following seven polynomial-time algorithms:

1. via $param \xleftarrow{\$} \mathrm{Setup}(1^\lambda)$, the trust initializer[1] outputs the system public parameters $param$. For brevity, the public parameter $param$ output by Setup is omitted below.
2. via $(PK_K, MK_K) \xleftarrow{\$} \mathrm{KKeyGen}()$, the KGC probabilistically outputs the public and private key pair PK_K, MK_K.
3. via $(PK_D, MK_D) \xleftarrow{\$} \mathrm{DKeyGen}()$, the data-storing center probabilistically outputs the public and private key pair PK_D, MK_D.
4. $\mathrm{KeyCom}_D(MK_D, ID_t) \leftrightarrow \mathrm{KeyCom}_K(MK_K, ID_t, aux_t)$ are two interactive algorithms which execute a user secret key generating protocol for a user u_t between the KGC and the data-storing center. The KGC takes as input its master secret key MK_K, a user identity ID_t, and the corresponding (personalized) secret information aux_t, and gets nothing as output. The data-storing center takes as input its master secret key MK_K and a user identity ID_t, and gets a personalized key component for a user u_t as output.
5. via $SK_{K,u_t} \xleftarrow{\$} \mathrm{IssueKey}_K(aux_t, S)$, the KGC takes as input the personalized secret information aux_t for a user u_t and a set of attributes S that describe the key, outputs a user secret key SK_{K,u_t}. SK_{K,u_t} consists of a set of attribute keys associated with S.
6. via $SK_{D,u_t} \xleftarrow{\$} \mathrm{IssueKey}_D()$, the data-storing center takes as input nothing, and outputs a personalized key component SK_{D,u_t} for a user u_t and a key encryption key (KEK) for the secure attribute group key distribution.

The first step of the key issuing protocol is to generate the user secret keys using secure 2PC protocol between the KGC and the data-storing center. When the KGC authenticates a user u_t who is entitled to a set S of attributes, the KGC starts to perform the secure 2PC protocol with the data-storing center. Then, the user receives two key components SK_{D,u_t} and SK_{K,u_t} from the data-storing center and KGC, respectively, as a result of the protocol. We require that the user can derive the whole secret key set SK_{u_t} using the two key components.

3.2.2 CP-ABE with User Revocation Capability

The CP-ABE scheme with user revocation capability is composed of the following three extra algorithms:

1. via $\mathrm{CT} \xleftarrow{\$} \mathrm{Encrypt}(\mathrm{PK}, M, \mathbb{A})$, anyone can encrypt a message M with public keys PK in the system under an access structure $\mathbb{A}$ over the universe of attributes, and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. CT implicitly contains $\mathbb{A}$.
2. via $\mathrm{CT}' \xleftarrow{\$} \mathrm{ReEncrypt}(\mathrm{CT}, G)$, the data-storing center can reencrypt a ciphertext CT for each attribute group in G. If the attribute groups appear in $\mathbb{A}$, the algorithm reencrypts CT for the attributes; else, returns $\perp$. Specifically, it outputs a reencrypted ciphertext CT' such that only a user who possesses a set of attributes that satisfies the access structure and has a valid membership for each of them (has valid attribute group keys for each of the attributes) at the same time will be able to decrypt the message.
3. via $M \xleftarrow{\$} \mathrm{Decrypt}(\mathrm{CT}', SK, K_\Lambda)$, a user with SK that satisfies the access structure embedded in CT' and a set of attribute group keys K_Λ for a set of attributes Λ will decrypt CT' and return a message M, iff Λ satisfies $\mathbb{A}$ and K_Λ is not revoked for any $\lambda \in \Lambda$.

4 Proposed CP-ABE Scheme

Since the first CP-ABE scheme proposed by Bethencourt et al. [5], dozens of the subsequent CP-ABE schemes have been suggested [2], [17], [18], [19], which are mostly motivated by more rigorous security proof in the standard model. However, most of the schemes failed to achieve the expressiveness of the Bethencourt et al.'s scheme, which described an efficient system that was expressive in that it allowed an encryptor to express an access predicate in terms of any monotonic formula over attributes. Therefore, in this section, we develop a variation of the CP-ABE algorithm partially based on (but not limited to) Bethencourt et al.'s construction in order to enhance the expressiveness of the

1. To prove security, we assume the parameters for the hash functions are setup by an honest party, which means the random oracles are not controlled by the adversary in the security proof [22], [23].

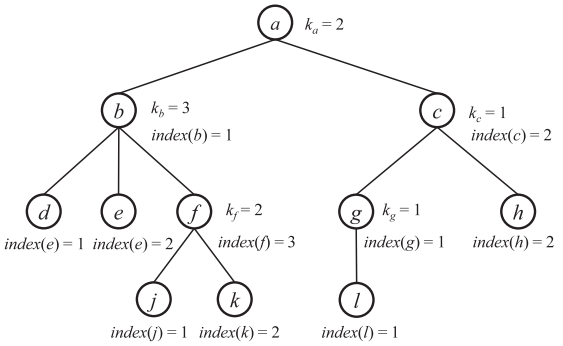


Fig. 2. An example of access tree.

access control policy instead of building a new CP-ABE scheme from scratch. Its key generation procedure is modified for our purpose of removing escrow. The proposed scheme is then built on this new CP-ABE variation by further integrating it into the proxy reencryption protocol for the user revocation.

To handle the fine-grained user revocation, the data-storing center must obtain the user access (or revocation) list for each attribute group, since otherwise revocation cannot take effect after all. This setting where the data-storing center knows the revocation list does not violate the security requirements, because it is only allowed to reencrypt the ciphertexts and can by no means obtain any information about the attribute keys of users. Since the proposed scheme is built on [5], we recapitulate some definitions in [5] to describe our construction in this section, such as access tree, encrypt, and decrypt algorithm definitions.

4.1 Access Tree

4.1.1 Description

Let $\mathcal{T}$ be a tree representing an access structure. Each nonleaf node of the tree represents a threshold gate. If num_x is the number of children of a node x and k_x is its threshold value, then $0 \leq k_x \leq num_x$. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$. λ_x denotes the attribute associated with the leaf node x in the tree. $p(x)$ represents the parent of the node x in the tree. The children of every node are numbered from 1 to num. The function $index(x)$ returns such a number associated with the node x. The index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Fig. 2 shows an example of the access tree structure. In Fig. 2, nodes a, b, f represent AND gates, and node c represents OR gate, respectively.

4.1.2 Satisfying an Access Tree

Let $\mathcal{T}_x$ be the subtree of $\mathcal{T}$ rooted at the node x. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. If x is a nonleaf node, evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of node x. $\mathcal{T}_x(\gamma)$ returns 1 iff at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 iff $\lambda_x \in \gamma$.

4.2 Scheme Construction

Let $\mathbb{G}_0$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_0$. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ denote the bilinear map. A security parameter, κ, will determine the size of the groups. We will also make use of Lagrange coefficients $\Delta_{i,\Lambda}$ for any $i \in \mathbb{Z}_p^*$ and a set, Λ, of elements in $\mathbb{Z}_p^*$: define $\Delta_{i,\Lambda}(x) = \prod_{j \in \Lambda, j \neq i} \frac{x-j}{i-j}$. We will additionally employ two hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ to associate each attribute with a random group element in $\mathbb{G}_0$, and $H_1 : \mathbb{G}_1 \rightarrow \mathbb{Z}_p^*$, which we will model as random oracles.

4.2.1 System Setup

Setup. The trust initializer chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g according to the security parameter. It also chooses hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_0$, $H_1 : \mathbb{G}_1 \rightarrow \mathbb{Z}_p^*$ from a family of universal one-way hash functions. The public parameter $param$ is given by $(\mathbb{G}_0, g, H, H_1)$. For brevity, the public parameter $param$ output by Setup is omitted below.

KKeyGen(). The KGC chooses a random exponent $\beta \in_R \mathbb{Z}_p^*$. It sets $h = g^\beta$. The master public and private key pair is given by $(PK_K = h, MK_K = \beta)$.

DKeyGen(). The data-storing center chooses a random exponent $\alpha \in_R \mathbb{Z}_p^*$. The master public and private key pair is given by $(PK_D = e(g,g)^\alpha, MK_D = g^\alpha)$. The data-storing center also chooses a random exponent $\gamma \in_R \mathbb{Z}_p^*$, and publishes $PK_D^{agree} = g^\gamma$ as another master public key while keep γ as a secret.

4.2.2 Key Generation

The KGC and the data-storing center are involved in the following key generation protocol. For brevity, the knowledge of proofs are omitted below.

$\mathbf{KeyCom}_D(MK_D, ID_t) \leftrightarrow \mathbf{KeyCom}_K(MK_K, ID_t, r_t)$:

1. When the KGC authenticates a user u_t, it selects a random exponent $r_t \in_R \mathbb{Z}_p^*$ for the user. This value is a personalized and unique secret to the user, which should be consistent for any further attribute additions to the user. Then, the KGC and the data-storing center engage in a secure 2PC protocol, where the KGC's private input is (r_t, β), and the data-storing center's private input is α. The secure 2PC protocol returns a private output $x = (\alpha + r_t)\beta$ to the data-storing center. This can be done via a general secure 2PC protocol for a simple arithmetic computation [22], [23]. Alternatively, we can do this more efficiently using the construction in [24].
2. The data-storing center randomly picks $\tau \in_R \mathbb{Z}_p^*$. Then, it computes $A = g^{\frac{x}{\tau}} = g^{\frac{(\alpha+r_t)\beta}{\tau}}$, and sends it to the KGC.
3. The KGC then computes $B = A^{1/\beta^2} = g^{\frac{\alpha+r_t}{\tau\beta}}$, and sends it to the data-storing center.
4. The data-storing center outputs a personalized key component $D = B^\tau = g^{\frac{(\alpha+r_t)}{\beta}}$.

Fig. 3 shows the protocol flows. In each step, PoK represents a proof of knowledge of the secret values used in the computation. The necessary proofs of knowledge for the above statements can be efficiently realized, e.g., via a Schnorr protocol. For simplicity, we have omitted the statement being proved.

Theorem 1. *The above key generation protocol is a secure 2PC protocol for computing $g^{(\alpha+r_t)/\beta}$ by the data-storing center,*

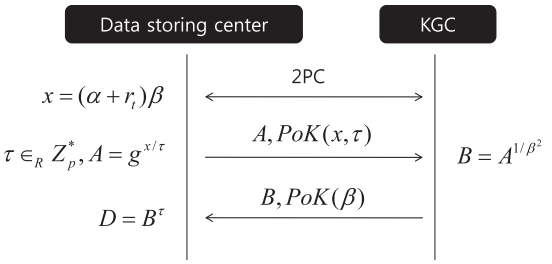


Fig. 3. Key generation protocol.

assuming that the underlying arithmetic 2PC and zero knowledge proofs are secure.

Proof. Proof can be found in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.78, of this paper. □

IssueKey$_K(r_t, S)$. The KGC takes as input a set of attributes S that a user u_t is entitled to have, and outputs a set of attribute keys identified with that set and the personalized secret value r_t. The KGC chooses random $r_j \in_R \mathbb{Z}_p^*$ for each attribute $j \in S$. Then, it computes the attribute keys and outputs them for a user u_t as

$$SK_{K,u_t} = (\forall j \in S : D_j = g^{r_t} \cdot H(j)^{r_j}, D'_j = g^{r_j}).$$

IssueKey$_D()$. The data-storing center takes as input nothing, and outputs a personalized key component SK_{D,u_t} for a user u_t as $SK_{D,u_t} = D$. Then, the user u_t can obtain its whole secret key set as

$$\begin{aligned} SK_{u_t} &= (SK_{D,u_t}, SK_{K,u_t}) \\ &= (D = g^{\frac{(\alpha+r_t)}{\beta}}, \\ &\quad \forall j \in S : D_j = g^{r_t} \cdot H(j)^{r_j}, D'_j = g^{r_j}). \end{aligned}$$

The data-storing center also outputs another KEK $SK_{u_t}^{agree} = H(ID_t)^\gamma = Q_t^\gamma$ for the user, which will be used for selective attribute group key distribution.

4.2.3 Data Encryption

When a data owner wants to upload its data M to the data-storing center for sharing, he defines the tree access structure $\mathcal{T}$ over the universe of attributes $\mathcal{L}$, and encrypts the data under $\mathcal{T}$ by running Encrypt(PK, M, $\mathcal{T}$) algorithm.

Encrypt(PK, M, $\mathcal{T}$). The algorithm chooses a polynomial q_x for each node x in the tree $\mathcal{T}$. These polynomials are chosen in a top-down manner, starting from the root node R.

For each node x in the tree $\mathcal{T}$, the algorithm sets the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. For the root node R, it chooses a random $s \in \mathbb{Z}_p^*$ and sets $q_R(0) = s$. Then, it sets d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{p(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.

Let Y be the set of leaf nodes in the access tree. To encrypt a message $M \in \mathbb{G}_1$ under the tree access structure $\mathcal{T}$, it constructs a ciphertext as

$$\begin{aligned} \mathrm{CT} = \big(\mathcal{T}, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(\lambda_y)^{q_y(0)}\big). \end{aligned}$$

After the construction of CT, the data owner sends it to the data-storing center.

4.2.4 Data Reencryption

Before distributing the ciphertext, the data-storing center reencrypts it by running ReEncrypt(CT, G) using a set of the membership information for each attribute group G that appears in the access tree of CT. The reencryption algorithm enforces user access control per each attribute group.

ReEncrypt(CT, G). The algorithm progresses as follows:

1. For all $G_y \subset G$, chooses a random $K_{\lambda_y} \in \mathbb{Z}_p^*$. Then, reencrypts CT and generates

$$\begin{aligned} \mathrm{CT}' = (\mathcal{T}, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, \\ C'_y = (H(\lambda_y)^{q_y(0)})^{K_{\lambda_y}}). \end{aligned}$$

2. Selects random $\rho, R \in_R \mathbb{Z}_p^*$; and $\forall u_t \in G$, computes $x_t = H_1(e(Q_t^\rho, PK_D^{agree}))$. It is important to note that each x_t can be precomputed in the system setup phase once and for all.
3. For all $G_y \subset G$, constructs the polynomial function $f^y(x) = \prod_{i=1}^m (x - x_i) = \sum_{i=0}^m a_i x^i \pmod p$, where $G_y = \{u_1, \ldots, u_m\}$; and the exponential function $\{P_0, \ldots, P_m\} \equiv \{g^{a_0}, \ldots, g^{a_m}\}$, where m represents the number of users in the attribute group.
4. Constructs $\mathrm{Hdr}_y = \{K_{\lambda_y} \cdot P_0^R, P_1^R, \ldots, P_m^R\}$, and generates a header message

$$\mathrm{Hdr} = (g^\rho, \forall y \in Y : \mathrm{Hdr}_y).$$

On receiving any data request query from a user, the data-storing center responds with (Hdr, CT′) to the user. It is important to note that the attribute group key distribution protocol through Hdr is a stateless approach. Thus, even if users cannot update their key states constantly, they will be able to decrypt the attribute group key from Hdr at any time they receive it, as long as they are not revoked from any of the attribute groups and authorized to decrypt it.

4.2.5 Data Decryption

Decrypt(CT′, SK, K_Λ). Data decryption phase consists of the attribute group key decryption from Hdr by exploiting the one-way anonymous key agreement protocol, followed by the message decryption from CT′.

Attribute group key decrypt. When a user receives the ciphertext (Hdr, CT′) from the data-storing center, he first obtains the attribute group keys for all attributes in Λ that the user holds from Hdr. If a user u_t is associated with an attribute λ_j (that is, $u_t \in G_j$), he can decrypt the attribute group key K_{λ_j} from Hdr_j as follows:

1. Computes $x_t = H_1(e(g^\rho, SK_{u_t}^{agree}))$.
2. Computes $K_{\lambda_j} \cdot P_0^R \cdot \prod_{i=1}^m (P_i^R)^{x_t^i} = K_{\lambda_j} \cdot g^{Rf^j(x_t)} = K_{\lambda_j}$, where $m = |G_j|$.

Then, u_t updates its secret key with the attribute group keys as follows:

$$SK_{u_t} = \Big(D = g^{(\alpha+r_t)/\beta}, \forall \lambda_j \in \Lambda : D_j = g^{r_t} \cdot H(\lambda_j)^{r_j}, D'_j = (g^{r_j})^{\frac{1}{K_{\lambda_j}}}\Big).$$

The above attribute group key computation protocol is secure, since the underlying one-way anonymous key agreement protocol guarantees the key secrecy [16]. Key secrecy means that it is infeasible for anyone other than $u_t \in G_j$ and the data-storing center to determine the secret key x_t generated during the protocol run. It implies that any user $u \notin G_j$ can by no means decrypt K_{λ_j} even if he colludes with other users $u' \notin G_j$. (We assume that $u_t \in G_j$ does not collude with other users $u' \notin G_j$ and give valid K_{λ_j} to u'.)

Message decrypt. After that, the user decrypts CT' with its secret key. The algorithm performs in a recursive way. We first define a recursive algorithm $DecryptNode(\mathrm{CT}', SK, x)$ that takes as inputs a ciphertext CT', a private key SK, which is associated with a set Λ of attributes, and a node x from the tree $\mathcal{T}$. It outputs a group element of $\mathbb{G}_0$ or $\perp$.

Without loss of generality, we suppose that a user u_t performs the decryption algorithm. If x is a leaf node then define as follows: If $\lambda_x \in \Lambda$ and $u_t \in G_x$, then

$$\begin{aligned} &DecryptNode(\mathrm{CT}', SK_{u_t}, x) \\ &= \frac{e(D_x, C_x)}{e(D'_x, C'_x)} = \frac{e(g^{r_t} \cdot H(\lambda_x)^{r_x}, g^{q_x(0)})}{e((g^{r_x})^{\frac{1}{K_{\lambda_x}}}, (H(\lambda_x)^{q_x(0)})^{K_{\lambda_x}})} \\ &= e(g,g)^{r_t q_x(0)}. \end{aligned}$$

If $u_t \notin G_x$, u_t cannot compute the desired value $e(g,g)^{r_t q_x(0)}$, as the exponent of D'_x in SK_{u_t} cannot contain the inverse of the exponent K_{λ_x} of C'_x. If $\lambda_x \notin \Lambda$ or $u_t \notin G_x$, we define $DecryptNode(\mathrm{CT}', SK, x) = \perp$.

We now consider the recursive case when x is a nonleaf node. The algorithm $DecryptNode(\mathrm{CT}', SK, x)$ then proceeds as follows: For all nodes z that are children of x, it calls $DecryptNode(\mathrm{CT}', SK, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \text{ where } \begin{array}{l} i = index(z), \\ S'_x = \{index(z) : z \in S_x\} \end{array} \\ &= \prod_{z \in S_x} \big(e(g,g)^{r_t \cdot q_z(0)}\big)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} \big(e(g,g)^{r_t \cdot q_{p(z)}(index(z))}\big)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} e(g,g)^{r_t \cdot q_x(i) \cdot \Delta_{i,S'_x}(0)} \\ &= e(g,g)^{r_t \cdot q_x(0)} \end{aligned}$$

and return the result.

The decryption algorithm begins by calling the function on the root node R of the access tree. We observe that $DecryptNode(\mathrm{CT}', SK, R) = e(g,g)^{r_t s}$ if the tree $\mathcal{T}$ is satisfied by Λ and the user has valid memberships for each attribute group G_i for all $\lambda_i \in \Lambda$. When we set $A = DecryptNode(\mathrm{CT}', SK, R) = e(g,g)^{r_t s}$, the algorithm decrypts the ciphertext by computing

$$\begin{aligned} \tilde{C}/(e(C,D)/A) &= \tilde{C}/\big(e\big(h^s, g^{(\alpha+r_t)/\beta}\big)/e(g,g)^{r_t s}\big) \\ &= \tilde{C}/\big(e\big(g^{\beta s}, g^{(\alpha+r_t)/\beta}\big)/e(g,g)^{r_t s}\big) \\ &= Me(g,g)^{\alpha s}/e(g,g)^{s\alpha} \\ &= M. \end{aligned}$$

4.3 Key Update

When a user comes to hold or drop an attribute, the corresponding key should be updated to prevent the user from accessing the previous or subsequent encrypted data for backward or forward secrecy, respectively.

The key update procedure is launched by the KGC when it receives a join or leave request for some attribute groups from a user. On receipt of the request, the KGC notifies the data-storing center of the event and sends the updated membership list of the attribute group to it. When the data-storing center receives the notification, it rekeys the corresponding attribute group key. Without loss of generality, suppose there is any membership change in G_i (e.g., a user comes to hold or drop an attribute λ_i at some time instance). Then, the key update procedure progresses as follows:

1. The data-storing center selects a random $s' \in \mathbb{Z}_p^*$, and a K'_{λ_i} which is different from the previous attribute group key K_{λ_i}. Then, it reencrypts the ciphertext using the public parameters PK as

$$\begin{aligned} \mathrm{CT}' &= \big(\mathcal{T}, \tilde{C} = Me(g,g)^{\alpha(s+s')}, C = h^{s+s'}, \\ &C_i = g^{q_i(0)+s'}, C'_i = (H(\lambda_i)^{q_i(0)+s'})^{K'_{\lambda_i}}, \\ &\forall y \in Y \setminus \{i\} : C_y = g^{q_y(0)+s'}, \\ &C'_y = \big(H(\lambda_y)^{q_y(0)+s'}\big)^{K_{\lambda_y}}\big). \end{aligned}$$

   For the other attribute groups that are not affected by the membership changes, the attribute group keys do not necessarily need to be updated.
2. The data-storing center generates a new polynomial function $f^i(x)$ with a new attribute group G_i including a new joining user who comes to hold an attribute λ_i (for backward secrecy) or excluding a leaving user who comes to drop an attribute λ_i (for forward secrecy). Then, it computes a new Hdr_i with K'_{λ_i} and generates a new header message as follow:

$$\mathrm{Hdr} = (g^{\rho}, \mathrm{Hdr}_i, \forall y \in Y \setminus \{i\} : \mathrm{Hdr}_y),$$

where the other Hdr_ys are the same as before. When users send request queries for the shared data afterward, the data storing center would respond with the above Hdr and ciphertext CT' encrypted under the updated keys.

Membership changes in some attribute groups affect all nonrevoked users in the groups, and require them to update their key components. For a user $u' \notin G_i$, u' does not need to update his key component since he is not affected by the membership change in G_i. For a user $u \in G_i$, u can decrypt and obtain a new K'_{λ_i} from the new Hdr message following the attribute group key decryption algorithm in Section 4.2.5. Then, he can update his key component

TABLE 1
Key Escrow and Revocation Comparison

Scheme	Revocation granularity	Key escrow
BSW [5]	timed attribute revocation	yes
BCP-ABE2 [9]	immediate user revocation	yes
YWRL [13]	immediate user revocation	yes
Proposed	immediate user revocation	no

$$D_i' = (g^{r_i})^{\frac{1}{K_{\lambda_i}}}$$

in SK_u as

$$D_i' = \left((g^{r_i})^{\frac{1}{K_{\lambda_i}}}\right)^{\frac{K_{\lambda_i}}{K'_{\lambda_i}}} = (g^{r_i})^{\frac{1}{K'_{\lambda_i}}}.$$

With this newly updated key component, u can decrypt the reencrypted CT$'$ following the message decryption algorithm in Section 4.2.5.

The above key update procedure guarantees the fine-grained user access control, that is an immediate user revocation in each attribute group. It can also trivially achieve the attribute revocation in an immediate way rather than the timed methods in [5], [8], [10] by blinding the attribute key component in the ciphertext with a random secret other than the updated attribute group key.

5 Scheme Analysis

In this section, we analyze and compare the efficiency of the proposed scheme with the previous CP-ABE schemes (that is, Bethencourt et al.'s scheme (BSW) [5], Attrapadung's scheme (BCP-ABE2) [9], and Yu et al.'s scheme (YWRL) [13]) in theoretical and practical aspects. Then, the efficiency of the proposed scheme is demonstrated in the network simulation in terms of the communication cost. We also discuss its efficiency when implemented with specific parameters and compare these results with those obtained by the other schemes.

5.1 Key Escrow and Revocation

Table 1 shows the revocation granularity and key escrow problem of each scheme. The rekeying in the proposed scheme can be done in an immediate way as opposed to BSW. Therefore, a user can be revoked at any time even before the expiration time which might be set to the attribute. This enhances security of the shared data in terms of the backward/forward secrecy by reducing the windows of vulnerability. In addition, the proposed scheme realizes more fine-grained user revocation for each attribute rather than for the whole system. Thus, even if a user drops some attributes during the service in the proposed scheme, he can still access the data with other attributes that he is holding as long as they satisfy the access policy. The proposed scheme also resolves the key escrow problem due to the escrow-free key issuing protocol exploiting secure 2PC protocol as opposed to the other schemes.

5.2 Efficiency

The theoretical efficiency comparison results among the schemes are summarized in Table 2. The notations used in the table are described as follows.

C_0	bit size of an element in $\mathbb{G}_0$
C_1	bit size of an element in $\mathbb{G}_1$
C_p	bit size of an element in $\mathbb{Z}_p^*$
$C_{\mathcal{T}}$	bit size of an access tree $\mathcal{T}$ in the ciphertext
C_k	bit size of a set of attributes associated with private key of a user
t	the number of attributes appeared in $\mathcal{T}$
T	the maximum size allowed for t (in [9])
m	the number of users in an attribute group
r	the number of revoked users
k	the number of attributes associated with private key of a user
K	the maximum size allowed for k (in [9])
u	the size of the attribute universe

In the comparison result, each scheme is compared in terms of ciphertext size, rekeying message size, private and public key size. Ciphertext size implies the communication cost that the data owner needs to send to data-storing center its data, or that the data-storing center needs to send to users (CT$'$ in the proposed scheme). Rekeying message size represents the communication cost that the KGC or the data-storing center needs to send so as to update non-revoked users' keys (Hdr in the proposed scheme) in an attribute group or to revoke an attribute. Private key size represents the storage cost required for each user to store secret keys. Public key size represents the size of the authorities' public keys in the system.

As shown in Table 2, the proposed scheme requires ciphertext size of $(2t+1)C_0 + C_1 + C_{\mathcal{T}}$, which is the same as that of BSW. The proposed scheme requires rekeying message (Hdr) size of $(m+2)C_0$ to realize the user revocation for each attribute in the system. In the proposed scheme, each user stores one more private KEK for decrypting the rekeying messages and obtaining attribute group keys than the basic BSW scheme. YWRL incurs high communication and storage overhead compared to the other schemes in all aspects, of which size are linear to the number of the whole attributes in the system. In YWRL, the KGC should send $2u$ proxy keys to the data server, and $2u$ key components to m users on every revocation in order to reencrypt the ciphertext and prevent any revoked users from decrypting it. Although BCP-ABE2 does not need to send additional rekeying message for user revocations as opposed to the other schemes, it requires ciphertext of

TABLE 2
Efficiency Comparison

System	Ciphertext size	Rekeying message	Private key size	Public key size
BSW [5]	$(2t+1)C_0 + C_1 + C_{\mathcal{T}}$	mC_0	$(2k+1)C_0$	$C_0 + C_1$
BCP-ABE2 [9]	$(t+2r+1)C_0 + C_1 + C_{\mathcal{T}}$	0	$(k+4)C_0$	$(K+T+7)C_0 + C_1$
YWRL [13]	$(u+1)C_0 + C_1 + C_{\mathcal{T}}$	$2umC_0 + 2uC_p$	$C_k + (2u+1)C_0$	$(3u+1)C_0 + C_1$
Proposed	$(2t+1)C_0 + C_1 + C_{\mathcal{T}}$	$(m+2)C_0$	$(2k+2)C_0$	$C_0 + C_1$

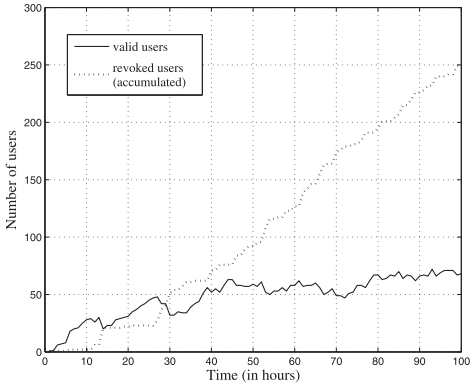


Fig. 4. The number of users in an attribute group.

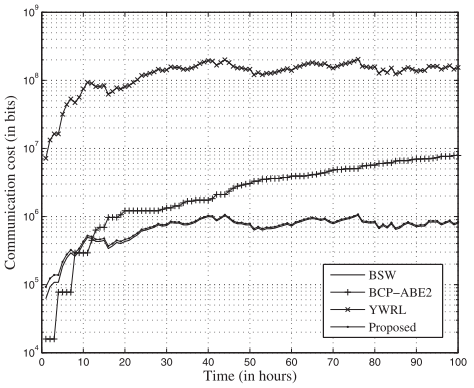


Fig. 5. Communication cost in the system.

which size increases in proportion to the number of revoked users in the system. The proposed scheme is as efficient as BSW in terms of the ciphertext and public key size, while guaranteeing immediate rekeying.

5.3 Simulation

Now, we measure the communication cost of the schemes. In this simulation, we consider the online data sharing system connected into the Internet. Almeroth and Ammar [21] demonstrated the group behavior in the Internet's multicast backbone network (MBone). They showed that the number of users joining a multicast group follows a Poisson distribution with rate $\tilde{\lambda}$, and the membership duration time follows an exponential distribution with a mean duration $1/\mu$. Since each attribute group can be seen as an independent network multicast group where the members of the group share a common attribute, we show the simulation result following this probabilistic behavior distribution [21].

We suppose that user join and leave events are independently and identically distributed in each attribute group in $\mathcal{G}$ following Poisson distribution. The membership duration time for an attribute is assumed to follow an exponential distribution. We set the interarrival time between users as 20 minutes ($\tilde{\lambda} = 3$) and the average membership duration time as 20 hours ($1/\mu = 20$). Fig. 4 represents the number of users in a single attribute group during 100 hours. The solid line and dotted line represent the number of current valid users and accumulated revoked users in an attribute group, respectively.

Fig. 5 shows the total communication costs in log scale that the data owner or the data-storing center needs to send on a membership change in the network system. It includes the ciphertext and rekeying messages for nonrevoked users. It is measured in bits. For a fair comparison with regard to the security perspective, we set the rekeying periods in BSW as $1/\tilde{\lambda}$ minutes. In this simulation, the total number of users in the network is 9,150 and the number of attributes to be updated in the system is $30(= t)$. We set $u = 100$. To achieve an 80-bit security level, we set $C_0 = 512$, $C_p = 160$. C_1 and $C_{\mathcal{T}}$ are not added to the simulation result because they are common in all of the schemes. As it is shown in Fig. 5, YWRL requires the largest amount of communication cost because the rekeying message increases linear to the size of the attribute universe in the whole system. The communication cost in BCP-ABE2 is the lowest in the beginning of the simulation time. However, as the time elapses, it increases conspicuously because the number of revoked users also increases accumulatively. The communication costs in BSW and proposed scheme are almost the same through the time.

5.4 Implementation

Next, we analyze and measure the computation cost for encrypting (by a data owner) and decrypting (by a user) a data. The decryption cost by a user includes the operations for decrypting the rekeying message as well as the data (in [13] and the proposed scheme). We used a Type A curve (in the pairing-based cryptography (PBC) library [20]) providing groups in which a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ is defined. Although such curves provide good computational efficiency (especially for pairing computation), the same does not hold from the point of view of the space required to represent group elements. Indeed each element of $\mathbb{G}_0$ needs 512 bits at an 80-bit security level and 1,536 bits when 128-bit of security are chosen.

Table 3 shows the computational time results. For each operation, we include a benchmark timing. Each cryptographic operation was implemented using the PBC library ver. 0.4.18 [20] on a 3.0 GHz processor PC. The public key

TABLE 3
Comparison of Computation Cost

Operation	Time (ms)	BSW [5]		BCP-ABE2 [9]		YWRL [13]		Proposed scheme	
		owner	user	owner	user	owner	user	owner	user
Pairing	2.9		$2k+1$		$2t+2r+1$		$u+1$		$2k+2$
Exp. in $\mathbb{G}_0$	1.0	$2t+1$		$(K+T+2)t$ $+3r+1$		$u+1$	k	$2t+1$	mk
Exp. in $\mathbb{G}_1$	0.2	1	log$t$	1	$t+r$	1		1	log$t$
Computation (ms)		$2t+1.2$	$5.8k+2.9$ $+0.2\log t$	$(K+T+2)t$ $+3r+1.2$	$6t+6r$ $+2.9$	$u+1.2$	$2.9u+k$ $+2.9$	$2t+1.2$	$(5.8+m)k+$ $0.2\log t+5.8$

parameters were selected to provide 80-bit security level. The implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. The computational cost is analyzed in terms of the pairing, exponentiation operations in $\mathbb{G}_0$ and $\mathbb{G}_1$. The comparatively negligible hash operations are ignored in the time result. In this analysis, we assume that the access tree in the ciphertext is a binary tree (in [5] and the proposed scheme).

Computation costs in Table 3 represent the upper bound of each cost in the worst case. We can see that the total computation time to encrypt data by a data owner in the proposed scheme is the same as BSW; while decryption time by a user may require at most mk exponentiations in $\mathbb{G}_0$ more, especially at the worst case (where all k attribute keys of a user need to be updated). This exponentiation operations are to realize the user revocation for each independent attribute group. Therefore, we can observe that there is a tradeoff between computational overhead and granularity of access control, which is closely related to the windows of vulnerability.

6 Security

In this section, we prove the security of the proposed scheme with regard to the security requirements discussed in Section 2.

6.1 Collusion Resistance

In the ciphertext-policy attribute-based encryption, the secret sharing must be embedded into the ciphertext instead to the private keys of users. Like the previous ABE schemes [3], [5], the private keys (SK) of users are randomized with personalized random values selected by the KGC such that they cannot be combined in the proposed scheme. In order to decrypt a ciphertext, the colluding attacker should recover $e(g,g)^{\alpha s}$. To recover this, the attacker must pair C_y from the ciphertext and D_y from the other colluding users' private keys for an attribute λ_y (we suppose that the attacker does not hold the attribute λ_y). However, this results in the value $e(g,g)^{\alpha s}$ blinded by some random r, which is uniquely assigned to each user, even if the attribute group keys for the attributes that the user holds are still valid.[2] This value can be blinded out if and only if the user has the enough key components to satisfy the secret sharing scheme embedded in the ciphertext. Therefore, the desired value $e(g,g)^{\alpha s}$ cannot be recovered by collusion attack since the blinding value is randomized from a particular user's private key.

6.2 Data Confidentiality

In our trust model, the KGC is no longer fully trusted as well as the data-storing center even if they are honest. Therefore, the plain data to be shared should be kept secret from them as well as from unauthorized users.

Data confidentiality on the shared data against outside users who have not enough attributes can be trivially guaranteed. If the set of attributes of a user cannot satisfy the access tree in the ciphertext, he cannot recover the desired value $e(g,g)^{rs}$ during the decryption process, where r is a random value uniquely assigned to him. On the other hand, when a user is revoked from some attribute groups that satisfy the access policy, he cannot decrypt the ciphertext either unless the rest of the attributes of him satisfy the access policy. In order to decrypt a node x for an attribute λ_x, the user needs to pair C'_x from the ciphertext and D'_x from its private key. However, this cannot result in the value $e(g,g)^{rq_x(0)}$, which is desired to generate $e(g,g)^{rs}$, since C'_x is blinded by the updated attribute group key that the revoked user from the attribute group can by no means obtain (by key secrecy property of the one-way key agreement protocol).

Another attack on the shared data can be launched by the data-storing center and the KGC. Since they cannot be totally trusted by users (suppose that the data-storing center could be compromised and the KGC tries to exploit private user data maliciously for its profit), the confidentiality for the shared data against them is another essential security criteria for secure data sharing. The KGC issues a set of attribute keys, $SK_{K,u}$, to an authenticated user u for the attributes that the user is entitled. The data-storing center issues a user a personalized secret key, $SK_{D,u}$, by performing a secure 2PC protocol with the KGC. As we discussed in Theorem 1, this key generation protocol discourages the two parties to obtain each other's master secret key and determine the secret key issued from each other. Therefore, they could not have enough information to decrypt the data. Even if the data-storing center manages each attribute group key, it cannot decrypt any of the nodes in the access tree in the ciphertext. This is because it is only authorized to reencrypt the ciphertext with each attribute group key, but is not allowed to decrypt it (that is, any of the attribute keys for the corresponding attributes in the ciphertext issued by the KGC are not given to the data-storing center). Therefore, data confidentiality against the honest-but-curious KGC and data-storing center is also guaranteed.

6.3 Backward and Forward Secrecy

When a user comes to hold a set of attributes that satisfy the access policy in the ciphertext at some time instance, the corresponding attribute group keys are updated and delivered to the valid attribute group members securely (including the user). In addition, all of the components encrypted with a secret key s in the ciphertext are reencrypted by the data-storing center with a new secret s', and the ciphertext components corresponding to the attributes are also reencrypted with the updated attribute group keys. Even if the user has stored the previous ciphertext before, he obtains the attribute keys and the holding attributes satisfy the access policy, he cannot decrypt the pervious ciphertext. This is because, even if he can succeed in computing $e(g,g)^{r(s+s')}$ from the current ciphertext, it would not help to recover the desired value $e(g,g)^{\alpha s}$ for the previous ciphertext since it is blinded by a random s'. Therefore, the backward secrecy of the shared data is guaranteed in the proposed scheme.

On the other hand, when a user comes to drop a set of attributes satisfying the access policy in the ciphertext at some time instance, the corresponding attribute group keys are also updated and delivered to the valid attribute group members securely (excluding the user). Then, all of the components encrypted with a secret key s in the ciphertext are reencrypted

2. Another collusion attack scenario is the collusion between revoked users in order to obtain the valid attribute group keys for some attributes that they are not authorized to have (e.g., due to revocation). The attribute group key distribution protocol in the proposed scheme is secure in terms of the (established) key secrecy as we discussed in Section 3.1.5. Thus, the colluding revoked users can by no means obtain any valid attribute group keys for attributes that they are not authorized to hold.

by the data-storing center with a new secret s', and the ciphertext components corresponding to the attributes are also reencrypted with the updated attribute group keys. Then, the user cannot decrypt any nodes corresponding to the attributes after his revocation due to the blindness resulted from newly updated attribute group keys. In addition, even if the user has recovered $e(g,g)^{\alpha s}$ before he was revoked from the attribute groups and stored it, it would not help to determine the desired value $e(g,g)^{\alpha(s+s')}$ in the subsequent ciphertext since it is also reencrypted with a new random s'. Therefore, the forward secrecy of the shared data is also guaranteed in the proposed scheme.

7 Conclusion

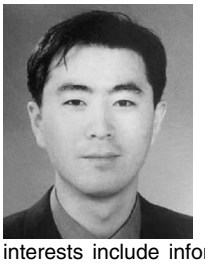
The enforcement of access policies and the support of policy updates are important challenging issues in the data sharing systems. In this study, we proposed a attribute-based data sharing scheme to enforce a fine-grained data access control by exploiting the characteristic of the data sharing system. The proposed scheme features a key issuing mechanism that removes key escrow during the key generation. The user secret keys are generated through a secure two-party computation such that any curious key generation center or data-storing center cannot derive the private keys individually. Thus, the proposed scheme enhances data privacy and confidentiality in the data sharing system against any system managers as well as adversarial outsiders without corresponding (enough) credentials. The proposed scheme can do an immediate user revocation on each attribute set while taking full advantage of the scalable access control provided by the ciphertext policy attribute-based encryption. Therefore, the proposed scheme achieves more secure and fine-grained data access control in the data sharing system. We demonstrated that the proposed scheme is efficient and scalable to securely manage user data in the data sharing system.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013R1A2A2A01005559).

References

[1] J. Anderson, "Computer Security Planning Study," Technical Report 73-51, Air Force Electronic System Division, 1972.
[2] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, and W. Jonker, "Mediated Ciphertext-Policy Attribute-Based Encryption and Its Application," *Proc. Int'l Workshop Information Security Applications (WISA '09)*, pp. 309-323, 2009.
[3] A. Sahai and B. Waters, "Fuzzy Identity-Based Encryption," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (Eurocrypt '05)*, pp. 457-473, 2005.
[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. ACM Conf. Computer and Comm. Security*, pp. 89-98, 2006.
[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. IEEE Symp. Security and Privacy*, pp. 321-334, 2007.
[6] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-Based Encryption with Non-Monotonic Access Structures," *Proc. ACM Conf. Computer and Comm. Security*, pp. 195-203, 2007.
[7] A. Lewko, A. Sahai, and B. Waters, "Revocation Systems with Very Small Private Keys," *Proc. IEEE Symp. Security and Privacy*, pp. 273-285, 2010.
[8] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-Based Encryption with Efficient Revocation," *Proc. ACM Conf. Computer and Comm. Security*, pp. 417-426, 2008.
[9] N. Attrapadung and H. Imai, "Conjunctive Broadcast and Attribute-Based Encryption," *Proc. Int'l Conf. Palo Alto on Pairing-Based Cryptography (Pairing)*, pp. 248-265, 2009.
[10] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure Attribute-Based Systems," *Proc. ACM Conf. Computer and Comm. Security*, 2006.
[11] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309-329, 2003.
[12] P. Golle, J. Staddon, M. Gagne, and P. Rasmussen, "A Content-Driven Access Control System," *Proc. Symp. Identity and Trust on the Internet*, pp. 26-35, 2008.
[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute Based Data Sharing with Attribute Revocation," *Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS '10)*, 2010.
[14] S.D.C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-Encryption: Management of Access Control Evolution on Outsourced Data," *Proc. Int'l Conf. Very Large Data Bases (VLDB '07)*, 2007.
[15] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO)*, pp. 213-229, 2001.
[16] A. Kate, G. Zaverucha, and I. Goldberg, "Pairing-Based Onion Routing," *Proc. Privacy Enhancing Technologies Symp.*, pp. 95-112, 2007.
[17] L. Cheung and C. Newport, "Provably Secure Ciphertext Policy ABE," *Proc. ACM Conf. Computer and Comm. Security*, pp. 456-465, 2007.
[18] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded Ciphertext Policy Attribute-Based Encryption," *Proc. Int'l Colloquium Automata, Languages and Programming (ICALP)*, pp. 579-591, 2008.
[19] X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably Secure and Efficient Bounded Ciphertext Policy Attribute Based Encryption," *Proc. Int'l Symp. Information, Computer, and Comm. Security (ASIACCS)*, pp. 343-352, 2009.
[20] The Pairing-Based Cryptography Library, http://crypto.stanford.edu/pbc/, 2012.
[21] K.C. Almeroth and M.H. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone (MBone)," *IEEE Comm. Magazine*, vol. 35, no. 6, pp. 124-129, June 1997.
[22] M. Chase and S.S.M. Chow, "Improving Privacy and Security in Multi-Authority Attribute-Based Encryption," *Proc. ACM Conf. Computer and Comm. Security*, pp. 121-130, 2009.
[23] S.S.M. Chow, "Removing Escrow from Identity-Based Encryption," *Proc. Int'l Conf. Practice and Theory in Public Key Cryptography (PKC '09)*, pp. 256-276, 2009.
[24] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Hysyanskaya, and H. Shacham, "Randomizable Proofs and Delegatable Anonymous Credentials," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '09)*, pp. 108-125, 2009.



Junbeom Hur received the BS degree in computer science from Korea University in 2001, and the MS and PhD degrees in computer science from KAIST in 2005 and 2009, respectively. He was with the University of Illinois at Urbana-Champaign as a post-doctoral researcher from 2009 to 2011. He is currently an assistant professor in the School of Computer Science and Engineering at the Chung-Ang University in Korea. His research interests include information security, mobile computing security, and cryptography.