TAAC: Temporal Attribute-based Access Control for Multi-Authority Cloud Storage Systems

Kan Yang*‡, Zhen Liu*†, Zhenfu Cao†, Xiaohua Jia*, Duncan S. Wong*, Kui Ren‡
* Dept. of CS, City University of Hong Kong
† Dept. of CS, Shanghai Jiao Tong University
‡ Dept. of CSE, University at Buffalo, The State University of New York
{kanyang3, zhenliu7}@student.cityu.edu.hk, zfcao@cs.sjtu.edu.cn, {csjia, duncan}@cityu.edu.hk, kuiren@buffalo.edu

***Abstract*—Data access control is an effective way to ensure the data security in the cloud. Due to data outsourcing and untrusted cloud servers, the data access control becomes a challenging issue in cloud storage systems. Ciphertext-Policy Attribute-based Encryption (CP-ABE), as a promising technique for access control of encrypted data, is very suitable for access control in cloud storage systems due to its high efficiency and expressiveness. However, the existing CP-ABE schemes cannot be directly applied to data access control for cloud storage systems because of the attribute revocation problem. In this paper, we consider the problem of attribute revocation in multi-authority cloud storage systems where the users' attributes come from different domains each of which is managed by a different authority. We propose TAAC (Temporal Attribute-based Access Control), an efficient data access control scheme for multi-authority cloud storage systems, where the authorities are independent from each other and no central authority is needed. TAAC can efficiently achieve temporal access control on attribute-level rather than on user-level. Moreover, different from the existing schemes with attribute revocation functionality, TAAC does not require re-encryption of any ciphertext when the attribute revocation happens, which means great improvement on the efficiency of attribute revocation. The analysis results show that TAAC is highly efficient, scalable, and flexible to applications in practice.**

***Index Terms*—Access Control, Temporal Revocation, CP-ABE, Cloud Storage, Multi-authority**

I. Introduction

Cloud storage is a promising technique that allows data owners to host their data in the cloud that provides data access service to users (data consumers) [1]. Because data owners cannot manage their outsourced cloud data as in their local storage systems, the data security becomes a significant issue in cloud storage systems. Data access control is an effective method to ensure the data security in the cloud. However, traditional server-based access control methods are no longer applicable to cloud storage systems, because the cloud server cannot be fully trusted by data owners. To keep data privacy on untrusted server, data owners usually encrypt the data and only the users who have valid keys can decrypt the data. However, the encryption-based methods may incur high key management overhead and require data owners to stay online all the time for the key distribution. Moreover, data owners may have to generate multiple copies of the encrypted data for the users with different keys. This may incur heavy storage overhead on the server, and data owners need to pay more for the storage space in pay-as-you-go cloud business model. Thus, the data access control is a challenging issue in cloud storage systems.

Ciphertext-Policy Attribute-based Encryption (CP-ABE) [2], [3] is regarded as one of the most suitable technologies for data access control in cloud storage systems, because it allows the data owner to define and enforce the access policy. In a CP-ABE system, each user is issued a decryption key that reflects his/her attributes (credentials), data owners encrypt data using access policies defined over attributes, so that only the users whose attributes satisfy the access policy can decrypt the ciphertext and access the data. There are two types of CP-ABE systems: single-authority CP-ABE [2]–[6] where all attributes are managed by a single authority, and multi-authority CP-ABE [7]–[10] where attributes are from different domains and managed by different authorities. Multi-authority CP-ABE is more appropriate for the access control of cloud storage systems, as the users may hold attributes issued by multiple authorities and data owners may share the data using access policy defined over attributes from different authorities. For example, data owners may share the data using access policy "Google.Engineer **AND** CityU.Alumni" where the attribute "Engineer" is issued by Google and the attribute "Alumni" is issued by CityU. However, due to the attribute revocation problem, these multi-authority CP-ABE schemes cannot be directly applied to data access control for such multi-authority cloud storage systems.

In multi-authority cloud storage systems, the users' attributes can be changed dynamically. Specifically, a user may be *entitled* some new attributes, or *revoked* some attributes, or *re-granted* some previously revoked attributes. Accordingly, the user's data access permission should be changed, e.g., when an attribute is revoked from a user, this user should not be able to decrypt any new ciphertexts which require the revoked attribute to decrypt. [1] However, all the aforementioned multi-authority CP-ABE schemes do not have appropriate mechanisms to handle such changes. Although some methods [11]–[14] were proposed to support such attribute revocation, they are not suitable for data access control in multi-authority cloud storage systems. Their schemes either rely on a trusted

[1]Note that revocation cannot obligate the revoked user to "forget" the data he has accessed before he is revoked, and that revocation is often to prevent the revoked user from accessing the "new" data after he is revoked, i.e., revocation is used to guarantee "backward security".

server to maintain the revocation list and enforce the key update or have to re-encrypt the ciphertexts in the cloud. However, the cloud server cannot be trusted by data owners in cloud storage systems, and the ciphertext re-encryption may incur a heavy burden to the system due to the large number of the ciphertexts. Thus, an efficient temporal attribute revocation method is desirable in order to design the data access control scheme for multi-authority cloud storage systems.

OUR CONTRIBUTIONS In this paper, we propose TAAC (Temporal Attribute-based Access Control), an efficient data access control scheme for multi-authority cloud storage systems, which allows data owners to define the access policy on the data and only users who possess sufficient attributes can decrypt the data. We solve the attribute revocation problem in multi-authority cloud storage systems, where each user in the system may be entitled attributes from multiple and different attribute domains and each authority manages a different attribute domain independently, even does not need to know the existence of others. Time in the system is slotted, and at each time slot each authority can revoke or re-grant any attribute in its domain from or to any user, without involving any other authorities, affecting any other users who also possesses the attribute, or affecting the decryption privilege of any other attributes possessed by the user.

In TAAC, only one copy of ciphertext for each data is stored, instead of multiple copies. TAAC can efficiently achieve temporal access control on attribute-level (when an attribute is revoked, it does not affect the user's decryption privilege of his other attributes) rather than on user-level (once an attribute is revoked, the user loses all the decryption privilege). Moreover, different from the existing schemes with attribute revocation functionality, TAAC does not require re-encryption of any ciphertext when the attribute revocation happens, which means great improvement on the efficiency of attribute revocation. We apply the tree structure for the user management of each attribute and propose an efficient algorithm to find the minimum set of nodes that covers all the non-revoked users, which can also improve the efficiency of our scheme. The analysis results show that TAAC is highly efficient, scalable, and flexible to applications in practice.

The remaining of this paper is organized as follows. We first give the definition of the system model, the framework and security model of TAAC in Section II. Then, we propose TAAC, a temporal attribute-based access control for multi-authority cloud storage systems with efficient revocation in Section III. In Section IV, we analyze TAAC in terms of security, scalability and performance by comparing with existing schemes. Section V gives the related work on data access control and the attribute revocation in ABE systems. Finally, the conclusion is given in Section VI.

II. SYSTEM MODEL AND SECURITY MODEL

A. Definition of System Model and Framework

We consider a cloud storage system with multiple authorities that defines a time space $\mathcal{T}$. Without loss of generality, $\mathcal{T}$ is defined as $\mathcal{T} = \{1, 2, \dots\}$, and the system initializes its time

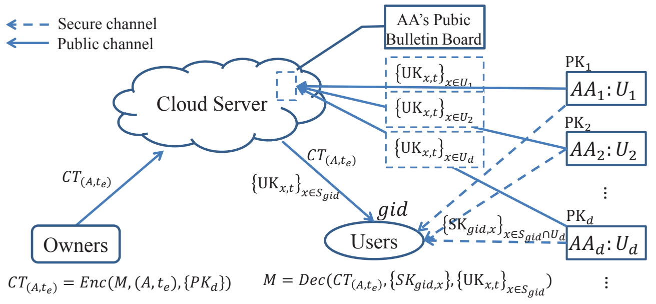

Secure channel
Public channel
AA's Pubic Bulletin Board
Cloud Server
$\{\mathrm{UK}_{x,t}\}_{x\in U_1}$
$\{\mathrm{UK}_{x,t}\}_{x\in U_2}$
$\{\mathrm{UK}_{x,t}\}_{x\in U_d}$
PK_1
$AA_1 : U_1$
PK_2
$AA_2 : U_2$
PK_d
$AA_d : U_d$
$CT_{(A,t_e)}$
$CT_{(A,t_e)}$
$\{\mathrm{UK}_{x,t}\}_{x\in S_{gid}}$
gid
$\{\mathrm{SK}_{gid,x}\}_{x\in S_{gid}\cap U_d}$
Owners
Users
$CT_{(A,t_e)} = Enc(M, (A, t_e), \{PK_d\})$
$M = Dec(CT_{(A,t_e)}, \{SK_{gid,x}\}, \{\mathrm{UK}_{x,t}\}_{x\in S_{gid}})$

Fig. 1. System Model of TAAC

to 0, and then increases it by 1 for the next time slot (for any time slot $t \in \mathcal{T}$, $t-1$ is its last time slot and $t+1$ is its next time slot). As shown in Fig.1, the system model consists of four types of entities: attribute authorities (AAs), the cloud server (server), data owners (owners) and data consumers (users). In addition, time in the system is slotted to support temporal data access control.

Every AA is an independent attribute authority that is responsible for entitling/revoking/re-granting attributes to/from/to users according to their role or identity in its domain. In TAAC, every attribute is associated with a single AA, but each AA can manage an arbitrary number of attributes. In practice, attributes belong to different authorities can be identified by encoding the attributes with different prefix, and in this paper, for simplicity, we will use a mapping $\phi : \mathcal{U} \mapsto \mathbb{D}$ to map any attribute to the index of corresponding authority. Every AA has full control over the structures and semantics of its attributes, and maintains a state and a revocation list for each attribute in its domain. Each AA is responsible for issuing secret keys to users when they are entitled attributes in its domain and publishing update keys for each attribute in its domain at each time slot to reflect the users' possessions of the attribute at the time slot.

Data owners define the access policies on attributes from multiple authorities and some time slots, then encrypt the data under the policies before hosting them on the cloud servers. They do not rely on the server to do data access control. Instead, the ciphertext can be accessed by all the legal users in the system, which means that any legal user who has been authenticated by the system somehow, he/she can freely download any interested ciphertexts from the server. But, the *access control happens inside the cryptography*. That is only the users who possess eligible attributes (satisfying the access policy $\mathbb{A}$) at a particular time slot t_e can decrypt the ciphertext associated with $(\mathbb{A}, t_e)$. The cloud server stores the owners' data and provides data access service to users.

Each user has a global identity gid in the system. A user gid may be entitled a set of attributes S_{gid} which may contain attributes from multiple authorities. As S_{gid} may change dynamically from time slot to time slot, $S_{gid,t}$ is used to denote the attribute set that *the user gid possesses at time slot t*. When a user gid is entitled an attribute x, he will be issued a corresponding *secret key* $\mathsf{SK}_{gid,x}$. However, the secret keys of a user *gid* is insufficient to decrypt a ciphertext encrypted under the policy $(\mathbb{A}, t_e)$ even when the corresponding attributes satisfy $\mathbb{A}$, and the user has to obtain a set of *update keys* at each time slot t (i.e. $\mathsf{UK}_{x,t}$) from the corresponding authorities. But the user are not required to be always online. Actually, the authorities revoke/re-grant attributes from/to users by publishing the update keys on the public bulletin boards (which can be stored on the cloud server), and only when the users need to access some data, they go online to obtain the released update keys for the passed time slots from the public bulletin boards. Note that such a revocation mechanism does not need secure channel or interaction (e.g., the user prove himself to the authorities) between the authorities and the users. The user can compute *decryption key*s for each time slot t from his secret keys and the received update keys, and uses the decryption keys to decrypt the ciphertext. [2]

B. Definition of TAAC Framework

Let $AA_1, AA_2, \ldots$ be attribute authorities and $\mathbb{D} = \{1, 2, \ldots\}$ be the index set of the AAs, that is, using $d \in \mathbb{D}$ to denote the index of attribute authority AA_d. Let $\mathcal{U}_d$ be the set of attributes managed by AA_d, where $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ for all $i \neq j \in \mathbb{D}$, and the attribute universe of the system is defined as $\mathcal{U} = \bigcup_{d \in \mathbb{D}} \mathcal{U}_d$. The AAs are independent from each other and do not need to know the existence of others. TAAC is defined as follows.

Definition 1 (TAAC)**.** *TAAC is a collection of the following algorithms:* GlobalSetup, AuthoritySetup, SKeyGen, UKeyGen, DKeyCom, Encrypt *and* Decrypt.

- $\mathsf{GlobalSetup}(\lambda) \to \mathsf{GPP}$. The global setup algorithm takes the security parameter λ as input. It outputs the global public parameters GPP.
- $\mathsf{AuthoritySetup}(\mathsf{GPP}, \mathcal{U}_d) \to (\mathsf{PK}_d, \mathsf{MSK}_d)$. The authority setup algorithm takes inputs as the global public parameters GPP, an attribute domain $\mathcal{U}_d$. It outputs the authority's public key PK_d and master secret key MSK_d. In addition, for each $x \in \mathcal{U}_d$, this algorithm initializes the state ST_x and initializes the revocation list of x to empty.
- $\mathsf{SKeyGen}(gid, x, \mathsf{ST}_x, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)}) \to (\mathsf{SK}_{gid,x}, \mathsf{ST}_x)$. The secret key generation algorithm takes as inputs the user's global identity *gid*, the state of the attribute ST_x, the global public parameters GPP and the master secret key $\mathsf{MSK}_{\phi(x)}$. The algorithm outputs a secret key $\mathsf{SK}_{gid,x}$ for the (attribute, identity) pair (x, gid), and an updated state ST_x.
- $\mathsf{UKeyGen}(t, x, \mathsf{ST}_x, \mathsf{RL}_{x,t}, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)}) \to (\mathsf{UK}_{x,t})$. At each time slot t, for each attribute $x \in \mathcal{U}_{\phi(x)}$, the authority $AA_{\phi(x)}$ runs the update key generation algorithm *once* by taking inputs as the state of the attribute ST_x, the revocation list of x at time slot t $\mathsf{RL}_{x,t}$, the global public parameters GPP, and its master secret key $\mathsf{MSK}_{\phi(x)}$. The algorithm outputs the *update key* $\mathsf{UK}_{x,t}$ for x at t, which will be published on the public bulletin board of the authority, which is stored on cloud servers.
- $\mathsf{DKeyCom}(\mathsf{SK}_{gid,x}, \mathsf{UK}_{x,t}) \to (\mathsf{DK}_{gid,x,t})$ or $\perp$. For any time slot t and any attribute x, a user *gid* can run the decryption key computation algorithm with secret key $\mathsf{SK}_{gid,x}$ and update key $\mathsf{UK}_{x,t}$ as input. The algorithm outputs a *decryption key* $\mathsf{DK}_{gid,x,t}$ implying that *gid* possesses x at t, or a special symbol $\perp$ implying that *gid* does not posses x at t. i.e., if and only if $x \in S_{gid,t}$, the user *gid* can compute a valid $\mathsf{DK}_{gid,x,t}$.
- $\mathsf{Encrypt}(M, t_e, \mathbb{A}, \mathsf{GPP}, \{\mathsf{PK}_d\}) \to (CT)$. The encryption algorithm takes as inputs a message M[3], a time slot t_e, an access policy $\mathbb{A}$ whose attributes can be from multiple authorities, the global public parameters GPP, and the public parameters $\{\mathsf{PK}_d\}$ related to $\mathbb{A}$. It outputs a ciphertext CT which includes $\mathbb{A}$ and t_e.
- $\mathsf{Decrypt}(CT, \mathsf{GPP}, \{\mathsf{PK}_d\}, \{\mathsf{DK}_{gid,x,t}\}_{x \in S_{gid,t}}) \to (M)$ or $\perp$. The decryption algorithm takes as input a ciphertext CT which includes access policy $\mathbb{A}$ and time slot t_e, the global public parameters GPP, the public parameters $\{\mathsf{PK}_d\}$ related to $\mathbb{A}$, and decryption keys $\{\mathsf{DK}_{gid,x,t}\}_{x \in S_{gid,t}}$ corresponding to a (global identity, time slot) pair (gid, t). The algorithm outputs a message M or a special symbol $\perp$ implying decryption failure.

C. Definition of Security Model

In cloud storage systems, we consider the case that the server may send the owners' data to the users who do not have access permission. The server is also curious about the content of the encrypted data. The users, however, are dishonest and may collude to obtain unauthorized access to data. Some of the *AAs* can be corrupted or compromised by the attackers.

The security of TAAC is defined by the following game run between a challenger and an adversary $\mathcal{A}$.

- Setup.
  1) The challenger runs GlobalSetup and gives the output GPP to $\mathcal{A}$.
  2) $\mathcal{A}$ specifies index set $\mathbb{D}_c \subset \mathbb{D}$ as the corrupt AAs. For good (non-corrupt) AAs in $\mathbb{D} \setminus \mathbb{D}_c$, the challenger runs the AuthoritySetup algorithm and gives the output $\mathsf{PK}_d (d \in \mathbb{D} \setminus \mathbb{D}_c)$ to $\mathcal{A}$.
- Phase 1. $\mathcal{A}$ can obtain secret keys and update keys by querying the following oracles:
  - $\mathsf{SKQ}(gid, x)$: $\mathcal{A}$ makes secret key queries by submitting pairs (gid, x), where *gid* is a global identity and x is an attribute belonging to some good authority (i.e., $\phi(x) \in \mathbb{D} \setminus \mathbb{D}_c$). The challenger runs $\mathsf{SKeyGen}(gid, x, \mathsf{ST}_x, \mathsf{GPP}, \mathsf{MSK}_{\phi(x)})$ to return a secret key $\mathsf{SK}_{gid,x}$ to $\mathcal{A}$ and update the state ST_x.

[2]In this paper, the term "*secret key*" is used to denote the key components that are issued to the user by the authority through a secure channel according to the entitled attributes to the user, the term "*update key*" is used to denote the key components that the authorities published on their public bulletin board to implement their control of revocation, and the term "decryption key" corresponds to the conventional secret key, i.e., the users in TAAC will use *decryption key* to decrypt the ciphertexts.

[3]In practice, data owners will first encrypt data with a content key by using symmetric encryption methods. Then, it apply TAAC to encrypt and control the content keys. Thus, in TAAC, the message M is the symmetric content key k.

  - $\mathsf{UKQ}(t,x,\mathsf{RL}_{x,t})$: $\mathcal{A}$ makes update key queries by submitting tuples $(t,x,\mathsf{RL}_{x,t})$, where $t \in \mathcal{T}$ is a time slot, x is an attribute belonging to some good authority (i.e., $\phi(x) \in \mathbb{D} \setminus \mathbb{D}_c$), and $\mathsf{RL}_{x,t}$ is a valid revocation list of x at t. The challenger runs $\mathsf{UKeyGen}(t,x,\mathsf{ST}_x,\mathsf{RL}_{x,t},\mathsf{GPP},\mathsf{MSK}_{\phi(x)})$ to return an update key $\mathsf{UK}_{x,t}$ to $\mathcal{A}$.
- Challenge Phase. $\mathcal{A}$ submits to the challenger two equal-length messages M_0, M_1, an access policy $\mathbb{A}^*$ over $\mathcal{U}$, and a time slot $t^* \in \mathcal{T}$. In addition, $\mathcal{A}$ must also give the challenger the public keys $\{\mathsf{PK}_d\}$ for any corrupt authorities whose attributes appear in $\mathbb{A}^*$. The challenger flips a random coin $\beta \in \{0,1\}$ and sends to $\mathcal{A}$ an encryption of M_β under $(\mathbb{A}^*,t^*)$.
- Phase 2. $\mathcal{A}$ makes further queries as in **Phase 1**.
- Guess. $\mathcal{A}$ submits a guess β' for β.

$\mathcal{A}$ wins the game if $\beta' = \beta$ under the following **restrictions**:

1) $\mathsf{UKQ}(t,\cdot,\cdot)$ can be queried on time slot which is greater than or equal to the time slot of all previous queries. i.e., the adversary is allowed to query only in non-decreasing order of time slot [4]. Also, for any pair (t,x), $\mathsf{UKQ}(t,x,\cdot)$ can be queried only once [5].
2) For any queried gid, S_{gid,t^*} does not satisfy $\mathbb{A}^*$.

The advantage of $\mathcal{A}$ is defined as $|\Pr[\beta = \beta'] - 1/2|$.

Definition 2. *TAAC is secure if for all polynomial-time adversary $\mathcal{A}$ in the game above, the advantage of $\mathcal{A}$ is negligible.*

III. TAAC: Temporal Attribute-based Access Control for Multi-Authority Cloud Storage Systems

In this section, we first give an overview of our solutions and then propose the detailed construction of TAAC and the efficient attribute revocation method.

A. Overview of Our Solutions

To realize the fine-grained access control, the owner first divides the data into several components according to the logic granularities and encrypts each data component with different content keys by using symmetric encryption methods. Then, the owner applies TAAC to encrypt each content key, such that only the users whose attributes at a time slot satisfy the access structure in the ciphertext can decrypt the content keys. Users with different attributes can decrypt different content keys and thus obtain different granularities of information from the same data.

To achieve temporal access control, the system defines a time space $\mathcal{T}$ which is slotted. For each attribute $x \in \mathcal{U}_d$, AA_d maintains a state ST_x and a revocation list RL_x, where ST_x is updated when a user is entitled x and RL_x (initially empty) is updated at each time slot if any user is revoked or re-granted x at that time slot. [6] For clarity, we denote $\mathsf{RL}_{x,t}$ as the revocation list of the attribute x at time slot t.

When a user gid is entitled an attribute $x \in \mathcal{U}_d$, AA_d issues a *secret key* $\mathsf{SK}_{gid,x}$ to gid and updates ST_x. When the owner encrypts a message M, besides specifying an access policy $\mathbb{A}$ over attribute universe $\mathcal{U}$, he also specifies a time slot $t_e \in \mathcal{T}$, which implies that only the users *who possess eligible attributes at time slot t_e* can decrypt the ciphertext. In TAAC, a user gid having secret keys $\{\mathsf{SK}_{gid,x}\}_{x\in S_{gid}}$ is unable to decrypt a ciphertext encrypted by $(\mathbb{A},t_e)$ even when S_{gid} satisfies $\mathbb{A}$. Besides, he has to obtain the *update key* for (x,t_e) (i.e. $\{\mathsf{UK}_{x,t_e}\}_{x\in S_{gid,t_e}}$) from the corresponding AAs. By using $\mathsf{SK}_{gid,x}$ and UK_{x,t_e}, the user gid can compute a *decryption key* DK_{gid,x,t_e} and use it to decrypt the ciphertext encrypted under $(\mathbb{A},t_e)$, if S_{gid,t_e} satisfies $\mathbb{A}$. At each time slot $t \in \mathcal{T}$, for any $x \in \mathcal{U}_d$, AA_d generates the $\mathsf{UK}_{x,t}$ according to ST_x and $\mathsf{RL}_{x,t}$ so that only the users who possess x at time slot t are able to obtain valid $\mathsf{UK}_{x,t}$. Thus, by setting the revocation list $\mathsf{RL}_{x,t}$ and publishing the corresponding $\mathsf{UK}_{x,t}$ the authority achieves revocation control of x.

B. Construction of TAAC

TAAC consists of four phases: System Initialization, Key Generation by AAs, Data Encryption by Owners and Data Decryption by Users.

Phase 1. System Initialization

The system is initialized by running the global setup algorithm $\mathsf{GlobalSetup}$. Let G be a bilinear group of order p and g be a generator of G. The global public parameter is published as

$$\mathsf{GPP} = (p,g,H),$$

where H is a hash function that will map global identities to elements of G.

Then, each authority is set up by running $\mathsf{AuthoritySetup}$. It takes the global public parameter GPP and the attribute set $\mathcal{U}_d$ as inputs.

For each attribute $x \in \mathcal{U}_d$, the corresponding authority $AA_{\phi(x)}$ will build up a binary tree T_x whose height is determined according to the possible number of users that possess x. When a user is entitled x, he will be associated to a leaf node of T_x, i.e., a binary tree of height h_x can allow 2^{h_x} users to be entitled x.

The nodes of T_x is encoded as

- The root node is on level 0;
- If the level of a non-leaf node is l, then the level of its children is $l+1$;
- For $l = 0$ to h_x, the nodes on level l are encoded/indexed from left to right to be $2^l, 2^l+1, \ldots, 2^{l+1}-1$.

Thus, the nodes in the tree T_x are uniquely encoded, and T_x can be represented as the set of indices of its nodes, i.e., $T_x = \{1,2,\ldots,2^{h_x+1}-1\}$. Fig. 2 shows a binary tree of height = 3.

Let $L_x = \{2^{h_x}, 2^{h_x}+1, \ldots, 2^{h_x+1}-1\}$ be the set of leaf nodes of T_x. For any leaf node $v_x \in L_x$, let $\mathsf{Path}(v_x)$ denote the set

[4]This captures the practical scenarios, where the authority will always generate and publish update keys for current time slot, so that the update keys can only be obtained in the non-decreasing order of time slot.

[5]This is because at each time slot t, for any attribute x, the corresponding authority will publish the corresponding update key $\mathsf{UK}_{x,t}$ only once.

[6]RL_x can be a part of ST_x, but we keep it explicit for clarity.

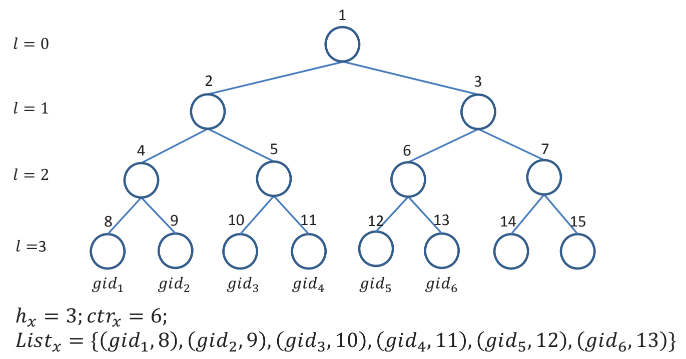

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
l = 0
l = 1
l = 2
l = 3
gid1
gid2
gid3
gid4
gid5
gid6
$h_x = 3; ctr_x = 6;$
$List_x = \{(gid_1, 8), (gid_2, 9), (gid_3, 10), (gid_4, 11), (gid_5, 12), (gid_6, 13)\}$

Fig. 2. A state tree of an attribute x. h_x is the hight of the state tree and ctr_x is the number of users in the state tree. $List_x$ stores the users in the state tree together with the node number that each user occupies in the tree.

of nodes on the path from v_x to the root node (both v_x and the root node inclusive), then we have

$$\mathsf{Path}(v_x) = \{\lfloor \frac{v_x}{2^l} \rfloor\}_{l=0}^{h_x}.$$

For any non-leaf node $v_x \in T_x \setminus L_x$, let $\mathsf{Lc}(v_x)$ and $\mathsf{Rc}(v_x)$ be the left child and right child of v_x respectively, then we have $\mathsf{Lc}(v_x) = 2v_x$ and $\mathsf{Rc}(v_x) = 2v_x + 1$.

Initially, all leaf nodes of T_x are set to be *empty* to imply that no user is entitled x. When a user gid is entitled x, the corresponding authority will choose an empty leaf node $u_{x,gid} \in L_x$ and assign it to gid by storing the pair $(gid, u_{x,gid})$ in the state of x, and $u_{x,gid}$ is said to be occupied by gid. Without loss of generality, and for the sake of efficiency, we assume that when a new user is entitled x, the authority will assign the most left empty leaf node of T_x to the user.

The authority setup algorithm is constructed as follows.

$\mathsf{AuthoritySetup}(\mathsf{GPP}, \mathcal{U}_d) \rightarrow (\mathsf{PK}_d, \mathsf{MSK}_d)$. For each attribute $x \in \mathcal{U}_d$, the algorithm chooses two random exponents $\alpha_x, \beta_x \in \mathbb{Z}_p$. Also, the algorithm determines an integer h_x to allow at most 2^{h_x} users to possess x, and sets $T_x = \{1, 2, \ldots, 2^{h_x+1} - 1\}$. For each node $v_x \in T_x$, the algorithm chooses a random element $R_{v_x} \in G$. Let $H_d : \mathcal{U}_d \times \mathcal{T} \mapsto G$ be a hash function that maps (attribute, time slot) paris in $\mathcal{U}_d \times \mathcal{T}$ to elements of G. The public key is set to

$$\mathsf{PK}_d = (\{e(g,g)^{\alpha_x}, g^{\beta_x}\}_{x \in \mathcal{U}_d}, H_d),$$

and the master secret key is set to

$$\mathsf{MSK}_d = \{\alpha_x, \beta_x,\ \{R_{v_x}\}_{v_x \in T_x}\}_{x \in \mathcal{U}_d}.$$

In addition, for each $x \in \mathcal{U}_d$, the algorithm initializes the state as $\mathsf{ST}_x = (x, h_x, ctr_x, List_x)$, where h_x is the height of the binary tree T_x, ctr_x is an integer (initial 0) that represents the number of users that has been entitled x, and $List_x$ (initially empty) is a set of (global identity, index of leaf node) pairs to record the assignments of leaf nodes to users.

Phase 2. Key Generation by AAs

The AAs will generate both secret keys and update keys for users.

1. *Secret Key Generation*

When a user gid is entitled an attribute $x \in \mathcal{U}_{\phi(x)}$, the corresponding authority $AA_{\phi(x)}$ runs the secret key generation algorithm $\mathsf{SKeyGen}$ to update ST_x and generate a secret key $\mathsf{SK}_{gid,x}$ for this user gid as follows.

1) Sets $u_{x,gid} = 2^{h_x} + ctr_x$, adds $(gid, u_{x,gid})$ to $List_x$ (assigning the leaf node $u_{x,gid}$, i.e., the most left empty leaf node, to gid), and sets $ctr_x = ctr_x + 1$,
2) Sets

$$\mathsf{SK}_{gid,x} = \{K_{gid,x,v_x} = g^{\alpha_x} H(gid)^{\beta_x} R_{v_x}\}_{v_x \in \mathsf{Path}(u_{x,gid})}.$$

Then, the authority $AA_{\phi(x)}$ sends the secret key $\mathsf{SK}_{gid,x}$ to the user gid. Note that only when a new attribute $x \in \mathcal{U}_{\phi(x)}$ is assigned to the user gid, a corresponding secret key $\mathsf{SK}_{gid,x}$ is issued to this user gid by the authority $AA_{\phi(x)}$. This secret key assignment happens at most *once* for each attribute on each user. When the attribute x is revoked or re-granted some time, the secret key $\mathsf{SK}_{gid,x}$ will not be removed or re-assigned to the user gid anymore.

2. *Update Key Generation*

At the beginning of each time slot t, for each attribute $x \in \mathcal{U}_{\phi(x)}$, the corresponding authority $AA_{\phi(x)}$ determines the revocation list $\mathsf{RL}_{x,t}$ and runs the update key generation algorithm $\mathsf{UKeyGen}$ *once* to generate and publish the update key $\mathsf{UK}_{x,t}$. The algorithm computes the minimum set of nodes $N_{x,t}$ that covers all the non-revoked users who possess x at t by running the key update nodes selection algorithm $\mathsf{KUNodes}$ (the algorithm will be described in Section III-C) , then for each $v_x \in N_{x,t}$ it chooses a random exponent $\xi_{v_x,t} \in \mathbb{Z}_p$. The update key for (x,t) is set to

$$\mathsf{UK}_{x,t} = \{(E_{v_x} = R_{v_x} H_{\phi(x)}(x,t)^{\xi_{v_x,t}},\ E'_{v_x} = g^{\xi_{v_x,t}})\}_{v_x \in N_{x,t}}.$$

Then, all the update keys $\{\mathsf{UK}_{x,t}\}$ are published on the public bulletin board of the corresponding authority $AA_{\phi(x)}$ *at the beginning of the time slot t*. All the users in the system can access these update keys from the public bulletin board of each authority, which is also stored in the cloud.

Phase 3. Data Encryption by Owners

The owner first encrypts the data component with a content key by using symmetric encryption methods. It then runs the encryption algorithm $\mathsf{Encrypt}$ to encrypt the content key k. The encryption algorithm is defined as follows.

$\mathsf{Encrypt}(k, t_e, \mathbb{A} = (A, \rho), \mathsf{GPP}, \{\mathsf{PK}_d\}) \rightarrow (CT)$. k is the content key, t_e is a time slot, $\mathbb{A}$ is the access policy which is expressed by an LSSS matrix (A, ρ) [7], where A is an $m \times n$ matrix and ρ maps each row A_i of A to an attribute $\rho(i)$, and $\{\mathsf{PK}_d\}$ are the public keys related to (A, ρ), where $d \in \{\phi(\rho(i)) | 1 \leq i \leq m\}$.
The algorithm chooses two random vectors $\vec{v}, \vec{u} \in \mathbb{Z}_p^n$, with s and 0 as the first entry respectively, and for each $i \in \{1, 2, \ldots m\}$, it randomly picks $r_i \in \mathbb{Z}_p$. Let $\lambda_i = A_i \cdot \vec{v}$ and

[7]The details of using LSSS to express access policy is referred to Appendix A.

$\mu_i = A_i \cdot \vec{u}$. The ciphertext of the content key is

$$CT = \langle\ (A,\rho), t_e,\ C = k\cdot e(g,g)^s,\ \{C_{i,1} = e(g,g)^{\lambda_i} e(g,g)^{\alpha_{\rho(i)} r_i},$$
$$C_{i,2} = g^{\mu_i} g^{\beta_{\rho(i)} r_i}, C_{i,3} = g^{r_i}, C_{i,4} = H_{\phi(\rho(i))}(\rho(i), t_e)^{r_i}\}_{i=1}^{m}\ \rangle.$$

Phase 4. Data Decryption by Users

Any legal user can download the ciphertexts he interested. But only the users who possess eligible attributes (satisfying the access policy $\mathbb{A}$) at a particular time slot t_e can decrypt the ciphertext associated with $(\mathbb{A}, t_e)$. The decryption phase consists two steps: Decryption Key Generation and Ciphertext Decryption.

1. *Decryption Key Generation*

At each time slot t, every user can get update keys for each attribute it possess at this time slot from the public bulletin boards of the authorities. For each attribute x it possesses at time slot t, the user gid computes a decryption key $\mathsf{DK}_{gid,x,t}$ for this attribute by running the algorithm $\mathsf{DKeyCom}$ constructed as follows.

$\mathsf{DKeyCom}(\mathsf{SK}_{gid,x}, \mathsf{UK}_{x,t}) \rightarrow (\mathsf{DK}_{gid,x,t})$ or $\perp$. If $x \in S_{gid,t}$, i.e., gid possesses x at t, then there exists a unique v_x such that $v_x \in \mathsf{Path}(u_{x,gid}) \wedge v_x \in N_{x,t}$ (this is guaranteed by the $\mathsf{KUNodes}$ algorithm), the algorithm sets the decryption key for attribute x of gid at t to

$$\mathsf{DK}_{gid,x,t} = (D_{gid,x,t} = K_{gid,x,v_x}/E_{v_x},\ D'_{gid,x,t} = E'_{v_x}),$$

so that

$$D_{gid,x,t} = g^{\alpha_x} H(gid)^{\beta_x} H_{\phi(x)}(x,t)^{-\xi_{v_x,t}},$$
$$D'_{gid,x,t} = g^{\xi_{v_x,t}}.$$

If $x \notin S_{gid,t}$, as there does not exist such a v_x (this is guaranteed by the KUNodes algorithm as well), the algorithm outputs $\perp$.

Only the users who possess x at t can compute a valid decryption key, i.e., a user gid who is entitled x at a later time slot $t' > t$ is unable to compute a valid $\mathsf{DK}_{gid,x,t}$ even he has $\mathsf{SK}_{gid,x}$ (issued at t') and $\mathsf{UK}_{x,t}$. However, in practice, sometimes such a user is allowed to obtain a valid $\mathsf{DK}_{gid,x,t}$ as a special case. Later we will also give a discussion how to efficiently handle such special cases.

2. *Ciphertext Decryption*

Each user can freely get the ciphertext from the server, but he can decrypt the ciphertext only when the attributes he possesses at time slot t_e satisfy the access policy defined in the ciphertext. Suppose the user gid has sufficient attributes at time slot t_e, it can generate sufficient valid decryption keys $\{\mathsf{DK}_{gid,x,t}\}_{x\in S_{gid,t}}$. By using these decryption keys, the user gid can decrypt the ciphertext by running the decryption algorithm $\mathsf{Decrypt}$ constructed as follows.

$\mathsf{Decrypt}(CT, \mathsf{GPP}, \{\mathsf{PK}_d\}, \{\mathsf{DK}_{gid,x,t}\}_{x\in S_{gid,t}}) \rightarrow (M)$ or $\perp$. Assume CT is associated with $\langle (A,\rho), t_e\rangle$ and the set of decryption keys $\{\mathsf{DK}_{gid,x,t}\}_{x\in S_{gid,t}}$ is associated with the pair (gid, t). If $t \neq t_e$ or $S_{gid,t}$ does not satisfy (A,ρ), the algorithm outputs $\perp$, otherwise, it decrypts the ciphertext as follows:

1) Finds a set $I = \{i|\rho(i) \in S_{gid,t}\}$ and computes corresponding constants $\{w_i | i \in I\}$ such that $\sum_{i\in I} w_i A_i = (1,0,\ldots,0)$.
2) For each $i \in I$, computes
$$\bar{C}_i = \frac{C_{i,1}\cdot e(H(gid), C_{i,2})}{e(D_{gid,\rho(i),t}, C_{i,3})\cdot e(D'_{gid,\rho(i),t}, C_{i,4})}$$
$$= e(g,g)^{\lambda_i}\cdot e(H(gid), g)^{\mu_i}.$$
3) Computes
$$\prod_{i\in I}\bar{C}_i^{w_i} = e(g,g)^{\sum_{i\in I} w_i\lambda_i}\cdot e(H(gid),g)^{\sum_{i\in I} w_i\mu_i}$$
$$= e(g,g)^s,$$
and recovers the content key k by $k = C/e(g,g)^s$.

Then, the user can use this content key to further decrypt the encrypted data.

C. *Efficient Attribute Revocation in TAAC*

To achieve efficient attribute revocation, each authority maintains a revocation list for each attribute and generates update keys according to the revocation list for each time slot $\mathsf{RL}_{x,t}$. In particular, $\mathsf{RL}_{x,t}$ is a set of global identities, and $gid \in \mathsf{RL}_{x,t}$ means that gid was entitled x at some time slot $t' \leq t$ but is revoked x at time slot t.

The attribute revocation scheme contains three phases: Revocation List Determination, Minimum Cover Set Selection and Update Key Generation.

Phase 1. Revocation List Determination

At any time slot t, for any attribute $x \in \mathcal{U}_d$, AA_d sets the elements of $\mathsf{RL}_{x,t}$ at its choice to determine who possess x at t. Note that it is not required that $\mathsf{RL}_{x,t} \subseteq \mathsf{RL}_{x,t+1}$, i.e., the system can support *temporal revocation*. For example, a user gid is revoked the decryption privilege of attribute x at time slot t (i.e., $gid \in \mathsf{RL}_{x,t}$), but at time slot $t+1$ he needs to be re-granted the decryption privilege of x. Such a case can be easily supported by removing gid from $\mathsf{RL}_{x,t+1}$ (i.e., $gid \notin \mathsf{RL}_{x,t+1}$).

Phase 2. Minimum Cover Set Selection

At the begging of each time slot $t \in \mathcal{T}$, for any attribute $x \in \mathcal{U}$, according to the revocation list $\mathsf{RL}_{x,t}$ for x at t and the current state ST_x, the corresponding authority $AA_{\phi(x)}$ finds the minimal set of nodes for which it publishes update keys so that only the users who possess x at t can make use of the published update keys to generate corresponding decryption keys. The key update nodes selection algorithm $\mathsf{KUNodes}(\mathsf{ST}_x, \mathsf{RL}_{x,t})$ is defined in Algorithm 1.

Note that a similar algorithm was used in [15], and the difference is that our algorithm will prevent a user who is entitled x at a later time slot $t' > t$ from making use of the update keys for (x,t). Fig. 3 and Fig. 4 show the examples where $\mathsf{RL}_{x,t} = \emptyset$ and $\mathsf{RL}_{x,t} = \{gid_3\}$ respectively. Also note that the algorithm can be efficiently implemented with the above encoding rules.

Phase 3. Update Key Generation

In this phase, $AA_{\phi(x)}$ generates the $\mathsf{UK}_{x,t}$ according to ST_x and $\mathsf{RL}_{x,t}$ so that only the users who possess x at time slot t are able to obtain valid $\mathsf{UK}_{x,t}$. For any attribute x, the corresponding authority runs $\mathsf{UKeyGen}$ only once at the

Algorithm 1 $\mathsf{KUNodes}(\mathsf{ST}_x, \mathsf{RL}_{x,t})$

```
1:  $X_e, X_r, N_{x,t} \leftarrow \emptyset$
2:  $u_e \leftarrow$ the most left empty leaf node
3:  $X_e \leftarrow \mathsf{Path}(u_e)$
4:  for each $u_r \in \mathsf{RL}_{x,t}$ do
5:      add $\mathsf{Path}(u_r)$ to $X_r$
6:  end for
7:  $X_r \leftarrow X_r \setminus X_e$
8:  for each $v_e \in X_e$ do
9:      if $v_e$ is not a leaf node then
10:         $v_{lc} \leftarrow$ left child of $v_e$
11:         if $v_{lc} \notin X_r \cup X_e$ then
12:             add $v_{lc}$ to $N_{x,t}$
13:         end if
14:     end if
15: end for
16: for each $v_r \in X_r$ do
17:     if $v_r$ is not a leaf node then
18:         $v_{lc} \leftarrow$ left child of $v_r$
19:         if $v_{lc} \notin X_r$ then
20:             add $v_{lc}$ to $N_{x,t}$
21:         end if
22:         $v_{rc} \leftarrow$ right child of $v_r$
23:         if $v_{rc} \notin X_r$ then
24:             add $v_{rc}$ to $N_{x,t}$
25:         end if
26:     end if
27: end for
28: if $N_{x,t} = \emptyset$ then
29:     add the root node $x$ to $N_{x,t}$
30: end if
31: Return $N_{x,t}$
```

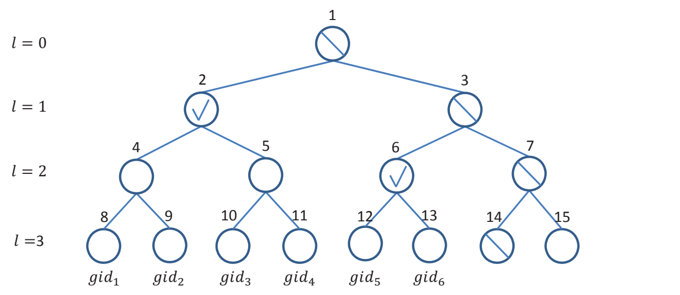
beginning of each time slot t, taking the current values of ST_x and $\mathsf{RL}_{x,t}$. Note that for any time slot t and attribute x, once $\mathsf{UKeyGen}(t, x, \mathsf{ST}_x, \mathsf{RL}_{x,t}, \mathsf{GPP}, \mathsf{MSK}_d) \rightarrow (\mathsf{UK}_{x,t})$ is run and $\mathsf{UK}_{x,t}$ is publish, the later changes of ST_x and $\mathsf{RL}_{x,t}$ will not affect $\mathsf{UK}_{x,t}$, and will be reflected at next time slot, i.e., $\mathsf{UK}_{x,t+1}$. Thus, by setting the revocation list $\mathsf{RL}_{x,t}$ and publishing the corresponding $\mathsf{UK}_{x,t}$ the authority achieves revocation control of x.

IV. Analysis of TAAC

A. Security Analysis

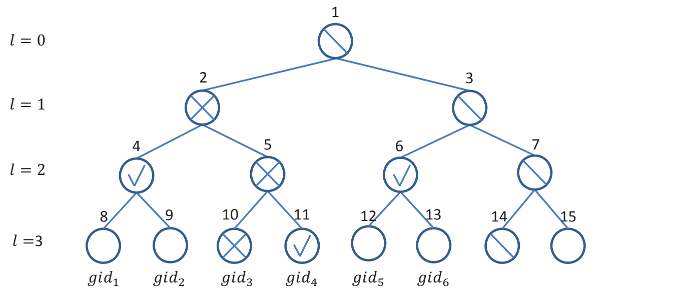
Similar to the underlying multi-authority CP-ABE scheme in Appendix D of [10], we will prove TAAC is secure in the generic bilinear group model previously used in [2], [16], [17], modeling H and $\{H_d\}_{d\in\mathbb{D}}$ as random oracles. Security in this model assures us that an adversary can not break TAAC with only black-box access to the group operations and $H, \{H_d\}_{d\in\mathbb{D}}$.

Theorem 1. *In the generic bilinear group model and random oracle model, no polynomial time adversary can break TAAC with non-negligible advantage in the security game of Sec. II-C.*



$h_x = 3; ctr_x = 6;$
$List_x = \{(gid_1, 8), (gid_2, 9), (gid_3, 10), (gid_4, 11), (gid_5, 12), (gid_6, 13)\}.$
$RL_{x,t} = \{\} \Rightarrow N_{x,t} = \{2,6\}.$

Fig. 3. $\mathsf{RL}_{x,t} = \emptyset$. In this example, the revocation list of attribute x at time slot t $RL_{x,t}$ is empty, and the minimum cover set is selected as $N_{x,t} = \{2,6\}$, i.e., the authority only needs to compute the update key $\mathsf{UK}_{x,t,2}$ for node 2 and $\mathsf{UK}_{x,t,6}$ for node 6 . The users $gid_1, gid_2, gid_3, gid_4$ can use the update key $\mathsf{UK}_{x,t,2}$ to compute their decryption keys at t and the users gid_5 and gid_6 can use the update key $\mathsf{UK}_{x,t,6}$ to compute their decryption keys at t.



$h_x = 3; ctr_x = 6;$
$List_x = \{(gid_1, 8), (gid_2, 9), (gid_3, 10), (gid_4, 11), (gid_5, 12), (gid_6, 13)\}.$
$RL_{x,t} = \{gid_3\} \Rightarrow N_{x,t} = \{4,6,11\}.$

Fig. 4. $\mathsf{RL}_{x,t} = \{gid_3\}$. In this example, gid_3 is in the revocation list of attribute x at time slot t $RL_{x,t}$, and the minimum cover set can be selected as $N_{x,t} = \{4,6,11\}$, i.e., the authority only needs to compute the update key $\{\mathsf{UK}_{x,t,4}, \mathsf{UK}_{x,t,6}, \mathsf{UK}_{x,t,11}\}$ for node 4, node 6 and node 11. The users gid_1, gid_2 can use the update key $UK_{x,t,4}$ to compute their decryption keys at t and the users gid_5 and gid_6 can use the update key $\mathsf{UK}_{x,t,6}$ to compute their decryption keys at t. For the user gid_4, it can also compute the decryption key by using $\mathsf{UK}_{x,t,4}$ at time slot t.

Proof: To cancel the $e(g,g)^s$ in the challenge ciphertext $C = M_b \cdot e(g,g)^s$, the adversary has to use the secret keys $\mathsf{SK}_{gid,x}$ in a way similar to that of the underlying multi-authority CP-ABE scheme in Appendix D of [10], i.e., for any $1 \le i \le m$, if the adversary has a corresponding secret key $\mathsf{SK}_{gid,\rho(i)}$, it can compute

$$\begin{aligned}\bar{C}_i =& C_{i,1} \cdot e(H(gid), C_{i,2}) / (e(K_{gid,\rho(i),v_{\rho(i)}}, C_{i,3}) \\ =& e(g,g)^{\lambda_i} \cdot e(H(gid), g)^{\mu_i} / e(R_{v_{\rho(i)}}, g^{r_i}).\end{aligned}$$

Note that if and only if the adversary has sufficient secret keys for some same gid, it is possible for him to use $e(g,g)^{\lambda_i} \cdot e(H(gid), g)^{\mu_i}$ to reconstruct $e(g,g)^s$. This is the key techniques used in TAAC to resist the attribute collusion attacks from different users. Furthermore, in TAAC, $e(g,g)^{\lambda_i} \cdot e(H(gid), g)^{\mu_i}$ is always blinded by an additional term $e(R_{v_{\rho(i)}}, g^{r_i})$ as shown above.

To cancel the term $e(R_{v_{\rho(i)}}, g^{r_i})$, the only way is to ob-

TABLE I
COMPARISON OF SCALABILITY

Scheme	Require Global CA?	Granularity
TAAC	No	Attribute Level
[18]	No	User Level
[19]	Yes	Attribute Level

tain corresponding update keys, as update keys are the only terms containing $R_{v_{\rho(i)}}$ except the secret keys. As the $R_{v_{\rho(i)}}$ in $E_{v_{\rho(i)}}$ (update key component) is always blinded by $H_{\phi(\rho(i))}(\rho(i),t)^{\xi_{v_{\rho(i)},t}}$, $e(R_{v_{\rho(i)}},g^{r_i})$ is always blinded by an additional term $e(H_{\phi(\rho(i))}(\rho(i),t)^{\xi_{v_{\rho(i)},t}},g^{r_i})$ due to $e(E_{v_{\rho(i)}},C_{i,3}) = e(R_{v_{\rho(i)}},g^{r_i})\cdot e(H_{\phi(\rho(i))}(\rho(i),t)^{\xi_{v_{\rho(i)},t}},g^{r_i})$. Note that the only ciphertext component related to $i,\rho(i),r_i,H_{\phi(\rho(i))}$ in the challenge ciphertext is for time slot t_e, i.e., $C_{i,4} = H_{\phi(\rho(i))}(\rho(i),t_e)^{r_i}$, we have that only if the adversary is using the update key components for the time slot t_e, can he cancel the term $e(H_{\phi(\rho(i))}(\rho(i),t)^{\xi_{v_{\rho(i)},t}},g^{r_i})$ and obtain $e(R_{v_{\rho(i)}},g^{r_i})$.

In summary, only if there exists a *gid*, such that the adversary has sufficient secret keys related to *gid* and sufficient update key components related to *gid* and time slot t_e, can the adversary cancel the $e(g,g)^s$. Note "sufficient" means that *gid* possesses eligible attributes (satisfying (A,ρ)) at time slot t_e. However, in the security game the adversary is restricted to make such key queries. This implies that the adversary cannot attain non-negligible advantage in the security game. The detailed proof is described in Appendix B. ■

B. Scalability Analysis

We conduct the scalability comparison of TAAC with two existing data access control schemes [18], [19] for multi-authority cloud storage systems. As shown in Table. I, TAAC does not require any global certificate authority and can support the access control on attribute level rather than on user level. That is in TAAC and [19], when an attribute is revoked from a user, the other attributes of this user are not affected, i.e., the user can use its remaining attributes to decrypt the ciphertexts. However, in [18], when an attribute is revoked from a user, the user lose the decryption privilege of all his attributes.

TAAC has high scalability also in the sense that it can be easily extended to support the following two scenarios:

Scenario 1: *In TAAC, a user gid who does not have sufficient attributes at time slot t cannot decrypt the ciphertext* CT *which is encrypted under the policy* $(\mathbb{A},t)$*, even when the user is entitled sufficient attributes at a later time slot* $t'(t<t')$*. However, in cloud storage systems, this user may be authorized to have the decryption privilege of the ciphertext* CT *due to some reasons.*

TAAC can easy handle such special case as follows. For every attribute x required for decryption at time slot t, the authority $AA_{\phi(x)}$ can publish a special update key

$$\mathsf{UK}_{x,t,gid} = (E_{u_{x,gid}} = R_{u_{x,gid}}H_{\phi(x)}(x,t)^{\xi_{u_{x,gid}}},\ E'_{u_{x,gid}} = g^{\xi_{u_{x,gid}}}),$$

where $u_{x,gid}$ is the leaf node assigned to *gid* and $\xi_{u_{x,gid}} \in \mathbb{Z}_p$ is randomly chosen. Note that such an update key is useless to other users, and the user *gid* can use this update key to obtain only the decryption privilege of x at time slot t.

Scenario 2: *The ciphertexts in TAAC only associated with attributes at* only one *time slot* t_e *and only the users who have sufficient attributes at this time slot* t_e *can decrypt them. However, in cloud storage systems, sometimes it allows the users who have sufficient attributes at multiple time slots or a time period to decrypt the ciphertexts.*

TAAC can also efficiently deal with this scenario by modifying the encryption algorithm. The encryption algorithm Encrypt can be modified to a general one which encrypts a message using $((A,\rho),T_e \subseteq \mathcal{T})$. In particular, $C_{i,4}$ can be modified to $C_{i,4} = \{H_{\phi(\rho(i))}(\rho(i),t_e)^{r_i}\}_{t_e\in T_e}$, and the ciphertext for $((A,\rho),T_e)$ contains $(3+|T_e|)m+1$ elements.

C. Performance Analysis

We conduct the performance analysis of TAAC by comparing with [18] and [19] under the metrics of *Storage Overhead*, *Communication Cost* and *Computation Cost*.

1) Storage Overhead

The storage overhead is one of the most significant issues of the access control scheme in cloud storage systems. The size of encrypted data and ciphertext contribute the main storage overhead on the server, which is almost the same in these schemes. For the storage overhead on the owner, besides the public keys, the scheme in [18] also requires the owner to store all the secret encryption exponent in order to re-encrypt the ciphertext during the revocation. The storage overhead on each user comes from the secret keys. In TAAC, each user should also hold the decryption key for every time slot. However, in [18], each non-revoked user should keep a component for each ciphertext which contains the revoked attribute. Due to the large number of ciphertexts, the scheme in [18] incurs a heavy storage overhead on each user. Compared with [18] and [19], TAAC requires each authority to store a set of values $\{R_{v_x}\}_{v_x\in T_x}$ for each attribute $x\in\mathcal{U}_{\phi(x)}$, but this storage overhead can be reduced by using Pseudo Random Function (PRF) and storing a seed s_x in $\mathsf{MSK}_{\phi(x)}$.

2) Communication Cost

The communication cost of normal access control is almost the same in these three access control schemes, which are all based on the CP-ABE methods. Here, we only consider the communication cost for the revocation of an attribute. Table II describes the comparison of communication cost for the revocation of attribute x. In TAAC, the communication cost comes from the update keys of each attribute from each *AA* to each user, which is linear to the number of total attributes that all the users possess in the system. The scheme in [18] requires the owner to transfer a new component of each ciphertext which is associated with the revoked attribute x to all the non-revoked users. The revocation communication cost in [18] is

TABLE II
COMPARISON OF COMMUNICATION COST FOR REVOCATION OF ATTRIBUTE x

Scheme	Owner-User	Owner-Server	User-Server	AA-Owner	AA-User
TAAC	None	None	None	None	$\sum_{k=1}^{n_u} 2\|S_{gid_k,t}\| \cdot \|p\|$
[18]	$n_{non,x} \sum_{i=1}^{n_o} n_{c,x,o_i} \cdot \|p\|$	$\sum_{i=1}^{n_o} n_{c,x,o_i} \cdot \|p\|$	None	None	None
[19]	None	$2n_o \cdot \|p\|$	None	$n_o \cdot \|p\|$	$(2n_{non,x} + r + \sum_{i=1}^{r} \|S'_{gid_i}\|) \cdot \|p\|$

$|p|$ is the size of elements in the group with order p; n_o is the number of owners and n_u is the number of users in the sytem; $n_{non,x}$ is the number of non-revoked users who possess x; n_{c,x,o_i} is the number of ciphertexts associated with x from generated by owner o_i; $|S_{gid_k,t}|$ is the number of attributes the user gid_k possesses at time slot t; $|S'_{gid_i}|$ is the number of attribute the revoked user gid_i holds after the revocation; r is the number of revoked users.

linear to the number of ciphertexts that are associated with the revoked attribute x. Due to the large number of ciphertexts in cloud storage systems, the scheme in [18] incurs a heavy communication cost. In [19], the revocation communication cost is much less than TAAC and [18], but it incurs a very heavy computation cost.

3) Computation Cost

Compared with [18] and [19], TAAC has the same complexity of encryption/decryption. Here we only compare the computation cost for the revocation of an attribute x, as shown in Table III. In TAAC, the revocation computation cost comes from the update key generation (which is linear to the number of the non-revoked users who possess x) and the decryption key computation (which is linear to the total number of attributes of all the users in the system at time slot t) in each time slot t. In [18], the revocation computation cost comes from the re-encryption of ciphertexts, which is linear to the number of ciphertexts associated with x. In [19], the revocation computation cost comes from two parts: 1) the key update for all the users who has any attribute from the authority $AA_{\phi(x)}$, which is linear to the total number of attributes in all the users which are issued by $AA_{\phi(x)}$; 2) the re-encryption for all the ciphertexts associated with any attribute from $AA_{\phi(x)}$, which is linear to the total number of attributes appeared in all the ciphertexts which are issued by $AA_{\phi(x)}$. In cloud storage systems, due to the large number of ciphertexts, [18] and [19] incur much more computation cost than TAAC.

To further reduce the revocation computation cost in TAAC, we design a key update nodes selection algorithm KUNodes to select the minimum set of nodes that covers all the non-revoked users who possess the revoked attribute at time slot t. Let $|N_{x,t}|$ denote the number of nodes in the minimum cover set $N_{x,t}$. Thus, we can reduce our revocation computation cost from $O(n_{non,x}) \cdot \textbf{Exp}. + O(n_{non,x} + n_{a,t}) \cdot \textbf{Mul}.$ to $O(|N_{x,t}|) \cdot \textbf{Exp}. + O(|N_{x,t}| + n_{a,t}) \cdot \textbf{Mul}.$ In practical cloud storage systems, the number of nodes in the minimum cover set $|N_{x,t}|$ is approximate to $r \log \frac{n_{u,x}}{r}$, where $n_{u,x}$ is the number of users who possess x and r is the number of revoked users.

TABLE III
COMPARISON OF COMPUTATION COST FOR REVOCATION OF ATTRIBUTE x

Scheme	Revocation Computation Cost
TAAC	$O(n_{non,x}) \cdot \textbf{Exp}. + O(n_{non,x} + n_{a,t}) \cdot \textbf{Mul}.$
[18]	$O(n_{c,x}) \cdot \textbf{Exp}. + O(n_{c,x}) \cdot \textbf{Mul}.$
[19]	$O(n_{a,u,\phi(x)}) \cdot \textbf{Exp}. + O(n_{a,c,\phi(x)}) \cdot \textbf{Mul}.$

Exp.: the exponentiate operation in the group; **Mul.**: the multiplication operation in the group; $n_{non,x}$: num of non-revoked users possess x; $n_{a,t}$: total num of attributes of all the users in the system at time slot t; $n_{c,x}$: num of ciphertexts associated with x; $n_{a,u,\phi(x)}/n_{a,c,\phi(x)}$: total num of attributes in all the users/ciphertexts issued by $AA_{\phi(x)}$.

V. RELATED WORK

Cryptographic techniques are well applied to access control for remote storage systems [20]–[22]. Traditional public key encryption (PKE) based schemes [23], [24] either incurs complicated key management or produces multiples copies of encrypted data with different user's keys. Some methods [25]–[27] deliver the key management and distribution from data owners to the remote server under the assumption that the server is trusted or semi-trusted. However, the server is not fully trusted in cloud storage systems and thus these methods cannot be applied to access control for cloud storage systems.

Attribute-based Encryption (ABE) is a promising technique that is designed for access control of encrypted data. After Sahai and Waters introduced the first ABE scheme [28], Goyal *et al.* [29] formulated the ABE into two complimentary forms: Key-Policy ABE (KP-ABE) and Ciphertext-Policy (CP-ABE). There are a number of works used ABE to realize fine-grained access control for outsourced data [2], [11], [12], [30]. In these schemes, a trusted single authority is used to manage the attributes and issue keys. However, in real storage systems, the authority can fail or be corrupted, which may leak out the data since the authority can decrypt all the encrypted data. Moreover, the authority may become the performance bottleneck in the large scale cloud storage systems.

Some new cryptographic methods are proposed to the multi-authority ABE problem [7]–[10], [31], [32]. Chase [7] proposed a solution that introduced a global identity to tie users' keys together. The proposed scheme also relies on a central authority to provide a final secret key to integrate the secret keys from different attribute authorities. However, the central authority would be able to decrypt all the ciphertexts in Chase's scheme, because it holds the master key of the system. Thus, the central authority would be a vulnerable point for security attacks and a performance bottleneck for large scale systems. Another limitation of Chase's scheme is that it can only express a strict "AND" policy over a pre-determined set of authorities. To improve the Chase's scheme, Muller *et al.* [8] proposed a multi-authority ABE scheme that can handle any expressions in LSSS access policy, but it also requires a centralized authority. Chase *et al.* [9] also proposed a method to remove the central authority by using a distributed PRF (pseudo-random function). But it has the same limitation to strict "AND" policy of pre-determined authorities. Lin *et al.* [31] proposed a decentralized scheme based on threshold mechanism. In their scheme, the set of authorities is pre-determined and it requires the interaction among the authorities during the system setup. Their scheme can tolerate collusion attacks for up to m colluding users, where m is a system parameter chosen at setup time. In [10], Lewko *et al.* proposed a new comprehensive scheme, which does not require any central authority. It is secure against any collusion attacks and it can process the access policy expressed in any Boolean formula over attributes. However, their method is constructed in composite order bilinear groups that incurs heavy computation cost. They also proposed a multi-authority CP-ABE scheme constructed in prime order group, but they did not consider attribute revocation, which is one of the major challenges in access control for multi-authority cloud storage systems.

There are a number of works about the revocation in ABE systems in the cryptography literature [2], [5], [11], [12], [33]–[36]. However, these methods either only support the user level revocation or rely on the server to conduct the attribute revocation. Moreover, these attribute revocation methods are designed only for ABE systems with single authority. In [36], Li *et al.* proposed an attribute revocation method for multi-authority ABE systems, but their methods is only for KP-ABE systems. Ruj *et al.* [18] designed a DACC scheme and proposed a revocation method for the Lewko and Waters' decentralized ABE scheme. However, their attribute revocation method incurs a heavy communication cost since it requires the data owner to transmit a new ciphertext component to every non-revoked user. In [19], the authors proposed an access control method for multi-authority systems in cloud storage, where a new multi-authority CP-ABE scheme and an attribute revocation method are proposed. However, their attribute revocation method is not efficient, because they require the re-encryption of a large number of ciphertexts. In our scheme, we propose an attribute revocation method without needing to re-encrypt any ciphertexts in the cloud.

VI. Conclusion

This paper proposed an efficient data access control scheme for multi-authority cloud storage systems denoted as TAAC, where the authorities are independent from each other and no central authority is needed. TAAC can efficiently achieve temporal access control on attribute-level rather than on user-level. Compared to the existing schemes with attribute revocation functionality, TAAC does not require re-encryption of any ciphertext when the attribute revocation happens, which can greatly reduce the computation cost of the attribute revocation. The analysis results show that TAAC is highly efficient, scalable, and flexible to multi-authority cloud storage systems in practice.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2009.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P'07).* IEEE Computer Society, 2007, pp. 321–334.

[3] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proceedings of the 4th International Conference on Practice and Theory in Public Key Cryptography (PKC'11).* Springer, 2011, pp. 53–70.

[4] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08).* Springer, 2008, pp. 579–591.

[5] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07).* ACM, 2007, pp. 195–203.

[6] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology - EUROCRYPT'10.* Springer, 2010, pp. 62–91.

[7] M. Chase, "Multi-authority attribute based encryption," in *Proceedings of the 4th Theory of Cryptography Conference on Theory of Cryptography (TCC'07).* Springer, 2007, pp. 515–534.

[8] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC'08).* Springer, 2008, pp. 20–36.

[9] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09).* ACM, 2009, pp. 121–130.

[10] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology - EUROCRYPT'11.* Springer, 2011, pp. 568–588.

[11] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10).* ACM, 2010, pp. 261–270.

[12] 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, 2011.

[13] S. Jahid, P. Mittal, and N. Borisov, "Easier: encryption-based access control in social networks with efficient revocation," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11).* ACM, 2011, pp. 411–415.

[14] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proceedings of the 32nd Annual International Cryptology Conference: Advances in Cryptology - CRYPTO'2012.* Springer, 2012, pp. 199–217.

[15] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *ACM Conference on Computer and Communications Security(CCS'08).* ACM, 2008, pp. 417–426.

[16] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proceeding of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology - EUROCRYPT '97*. Springer, 1997, pp. 256–266.

[17] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proceeding of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology - EUROCRYPT '05*. Springer, 2005, pp. 440–456.

[18] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed Access Control in Clouds," in *Proceeding of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'11)*. IEEE, 2011, pp. 91–98.

[19] K. Yang and X. Jia, "Attribute-based access control for multi-authority systems in cloud storage," in *Proceedings of the 32th IEEE International Conference on Distributed Computing Systems (ICDCS'12)*. IEEE, 2012, pp. 1–10.

[20] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST'03)*. USENIX, 2003.

[21] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage," in *Proceedings of the Network and Distributed System Security Symposium (NDSS'03)*. The Internet Society, 2003.

[22] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," *Electronic Colloquium on Computational Complexity (ECCC)*, no. 043, 2002.

[23] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," in *Proceedings of the first ACM Cloud Computing Security Workshop (CCSW'09)*. ACM, 2009, pp. 103–114.

[24] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *Journal of Computer Security*, vol. 19, no. 3, pp. 367–397, 2011.

[25] E. Damiani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Key management for multi-user encrypted databases," in *Proceedings of the 2005 ACM Workshop On Storage Security And Survivability (StorageSS'05)*. ACM, 2005, pp. 74–83.

[26] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB'07)*. ACM, 2007, pp. 123–134.

[27] W. Wang, Z. Li, R. Owens, and B. K. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the first ACM Cloud Computing Security Workshop (CCSW'09)*. ACM, 2009, pp. 55–66.

[28] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology - EUROCRYPT'05*. Springer, 2005, pp. 457–473.

[29] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*. ACM, 2006, pp. 89–98.

[30] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM'10)*. IEEE, 2010, pp. 534–542.

[31] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Inf. Sci.*, vol. 180, no. 13, pp. 2618–2632, 2010.

[32] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11)*. ACM, 2011, pp. 386–390.

[33] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure attribute-based systems," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*. ACM, 2006, pp. 99–112.

[34] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proceedings of the Third International Conference on Pairing-Based Cryptography (Pairing'09)*. Springer, 2009, pp. 248–265.

[35] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "An efficient and secure user revocation scheme in mobile social networks," in *Proceedings of the Global Communications Conference (GLOBECOM'11)*. IEEE, 2011, pp. 1–5.

[36] 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 Transactions on Parallel and Distributed Systems*, 2012.

[37] A. Beimel, "Secure schemes for secret sharing and key distribution," *DSc dissertation*, 1996.

Appendix A
Access Policy

Definition 3 (Access Structure [37])**.** *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 non-empty 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 ABE, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ contains the authorized sets of attributes. As of previous work in ABE, we focus on monotone access structures in this paper. It is shown in [37] that any monotone access structure can be realized by a linear secret sharing scheme. Here we use the definition from [3], [37].

Definition 4 (Linear Secret-Sharing Schemes (LSSS) [3])**.** *A secret sharing scheme Π over a set of parties $\mathbb{P}$ is called linear (over $\mathbb{Z}_p$) if*

1) *The shares for each party form a vector over $\mathbb{Z}_p$.*
2) *There exists a matrix A called the share-generating matrix for Π. The matrix A has l rows and n columns. For $i = 1, \ldots, l$, the i^{th} row of A is labeled by a party $\rho(i)$(ρ is a function from $\{1, \ldots, l\}$ to $\mathbb{P}$). When we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $A\vec{v}$ is the vector of l shares of the secret s according to Π. The share $(A\vec{v})_i$ belongs to party $\rho(i)$.*

It is shown in [37] that every linear secret-sharing scheme according to the above definition also enjoys the linear reconstruction property, defined as follows: Suppose that Π is an LSSS for access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. There exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A. For any unauthorized set, no such constants exist. In this paper, we use LSSS matrix (A, ρ) to express an access policy.

Appendix B
Proof of Theorem 1

Similar to the proof of the underlying multi-authority CP-ABE scheme in Appendix D of [10], we will now prove our scheme is secure in the generic bilinear group model

TABLE IV
Possible query terms from the adversary

α_x		
β_x	$\beta_x\beta_y$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})(\alpha_y+h_{gid'}\beta_y+\gamma_{v_y})$
h_{gid}	$\beta_x h_{gid}$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})\xi_{v_y,t}$
$h_{x,t}$	$\beta_x h_{y,t}$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})(\gamma_{v_y}+h_{y,t}\xi_{v_y,t})$
$\alpha_x+h_{gid}\beta_x+\gamma_{v_x}$	$\beta_x(\alpha_y+h_{gid}\beta_y+\gamma_{v_y})$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})r_i$
$\xi_{v_x,t}$	$\beta_x\xi_{v_y,t}$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})(\mu_i+\beta_{\rho(i)}r_i)$
$\gamma_{v_x}+h_{x,t}\xi_{v_x,t}$	$\beta_x(\gamma_{v_y}+h_{y,t}\xi_{v_y,t})$	$(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})h_{\rho(i),t_e}r_i$
r_i	$\beta_x r_i$	
$\mu_i+\beta_{\rho(i)}r_i$	$\beta_x(\mu_i+\beta_{\rho(i)}r_i)$	$\xi_{v_x,t}\xi_{v_y,t'}$
$h_{\rho(i),t_e}r_i$	$\beta_x h_{\rho(i),t_e}r_i$	$\xi_{v_x,t}(\gamma_{v_y}+h_{y,t}\xi_{v_y,t'})$
$\lambda_i+\alpha_{\rho(i)}r_i$		$\xi_{v_x,t}r_i$
	$h_{gid}h_{gid'}$	$\xi_{v_x,t}(\mu_i+\beta_{\rho(i)}r_i)$
	$h_{gid}h_{x,t}$	$\xi_{v_x,t}h_{\rho(i),t_e}r_i$
	$h_{gid}(\alpha_x+h_{gid}\beta_x+\gamma_{v_x})$	
	$h_{gid}\xi_{v_x,t}$	$(\gamma_{v_x}+h_{x,t}\xi_{v_x,t})(\gamma_{v_y}+h_{y,t'}\xi_{v_y,t'})$
	$h_{gid}(\gamma_{v_x}+h_{x,t}\xi_{v_x,t})$	$(\gamma_{v_x}+h_{x,t}\xi_{v_x,t})r_i$
	$h_{gid}r_i$	$(\gamma_{v_x}+h_{x,t}\xi_{v_x,t})(\mu_i+\beta_{\rho(i)}r_i)$
	$h_{gid}(\mu_i+\beta_{\rho(i)}r_i)$	$(\gamma_{v_x}+h_{x,t}\xi_{v_x,t})h_{\rho(i),t_e}r_i$
	$h_{gid}h_{\rho(i),t_e}r_i$	
		r_ir_j
	$h_{x,t}h_{y,t'}$	$r_i(\mu_j+\beta_{\rho(j)}r_j)$
	$h_{x,t}(\alpha_y+h_{gid}\beta_y+\gamma_{v_y})$	$r_ih_{\rho(j),t_e}r_j$
	$h_{x,t}\xi_{v_y,t'}$	
	$h_{x,t}(\gamma_{v_y}+h_{y,t'}\xi_{v_y,t'})$	$(\mu_i+\beta_{\rho(i)}r_i)(\mu_j+\beta_{\rho(j)}r_j)$
	$h_{x,t}r_i$	$(\mu_i+\beta_{\rho(i)}r_i)h_{\rho(j),t_e}r_j$
	$h_{x,t}(\mu_i+\beta_{\rho(i)}r_i)$	
	$h_{x,t}h_{\rho(i),t_e}r_i$	$h_{\rho(i),t_e}r_ih_{\rho(j),t_e}r_j$

previously used in [2], [16], [17], modeling H and $\{H_d\}_{d\in\mathbb{D}}$ as random oracles. Security in this model assures us that an adversary can not break our scheme with only black-box access to the group operations and $H,\{H_d\}_{d\in\mathbb{D}}$.

Proof: We describe the generic bilinear model as [10], [17]. Let ψ_0 and ψ_1 be two random encodings of the additive group $\mathbb{Z}_p$. In particular, each of ψ_0,ψ_1 is an injective map from $\mathbb{Z}_p$ to $\{0,1\}^{\bar{\lambda}}$ for $\bar{\lambda} > 3\log(p)$. We define the groups $G_0=\{\psi_0(x): x\in\mathbb{Z}_p\}$ and $\{G_1=\psi_1(x): x\in\mathbb{Z}_p\}$, and assume we have access to oracles which compute the induced group operations in G_0 and G_1 and an oracle with computes a non-degenerate bilinear map $e: G_0\times G_0\to G_1$. We refer to G_0 as a generic bilinear group.

In our security game, the adversary must distinguish between $C=M_0e(g,g)^s$ and $C=M_1e(g,g)^s$. We can alternatively consider a modified game, where the attacker must distinguish between $C=e(g,g)^s$ and $C=e(g,g)^{\bar{s}}$ for $\bar{s}$ chosen uniformly randomly from $\mathbb{Z}_p$. This is same modification employed in [2], and it is justified by a simple hybrid argument.

We will simply our notation as follows: let g denote $\psi_0(1)$, g^x denote $\psi_0(x)$, $e(g,g)$ denote $\psi_1(1)$, and $e(g,g)^y$ denote $\psi_1(y)$.

We now simulate the modified security game in the generic bilinear group model where C is set to $e(g,g)^{\bar{s}}$.

Setup.

1) The simulator $\mathcal{B}$ runs the GlobalSetup algorithm, and gives g to the adversary $\mathcal{A}$.
2) $\mathcal{A}$ gives $\mathcal{B}$ an index set $\mathbb{D}_c\subset\mathbb{D}$ as the corrupt AAs.
3) For each attribute $x\in\cup_{d\in\mathbb{D}\setminus\mathbb{D}_c}\mathcal{U}_d$:
    a) $\mathcal{B}$ randomly chooses values $\alpha_x,\beta_x\in\mathbb{Z}_p$, queries the group oracles for $e(g,g)^{\alpha_x},g^{\beta_x}$, and gives these to $\mathcal{A}$.
    b) $\mathcal{B}$ determines an integer h_x to allow at most 2^{h_x} users to possess x, and sets $T_x=\{1,2,\ldots,2^{h_x+1}-1\}$. And for each $v_x\in T_x$ $\mathcal{B}$ randomly chooses $\gamma_{v_x}\in\mathbb{Z}_p$, queries the group oracle for $R_{v_x}=g^{\gamma_{v_x}}$, and stores them.
    c) $\mathcal{B}$ initializes the state $\mathsf{ST}_x=(x,h_x,ctr_x=0,List_x=\emptyset)$.

Phase 1.

- Hash Function H: If $\mathcal{A}$ requests $H(gid)$ from some gid first time, $\mathcal{B}$ chooses a random value $h_{gid}\in\mathbb{Z}_p$, queries the group oracle for $g^{h_{gid}}$, and gives this value to $\mathcal{A}$ as $H(gid)$. $\mathcal{B}$ stores this value so that it can reply consistently to any subsequent requests fro $H(gid)$.

- Hash Functions $\{H_d\}$: If $\mathcal{A}$ requests $H_{\phi(x)}(x,t)$ from some pair (x,t) first time, $\mathcal{B}$ chooses a random value $h_{x,t} \in \mathbb{Z}_p$, queries the group oracle for $g^{h_{x,t}}$, and gives this value to $\mathcal{A}$ as $H_{\phi(x)}(x,t)$. $\mathcal{B}$ stores this value so that it can reply consistently to any subsequent requests for $H_{\phi(x)}(x,t)$.
- $\mathsf{SKQ}(gid,x)$: When $\mathcal{A}$ requests a secret key for some pair (gid,x) where $x \in \cup_{d\in\mathbb{D}\setminus\mathbb{D}_c}\mathcal{U}_d$, $\mathcal{B}$ adds $(gid, u_{x,gid} = 2^{h_x} + ctr_x)$ to $List_x$, computes $\mathsf{SK}_{gid,x} = \{K_{gid,x,v_x} = g^{\alpha_x}H(gid)^{\beta_x}R_{v_x}\}_{v_x\in\mathsf{Path}(u_{x,gid})}$ using the group oracle and stored $\{R_{v_x}\}$, and sets $ctr_x = ctr_x + 1$. Then $\mathcal{B}$ gives $\mathsf{SK}_{gid,x}$ to $\mathcal{A}$. If $H(gid)$ has not been requested before, it is determined as above.
- $\mathsf{UKQ}(t,x,\mathsf{RL}_{x,t})$: When $\mathcal{A}$ requests a update key form some valid tuple $(x,t,\mathsf{RL}_{x,t})$ where $x \in \cup_{d\in\mathbb{D}\setminus\mathbb{D}_c}\mathcal{U}_d$ and $(x,t,\mathsf{RL}_{x,t})$ satisfies the restriction 1 in the definition of security game, $\mathcal{B}$ computes the node set $N_{x,t} \leftarrow \mathsf{KUN}(\mathsf{ST}_x, \mathsf{RL}_{x,t})$, and for each $v_x \in N_{x,t}$, it chooses a random exponent $\xi_{v_x,t} \in \mathbb{Z}_p$. Then $\mathcal{B}$ computes $\mathsf{UK}_{x,t} = \{(E_{v_x} = R_{v_x}H_{\phi(x)}(x,t)^{\xi_{v_x,t}}, E'_{v_x} = g^{\xi_{v_x,t}})\}_{v_x\in N_{x,t}}$ using the group oracle and stored $\{R_{v_x}\}$, and gives $\mathsf{UK}_{x,t}$ to $\mathcal{A}$. If $H_{\phi(x)}(x,t)$ has not been requested before, it is determined as above.

Challenge Phase. $\mathcal{A}$ submits to $\mathcal{B}$ an LSSS matrix (A,ρ) and a time slot t_e for the challenge ciphertext and additionally supplies $\mathcal{B}$ with $e(g,g)^{\alpha_x}, g^{\beta_x}$ values for any attribute x controlled by corrupt authorities that appear in (A,ρ). $\mathcal{B}$ then checks that these are valid group elements by query the group oracles.

$\mathcal{B}$ chooses random vectors $\vec{v} = (s, v_2, \ldots, v_n), \vec{u} = (0, u_2, \ldots, u_n) \in \mathbb{Z}_p^n$. For each $i \in \{1,2,\ldots,m\}$, $\mathcal{B}$ computes $\lambda_i = A_i \cdot \vec{v}, \mu_i = A_i \cdot \vec{u}$, and chooses a random $r_i \in \mathbb{Z}_p$. Then $\mathcal{B}$ chooses a random $\bar{s} \in \mathbb{Z}_p$ and (using the group oracles) computes the challenge cipherttext:

$$C = e(g,g)^{\bar{s}},\ \{C_{i,1} = e(g,g)^{\lambda_i}e(g,g)^{\alpha_{\rho(i)}r_i},\ C_{i,2} = g^{\mu_i}g^{\beta_{\rho(i)}r_i},\ C_{i,3} = g^{r_i},\ C_{i,4} = H_{\phi(\rho(i))}(\rho(i),t_e)^{r_i}\}_{i=1}^m.$$

The challenge ciphertext is given to $\mathcal{A}$.

Phase 2. Same to Phase 1.

We will argue that will all but negligible probability, $\mathcal{A}$'s view in the above simulation is identically distributed to what its view would have been if C had been set to $e(g,g)^s$ instead of $e(g,g)^{\bar{s}}$. This shows that $\mathcal{A}$ cannot attain a non-negligible advantage in the modified security game, and hence cannot attain a non-negligible advantage in the real security game.

We condition the event that each of $\mathcal{A}$'s queries to the group oracles have input values that were given to $\mathcal{A}$ during the simulation or were received from the oracles in response to previous queries. This event occurs with high probability. Since each ψ_0, ψ_1 is a random injective map from $\mathbb{Z}_p$ into a set of $> p^3$ elements, the probability of $\mathcal{A}$ being able to guess an element in the images of ψ_0, ψ_1 which it has not previously obtained is negligible.

Under this condition, we can think of $\mathcal{A}$'s queries as a multi-variate polynomial in the variables $\bar{s}, \alpha_x, \beta_x, \gamma_{v_x}, \xi_{v_x,t}, \lambda_i, \mu_i, r_i, h_{gid}, h_{x,t}$, where x ranges over the attributes controlled by uncorrupt authorities, v_x ranges over the nodes of binary tree corresponding to x where x ranges over the attributes controlled by uncorrupt authorities, i ranges over the rows of the challenge access matrix, gid ranges over the allowed identities, and (x,t) ranges over the allowed $(attribute, timeslot)$ pairs. (We can also think of λ_i, μ_i as linear combinations of the variables $s, v_2, \ldots, v_n, u_2, \ldots, u_n$.)

We now further condition on the event that for each pair of the queries $\mathcal{A}$ makes corresponding to different polynomials, $\mathcal{A}$ receives different answers. In other words, we are conditioning on the event that our random assignment of values to the variables $\bar{s}, \alpha_x, \beta_x, \gamma_{v_x}, \xi_{v_x,t}, \lambda_i, \mu_i, r_i, h_{gid}, h_{x,t}$ does not happen to be a zero of the difference of two query polynomials. (Here, we are treating λ_i as a linear combination of variables $s, v_2, \ldots, v_m$ and μ_i as a linear combination of the variables $u_2, \ldots, v_n$.) This event occurs with high probability, which we can see by using the Schwartz-Zippel lemma and a union bound, since our polynomials have degree at most 4 (which we will see below when we enumerate all the types of queries $\mathcal{A}$ can make).

Since $\bar{s}$ only appears as $e(g,g)^{\bar{s}}$, the only queries $\mathcal{A}$ can make involving $\bar{s}$ are of the form $ct + other\ terms$, where c is a constant. $\mathcal{A}$'s view can only differ when $\bar{s} = s$ if $\mathcal{A}$ can make two queries f and f' into G_1 where these are unequal as polynomials but become the same when we substitute s for $\bar{s}$. This implies $f - f' = cs - c\bar{s}$ from some constant c. We may conclude that $\mathcal{A}$ can then make the query cs.

We will show that $\mathcal{A}$ cannot make a query of the form cs, and therefore arrive at a contradiction. By examining the values given to $\mathcal{A}$ doing the simulation, we see that $\mathcal{A}$ can only form queries which are linear combinations of $1, \bar{s}$, and the terms appearing in Table IV.

We note that $\mathcal{A}$ additionally knows the values of α_x, β_x for attributes x which are controlled by corrupt authorities, so these are known constants which can appear in coefficients of the terms in Table IV in a linear combination.

We recall that $\lambda_i = A_i\vec{v}$ where $\vec{v} = (s, v_2, \ldots, v_n)$. Since these are the only appearances of s in the above table, in order to form a query cs $\mathcal{A}$ must choose constants $\{w_i\}$ such that $\sum_i w_i\lambda_i = cs$ and form :

$$\sum_i w_i(\lambda_i + \alpha_{\rho(i)}r_i).$$

For any terms $w_i\alpha_{\rho(i)}r_i$ where $\rho(i)$ is an attribute controlled by a corrupt authority, $\mathcal{A}$ knows the value $\alpha_{\rho(i)}$, and so can form the term $-w_i\alpha_{\rho(i)}r_i$ in order to cancel this from the above polynomial. For terms $w_i\alpha_{\rho(i)}r_i$ where $\rho(i)$ is an attribute controlled by an uncorrupt authority, $\mathcal{A}$ must cancel this term by using:

$$-w_i \cdot (\alpha_{\rho(i)} + h_{gid}\beta_{\rho(i)} + \gamma_{v_{\rho(i)}})r_i,$$

which leaves an additional term of $-w_i(h_{gid}\beta_{\rho(i)} + \gamma_{v_{\rho(i)}})r_i$ to be canceled. We also note that $\mathcal{A}$ only has access to a term $(\alpha_{\rho(i)} + h_{gid}\beta_{\rho(i)} + \gamma_{v_{\rho(i)}})r_i$ if it requests a secret key for the pair $(gid, \rho(i))$.

1) The term $-w_i \cdot \gamma_{v_{\rho(i)}} r_i$ can only be canceled by using

$$w_i \cdot (\gamma_{v_{\rho(i)}} + h_{\rho(i),t} \xi_{v_{\rho(i)},t}) r_i,$$

which incurs an addition term of $w_i h_{\rho(i),t} \xi_{v_{\rho(i)},t} r_i$ to be canceled. $\mathcal{A}$ must cancel this term by using

$$-w_i \cdot \xi_{v_{\rho(i)},t} h_{\rho(i),t_e} r_i,$$

where it is required that $t = t_e$. Thus, We have that $\mathcal{A}$ only can cancel $-w_i \cdot \gamma_{v_{\rho(i)}} r_i$ if it requests an update key for $(t_e, \rho(i), \mathsf{RL}_{\rho(i),t_e})$ under the condition that *gid* possesses $\rho(i)$ at t_e.

2) The term $-w_i h_{gid} \beta_{\rho(i)} r_i$ can only be canceled by using

$$w_i \cdot h_{gid}(\mu_i + \beta_{\rho(i)} r_i),$$

which leaves behind the term $w_i h_{gid} \mu_i$.

The collection of these terms of for each *gid* will only cancel if vector $(1, 0, \ldots, 0) \in \mathbb{Z}_p^m$ is in the span of rows A_i of A belonging to corrupt authorities or for which $\mathcal{A}$ obtained secure keys for $(gid, \rho(i))$ and update keys for $(t_e, \rho(i), \mathsf{RL}_{\rho(i),t})$ under the condition that *gid* possesses $\rho(i)$ at t_e. However, in the security game $\mathcal{A}$ is restricted to make such queries.

Hence, we have shown that $\mathcal{A}$ can not construct a query of the form cs for a constant c. Therefore, under conditions that hold with all but negligible probability, $\mathcal{A}$'s view when $\bar{s}$ is random is the same as $\mathcal{A}$'s view when $\bar{s} = s$. This proves that $\mathcal{A}$ cannot attain non-negligible advantage in the security game. ■