This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Secure and Efficient Attribute-Based Access Control for Multiauthority Cloud Storage

Jianghong Wei, Wenfen Liu, and Xuexian Hu

***Abstract*—Cloud storage facilitates both individuals and enterprises to cost effectively share their data over the Internet. However, this also brings difficult challenges to the access control of shared data since few cloud servers can be fully trusted. Ciphertext-policy attribute-based encryption (CP-ABE) is a promising approach that enables the data owners themselves to place fine-grained and cryptographically-enforced access control over outsourced data. In this paper, we present secure and cost-effective attribute-based data access control for cloud storage systems. Specifically, we construct a multiauthority CP-ABE scheme that features: 1) the system does not need a fully trusted central authority, and all attribute authorities independently issue secret keys for users; 2) each attribute authority can dynamically remove any user from its domain such that those revoked users cannot access subsequently outsourced data; 3) cloud servers can update the encrypted data from the current time period to the next one such that the revoked users cannot access those previously available data; and 4) the update of secret keys and ciphertext is performed in a public way. We show the merits of our scheme by comparing it with the related works, and further implement it to demonstrate its practicality. In addition, the proposed scheme is proven secure in the random oracle model.**

***Index Terms*—Access control, cloud storage, multiauthority ciphertext-policy attribute-based encryption (CP-ABE), public update, revocation.**

I. Introduction

CLOUD storage is one of the major services provided by cloud computing. It enables data owners to remotely host their data by outsourcing them to cloud servers, which brings great convenience for both individuals and enterprises to share data over the Internet. However, this new paradigm challenges the approaches of traditional data access control scenarios, where a fully trusted server is in charge of implementing access control mechanisms, since the outsourced data (e.g., patients' health records and enterprises' development projects) might be sensitive and valuable for data owners, and few cloud servers can be fully trusted. Thus, to protect the security of outsourced data, data owners would like to place access policies over their data before outsourcing them to cloud servers.

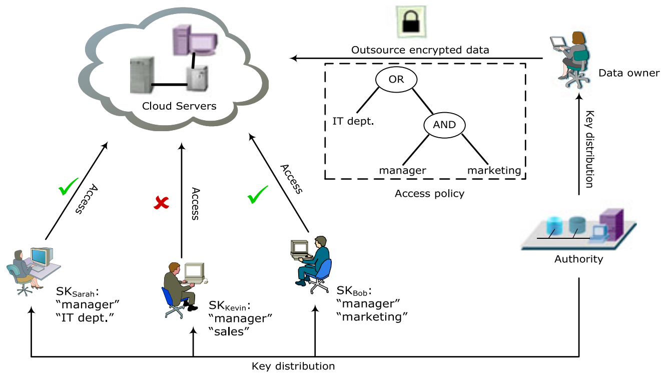


Fig. 1. Setting of attribute-based access control.

Among various solutions suggested for securing data sharing in cloud storage systems, attribute-based access control, which employs ciphertext-policy attribute-based encryption (CP-ABE), is rather promising. In this setting, each user is described by a set of personal attributes and holds a secret key issued by an authority according to his/her attributes. A data owner defines an access policy over attributes and encrypts the data to be outsourced under this policy. Consequently, after outsourcing the encrypted data to cloud servers, only those users whose attributes satisfy the access policy can decrypt the outsourced data. This effectively prevents unauthorized users (including cloud servers) from accessing the outsourced data. For example, as indicated in Fig. 1, Sarah and Bob can access the outsourced data since their attributes satisfy the access policy, but Kevin will fail to access the outsourced data since his attributes do not satisfy the access policy. Depending on the fact that attributes are managed by a single authority or different authorities, CP-ABE falls into two categories: single-authority CP-ABE and multiauthority CP-ABE. From the perspective of practicability, the latter is of course more desirable for cloud storage systems.

However, the original multiauthority CP-ABE cannot be directly deployed in cloud storage systems because of the following two issues. First, the public parameter of these schemes is dependent on the attribute universe, which implies that when the system completes initialization, the attribute universe is completely fixed. But in practical applications, the requirement of adding new attributes into the system is very common. For example, an enterprise might establish a new department and

Manuscript received June 27, 2016; revised October 8, 2016; accepted November 26, 2016. This work was supported in part by the National Nature Science Foundation of China under Grant 61502527 and Grant 61379150 and in part by the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2016-2-22.

J. Wei is with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China, and also with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100088, China (e-mail: jianghong.wei.xxgc@gmail.com).

W. Liu and X. Hu are with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China (e-mail: wenfenliu@sina.com; xuexian_hu@hotmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org. This includes a security proof of a theorem in this paper. This material is 96 kB in size.

Digital Object Identifier 10.1109/JSYST.2016.2633559

1937-9234 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the employees attached to this department will possess new attributes different from available system attributes. Second, in a cloud storage system, system users would dynamically join or leave. This requires that these users' authorization should also be changed correspondingly. Specifically speaking, the leaving users not only should be prevented from accessing those outsourced data that were previously accessible for them (forward security), but also cannot access those subsequently outsourced data (backward security), even if their attributes satisfy the corresponding access policies. In this study, we will try to overcome these issues in cloud storage systems by using multiauthority CP-ABE.

A. Related Work

Sahai and Waters [1] first put forward the notion of ABE that provides fine-grained access control over encrypted data. Goyal *et al.* [2] further divided it into two complementary types: key-policy ABE (KP-ABE) and CP-ABE. In the setting of KP-ABE, secret keys are associated with access policies and encrypted data are described by attributes, whereas CP-ABE has an opposite scenario. Due to their different characteristics, KP-ABE is widely applied to query applications such as subscription-video services and database access, and CP-ABE is quietly suitable for access control applications, e.g., E-health system and social network.

The revocation problem in ABE comes into two categories: attribute revocation and user revocation. The first one considers the case that the number of a user's attributes might decrease with time, and focuses on how to dynamically limit these users' permissions. The second one covers the case that users might leave the system, and is aimed at directly abolishing revoked users' permissions in the system. Obviously, user revocation can be achieved by revoking all of his/her attributes. However, this solution is rather inefficient as we will demonstrate below.

In early works [3], [4], the authors realized attribute revocation by associating each attribute with a time frame within which it is valid. For example, the attribute "*Manager-December 31st 2015*" indicates that the uselessness of this attribute expires at the end of 2015. Obviously, such a kind of revocation manner cannot duly reflect the changes of users' permissions since a revoked user can still access those previously encrypted data until the lifecycle of his/her attributes expires. On the other hand, the authority needs to periodically update the public parameter of the system and issue secret keys for nonrevoked users, which brings tremendous workload for the authority.

Yu *et al.* [5] proposed an immediate attribute revocation method by integrating the technique of proxy re-encryption with CP-ABE. Their scheme can only deal with the simplest access policies ''AND" gate. Most recently, Chow [6] introduced a general framework of multiauthority CP-ABE with outsourcing and attribute revocation. In this framework, the author utilizes the same manner to achieve attribute revocation as in [5]. Specifically speaking, the authority immediately generates an update key when a user's some attributes are revoked, and sends it to each nonrevoked user in a secure way. However, this implies that a malicious nonrevoked user can share the update key with a revoked user. As a consequence, the revoked user can also correctly update his/her secret key. Hur and Noh [7] utilized another manner, selective group key distribution, to achieve scalable attribute revocation. In their scheme, they assumed that cloud servers are fully trusted, which do not meet the real situation. Fan *et al.* [8] proposed an arbitrary-state ABE scheme with dynamic membership. In this scheme, each user can update the values of his/her attributes by interacting with the authority. Meanwhile, by updating the public parameter, this scheme also achieves attribute-level revocation.

Yang *et al.* [9] constructed a multiauthority CP-ABE with attribute revocation for cloud storage systems to improve the efficiency of previous schemes [10], [11]. Unfortunately, Hong *et al.* [12] found that this scheme cannot resist collusion attacks, that is, a revoked user can still decrypt subsequently encrypted data. Chen and Ma [13] proposed a new scheme to fix this pitfall. In their scheme, the update key is the same for all nonrevoked users. Thus, a nonrevoked user can share it with a revoked user, and their scheme naturally cannot withstand collusion attacks. Yang and Jia [14] yet proposed another more efficient scheme. However, this scheme also suffers from collusion attacks, which means it cannot provide forward and backward security. Li *et al.* [15] recently proposed a multiauthority CP-ABE scheme supporting attribute revocation and decryption outsourcing, and proved the adaptive security of the proposed scheme in the setting of bilinear groups with composite order. Focusing on attribute revocation, this scheme utilizes a private manner to update nonrevoked users' private keys. In addition, there are several works [16], [17] that consider the problem of attribute revocation in hierarchical single-authority CP-ABE.

To summarize, the above-mentioned approaches for attribute revocation (except Li *et al.*'s [15]) all require the authority to publish new public parameter of the system when users' attributes are revoked. In addition, to update the encrypted data, either the data owner or the authority has to generate an update key by using their secret keys. The involvement of secret keys in this process makes this process troublesome and exposes the whole system to problematic new vulnerabilities. In general, we want to limit the use of secret keys to only decryption (for user) or key generation (for authority) and not to database upkeep.

Liang *et al.* [18] constructed the first single-authority CP-ABE with efficient user revocation. However, their scheme does not enjoy forward security, that is, a revoked user can still decrypt previously accessible data. Sahai *et al.* [19] provided a generic construction of single-authority CP-ABE supporting user revocation and ciphertext update, which provides both forward and backward security. Their construction is built upon composite order (product of three primes) bilinear groups, and thus has a lower implementation efficiency than prime order groups implementation. To illustrate this point, we give the benchmark results of the main group operations (exponentiation and pairing) in Table I. These experiments are conducted with PBC 0.5.14[1] under the platform introduced in Section V. We can see that there is a significant efficiency gap between the two implementations for the same security level. In addition, we note that the above two schemes cannot withstand decryption key exposure attack [20], a realistic attack against revocable

[1][Online]. Available: http://crypto.stanford.edu/pbc/

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
AVERAGE TIMING RESULTS OF OPERATIONS IN PRIME AND COMPOSITE ORDER GROUPS (UNIT: ms)

Operations	Prime order	Composite order	Ratios
Pairing	5.12	421.43	82.3
Exp. in $\mathbb{G}$	4.21	173.08	42.0
Exp. in $\mathbb{G}_T$	0.62	31.93	51.5

cryptographic primitives, that is, when a user's decryption key is compromised, the adversary can extract the secret key from it and further combine it with subsequent update keys to access the encrypted data.

B. Our Contribution

In this paper, we construct a novel multiauthority CP-ABE as a core building of the proposed multiauthority attribute-based data access control for cloud storage systems. Specifically, our scheme has the following merits.

1) Our scheme supports scalable and dynamic user revocation, namely, the overhead of revocation is linear in the logarithm of the number of revoked users, which is more efficient than the revocation mechanisms in [3]–[5], [9], and [14], within which the workload of revocation is linear in the number of users possessing those nonrevoked attributes.
2) Throughout the life cycle of our system, the system public parameter remains unchanged, regardless of removing a user from the system or adding new attributes into the system.
3) Our scheme enjoys the desired property of public update in terms of secret key and ciphertext. First, the update key generated by the authority is available to all users, but only nonrevoked users can utilize it to update their secret keys. Second, the cloud server can update the ciphertext by just using the public parameter of the system, without the help of the data owner and the authority.
4) With regard to security, our scheme provides both forward security and backward security, that is, a revoked user cannot access those encrypted data that were previously accessible for him/her, and also cannot decrypt those subsequently encrypted data. In addition, we formally prove the security of the proposed multiauthority CP-ABE scheme.
5) Our scheme is built upon prime order bilinear groups, which enables the system to be efficient enough for practical applications. To show its practicality, we implement it in Charm,[2] a framework for rapid prototyping of cryptographic primitives.

C. Outline

This paper is structured as follows. In Section II, we introduce the proposed framework of attribute-based data access control for cloud storage systems. In Section III, we review the preliminaries used throughout this paper. In Section IV, we present a concrete construction of multiauthority CP-ABE that achieves the intended security requirements of the proposed framework. In Section V, we implement the proposed scheme to show its practicability. In Section VI, we discuss the possible extensions of the proposed scheme. The conclusion is given in Section VII. In the supplementary file,[3] we provide the corresponding security model, and formally prove the security of the proposed scheme.

II. FRAMEWORK OF OUR SYSTEM

A. Overview

In this section, we provide a high-level description of our framework of multiauthority attribute-based data access control for cloud storage systems. The system involves the following entities.

1) *Semitrusted third party:* It is just in charge of producing the global public parameter of the system, which is shared among all authorities and users in the system. Particularly, it does not keep any secret key, and also does not generate any secret keys for authorities and users. Thus, it makes no difference on the security of the system.
2) *Attribute authority:* Each attribute authority independently manages its own attribute universe and sets up its own public parameter and master secret key. It is responsible for checking the validity of a user's attributes belonging to its domain. If yes, it issues a secret key component to the user according to his/her attributes. In addition, it is also in charge of periodically generating an update key for users that are not revoked in its domain. In our system, any string can be an attribute, and each attribute belongs to only one authority.
3) *Data owner:* The data owner is an entity that owns data and would like to share his/her data by outsourcing them to cloud servers managed by cloud service providers. A data owner first defines an intended access policy over attributes, and enforces it over the data by calling the proposed multiauthority CP-ABE scheme. Then, the owner sends the encrypted data to cloud servers.
4) *User:* Each user owns a unique global identifier in the system, and possesses a set of attributes and the corresponding secret key, which consists of all secret key components issued by different attribute authorities (AAs). If a user is not revoked by an authority at some time period, he/she can utilize the published update key to update the corresponding secret key component.
5) *Semitrusted cloud server:* A cloud server is in charge of storing and updating those encrypted data from data owners. Here, we emphasize that the update procedure can be done by just using the public parameter of the system (including the global public parameter and the public parameter of all authorities), without the involvement of secret information. As in previous works [5], [7], [9], we assume that it is semitrusted (curious but honest), namely, it will honestly perform the valid assignments, but will attempt to learn information about the outsourced data as much as possible.

[2][Online]. Available: http://charm-crypto.com

[3]Supplemental material for the reader can be downloaded online at http://ieeexplore.ieee.org/

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

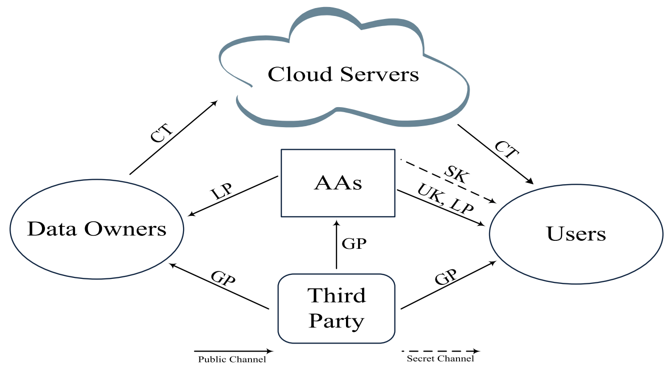


Fig. 2. Our framework of multiauthority attribute-based data access control for cloud storage systems.

B. Details of the Framework

In our framework, there are a third party, a collection $\mathcal{A}$ of AAs, multiple data owners, and multiple users. The core building of the framework is our multiauthority CP-ABE scheme described in later section. As indicated in Fig. 2, this framework is comprised of the following phases.

System setup: Initially, the third party and each authority set up the public parameter of the system by running the following algorithms.

$\mathsf{GlobalSetup}$ $(\lambda) \rightarrow \mathrm{GP}$: This algorithm, which takes as input a security parameter λ, is executed by the third party. It outputs the global public parameter GP of the system, which is available for all AAs and users in the system.

$\mathsf{AuthSetup}$ $(\mathrm{GP}, T, N) \rightarrow (\mathrm{LP}_\delta, \mathrm{MSK}_\delta, \mathrm{RL}_\delta, \mathrm{st}_\delta)$: This algorithm, which takes as input the global public parameter GP, the total number of system time periods T, and the number of system users, is run by each attribute authority $\delta \in \mathcal{A}$. It produces the local public parameter LP_δ and the master secret key MSK_δ of this authority. In addition, it creates a revocation list RL_δ initialized as an empty set and the state information st_δ used to manage users' identifiers.

The public parameter of the overall system is in the form of $\mathrm{PP} = \{\mathrm{GP}, \{\mathrm{LP}_\delta\}_{\delta \in \mathcal{A}}\}$. We note that the two algorithms do not take the attribute universe as input, which implies that the public parameter of the system is independent of the attribute universe, and any string can be an attribute.

Key distribution: For a user with a unique global identifier GID and an attribute set $S = \bigcup_{\delta \in \mathcal{A}} S_\delta$, where S_δ is a subset of attributes belonging to the authority δ, he/she would require a secret key component $\mathrm{SK}_{\mathrm{GID}, S_\delta}$ from each authority $\delta \in \mathcal{A}$. Each attribute authority δ generates the corresponding secret key component for such a user by carrying out the following algorithm.

$\mathsf{SKeyGen}$ $(\mathrm{PP}, \mathrm{MSK}_\delta, \mathrm{GID}, S_\delta, \mathrm{st}_\delta) \rightarrow \mathrm{SK}_{\mathrm{GID}, S_\delta}$: This algorithm takes as input the public parameter PP (GP and LP_δ), the master secret key MSK_δ, the user's identifier GID, the corresponding attribute set S_δ, and the current state information st_δ. The algorithm first checks the validity of the user's identifier and attributes. If they pass through the verification, it generates a secret key component $\mathrm{SK}_{\mathrm{GID}, S_\delta}$ for the user, and also outputs an updated state information st_δ.

Then, the authority δ would send the secret key component to the user through a secure way. After getting all secret key components from all AAs, the user structures his/her complete secret key in the form of $\mathrm{SK}_{\mathrm{GID}, S} = \{\mathrm{SK}_{\mathrm{GID}, S_\delta}\}_{\delta \in \mathcal{A}}$.

Data encryption: Before data owners upload their data to a cloud server, they first define access policies and implement them over these data by calling the following encryption algorithm.

$\mathsf{Encrypt}$ $(\mathrm{PP}, \mathbb{A}, file, t) \rightarrow \mathrm{CT}_t$: This algorithm takes as input the system public parameter PP, an access policy $\mathbb{A}$, the data file $file$ to be encrypted, and a time period t. It outputs a ciphertext CT_t of $file$ at the time period t. We will assume that $\mathbb{A}$ and t are implicitly attached to the ciphertext.

Update key generation: Each attribute authority δ periodically produces an update key such that those nonrevoked users can utilize it to update the corresponding secret key component $\mathrm{SK}_{\mathrm{GID}, \delta}$. Specifically, the attribute authority runs the following algorithm to accomplish the above assignment.

$\mathsf{UKeyGen}$ $(\mathrm{PP}, \mathrm{MSK}_\delta, \mathrm{RL}_\delta, \mathrm{st}_\delta, t) \rightarrow \mathrm{UK}_{\delta, t}$: This algorithm takes as input the public parameter PP (GP and LP_δ), the master secret key MSK_δ of the authority, the current revocation list RL_δ, and the update time period t. The algorithm produces an update key component $\mathrm{UK}_{\delta, t}$, which is available to all users. However, it is only useful for nonrevoked users (i.e., users that do not appear in RL_δ).

The update key UK_t of the whole system at the time period t consists of all update key components generated by these AAs, i.e., $\mathrm{UK}_t = \{\mathrm{UK}_{\delta, t}\}_{\delta \in \mathcal{A}}$.

Decryption key generation: When the update key is published, those nonrevoked users need to utilize it to update their secret keys. It is a considerable scenario that a user might be revoked by partial authorities. For a system user, we denote by $\mathcal{A}_n \subseteq \mathcal{A}$ the set of authorities that have not removed the user from their domains. The update procedure is executed in a piecemeal manner, as in the key generation process, that is, for each attribute authority $\delta \in \mathcal{A}_n$, a nonrevoked user runs the following algorithm.

$\mathsf{DKeyGen}$ $(\mathrm{PP}, \mathrm{SK}_{\mathrm{GID}, S_\delta}, \mathrm{UK}_{\delta, t}) \rightarrow \mathrm{DK}^t_{\mathrm{GID}, S_\delta}$: This algorithm takes as input the public parameter PP (GP and LP_δ), the secret key component $\mathrm{SK}_{\mathrm{GID}, S_\delta}$, and the update key component $\mathrm{UK}_{\delta, t}$. It derives a decryption key component $\mathrm{DK}^t_{\mathrm{GID}, S_\delta}$ for the nonrevoked user.

Finally, the user would hold a decryption key in the form of $\{\mathrm{DK}^t_{\mathrm{GID}, S_\delta}\}_{\delta \in \mathcal{A}_n}$.

Ciphertext update: By using the public parameter, the cloud server periodically updates the encrypted data. It finishes this task by calling the following algorithm.

$\mathsf{CTUpdate}$ $(\mathrm{PP}, \mathrm{CT}_t, t') \rightarrow \mathrm{CT}_{t'}$: This algorithm takes as input the public parameter PP, a ciphertext CT_t, and a new time period $t' > t$. It updates the original ciphertext to the new time period, and outputs a new ciphertext $\mathrm{CT}_{t'}$, whereas, the original ciphertext CT_t is erased.

User revocation: When a user's permission authorized by an authority δ expires, the authority δ removes him/her from its domain. It achieves this by running the following algorithm.

Revoke $(\mathrm{GID}, \mathrm{RL}_\delta, \mathrm{st}_\delta, t) \rightarrow \mathrm{RL}_\delta$: This algorithm takes as input the user's identifier GID to be revoked, the current revocation list RL_δ, and state information st_δ as well as a revocation time t. It outputs an updated revocation list.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Data decryption: In case that a user's attributes satisfy the access policy of the outsourced data, he/she can download the data and then decrypt it by running the following algorithm.

Decrypt $(\mathrm{PP}, \mathrm{CT}_t, \mathrm{DK}^t_{\mathrm{GID},S})$: This algorithm takes as input the public parameter PP, a ciphertext CT_t, and the decryption key $\mathrm{DK}^t_{\mathrm{GID},S}$. It recovers the original data file $file$ shared by the data owner.

III. Preliminaries

A. Notation

For a natural number $n \in \mathbb{N}$, we define $[n] = \{1, 2, \ldots, n\}$. Similarly, for natural numbers $n_1, n_2, \ldots, n_k \in \mathbb{N}$, we let $[n_1, n_2, \ldots, n_k] = [n_1] \times [n_2] \times \cdots \times [n_k]$. Denote by $\mathbb{Z}_p^{l\times n}$ the set of $l \times n$ matrices with elements in $\mathbb{Z}_p$. Given a row vector $\vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_p^n$, $\vec{x}^\top$ means its transform, a column vector. For a binary string $b \in \{0,1\}^l$, denote by $|b|$ its length, and $b[j]$ $(j \le l)$ its jth bit. For a time period $t \in [T]$, denote by $t[j]$ the jth bit of the binary expression of t. PPT means probabilistic polynomial time.

B. Cryptographic Background

In this section, we briefly review several concepts involved in the construction of our multiauthority CP-ABE scheme, and a complexity assumption employed to prove the security of our scheme.

Bilinear mappings: Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic (multiplicative) groups with prime order p. A bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a mapping with the following properties.

1) *Bilinearity:* For any $g \in \mathbb{G}$ and any $\alpha, \beta \in \mathbb{Z}_p$, $e(g^\alpha, g^\beta) = e(g,g)^{\alpha\beta}$.
2) *Nondegeneracy:* For a generator g of $\mathbb{G}$, $e(g,g) \neq 1_{\mathbb{G}_T}$.
3) *Computability:* For any $g \in \mathbb{G}$ and any $\alpha, \beta \in \mathbb{Z}_p$, there exists a polynomial-time algorithm to compute $e(g^\alpha, g^\beta)$.

Access structures: We slightly modify the definition of access structures introduced in [21] to match our setting.

Definition 1 (Access structure [21]): Let $\mathcal{U}$ be the attribute universe. An access structure defined over $\mathcal{U}$ is a collection $\mathbb{A}$ of nonempty attribute subsets, i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets. In addition, an access structure $\mathbb{A}$ is called monotone, if $\forall B, C \in \mathbb{A}$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$.

In this paper, we only consider monotone access structures. From now on, by an access structure we will refer to a monotone one.

Linear Secret-Sharing Schemes (LSSS): As proved in [21], a monotone access structure can be realized by an LSSS, which is defined as follows.

Definition 2 (LSSS [21]): Let $\mathcal{U}$ be the attribute universe and p a prime. A secret-sharing scheme Π realizing access structures on $\mathcal{U}$ is called linear over $\mathbb{Z}_p$, if the following conditions hold.

1) The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.
2) For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a matrix $A \in \mathbb{Z}_p^{l\times n}$ called the share-generating matrix. For $i = 1$ to l, the ith row A_i of A is labeled by an attribute $\varphi(i) \in \mathcal{U}$, where φ is a function defined as $\varphi : [l] \to \mathcal{U}$. When we consider the vector $\vec{v} = (s, s_2, \ldots, s_n)$, where $s_2, \ldots, s_n$ are random chosen from $\mathbb{Z}_p$, then $A\vec{v}^\top$ is the vector of l shares of the secret s according to A. The share $\lambda_i = A_i\vec{v}^\top$ where $i \in [l]$ belongs to the attribute $\varphi(i)$.

Each LSSS according to the above definition enjoys the *linear reconstruction* property, that is, given an LSSS Π for the access structure $\mathbb{A}$, if $S \in \mathbb{A}$, then there exist constants $\{\xi_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} \xi_i\lambda_i = s$, where $I = \{i | i \in [l] \wedge \varphi(i) \in S\}$ and $\{\lambda_i\}_{i\in I}$ are valid shares of s. On the other hand, for any set $S \notin \mathbb{A}$ no such constants $\{\xi_i\}$. Moreover, there will exist a vector $\vec{w} \in \mathbb{Z}_p^n$ such that $\vec{w} \cdot (1, 0, \ldots, 0)^\top = 1$ and $A_i\vec{w}^\top = 0$ for all $i \in I$.

Complexity assumption: The security of our multiauthority CP-ABE scheme is built upon a q-type assumption, which is a slightly modified version of q-decisional parallel bilinear Diffie–Hellman exponent (q-DPBDHE) [22]. We denote by such an assumption q-DPBDHE-v. Rouselakis and Waters [23], proved the generic security of this assumption.

Choose a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ according to the security parameter, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p. Let g be a random generator of $\mathbb{G}$, randomly select $s, a, b_1, \ldots, b_q \in \mathbb{Z}_p$. Given a tuple D consists of the following terms:

$$g, g^s;\ g^{a^i},\ g^{b_j a^i}\ \forall (i,j) \in [2q, q] \text{ with } i \neq q+1$$

$$g^{s/b_i}\ \forall i \in [q];\ g^{s a^i b_j / b_{j'}}\ \forall (i,j,j') \in [q+1, q, q] \text{ with } j \neq j'.$$

The q-DPBDHE-v assumption states that it must be intractable for a PPT adversary to distinguish the challenge term $e(g,g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element $R \in \mathbb{G}_T$. An algorithm $\mathcal{B}$ that outputs a bit $\theta \in \{0,1\}$ has advantage ϵ of solving the q-DPBDHE-v problem in $\mathbb{G}$ provided that

$$\left| \Pr[\mathcal{B}(D, e(g,g)^{a^{q+1}s}) = 0] - \Pr[\mathcal{B}(D, R) = 0] \right| \ge \epsilon.$$

Definition 3 (q-DPBDHE-v assumption [23]): We say that the q-DPBDHE-v assumption holds if for any PPT algorithm, its advantage of solving the q-DPBDHE-v problem is negligible in the security parameter.

C. KUNodes Algorithm

To achieve scalable user revocation, we mainly follow Boldyreva *et al.*'s [24] strategy that defines a KUNode algorithm by using a binary. For convenience, we first give several notations. Let $\mathcal{BT}$ be a binary tree with N leaf nodes and root be the root node. For each nonleaf node θ, denote by θ_l and θ_r its left and right child, respectively. Each user is associated with a leaf node η of $\mathcal{BT}$, and denote by Path(η) the set of all nodes over the path from root to η (including root and η).

The KUNode algorithm takes as input a binary tree $\mathcal{BT}$, a revocation list RL consisting of two tuples recorded in the form of (η_i, t_i) as well as a time period t. It outputs a minimal set Y of nodes of $\mathcal{BT}$ such that for any leaf node η listed in RL, it holds that Path$(\eta) \cap \mathsf{Y} = \emptyset$. On the other hand, for each nonrevoked leaf node there exactly exists a node $\theta \in \mathsf{Y}$ such that θ is an ancestor of this leaf node. For an exemplary purpose, we give two instances of this algorithm as indicated in Fig. 3.

Below we give the details of this algorithm.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

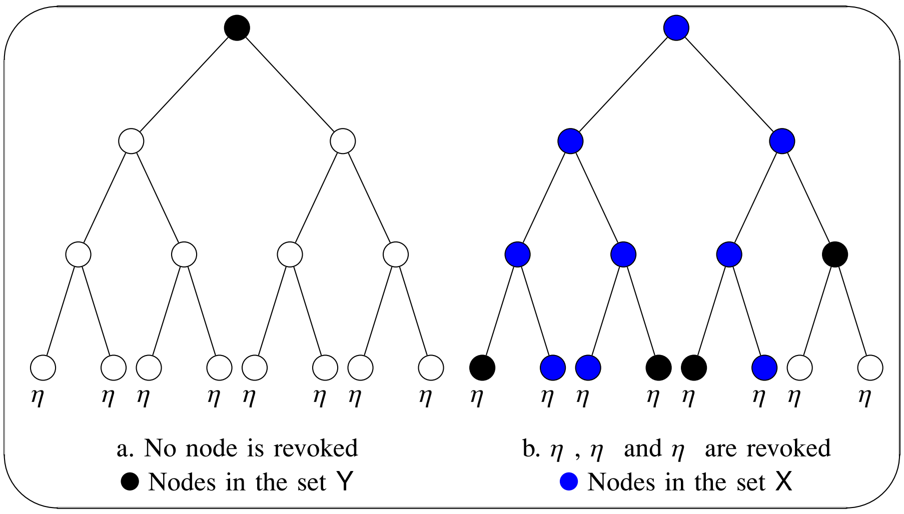


Fig. 3. Two instances of the KUNodes algorithm.

```
KUNode $(\mathcal{BT}, RL, t)$:
        X, Y$\leftarrow \emptyset$
        $\forall(\eta_i, t_i) \in RL$
                if $t_i \leq t$ then add Path$(\eta_i)$ to X
        $\forall\theta \in$ X
                if $\theta_l \notin$ X then add $\theta_l$ to Y
                if $\theta_r \notin$ X then add $\theta_r$ to Y
        If Y $= \emptyset$ then add root to Y
        Return Y
```

IV. Our Multiauthority CP-ABE Scheme

In this section, we give the core building of our data access control scheme for cloud storage systems, a multiauthority CP-ABE scheme supporting efficient user revocation, and public ciphertext update.

Our technique: A desirable multiauthority CP-ABE scheme should have the following features.

1) The public parameter should remain unchanged, and any string can be an attribute.
2) The revoked users can no longer decrypt those previously accessible data and subsequently encrypted data, namely, it should provide forward security and backward security simultaneously.
3) The ciphertext update should not involve secret information.

To this end, we build our scheme upon large attribute universe CP-ABE schemes [23], [25], [26], and utilize a binary tree to manage users' identifiers. Meanwhile, the lifetime of the system is divided into numerous discrete time periods, and each ciphertext is associated with a time period. We employ another binary tree in a different way to handle these time periods such that the ciphertext can be publicly updated from the current time period to the next one.

Specifically speaking, each attribute authority δ maintains a binary tree $\mathcal{BT}_\delta$ with N leaf nodes. When a user with identifier GID requires for a secret key component from δ, the authority δ will assign an unused leaf node η to GID. For each node $\theta \in \mathsf{Path}(\eta)$, the authority δ would generate a key component SK_θ and then return all the key components $\{\mathrm{SK}_\theta\}_{\theta\in\mathsf{Path}(\eta)}$ to the user. Given a revocation list RL and a time period t, the attribute authority runs the algorithm KUNode to get a node set Y. For each node $\theta \in \mathsf{Y}$, the authority further creates an update key component $\mathrm{UK}_{\theta,\delta}$ and publishes the update key components $\{\mathrm{UK}_\theta\}_{\theta\in\mathsf{Y}}$. Consequently, for a user that is not revoked at the time period t, there would exist a node $\theta' \in \mathsf{Y} \cap \mathsf{Path}(\eta)$. Thus, this user can utilize $\mathrm{UK}_{\theta'}$ to update his/her secret key component $\mathrm{SK}_{\theta'}$. However, by the principle of algorithm KUNode, we know that $\{\mathrm{UK}_\theta\}_{\theta\in\mathsf{Y}}$ is useless for those users that were revoked before the time period t.

With regard to forward security, we follow the idea of forward-secure public-key encryption [27],[4] that is, the whole lifetime of the system is divided into T time periods: $0, 1, \ldots, T-1$. Each time period t is assigned to a leaf node ζ_t of a binary tree $\mathcal{T}$ with T leaves in a natural order from left to right. For a node ζ of $\mathcal{T}$, we denote by $\mathsf{R}(\zeta)$ its right child,[5] and b_ζ the binary string corresponding to the path from the root node to ζ, where a 0 and a 1 indicate the path traverses the left and right child of the previous node, respectively. For example, we use a binary tree with depth d to manage 2^d time periods. Then, the leftmost leaf node (0^d) is assigned to the time period 0, and the rightmost leaf node (1^d) is assigned to $2^d - 1$. Furthermore, for a time period t, we define a set $\mathcal{T}_t = \{\mathsf{R}(\zeta) | \zeta \in \mathsf{Path}(\zeta_t) \wedge \mathsf{R}(\zeta) \notin \mathsf{Path}(\zeta_t)\} \bigcup \{\zeta_t\}$. It is easy to prove that such defined sets have the following property.

Property 1: Give two time periods t and t' such that $t' > t$, for each node $\zeta' \in \mathcal{T}_{t'}$, there exists a node $\zeta \in \mathcal{T}_t$ such that b_ζ is a prefix of $b_{\zeta'}$.

[4]In a forward-secure public-key encryption, it is the *secret key* that is periodically updated, and the secret key at the current time period cannot be used to decrypt previously generated ciphertext.

[5]For a leaf node η, we assume that $\mathsf{R}(\zeta) = \zeta$.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

For each node $\zeta \in \mathcal{T}_t$, we produce a corresponding ciphertext component, which consists of two parts: one part is regarded as an essential input of the decryption algorithm, which guarantees that only those users that are not revoked at the time period t can decrypt the ciphertext, and another part is just used to update the ciphertext from the current time period to the next one.

A. Construction

Our multiauthority CP-ABE scheme consists of the following algorithms.

GlobalSetup (λ): Given a security parameter λ, this algorithm produces the global public parameter as follows.

1) Choose two cyclic groups $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ of prime order p, and a bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Pick a random group element $g \in \mathbb{G}$.
2) Let $\mathcal{I}$ be the identifier space, $\mathcal{A}$ be the set of authorities, and $\mathcal{U}$ be the attribute universe. For each authority $\delta \in \mathcal{A}$, denote by $\mathcal{U}_\delta$ the attribute universe managed by δ.[6] Select two hash functions: $H : \mathcal{I} \to \mathbb{G}$ and $F : \mathcal{U} \to \mathbb{G}$.
3) Publish the global public parameter as
$$\mathrm{GP} = \{\mathbb{G}, \mathbb{G}_T, e, p, g, e(g,g), H, F\}.$$

AuthSetup (GP, N, T): The authority setup algorithm takes as input the global parameter GP, the number of system users N, and the number of time periods $T = 2^d$. For an authority δ, it generates the public parameter and master secret key by conducting the following steps.

1) Generate a binary tree $\mathcal{BT}_\delta$ with N leaf nodes, and initialize $\mathrm{RL}_\delta = \emptyset$ and $\mathrm{st}_\delta = \mathcal{BT}_\delta$. Denote by $\mathcal{N}_\delta$ the set of all nodes of $\mathcal{BT}_\delta$. For each node $\theta \in \mathcal{N}_\delta$, select a random integer $r_\theta \in \mathbb{Z}_p$.[7]
2) Choose random exponents $\alpha_\delta, \beta_\delta \in \mathbb{Z}_p$ and random group elements $f_{\delta,0}, f_{\delta,1}, \ldots, f_{\delta,d} \in \mathbb{G}$, set the public parameter and master secret key as
$$\mathrm{LP}_\delta = \{e(g,g)^{\alpha_\delta}, g^{\beta_\delta}, f_{\delta,0}, f_{\delta,1}, \ldots, f_{\delta,d}\}$$
and
$$\mathrm{MSK}_\delta = \{\alpha_\delta, \beta_\delta, \{r_\theta\}_{\theta\in\mathcal{N}_\delta}\}.$$

In addition, for each authority $\delta \in \mathcal{A}$, we define a function $W_\delta(t) : \{0, \ldots, T-1\} \to \mathbb{G}$ as follows:
$$W_\delta(t) = f_{\delta,0} \prod_{j=1}^{d} f_{\delta,j}^{t[j]}$$
where $t[j]$ is the jth bit of t in the form of binary string.[8]

SKeyGen $(\mathrm{GP}, \mathrm{LP}_\delta, \mathrm{MSK}_\delta, \mathrm{GID}, S_\delta, \mathrm{st}_\delta)$: The secret key generation algorithm take as input the global public parameter GP, the local public parameter LP_δ, the master secret key MSK_δ, a global identifier GID, an attribute set $S_\delta \subseteq \mathcal{U}_\delta$, and the state information st_δ. It produces a secret key for (GID, S_δ) as follows.

1) Select an unassigned leaf node η of $\mathcal{BT}_\delta$ and store GID in it.
2) For each node $\theta \in \mathsf{Path}(\eta)$, choose a random integer $r_u \in \mathbb{Z}_p$ for each attribute $u \in S_\delta$, and create a key component $\mathrm{SK}_\theta = \{(K_{\theta,\delta,u,\mathrm{GID}}, K'_{\theta,\delta,u,\mathrm{GID}})\}_{u\in S_\delta}$, where
$$K_{\theta,\delta,u,\mathrm{GID}} = g^{\alpha_\delta - r_\theta} H(\mathrm{GID})^{\beta_\delta} F(u)^{r_u}$$
and
$$K'_{\theta,\delta,u,\mathrm{GID}} = g^{r_u}.$$
3) Output the secret key $\mathrm{SK}_{\mathrm{GID},S_\delta} = \{\mathrm{SK}_\theta\}_{\theta\in\mathsf{Path}(\eta)}$ and updated state information st_δ.

UKeyGen $(\mathrm{GP}, \mathrm{LP}_\delta, \mathrm{MSK}_\delta, \mathrm{RL}_\delta, \mathrm{st}_\delta, t)$: The update key generation algorithm takes as input the global public parameter GP, the local public parameter LP_δ, the master secret key MSK_δ, the current revocation list RL_δ and state information st_δ, and the update time period t. It generates an update key for nonrevoked users as follows.

1) Run the algorithm $\mathsf{KUNode}(\mathcal{BT}_\delta, \mathrm{RL}_\delta, t)$ to get a node set $\mathsf{Y}_\delta \subseteq \mathcal{N}_\delta$.
2) For each node $\theta \in \mathsf{Y}_\delta$, choose a random exponent $\gamma_t \in \mathbb{Z}_p$ and compute
$$\mathrm{UK}_{\theta,\delta} = (U_{\theta,\delta}, U'_{\theta,\delta}) = (g^{r_\theta} \cdot W_\delta(t)^{\gamma_t}, g^{\gamma_t}).$$
3) Output the update key $\mathrm{UK}_{\delta,t} = \{\mathrm{UK}_{\theta,\delta}\}_{\theta\in\mathsf{Y}_\delta}$.

DKeyGen $(\mathrm{GP}, \mathrm{LP}_\delta, \mathrm{SK}_{\mathrm{GID},S_\delta}, \mathrm{UK}_{\delta,t})$: The decryption key generation algorithm takes as input the global public parameter GP, the local public parameter LP_δ, a secret key $\mathrm{SK}_{\mathrm{GID},S_\delta}$, and an update key $\mathrm{UK}_{\delta,t}$. It generates a decryption key by conducting the following steps.

1) Find out $\theta \in \mathsf{Y}_\delta \bigcap \mathsf{Path}(\eta)$, and parse the two components as $\mathrm{SK}_\theta = \{(K_{\theta,\delta,u,\mathrm{GID}}, K'_{\theta,\delta,u,\mathrm{GID}})\}_{u\in S_\delta}$ and $\mathrm{UK}_{\theta,\delta} = (U_{\theta,\delta}, U'_{\theta,\delta})$.
2) Choose a random exponent $\gamma'_t \in \mathbb{Z}_p$, and for each attribute $u \in S_\delta$ compute
$$D_{\delta,u,\mathrm{GID}} = K_{\theta,\delta,u,\mathrm{GID}} \cdot U_{\theta,\delta} \cdot W_\delta(t)^{\gamma'_t}$$
and
$$D'_{\delta,u,\mathrm{GID}} = K'_{\theta,\delta,u,\mathrm{GID}},\ D_{\delta,t} = U'_{\theta,\delta} \cdot g^{\gamma'_t}.$$
3) Return the decryption key
$$\mathrm{DK}^t_{\mathrm{GID},S_\delta} = \{D_{\delta,t}, \{D_{\delta,u,\mathrm{GID}}, D'_{\delta,u,\mathrm{GID}}\}_{u\in S_\delta}\}.$$

Encrypt $(\mathrm{GP}, \{\mathrm{LP}_\delta\}_{\delta\in\mathcal{A}}, (A, \rho, \varphi), m, t)$: The encryption algorithm takes as input the global public parameter GP, all local public parameter $\{\mathrm{LP}_\delta\}_{\delta\in\mathcal{A}}$, an $l \times n$ access matrix A associated with two mappings[9] $\rho : [l] \to \mathcal{A}$ and $\varphi : [I] \to \mathcal{U}$ [i.e., the attribute $\varphi(i)$ belongs to the authority $\rho(i)$], a message m to be encrypted, and a time period t. It produces the corresponding ciphertext as follows.

[6]For two authorities $\delta_1, \delta_2 \in \mathcal{A}$, we assume that $\mathcal{U}_{\delta_1} \bigcap \mathcal{U}_{\delta_2} = \emptyset$, that is, each attribute belongs to only one authority.

[7]The authority can use a pseudorandom generator instead of choosing and storing r_θ.

[8]We use a binary tree with depth d to manage time periods, and each time period can be represented with a binary string with fixed length d.

[9]We assume there exists a mapping that can easily map an attribute to the corresponding authority. In the implementation, the attribute is structured as Attribute@Authority.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

1) Choose two random vectors $\vec{y} = (s, y_2, \dots, y_n)$ and $\vec{w} = (0, w_2, \dots, w_n) \in \mathbb{Z}_p^n$, and compute the shares of s and 0 as
$$\vec{\lambda} = (\lambda_1, \dots, \lambda_l)^\top = A\vec{y}^\top \quad \vec{\chi} = (\chi_1, \dots, \chi_l)^\top = A\vec{w}^\top.$$
2) Pick a random exponent $z_i \in \mathbb{Z}_p$ for each $i \in [l]$ and compute
$$C_0 = m \cdot e(g,g)^s \quad C_{i,1} = e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\rho(i)} z_i}$$
$$C_{i,2} = g^{-z_i} \quad C_{i,3} = g^{\beta_{\rho(i)} z_i} g^{\chi_i} \quad C_{i,4} = F(\varphi(i))^{z_i}$$
and for each node $\zeta \in \mathcal{T}_t$
$$C_{i,\zeta} = (C_{i,\zeta,0}, C_{i,\zeta,|b_\zeta|+1}, \dots, C_{i,\zeta,d})$$
where
$$C_{i,\zeta,0} = \left(f_{\rho(i),0} \prod_{j=1}^{|b_\zeta|} f_{\rho(i),j}^{b_\zeta[j]} \right)^{z_i}$$
and
$$C_{i,\zeta,k} = f_{\rho(i),k}^{z_i} \text{ for } k \in [|b_\zeta| + 1, d].$$
3) Output the ciphertext CT_t
$$\{C_0, \{C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, \{C_{i,\zeta}\}_{\zeta \in \mathcal{T}_t}\}_{i \in [l]}, (A, \rho, \varphi), t\}.$$

CTUpdate $(\mathrm{GP}, \{\mathrm{LP}_\delta\}_{\delta \in \mathcal{A}}, \mathrm{CT}_t, t')$: The ciphertext update algorithm takes as input the global public parameter GP, all local public parameter $\{\mathrm{LP}_\delta\}_{\delta \in \mathcal{A}}$, a ciphertext CT_t produced at a time period t, and a new time period $t' > t$. It updates the ciphertext to the new time period as follows.

1) Let the corresponding access policy be (A, ρ, φ), where A is an $l \times n$ matrix. Choose two random vectors $\vec{y}' = (s', y_2', \dots, y_n')$, $\vec{w}' = (0, w_2', \dots, w_n') \in \mathbb{Z}_p^n$, and compute the shares of s' and 0 as
$$\vec{\lambda}' = (\lambda_1', \dots, \lambda_l')^\top = A\vec{y}'^\top, \vec{\chi}' = (\chi_1', \dots, \chi_l')^\top = A\vec{w}'^\top.$$
2) For each $i \in [l]$, choose a random exponent $z_i' \in \mathbb{Z}_p$, and compute
$$C_0' = C_0 \cdot e(g,g)^{s'} \quad C_{i,1}' = C_{i,1} \cdot e(g,g)^{\lambda_i'} e(g,g)^{\alpha_{\rho(i)} z_i'}$$
$$C_{i,2}' = C_{i,2} \cdot g^{-z_i'} \quad C_{i,3}' = C_{i,3} \cdot g^{\beta_{\rho(i)} z_i'} g^{\chi_i'}$$
$$C_{i,4}' = C_{i,4} \cdot F(\varphi(i))^{z_i'}$$
and for each node $\zeta' \in \mathcal{T}_{t'}$, find a node $\zeta \in \mathcal{T}_t$ such that b_ζ is a prefix of $b_{\zeta'}$, and then compute
$$C_{i,\zeta'} = (C_{i,\zeta',0}, C_{i,\zeta',|b_\zeta'|+1}, \dots, C_{i,\zeta',d})$$
where
$$C_{i,\zeta',0} = C_{i,\zeta,0} \prod_{j=|b_\zeta|+1}^{|b_{\zeta'}|} (C_{i,\zeta,j})^{b_{\zeta'}[j]} \left(f_{\rho(i),0} \prod_{j=1}^{|b_{\zeta'}|} f_{\rho(i),j}^{b_{\zeta'}[j]} \right)^{z_i'}$$
$$C_{i,\zeta',k} = C_{i,\zeta,k} \cdot f_{\rho(i),k}^{z_i'}, \text{ for } k \in [|b_{\zeta'}| + 1, d].$$
3) Return the updated ciphertext $\mathrm{CT}_{t'}$
$$\{C_0', \{C_{i,1}', C_{i,2}', C_{i,3}', C_{i,4}', \{C_{i,\zeta'}\}_{\zeta' \in \mathcal{T}_{t'}}\}_{i \in [l]}\}$$
associated with (A, ρ, φ) and t'. The original ciphertext CT_t is then erased.

Decrypt $(\mathrm{GP}, \mathrm{CT}_t, \mathrm{DK}_{\mathrm{GID},S}^t)$: The decryption algorithm takes as input the global public parameter GP, a ciphertext CT_t, and a decryption key $\mathrm{DK}_{\mathrm{GID},S}^t$. It recovers the original message as follows.

1) Identify the index subset $I = \{i \in [l] : \varphi(i) \in S\}$, and compute these constants $\{\xi_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} \xi_i A_i = (1, 0, \dots, 0)$, where A_i is the ith row of A.
2) For each $i \in I$, calculate
$$\begin{aligned} C_i &= C_{i,1} \cdot e(D_{\rho(i),\varphi(i),\mathrm{GID}}, C_{i,2}) \cdot e(H(\mathrm{GID}), C_{i,3}) \\ &\quad \cdot e(D'_{\rho(i),\varphi(i),\mathrm{GID}}, C_{i,4}) \cdot e(D_{\rho(i),t}, C_{i,\zeta_t,0}) \\ &= e(g,g)^{\lambda_i} \cdot e(H(\mathrm{GID}), g)^{\chi_i} \end{aligned}$$
and
$$C = \prod_{i \in I} C_i^{\xi_i} = e(g,g)^s.$$
3) Return the message $m = C_0 / C$.

Revoke $(\mathrm{GID}, \mathrm{RL}_\delta, \mathrm{st}_\delta, t)$: The revocation algorithm takes as input an identifier GID to be revoked at a time period t, the current revocation list RL_δ, and the state information st_δ. It returns the updated revocation list $\mathrm{Rl}_\delta \leftarrow \mathrm{RL}_\delta \bigcup \{(\eta, t)\}$, where η is the leaf node of $\mathcal{BT}_\delta$ used to store GID.

B. Security Analysis

In the supplementary file, we formally define the static security of the proposed scheme, and capture its security through the following theorem.

Theorem 1: If the q-DPBDHE-v assumption holds, then our multiauthority CP-ABE scheme is statically secure against any PPT adversary with a challenge matrix of size at most $q \times q$ in the random oracle model.

Proof: The detailed proof is provided in the supplementary file. ■

Here we give an intuitive and compact security analysis of our construction in terms of collusion attacks, backward security, and forward security.

Collusion attacks: In our construction, each user has a global identifier GID. This global identifier enables AAs to *independently* generate secret key components for the user, that is, the AAs neither have to cooperate with each other, nor need to interact with a central authority. On the other hand, the hash value of the identifier makes the user's key components issued by different AAs consistent with each other, but not with the key components issued to another user, namely, users with different identifiers cannot conclude with each other.

Backward security: Our construction guarantees the backward security of the ciphertext by providing the functionality of user revocation. In our scheme, we note that only a correct *decryption key*, which is derived from the original secret key and the current update key in a random manner, can be used to correctly decrypt the corresponding ciphertexts. When a user is removed from an attribute domain $\mathcal{U}_\delta$, by the KUNode algorithm, we know that those subsequently published update

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
COMPARISONS OF COMPUTATION COMPLEXITY WITH PREVIOUS WORKS

Schemes	GlobalSetup	AuthSetup	SKeyGen	UKeyGen	DKeyGen	Encrypt	CTUpdate	Decrypt
[9]	$\mathcal{O}(N)\mathsf{e}$	$\mathcal{O}(\lvert\mathcal{U}\rvert)\mathsf{e}+\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert S\rvert)\mathsf{e}$	$\mathcal{O}(N-r)\mathsf{e}$	-	$\mathcal{O}(l+k)\mathsf{e}$	$\mathcal{O}(a)\mathsf{e}$	$\mathcal{O}(\lvert I\rvert)\mathsf{e}+\mathcal{O}(k+\lvert I\rvert)\mathsf{p}$
[13]	$\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert\mathcal{U}\rvert)\mathsf{e}+\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert S\rvert)\mathsf{e}$	$\mathcal{O}(a)\mathsf{e}$	-	$\mathcal{O}(l+k)\mathsf{e}$	$\mathcal{O}(k+l)\mathsf{e}$	$\mathcal{O}(\lvert I\rvert)\mathsf{e}+\mathcal{O}(\lvert I\rvert)\mathsf{p}$
[14]	$\mathcal{O}(N)\mathsf{e}$	$\mathcal{O}(\lvert\mathcal{U}\rvert)\mathsf{e}+\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert S\rvert)\mathsf{e}$	$\mathcal{O}(N-r)\mathsf{e}$	-	$\mathcal{O}(l)\mathsf{e}$	$\mathcal{O}(a)\mathsf{e}$	$\mathcal{O}(\lvert I\rvert)\mathsf{e}+\mathcal{O}(k+\lvert I\rvert)\mathsf{p}$
[18]	-	$\mathcal{O}(1)\mathsf{e}+\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert S\rvert\cdot\log N)\mathsf{e}$	$\mathcal{O}(r\log(N/r))\mathsf{e}$	-	$\mathcal{O}(l)\mathsf{e}$	-	$\mathcal{O}(\lvert I\rvert)\mathsf{e}+\mathcal{O}(\lvert I\rvert)\mathsf{p}$
Ours	$\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(1)\mathsf{e}+\mathcal{O}(1)\mathsf{p}$	$\mathcal{O}(\lvert S\rvert\cdot\log N)\mathsf{e}$	$\mathcal{O}(r\log(N/r))\mathsf{e}$	$\mathcal{O}(1)\mathsf{e}$	$\mathcal{O}(l\cdot\log T)\mathsf{e}$	$\mathcal{O}(l\cdot\log T)\mathsf{e}$	$\mathcal{O}(\lvert I\rvert)\mathsf{e}+\mathcal{O}(\lvert I\rvert)\mathsf{p}$

$^*N$ = the number of users in the system. r = the number of users to be revoked. T = the total number of time periods. k = the number of authorities involved in encryption and decryption. l = the number of rows of LSSS matrix. $|I|$ = the number of attributes involved in decryption. $|\mathcal{U}|$ = the size of attributes managed by an authority. $|S|$ = the size of a user's attribute set. a = the number of attributes to be revoked. e = one exponenation operation. p = one pairing operation.

keys are useless for him/her, since $\mathsf{Path}(\eta)\bigcap \mathsf{Y}_\delta = \emptyset$. Therefore, without help of the update key, the user can no longer get a correct decryption key, and thus cannot decrypt those subsequently produced ciphertexts.

Forward security: The forward security of our scheme comes from public ciphertext update. More precisely, among the ciphertext components $\{C_{i,\zeta}\}_{i\in[l],\zeta\in\mathcal{T}_t}$, we note that the term C_{i,ζ_t} is mandatorily required for the decryption procedure, and other terms in fact are only used to update C_{i,ζ_t} to $C_{i,\zeta_{t'}}$ for a new time period $t' > t$. By the construction of $\mathcal{T}_t$, we know that this update procedure is unidirectional, that is, C_{i,ζ_t} cannot be conversely derived from C_{i,ζ'_t}. Therefore, when the ciphertext CT_t is updated to $\mathrm{CT}_{t'}$ and then erased, those previously generated decryption keys naturally expire.

C. Performance Discussions

In this section, we theoretically analyze the performance of our multiauthority CP-ABE scheme in terms of computation complexity and feature.

In Table II, we compare the computation complexity of the proposed scheme with several recently proposed CP-ABE schemes[10] supporting attribute/user revocation and ciphertext update. In Yang *et al.*'s schemes [9], [14], the central authority needs to produce a key component related to each user's global identity. Thus, the complexity of algorithm $\mathsf{GlobalSetup}$ in their schemes is linear to the number of system users. On the other hand, the central authority in Chen and Ma's [13] scheme and our scheme only needs to produce global public parameter, and thus its computation cost is constant. We can see that in Liang *et al.* [18] scheme and our scheme, the computation cost of the attribute authority generating the public parameter is also constant. This is because that the public parameter of our scheme is independent of the attribute universe, while Liang *et al.*'s [18] scheme chooses a random group element for each attribute. In general, the computation costs of the algorithms $\mathsf{SKeyGen}$ and $\mathsf{Encrypt}/\mathsf{CTUpdate}$ are linear in the size of a user's attribute set and the number of rows of an LSSS matrix, respectively, such as [9], [13], and [14]. However, to achieve scalable user revocation and public update of secret key and ciphertext, these algorithms in our scheme consume additional but acceptable computation costs, which are upper bounded by $\log N$ and $\log T$, respectively. Focusing on the update key generation, we note that the computation complexity of the algorithm $\mathsf{UKeyGen}$ in [9], [13], and [14] is linear in the number of nonrevoked users (or users whose attributes are not revoked). Compared with our scheme and Liang *et al.*'s [18] scheme, such a revocation manner is not desirable. With regard to the complexity of algorithm $\mathsf{DKeyGen}$, we note only our scheme needs insignificant computation cost. This is because that we combine a secret key and an update key in a rerandom manner to derive a decryption such that our scheme can resist decryption key exposure attack. However, the schemes in [9], [13], and [14] do not involve the derivation of decryption key. Liang *et al.*'s [18] scheme derives the decryption key in an intuitive manner, i.e., directly combing the original secret key with the update key, without additional computation. As in original ABE scheme, the computation complexity of the decryption procedure in all listed schemes grows linearly with the number of attributes satisfying the corresponding access policy.

Table III lists the features of these schemes. We can see that the two schemes in [9] and [14] suffer from collusion attacks such that revoked users can still access the encrypted data. Thus, they naturally cannot provide forward security and backward security. In Chen and Ma's [13] scheme, all nonrevoked users share the same update key. Therefore, their scheme also fails to achieve forward and backward security. In addition, the abovementioned schemes all update users' secret keys or ciphertexts in a private way, rather than public update as in [18] and [19] and our scheme. With the exception of our scheme, other schemes all cannot withstand decryption key exposure attack since a decryption key in their schemes is a direct combination of a secret key and an update key. Moreover, our scheme uniquely features that the public parameter is independent of the attribute universe (i.e., large universe). In other words, throughout the lifetime of the system, the public parameter of the system unchanged, and any string can be an attribute.

In conclusion, at the cost of additional computation and communication overhead, the proposed scheme provides desirable security properties for practical applications. However, compared with those constructions [15], [19] providing the same security properties but built upon bilinear groups of composite order, our scheme is more efficient. The below experiments show that the proposed scheme is indeed efficient enough for practical applications.

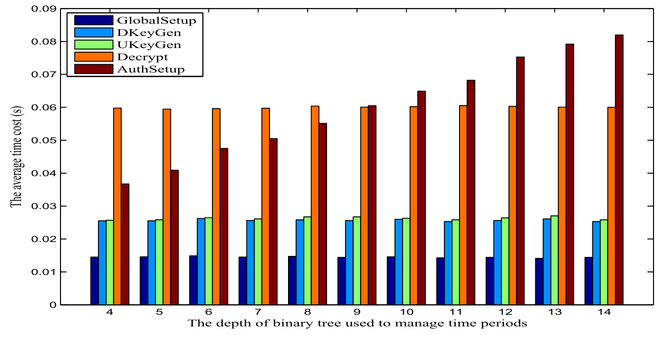
[10]Since the scheme in [19] is a generic construction with these functionalities and the scheme in [15] is built upon bilinear groups of composite order, we thus exclude them from the comparison.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

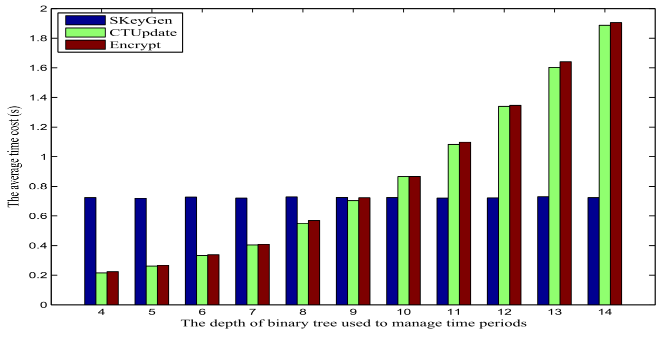
TABLE III
COMPARISONS OF FUNCTIONALITIES WITH PREVIOUS WORKS

Schemes	Authority	Attribute universe	Revocation level	Forward security	Backward security	Public update	DKE
[9]	Multiple	Small	Attribute	✗	✗	✗	✗
[13]	Multiple	Small	Attribute	✗	✗	✗	✗
[14]	Multiple	Small	Attribute	✗	✗	✗	✗
[18]	Single	Small	User	✗	√	√	✗
[19]	Single	Small	User	√	√	√	✗
Ours	Multiple	Large	User	√	√	√	√

*DKE = decryption key exposure.



(a)



(b)

Fig. 4. Average time costs of all algorithms for different numbers of time periods.

V. IMPLEMENTATION AND EVALUATION

To demonstrate the practicability of the proposed multi-authority CP-ABE scheme, we implement it in Charm, a programming framework for cryptographic primitives using the Python language. Since the routines provided by this framework all use asymmetric groups (although the groups used in the scheme might be symmetric), we first translate our scheme to the asymmetric setting, namely, we employ an asymmetric bilinear mapping: $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are different cyclic groups. The complexity assumption and security proof can be translated to the asymmetric setting in a generic way.

We implemented the proposed scheme on a super-singular (SS) symmetric elliptic curve group ("SS512") supported by Charm. This curve group has a 160-b order and 1024-b security. We conducted all experiments on a PC equipped with a dual core Intel Core CPU E7300@2.66 GHz with 2.0 GB RAM running 32-b Linux Mint 17.1 operating system. For all tests we used Python 3.4 and Charm 0.43.

In Fig. 4, we present the average time costs of all algorithms in our scheme for different choices of the total number of time periods $T \in \{2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}\}$. In this experiment, we fixed the number of system users as $N = 8$ and the number of revoked users as $r = 3$. The access policy is set as ((A@Auth1 OR" B@Auth2) AND" (C@Auth1 OR" D@Auth2)) and the user's attributes are in the form {A@Auth1, D@Auth2, E@Auth2, F@Auth3, G@Auth3}. Note that the user's attributes always satisfy the access policy and the decryption algorithm only uses two rows of the LSSS matrix. To produce these experiment results, we perform in the following manner: generate a secret key, then, periodically update the secret key, encrypt a message and update the ciphertext, and decrypt the updated ciphertext. From Fig. 4(a), we can see that the average time costs of the algorithms GlobalSetup, AuthSetup, UKeyGen, DKeyGen, and Decrypt are all less than 100 ms for different numbers of time periods. Fig. 4(b) indicates that the algorithms SKeyGen, CTUpdate, and Encrypt consume more computation costs. Specifically, the time overhead of the algorithm SKeyGen is independent of the number of time periods, and is upper bounded by 800 ms. Although the time costs of the other two algorithms are linear in the number of time periods, for a small but practical number of time periods $T \in \{2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}\}$, the average time costs of the two algorithms are all under 1 s.

In Fig. 5(a), we show that the number of attributes has an influence on the efficiency of these algorithms that involve attribute operation, i.e., SKeyGen, DKeyGen, Encrypt, CTUpdate, and Decrypt. In this experiment, we fixed the number of system users as $N = 8$, the total number of time periods as $T = 2^8$, and no users are revoked. The number of attributes ranges from 5 to 20 and the access policy is set as (A_1 AND", ..., AND" A_i) for $i = 5 - 20$. We can see that the time cost of the algorithm Decrypt grows linearly in the number of attributes at a rapid pace. And the time overhead of the algorithms SKeyGen and CTUpdate is nearly the same. Particularly, the computation cost of the Encrypt algorithm slightly depends on the number of attributes, and is always under 500 ms. In addition, we also tested the computation consumption of the UKeyGen algorithm. As analyzed, it is independent of the number of attributes, and is upper bounded by 15 ms.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

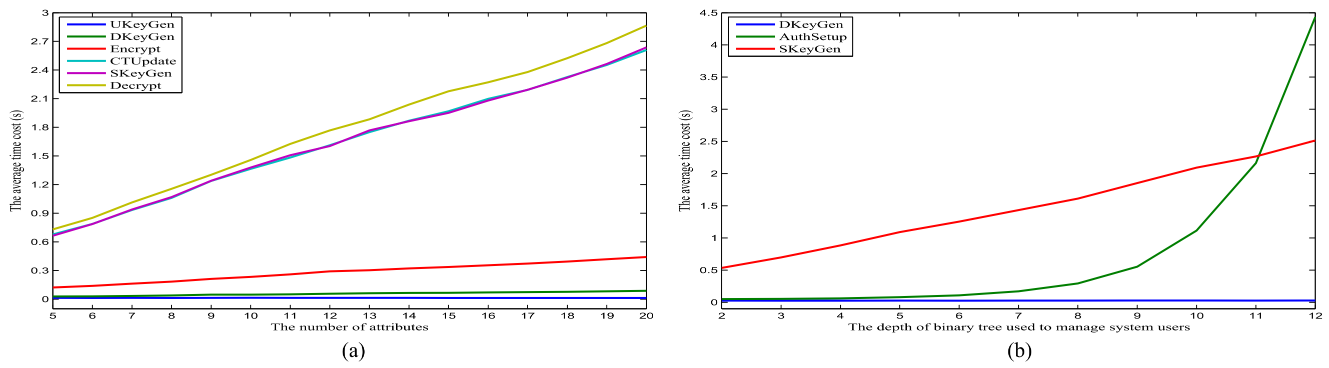


Fig. 5. Average time costs of several algorithms for different choices of (a) the number of attributes and (b) the number of system users.

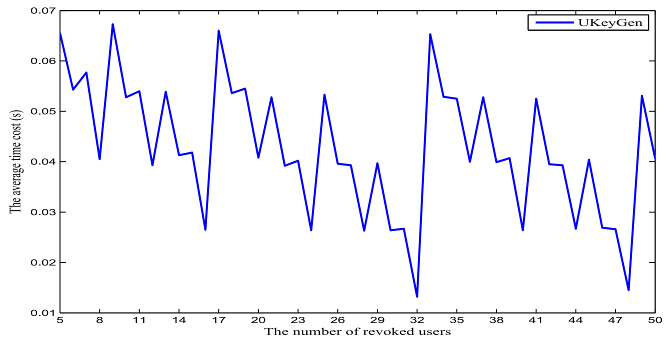


Fig. 6. Average time cost of the algorithm $\mathsf{UKeyGen}$ under different numbers of users to be revoked.

Fig. 5(b) shows the time costs of the algorithms $\mathsf{AuthSetup}$, $\mathsf{SKeyGen}$, and $\mathsf{DKeyGen}$ versus the number of system users. In this experiment, we fixed the total number of time periods as $T = 2^8$, the number of attributes as $u = 5$, and no users are revoked. The number of system users ranges from 2^2 to 2^{12}. We can see that the computation overhead of the algorithm $\mathsf{AuthSetup}$ grows exponentially in the depth of the binary tree used to manage system users (i.e., it is linear in the number of system users). This is because the authority needs to generate a random value for each node of the binary tree. On the other hand, the computation overhead of the algorithm $\mathsf{SKeyGen}$ is linear in the depth of the binary tree, and the time cost of the algorithm $\mathsf{DKeyGen}$ is unaffected by the number of system users.

Fig. 6 demonstrates the computation cost of the algorithm $\mathsf{UKeyGen}$ versus the number of users to be revoked. In this experiment, we fixed the number of system users as $N = 64$ and the total number of time periods as $T = 2^8$. The number of revoked users ranges from 5 to 50. We can see that the time over head of this algorithm in all cases is upper bounded by 70 ms. As mentioned, when $r \geq N/2$, the theoretical computation complexity is $\mathcal{O}(r \log(N/r))$ and $\mathcal{O}(N - r)$ otherwise. So, we can see that when $r > 32$, the time cost decreases.

VI. Extensions

Compared to general CP-ABE scheme, the proposed multiauthority CP-ABE scheme in this work additionally provides the functionalities of user revocation and ciphertext update. To meet more security and efficiency requirements of practical applications, the proposed scheme can be extended from the following aspects.

Combining CP-ABE with two-factor authentication: We note that user access control based on CP-ABE in fact achieves fuzzy authentication between a user and the cloud server, and thus preserves the user's privacy. However, in some cases the two sides need to authenticate each other exactly for consistent communications. On the other hand, as a kind of mainline cryptographic primitives for realizing user access control in clouding, two-factor authentication schemes [28]–[30] can provide precise authentication from end to end. Thus, it seems that we can combine the proposed scheme with various two-factor authentication schemes [31]–[33] to provide a kind of fine-grained and precise authentication mechanism with privacy preserving.

Outsourcing decryption: Green *et al.* [34] proposed the original work on outsourcing the decryption of ABE ciphertexts to a computation-efficient third party. As in Chow's [6] framework, we can add the functionality of outsourcing decryption to our scheme in a general way. As a result, the semitrusted cloud server would not only be in charge of periodically updating ciphertexts, but also partially decrypting ciphertexts for users, which will greatly reduce the computation cost of each user.

VII. Conclusion

In this paper, to build a secure and cost-effective multiauthority attribute-based access control scheme for data sharing in cloud storage systems, we proposed a multiauthority CP-ABE scheme supporting scalable user revocation and public ciphertext update. The proposed scheme achieves the intended security properties of forward security and backward security, and can also withstand decryption key exposure. We proved the security of the proposed scheme in the random oracle model. Both performance discussions and implementation experiments show that our scheme is more desirable for practical applications.

Acknowledgment

The authors would like to thank anonymous reviewers for their valuable comments.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Adv. Cryptol.—EUROCRYPT 2005*. New York, NY, USA: Springer, 2005, pp. 457–473.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Security Privacy 2007*, 2007, pp. 321–334.

[4] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 99–112.

[5] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Security 2010*, 2010, pp. 261–270.

[6] S. S. M. Chow, "A framework of multi-authority attribute-based encryption with outsourcing and revocation," in *Proc. 21st ACM Symp. Access Control Models Technol.*, 2016, pp. 215–226.

[7] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.

[8] C.-I. Fan, V. S.-M. Huang, and H.-M. Ruan, "Arbitrary-state attribute-based encryption with dynamic membership," *IEEE Trans. Comput.*, vol. 63, no. 8, pp. 1951–1961, Aug. 2014.

[9] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[10] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed access control in clouds," in *Proc. 2011 IEEE 10th Int. Conf. Trust, Security Privacy Comput. Commun.*, 2011, pp. 91–98.

[11] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[12] J. Hong, K. Xue, and W. Li, "Security analysis of attribute revocation in multiauthority data access control for cloud storage systems," *IEEE Trans. Inf. Forens. Security*, vol. 10, no. 6, pp. 1315–1317, Jun. 2015.

[13] J. Chen and H. Ma, "Efficient decentralized attribute-based access control for cloud storage with user revocation," in *Proc. 2014 IEEE Int. Conf. Commun.*, 2014, pp. 3782–3787.

[14] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.

[15] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, "Secure, efficient and revocable multi-authority access control system in cloud storage," *Comput. Security*, vol. 59, pp. 45–59, 2016.

[16] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 735–737.

[17] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forens. Security*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[18] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "An efficient and secure user revocation scheme in mobile social networks," in *Proc. IEEE GLOBECOM 2011*, 2011, pp. 1–5.

[19] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. Adv. Cryptol.—CRYPTO 2012*. New York, NY, USA: Springer, 2012, pp. 199–217.

[20] J. H. Seo and K. Emura, "Revocable identity-based encryption revisited: Security model and construction," in *Proc. Public-Key Cryptography 2013*. New York, NY, USA: Springer, 2013, pp. 216–234.

[21] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Faculty Comput. Sci., Technion–Israel Inst. Technol., Haifa, Israel, 1996.

[22] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography 2011*. New York, NY, USA: Springer, 2011, pp. 53–70.

[23] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financial Cryptography Data Security*. New York, NY, USA: Springer, 2015, pp. 315–332.

[24] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 417–426.

[25] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Adv. Cryptol.–EUROCRYPT 2011*. New York, NY, USA: Springer, 2011, pp. 568–588.

[26] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2013, pp. 463–474.

[27] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," *J. Cryptol.*, vol. 20, no. 3, pp. 265–294, 2007.

[28] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: Efficient and provably secure two-factor authentication scheme with user anonymity," *Inf. Sci.*, vol. 321, pp. 162–178, 2015.

[29] D. He, S. Zeadally, N. Kumar, and J.-H. Lee, "Anonymous authentication for wireless body area networks with provable security," *IEEE Syst. J.*, 2016, doi:10.1109/JSYST.2016.2544805.

[30] D. He, S. Zeadally, N. Kumar, and W. Wu, "Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures," *IEEE Trans. Inf. Forens. Security*, vol. 11, no. 9, pp. 2052–2064, Sep. 2016.

[31] X. Huang, X. Chen, J. Li, Y. Xiang, and L. Xu, "Further observations on smart-card-based password-authenticated key agreement in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1767–1775, Jul. 2014.

[32] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 4, pp. 428–442, Jul./Aug. 2015.

[33] D. He, N. Kumar, H. Wang, L. Wang, K.-K. R. Choo, and A. Vinel, "A provably-secure cross-domain handshake scheme with symptorms-matching for mobile healthcare social network," *IEEE Trans. Depend. Sec. Comput.*, 2016, doi:10.1109/TDSC.2016.2596286.

[34] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Conf. Security 2011*, 2011, p. 34.

Jianghong Wei received the B.S. degree in information security from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2009. He is currently working toward the Ph.D. degree at the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China.

His research interests include applied cryptography and network security.

Wenfen Liu received the Ph.D. degree in mathematics from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 1995.

She is a Full Professor with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China, and serves as the Head of probability statistics. Her research interests include probability statistics, network communications, and information security.

Xuexian Hu received the Ph.D. degree in information security from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2009.

He is a Lecturer with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China. His research interests include applied cryptography and network security.