This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

An Attribute-based Controlled Collaborative Access Control Scheme for Public Cloud Storage

Yingjie Xue, Kaiping Xue, *Senior Member, IEEE,* Na Gai, Jianan Hong,
David S.L. Wei, *Senior Member, IEEE,* and Peilin Hong

***Abstract*—In public cloud storage services, data are outsourced to semi-trusted cloud servers which are outside of data owners' trusted domain. To prevent untrustworthy service providers from accessing data owners' sensitive data, outsourced data are often encrypted. In this scenario, how to conduct access control over these data becomes a challenging issue. Attribute Based Encryption (ABE) has been proven to be a powerful cryptographic tool to express access policies over attributes, which can provide a fine-grained, flexible, and secure access control over outsourced data. However, existing ABE-based access control schemes do not support users to gain the access permission by collaboration. In this paper, we explore a special attribute-based access control scenario where multiple users having different attribute sets can collaborate to gain access permission if the data owner allows their collaboration in the access policy. Meanwhile, the collaboration that is not designated in the access policy should be regarded as a collusion and the access request will be denied. We propose an attribute-based controlled collaborative access control scheme through designating translation nodes in the access structure. Security analysis shows that our proposed scheme can guarantee data confidentiality and has many other critical security properties. Extensive performance analysis shows that our proposed scheme is efficient in terms of storage and computation overhead.**

***Index Terms*—Public Cloud Storage, Access Control, CP-ABE, Collaboration**

I. Introduction

Cloud computing has emerged as the natural evolution and integration of advances in several fields, including utility computing, distributed computing, grid computing, and service oriented architecture [1]. It promotes the concept of leasing remote resources rather than buying hardwares, which frees cloud customers (such as enterprises and individuals) from maintenance expenses. Cloud customers are able to utilize cloud services on a pay-as-you-use basis, where the price is relatively low. What's more, since services are provided via the Internet, customers can access applications and data anywhere and anytime. To benefit from the above advantages, but not limited to, an increasing number of enterprises and individuals are willing to outsource their data and applications to cloud platforms.

Despite many advantages of cloud computing, there still remain various challenging issues that impede cloud computing from being widely adopted, among which, privacy and security of users' data have been the major issues. Traditionally, a data owner stores his/her data in trusted servers which are generally controlled by a fully trusted administrator. However, in public cloud storage, which is a popular service model in cloud computing, data are usually stored and managed on remote cloud servers which are administrated by a semi-trusted third party, i.e. the cloud service provider. Data are no longer in data owners' trusted domains and they cannot trust cloud servers to conduct secure data access control. Therefore, the secure access control has become a challenging issue in public cloud storage, in which traditional security technologies cannot be directly applied.

In recent years, many researches have been devoted on data access control in public cloud storage, such as the work in [2–7]. Among those literatures, Ciphertext-policy Attribute-based Encryption (CP-ABE) is regarded as one of the most suitable schemes due to the fact that it can guarantee data owners' direct control over their data and provide a fine-grained access control service. In CP-ABE schemes, each user is associated with a set of attributes and every ciphertext is embedded with an access structure over some chosen attributes. The access structure is used to express the specific access policy that should be satisfied to access data contents. Only if a user's attribute set satisfies the access structure embedded in the ciphertext can he/she decrypt the ciphertext. Therefore, by using access structures over attributes to express access policies, CP-ABE is a promising tool to provide fine-grained, flexible, and secure data access control in public cloud storage.

Nevertheless, the existing CP-ABE schemes can merely assign access permission to individuals who own attribute sets satisfying the access policy. However, in many scenarios, the secret information cannot be obtained individually by a single user alone. For example, in enterprises and organizations, some important files/documents are shared among multiple users who have distinct responsibilities according to their positions, but have the same goal to protect data confidentiality. A data access request may be permitted only when multiple users with different responsibilities collaborate. Such requirement of collaboration to access secret data has been widely studied in secret sharing schemes [8, 9], where data can only be accessed by a number of participants together and the number is no less than a given threshold. Unfortunately, the existing secret sharing schemes cannot express access policies in a fine-grained and flexible way. For a fine-grained access control, we can label each user by an attribute set where his/her responsibility/role can be expressed as a single attribute or a set of attributes. To design a general access control system, we

Y. Xue, K. Xue, N. Gai, J. Hong and P. Hong are with Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (K. Xue, kpxue@ustc.edu.cn).

D. Wei is with the Computer and Information Science Department, Fordham University, NY 10458, USA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

let data be accessed on one of the following two conditions: 1) (Non-collaboration scenario) As the existing CP-ABE schemes do, an individual who has a sufficiently powerful attribute set satisfying the access structure can access data individually; 2) (Collaboration scenario) For a user whose attribute set doesn't have sufficient authority to gain access individually, he/she can collaborate with other users who have some different attributes such that their integrated attribute set has sufficient authority to access data. Here, we take an example to illustrate the scenario. In a company, the data owner requires a financial document to be accessed on the following circumstances, as shown in Fig. 1: The left side with two boxes shows that two policies can be utilized to access data. The word "OR" between the two boxes means that the data users can access the data if they can satisfy either sub-policy A or sub-policy B. Sub-policy A denotes the case that the policy tree must be satisfied by an independent user, and sub-policy B denotes the case that the ciphertext can be accessed by collaboration on the condition that one user has an attribute set that can satisfy the tree on the left side (in the box denoting sub-policy B) and the other user has the attribute '*Auditor*'. The word "AND" means the mentioned two users can collaborate to satisfy sub-policy B. We observe that it is usually impossible for a user to have both attribute sets {'*Manager*'/'*Accountant*'} and {'*Auditor*'}. Thus, sub-policy A and sub-policy B can be expressed in a more efficient way by a compound policy tree as shown on the right side in Fig.1. The node denoted '*Auditor*' has two circular edges to indicate that it is a special node that allows collaboration to be performed on it.

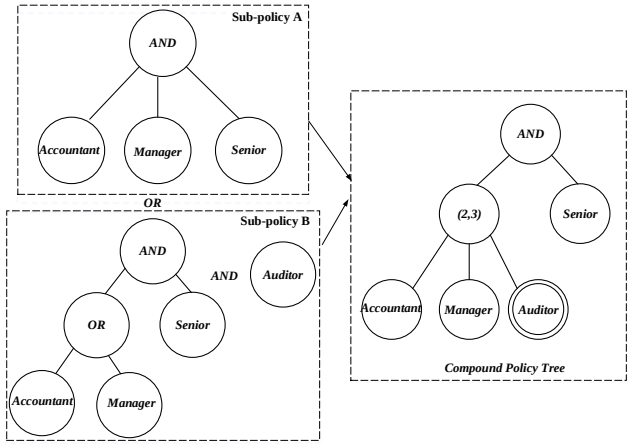


Fig. 1: An Example of An Access Policy

It is not straightforward to design a suitable mechanism to allow collaboration embedded in the access policy. To tackle the above problem, Li et al. [10] firstly addressed the collaboration problem among users with different attribute sets, and they proposed a Group-Oriented Attribute-Based Encryption (GO-ABE) scheme. Constructed on CP-ABE, GO-ABE further divides users into groups and allows users from the same group to collaborate to access data. Allowing users within the same group to collaborate is reasonable, since users responsible for one project may have more motivations to protect their data. However, their scheme allows all attributes to be collaborated within the same group, even if the data owner disallows such data access. That is to say, in the example of Fig. 1, a user with the attribute set {'*Junior*', '*Accountant*', '*Manager*'} can also collaborate with a user with the attribute set {'*Senior*', '*Programmer*'} to gain the permission to access data, by collaborating on the attribute 'Senior'. However, this should be considered as a malicious behavior from the viewpoint of data owners. Thus, it is a challenging issue to design such mechanism that allows expected collaboration among honest users and also simultaneously resist unwanted collusion among curious users. As the access policy specified in Fig 1, only a user with the attribute set {'*Senior*', '*Accountant*'} or {'*Senior*', '*Manager*'} collaborating with a user with the attribute *'Auditor'* is regarded as valid collaboration to access data. Any other kind of collaboration should be considered as malicious and the access is not permitted.

In this paper, we address the collaboration issue in practical scenarios and propose an attribute-based controlled collaborative access control scheme for public cloud storage. Specifically, like GO-ABE [10], we restrict user collaboration in the same group that corresponds to the same project for which the involved people are responsible. Thus, in our work, in order to provide both data confidentiality and collaborative access control, only people who are in charge of the same project are allowed to collaborate. Technically, data owners allow expected collaboration by designating translation nodes in the access structure. In this way, unwanted collusion can be resisted if the attribute sets by which users are collaborating are not corresponded to translation nodes. For each translation node, an additional translation value is generated. Using this translation value and special translation keys embedded in users' secret keys, users within the same group can collaborate to satisfy the access structure and gain the data access permission. For colluding users across groups, their access is not permitted as their secret keys do not correspond to the same group. The main contributions of this work can be summarized as follows.

1) We address the problem of data access control in collaboration scenarios and propose an attribute-based controlled collaborative access control scheme. Data owners can specify expected collaboration among users when they define access policies. Meanwhile, unwanted collusion can be denied to access the data.
2) We design a mechanism to achieve our goal by designating translation nodes in policy trees and modifying secret keys and ciphertexts. More specifically, our approach embeds a translation key inside the secret key of BSW scheme [11] and adds a translation value in the ciphertext for each translation node. The combination of translation keys and translation values enables users to collaborate to satisfy a policy tree.
3) Users are divided into groups in a way such that the collaboration is restricted and secure. That is to say, only users responsible for the same project are allowed to collaborate in case that malicious users who are not responsible for the project collude. Extensive security analysis is given to show the security properties of our proposed scheme.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

The rest of this paper is organized as follows. In Section II, we introduce some related works about access control schemes in public cloud storage, along with the researches on the topic of collaborative access control. In Section III, technical preliminaries are presented, and the definition of the system model and security assumptions are presented in Section IV. We introduce our proposed attribute-based controlled collaborative access control scheme for public cloud storage in Section V. In Sections VI and VII, we analyze our proposed scheme in terms of security and performance, respectively. Finally, the conclusion is given in Section VIII.

II. Related work

The motivation of our work can be dated back to the access control of data stored in untrusted servers and collaborative access control. To address access control for data stored in untrusted servers, many works using cryptographic technologies have been proposed, such as the literatures [12, 13]. However, they are either coarse-grained or short of scalability as the number of users increases. Ciphertext-policy Attribute-based Encryption (CP-ABE) is regarded as a promising technique to provide fine-grained, flexible, and secure access control of outsourced data in public cloud storage. The first CP-ABE scheme was designed by Bethencourt in [11], and subsequently some literatures were proposed to improve its security [14–16] and efficiency [17–19]. To improve its expressiveness, the work of [20] has been proposed, and then the work of [21] further improves the scalability of [20]. Considering that users may hold attributes from multiple authorities, some multi-authority schemes, such as the works in [2, 22], have been proposed. Recently, in [23], the authors pointed out that ABE schemes cannot express access control rules like role hierarchy and object hierarchy. Consequently, they proposed a secure role-based access control scheme to address the problem.

Another research area related to our work is collaborative access control. However, in most of the existing works [24, 25], the word "collaborative" denotes that multiple authorized users are able to work on the same document and edit the document collaboratively. Their main goal is to assign different privileges (such as read, write, and grant) to different users. A user can obtain privilege independently. The works that have the concept of collaboration that are most similar to ours are found in the research area of online social networks. In [26, 27], a user is allowed to access data only when he/she get permission from multiple users, suggesting that multiple users need to collaborate to satisfy the policy. Unfortunately, their schemes cannot support fine-grained access control since they mainly label users by trust levels.

Secret sharing scheme is a feasible solution to support collaboration among users and it has been adopted in collaborative access control [28]. It specifies that a secret can only be obtained if the number of valid participants is no less than a threshold. In 1979, Shamir [8] and Blackly [29] proposed *(t,n)* threshold secret sharing schemes which are based on Lagrange interpolation and multi-dimensional space mapping, respectively. Such schemes are referred to as threshold secret sharing schemes, in which each of the participants has equal right. Weighted threshold secret sharing schemes [30] are natural generalizations of threshold secret sharing schemes, where each participant is assigned a weight depending on his/her importance in the group of all participants. For example, in a bank, the tellers and directors have different weights as to the rights to reconstruct the key of bank vault. The secret can be reconstructed if and only if the sum of the weights assigned to a set of participants is greater than or equal to a fixed threshold. A variant of weighted secret sharing is multi-level secret sharing schemes, such as [9], where participants are partitioned into levels. Generally speaking, those schemes distinguish a user by only one factor (e.g. importance, role, or level). Our scheme is more expressive, as we label a user by a set of attributes.

The works presented in the literatures [10] and [20] are the most related to our work. However, our work is different from theirs in several aspects. Compared with the work in [10], we give data owners the privilege to specify inside the policy tree whether a collaboration is allowed and on which nodes a collaboration is allowed. Although we use the concept of translation nodes proposed in [20], we consider a different problem from that in [20] and our approach is also different. We consider the collaboration between different users to decrypt a ciphertext, whereas the work in [20] focuses on restricting the combination of an independent user's attributes inside his/her attribute set to satisfy a policy tree. Technically, the scheme in [20] organizes a user's attributes into a recursive set based structure. According to the structure, only the attributes inside the same set can be combined to satisfy a policy tree. Attributes from different sets can be combined to satisfy a policy tree only if there exists corresponding translation nodes. Different from [20], in our scheme, users' attributes do not have any recursive structures. In addition, in our scheme, collaboration is allowed only among the users from the same group.

III. Preliminaries and Definitions

In this section, we first give a brief review of background information on bilinear maps. Then we describe CP-ABE on which our scheme is based.

A. Bilinear Maps

Here, we briefly review the facts about groups with efficiently computable bilinear maps. We refer readers to the literature [31] for more details.

Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative cyclic groups of the same prime order p and g be a generator of $\mathbb{G}$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ defined on $\mathbb{G}$ has the following three properties:

1) *Bilinearity*: $\forall a, b \in \mathbb{Z}_p$ and $g_1, g_2 \in \mathbb{G}$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2) *Non-degeneracy*: $\forall g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2) \neq 1$, which means that the map does not send any pair in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$.
3) *Computability*: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

B. Ciphertext-policy Attribute-based Encryption (CP-ABE)

Although the definitions and constructions of different CP-ABE schemes are not always consistent, the uses of the access structure in **Encrypt** and **Decrypt** algorithms are nearly the same. Here we adopt the definition and construction of the first CP-ABE scheme [11] to illustrate the construction of CP-ABE, as most of the state-of-the-art literatures adopting CP-ABE for data access control are based on this construction.

A CP-ABE scheme consists of four algorithms: **Setup**, **Encrypt**, **KeyGen** (Key Generation), and **Decrypt**.

Setup$(\lambda, U) \rightarrow (PK, MSK)$. The setup algorithm takes the security parameter λ and the attribute universe description U as the input. It outputs the public parameters PK and a master secret key MSK.

Encrypt$(PK, M, \mathbb{A}) \rightarrow CT$. The encryption algorithm takes the public parameters PK, a message M, and an access structure $\mathbb{A}$ over the universe of attributes as the input. The algorithm will encrypt M and produce a ciphertext CT such that only a user whose attribute set satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains $\mathbb{A}$.

KeyGen$(MSK, S) \rightarrow SK$. The key generation algorithm takes the master secret key MSK and a set of attributes S as the input. It outputs a secret key SK.

Decrypt$(PK, CT, SK) \rightarrow M$. The decryption algorithm takes the public parameters PK, a ciphertext CT which contains an access structure $\mathbb{A}$, and a secret key SK as the input, where SK is a secret key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathbb{A}$, the algorithm will decrypt the ciphertext and return a message M.

Please refer to [11] for more details about CP-ABE. Furthermore, the literatures, such as [2, 21], have introduced CP-ABE to construct fine-grained access control frameworks in public cloud storage.

IV. System Model and Security Assumptions

In this section, we give the definitions of the system model, the security assumptions and requirements for our proposed data access control scheme.

A. System Model

The system model of our design is defined in Fig. 2, which consists of four entities: a ***central authority (CA)***, many ***data owners (Owners)***, many ***data consumers (Users)***, and a ***cloud service provider with multiple cloud servers*** (called ***cloud servers*** from here on).

- ***The central authority (CA)*** is the administrator of the whole system. Particularly, it sets up the system parameters for the access control implementation and distributes secret keys for users.
- ***The data owner (Owner)*** is the entity who outsources his/her data to cloud servers. To share his/her data with other intended entities, he/she defines access policies for data. The access policy is represented by an access structure over attributes. Data contents are encrypted under access structures before being uploaded to cloud servers.

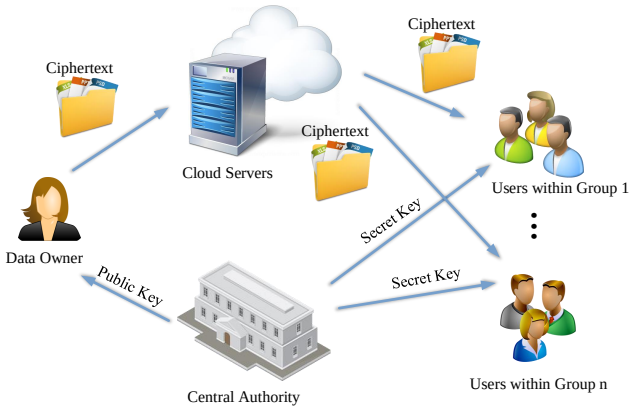


Fig. 2: System Model

- ***The data consumer (User)*** is the entity who is interested in data contents. In our controlled collaborative access control scheme, each user is assigned to a group related to the project for which he/she is responsible. He/She possesses a set of attributes and is equipped with a secret key associated with his/her attribute set. The user can freely get any encrypted data that he/she is interested in from cloud servers.Then, he/she can decrypt the encrypted data on either conditions: (1) His/Her attribute set independently satisfies the access structure embedded inside the encrypted data; (2) If the policy allows/specifies some kinds of collaboration, he/she can collaborate with other valid users to decrypt the data.
- ***Cloud servers*** provide a public platform for owners to store and share their encrypted data. They do not conduct data access control for owners. The encrypted data stored in cloud servers can be downloaded freely by any data consumer.

B. Security Assumptions and Requirements

In our proposed scheme, the security assumptions of the four roles can be defined as follows. Cloud servers are always online and are managed by the cloud provider who is usually assumed to be "honest-but-curious". It means that cloud servers will correctly execute the tasks assigned to them for profits, but they would try to obtain as much secret information as possible based on data owners' inputs and outsourced data. *CA* is assumed to be fully trusted, which will not collude with any entity to peep data contents. Owners have access control over their own outsourced data which are protected by specific policies. We assume that owners will not do harm to their data confidentiality. Users are assumed to be dishonest and curious, who may collude with each other to gain unauthorized access. What's more, although data owners may allow some kind of collaboration among users by combining their attribute sets to satisfy the access structure, it's not that any combination of attribute sets can be authorized. Only the collaboration specified in the access policy/structure is valid for accessing data. Any other kind of collaboration is treated as illegal and cannot succeed to get data contents.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

Towards controlled collaborative access control in public cloud storage, we have at least six basic security requirements as follows:

- ***Data confidentiality***. Data contents must be kept confidential to unauthorized individuals and collaborating users, including the curious cloud servers.
- ***User collusion resistance***. For users within the same group, collaboration is allowed only upon translation nodes. Collaboration on non-translation nodes is resisted.
- ***Controlled collaboration within the same group***. Users from different groups cannot decrypt the ciphertext by collaboration.
- ***Secret key privacy***. All users' secret keys that are related to attributes are kept secret from other users in the collaboration.
- ***Secure revocation of the collaboration***. The collaboration privilege can be revoked by data owners or users.
- ***Non-reusability of intermediate results***. Each collaboration is only useful to decrypt one ciphertext.

V. An Attribute-based Controlled Collaborative Access Control Scheme

This section first gives an overview of our proposed scheme, then describes the scheme in detail, which mainly consists of four phases: ***System Initialization***, ***Key Generation***, ***Encryption***, and ***Decryption***.

A. Overview

The main goal of this work is to allow expected user collaboration for data access. For example in Fig. 1, to access a financial record, a user with the attribute set {'*Senior*', '*Accountant*', '*Manager*' } is able to access the data independently. However, with the data owner's specification of collaboration, if a user only has the attribute set {'*Senior*', '*Accountant*'}, he/she can also access with the help of a user possessing the attribute '*Auditor*'. On this circumstance, the attribute '*Auditor*' can be treated as a "high-level" attribute that has the power of supervision to protect confidential data.

Our work is a challenging one in that while we allow authorized user collaboration, we also have to prevent unauthorized user collusion. The authorized collaboration should conform to both of the following two rules: 1) The collaborating users must be from the same group; 2) Their combined attribute set should satisfy the access structure and the collaboration should be specified within the access policy. Users in the same group are responsible for the same project. The first rule means that only users responsible for the same project can gain data access permission by collaboration. A user cannot be permitted to access data when he/she colludes with the users from a different group. The second rule indicates that the access policy will designate the condition on which users' attribute sets can collaborate. Any other ways of collaboration should be considered as collusion and be prohibited.

According to the left part in Fig. 1, we illustrate a trivial solution in a simple case where all users are in the same group. Data are encrypted under sub-policy A and sub-policy B separately. The generated ciphertext will contain two pieces of ciphertext according to each sub-policy. For an independent user whose attribute set satisfies sub-policy A, he/she can decrypt individually without collaboration with other users. The encryption and decryption are just the same as the phases of CP-ABE schemes such as that of [11]. Collaborating users can decrypt by satisfying sub-policy B. In sub-policy B, for simplicity, we assume that the left tree in sub-policy B can be satisfied by user U_1 and the right tree (a tree containing only one node 'Auditor' is also considered as a tree) can be satisfied by user U_2. In the encryption phase, for sub-policy B, s is randomly chosen, and s_1 and s_2 are assigned to the root node of the right tree and the left tree, respectively, such that $s = s_1 + s_2$. The parameters s_1 and s_2 are encrypted in the way as in CP-ABE schemes, e.g., [11]. The ciphertext can be decrypted if and only if the left tree and right tree are satisfied independently by U_1 and U_2, respectively. The total ciphertext size nearly doubles since each ciphertext associated with its policy will contain components corresponding to nearly all attributes. If we define more complex collaboration, there will be more ciphertexts and the total ciphertext size will grow rapidly.

Therefore, to allow collaboration as well as to avoid the rapid growth of the ciphertext size, we refer to the basic idea of [20] and define translation nodes in the policy tree (such as Fig. 1) to allow authorized collaboration and forbid unauthorized collusion. A translation node can be either a leaf node or a non-leaf node, and users can collaborate on those nodes. For each translation node, a translation value is generated for it. Furthermore, to restrict collaboration to be within the same group, a random group secret key is embedded in each user's secret key to assign a user to a certain group. In addition, a translation key is also added to each user's secret key. For the users in the same group, with the translation value and the translation key, one user can translate the secret share of his/her attribute sets in a policy tree to another one's. In this way, collaborating users can be viewed as an individual who has all required attributes satisfying the policy tree, and thus he/she can further decrypt the ciphertext. Since a translation value is only appended on translation nodes, the collaboration can only happen on those nodes.

B. Access Structure

We build our scheme on the access structure used in [11], which is a policy tree of several nodes. Let $\mathcal{T}$ be a policy tree with the root node r and Let $\mathcal{T}_x$ denote a subtree of $\mathcal{T}$ rooted at the node x. Each leaf node y of the tree is associated with an attribute which is denoted as $att(y)$. Each non-leaf node of the tree is a threshold gate, which can be defined by its children and a threshold value. Take the node x as an example, we describe the features of $\mathcal{T}$. Let nc_x and k_x respectively denote the number of children and the threshold value of the node x, where $0 < k_x \leq nc_x$. The node x is denoted as (k_x, nc_x). When $k_x = 1$, the threshold gate is an "OR" gate and when $k_x = nc_x$, it is an "AND" gate. The parent of the node x in the tree is denoted as $parent(x)$. The policy tree $\mathcal{T}$ also defines an ordering between the children of every non-leaf node, that is, the children of a non-leaf node x are numbered from 1 to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

nc_x in an arbitrary manner. For each of x's child nodes (y), this number is denoted as $index(y)$.

Let $\mathcal{T}_x()$ denote the tree satisfaction algorithm, where the input is the individual/combined attribute set and x denotes the root node of the sub-tree $\mathcal{T}_x$ of $\mathcal{T}$. First, we define the condition under which a user's attribute set is said to satisfy a given policy tree. The definition is modified a little from the basic one of [11]. We assign each user an identifier (e.g. U_i with u_i) for ease of description. For a user U_i , we denote his/her attribute set S_i satisfies the policy tree $\mathcal{T}_x$ if and only if $\mathcal{T}_x(S_i) = u_i$. We compute $\mathcal{T}_x(S_i)$ as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(S_i)$ for all children x' of the node x. $\mathcal{T}_x(S_i)$ returns u_i if and only if at least k_x children return u_i. If x is a leaf node, then $\mathcal{T}_x(S_i)$ returns u_i if and only if $att(x) \in S_i$.

Here, we further extend the definition to consider the introduction of translation nodes in order to provide controlled collaboration. If there are designated translation nodes in the policy tree, $\mathcal{T}_x()$ can be further modified as follows. Firstly, assume that the users come from the same group. We define γ as a collection of multiple users' attribute sets:

$$\begin{aligned}\gamma &= \{S_1, \ldots, S_k\} \\ &= \{\{att_{1,1}, \ldots, att_{1,n_1}\}_{u_1}, \ldots, \{att_{k,1}, \ldots, att_{k,n_k}\}_{u_k}\},\end{aligned} \tag{1}$$

where $u_1, \ \ldots, \ u_k$ denotes the unique identifiers of k different users, S_i denotes the attribute set of the user with the identifier u_i, and n_i is the total number of attributes in S_i. If x is a leaf node, $\mathcal{T}_x(\gamma)$ returns a set U_x of users' identifers. $u_i \in U_x$ if $att(x) \in S_i$. When x is a non-leaf node, we evaluate $\mathcal{T}_{x'}(\gamma)$ for all children x' of x. $\mathcal{T}_x(\gamma)$ returns a set U_x of users' identifers, where for each $u_i \in U_x$ there exists no less than k_x children that satisfy the following requirement: For each one of these children x', $U_{x'}$ either contains the identifier u_i or x' is a translation node and $U_{x'} \neq \oslash$.

Thus, if the node x is a designated translation node (Note that translation nodes can be set upon leaf nodes and non-leaf nodes.), even though the attributes used to satisfy the predicate represented by x comes from S_i, and the attributes used to satisfy the predicate represented by x's siblings comes from S_j where $i \neq j$, they can collaborate to satisfy the predicate represented by their common parent node. Accordingly, a collection of attribute sets γ is said to satisfy the policy tree $\mathcal{T}$ if and only if $\mathcal{T}_r(\gamma)$ returns a non-empty set U_r.

C. Details of Our Proposed Scheme

1) System Initialization: Firstly, *CA* chooses two multiplicative cyclic groups $\mathbb{G}$ (the parameter g is a generator of $\mathbb{G}$) and $\mathbb{G}_T$ with the same prime order p, and defines a binary map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ on $\mathbb{G}$. Let $H : \{0,1\}^* \to \mathbb{G}$ be a hash function that maps any arbitrary string to a random group element. This function is used to map attributes described as arbitrary strings to group elements. *CA* randomly chooses $\alpha, \beta_1, \beta_2 \in \mathbb{Z}_p$ as the group secret key. Besides, in our work, we assume that there are n projects and each of which associates with a different group. For every group m, a unique master secret key $\theta_m \in \mathbb{Z}_p$ is chosen to implicitly indicate to which group a specific user belongs to. The published public key PK is:

$$\begin{aligned}\mathbb{G}_T, \mathbb{G}, H, g, h_1 = g^{\beta_1}, f_1 = g^{1/\beta_1}, \\ h_2 = g^{\beta_2}, f_2 = g^{1/\beta_2}, e(g,g)^{\alpha},\end{aligned}$$

and the master secret key MSK is:

$$g^{\alpha}, \beta_1, \beta_2, g^{\theta_1}, \ldots, g^{\theta_n},$$

which implicitly exists in the system, but doesn't need to be obtained by any other entity. In addition, *CA* issues a unique identifier u_i to each user. For ease of description, we denote the user with the identifier u_i as U_i.

2) Key Generation: Let $S_i = \{a_{i,1}, \ldots, a_{i,n_i}\}$ be the attribute set of the user U_i, where $a_{i,j}$ denotes the jth attribute appearing in the set S_i, and n_i denotes the number of attributes in S_i. *CA* generates a secret key for the user U_i as follows. Firstly, *CA* checks which group he/she belongs to and finds out the corresponding group secret key θ_m. Then *CA* randomly chooses $r_i \in \mathbb{Z}_p$ for the user U_i, and $r_{i,j} \in \mathbb{Z}_p, 1 \leq j \leq n_i$ for each attribute $a_{i,j}$. Then the secret key is issued to the user U_i as:

$$\begin{aligned}SK_i = \{&D_i = g^{\frac{\alpha+\theta_m}{\beta_1}}, \\ &D_{i,j} = g^{r_i} H(a_{i,j})^{r_{i,j}}, D'_{i,j} = g^{r_{i,j}}, 1 \leq j \leq n_i\} \\ &E_i = g^{\frac{\theta_m + r_i}{\beta_2}},\end{aligned}$$

where E_i is a translation key used for collaboration.

3) Encryption: Let M denote the plaintext of the data file. To improve the system's performance, the owner first chooses a random number $\kappa \in \mathbb{Z}_p$ as the symmetric key and uses it to encrypt the plaintext with a symmetric encryption algorithm, such as *AES*. The encrypted data can be denoted as $E_{\kappa}(M)$, then the owner encrypts the symmetric key κ under the access policy using the following algorithm. Here, a policy tree $\mathcal{T}$ is defined by the data owner to express the access policy. The data owner sets some nodes on the policy tree as translation nodes, based on which he/she can allow multiple users to collaborate to satisfy the policy tree.

The algorithm first chooses a polynomial q_x for each node x in the policy tree $\mathcal{T}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node r. For each non-leaf node x in the tree, the degree d_x is set to be one less than the threshold value k_x, i.e. $d_x = k_x - 1$. Starting with the root node of the policy tree $\mathcal{T}$, the algorithm randomly chooses $s \in \mathbb{Z}_p$ and sets $q_r(0) = s$. Then, it randomly chooses d_r other points of the polynomial q_r to define it completely. For any other node x (including both leaf nodes and non-leaf nodes), it sets $q_x(0) = q_{parent(x)}(index(x))$ and randomly chooses d_x other points to completely define q_x. The degree of leaf nodes is set to be 0.

Let $\mathbb{Y}$ denote the set of leaf nodes in $\mathcal{T}$, and $\mathbb{X}$ denote the set of the defined translation nodes in $\mathcal{T}$. Then ciphertext CT is constructed with $\mathcal{T}$ and computed as:

$$\begin{aligned}CT = \{&\mathcal{T}, \tilde{C} = \kappa \cdot e(g,g)^{\alpha s}, C = h_1^s, \bar{C} = h_2^s, \\ &\forall y \in \mathbb{Y} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}, \\ &\forall x \in \mathbb{X} : \hat{C}_x = h_2^{q_x(0)}\},\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

where $\hat{C}_x$ is denoted as the translation value.

4) Decryption: Generally speaking, in real scenarios, a user (denoted as the data requestor) has the intention to access data and he/she asks other users (denoted as collaborators) to collaborate if he/she is not allowed to decrypt the data independently. We assume there are $k \geq 1$ users and they are within the same group. In this setting, the input of the algorithm is actually a combined attribute set γ (as defined in Eq. 1) which contains the attribute set of those users. Specifically, only the whole attribute set of the data requester and partial attribute subsets of collaborators are included in γ. That is to say, collaborators only input the necessary attribute subset of theirs which are related to the collaboration (translation nodes) into set γ (e.g. as shown in Fig. 1, a senior auditor will only contribute the attribute 'Auditor' to collaborate, while his/her attribute 'Senior' will not appear in the decryption.).

Let $S_i \in \gamma$ denote the attribute subset of the user U_i. The decryption algorithm first runs the tree satisfaction algorithm $\mathcal{T}(\gamma)$ on the policy tree with the collection (γ) of attribute subsets (as defined in Eq. 1), and stores the results of each recursive call in policy tree $\mathcal{T}$. That is, each node x in the tree is associated with a set (U_x) of identifiers returned by $\mathcal{T}_x(\gamma)$. If $u_i \in U_x$, we denote that u_i is associated with the node x. An identifier $u_i \in U_x$ means that the user U_i can be selected as a representative to successfully decrypt the node x. If the combined attribute set γ can satisfy the policy tree $\mathcal{T}$, the identifier of the data requester (e.g. u_i) will be included in U_r by $T_r(\gamma)$. Then, the data requester will call a recursive function $DecryptNode(CT, \gamma, x, u_i)$ on root node r of the tree. Here, x is a node in $\mathcal{T}$ and u_i denotes the identifier of a user who acts as a representative to decrypt the node x. If the combined attribute set γ does not satisfy policy tree $\mathcal{T}$, $U_r = \oslash$ and the decryption algorithm will return $\perp$.

When $x \in \mathbb{Y}$ is a leaf node, the recursive function $DecryptNode(CT, \gamma, x, u_i)$ is defined as follows. If $u_i \notin U_x$, i.e. $att(x) \notin S_i$, $DecryptNode(CT, \gamma, x, u_i) = \perp$. Otherwise, the algorithm gets $att(x) = a_{i,j} \in S_i$. Then, $DecryptNode(CT, \gamma, x, u_i)$ is computed as:

$$DecryptNode(CT, \gamma, x, u_i) = \frac{e(D_{i,j}, C_x)}{e(D'_{i,j}, C'_x)}$$
$$= \frac{e(g^{r_i} H(a_{i,j})^{r_{i,j}}, g^{q_x(0)})}{e(g^{r_{i,j}}, H(a_{i,j})^{q_x(0)})} = e(g,g)^{r_i q_x(0)}.$$

Note that the set from which the satisfying attribute $a_{i,j}$ is picked is implicitly in the result $e(g,g)^{r_i q_x(0)}$ (indicated by r_i).

Then, when $x \notin \mathbb{Y}$, i.e., the node x is a non-leaf node, then $DecryptNode(CT, \gamma, x, u_i)$ runs recursively as follows:

i) The algorithm first computes k_x sized child nodes (denoted as z) of x, and stores them in a set B_x. Those child nodes z are selected by the following restrictions. If it is not a translation node, node z must be associated with u_i. If it is a translation node, node z must be associated with some $u_{i'}$ where $i \neq i'$. The computation of B_x intuitively means the selection of satisfying child nodes to satisfy the predicate represented by the node x. If there are no such sized k_x child nodes z, the algorithm return $\perp$.

ii) For each node $z \in B_x$ which is not a translation node, i.e. $u_i \in U_z$, the algorithm further calls $DecryptNode(CT, \gamma, z, u_i)$ and stores output in F_z. This means that the representative (the user U_i) will decrypt the secret of those child nodes z.

iii) For each node $z \in B_x$ which is a translation node, i.e. $u_{i'} \in U_z$ and $i' \neq i$, the algorithm further calls $DecryptNode(CT, \gamma, z, u_{i'})$ and stores the output in F'_z. This means that another user $U_{i'}$ (a collaborator) is selected as a new representative to decrypt the node z. Then, the user U'_i translates the output (denoted as F'_z) to F_z as follows:

$$F_z = e(\hat{C}_z, E_i / E_{i'}) \cdot F'_z$$
$$= e(g^{\beta_2 q_z(0)}, g^{\frac{\theta_m + r_i - (\theta_m + r_{i'})}{\beta_2}}) \cdot e(g,g)^{r_{i'} q_z(0)}$$
$$= e(g,g)^{r_i \cdot q_z(0)}.$$

We let the user U_i broadcast its translation key E_i implicitly. Then F_z is transmitted to the user U_i so that the user U_i can gather the secret of k_x child nodes z of x to act as a representative to decrypt the node x. In this way, it is equivalent to the fact that the user U_i owns all attributes that satisfy the predicate represented by x and he/she is able to decrypt the node x. Now F_z is related to the random number r_i of the user U_i.

iv) Then, the user U_i computes F_x using exponential polynomial interpolation as follows:

$$F_x = \Pi_{z \in B_x} F_z^{\Delta_{k, B'_z}(0)} = e(g,g)^{r_i q_x(0)},$$

where $k = index(z), B'_z = \{index(z) : z \in B_x\}$ and Lagrange coefficient $\Delta_{i,S}(x) = \Pi_{j \in S, j \neq i} \frac{x-j}{i-j}$. In this way, by running the algorithm recursively, one user with identifier u_i who is associated with the root node r can decrypt the root node and get $F_r = DecryptNode(CT, \gamma, r, u_i) = e(g,g)^{r_i \cdot q_r(0)} = e(g,g)^{r_i \cdot s}$. The decryption proceeds and outputs F as

$$F = \frac{e(\bar{C}, E_i)}{F_r} = \frac{e(g^{\beta_2 \cdot q_r(0)}, g^{\frac{\theta_m + r_i}{\beta_2}})}{e(g,g)^{r_i \cdot q_r(0)}} = e(g,g)^{\theta_m \cdot s}.$$

Finally, the decryption algorithm performs following computation to obtain κ:

$$\kappa = \frac{\tilde{C} \cdot F}{e(C, D_i)} = \frac{\kappa \cdot e(g,g)^{\alpha \cdot s} \cdot e(g,g)^{\theta_m \cdot s}}{e(g^{s \cdot \beta_1}, g^{\frac{\theta_m + \alpha}{\beta_1}})}. \quad (2)$$

Then κ is used to decrypt $E_\kappa(M)$ by symmetric encryption algorithms and the user/users finally get M.

As shown in the above process, with translation keys E_i and $E_{i'}$ in users' secret keys and the translation value $\hat{C}_x$ at the translation node x, one can translate the secret with $r_{i'}$ to r_i such that the collaboration is allowed. Note that only translation keys are exposed to help translate the output associated with the identifier $u_{i'}$ to the user U_i. The secret key components that are related to attributes are not disclosed. In the decryption of every non-leaf node x, one user with identifier u_i ($u_i \in U_x$) is selected to decrypt that node. The user exposes his/her translation key E_i to other participants who decrypt child nodes of x. Then, each participant $U_{i'}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

decrypts translation nodes according to his/her attribute subset and then helps translate the computed output to the user U_i. Each output $e(g,g)^{r_{i'}q_x(0)}$ is translated into $e(g,g)^{r_i q_x(0)}$ and is transferred to the user U_i. Finally, the user U_i ($u_i \in U_r$) gathers output from every collaborators and gets the secret of root node r.

It is also possible that there is not an explicit data requester in the collaboration scenario but multiple users simultaneously have the intention to get access to the data. In that case, the combined attribute set γ includes all their attributes so that some attributes may appear in multiple attribute subsets S_i. However, by running the tree satisfaction algorithm $\mathcal{T}_r(\gamma)$, the results of each recursive call in the policy tree $\mathcal{T}$ imply which attribute subset S_i can be used to satisfy the predicate represented by each node. Thus, even if U_x returned by the node x may contain multiple identifiers, as long as the identifier (e.g. u_i) is associated with the node x, it can be selected to decrypt the node x. The algorithm randomly selects a user with identifier u_i ($u_i \in U_r$) and starts the decryption algorithm from the root node r. The role of the chosen user with identifier u_i is the same as that of the data requester, as we have mentioned. The remaining recursive call procedure is also the same as the procedure we described above. The person who is chosen as a representative to decrypt the node is chosen in the top-down manner implicitly in the process of calling $DecryptNode(CT, \gamma, x, u_i)$ with the information returned by tree satisfaction algorithm. Thus, it is not necessary for users to worry about whom to be selected to decrypt.

D. A Summary of Our Proposed Approach

Fig. 3 illustrates the overall procedure of our scheme. A set of n users from the same group (e.g. group k) with different attribute sets firstly share their own attribute sets S_i to form a combined attribute set $\gamma = \{S_1, S_2, \cdots, S_n\}$, which is then inputted into the tree satisfaction algorithm $\mathcal{T}_r(\gamma)$. $\mathcal{T}_r(\gamma)$ labels each node of the policy tree with one or more specific users' identifiers, which denotes that the node can be satisfied by the specific user with the labelled identifier. We denote the labelled policy tree as an expanded policy tree. With this expanded policy tree, user U_i can try to decrypt the nodes labelled with u_i recursively, starting from the root node. If the function $DecryptNode(CT, x, \gamma, u_i)$ to decrypt the secret of the node x needs to call $DecryptNode(CT, z, \gamma, u_j)$ where $i \neq j$ and z is the child of x, then user U_j is responsible to run $DecryptNode(CT, z, \gamma, u_j)$. Furthermore, user U_j translates the output $e(g,g)^{r_j q_z(0)}$, which is computed by $DecryptNode(CT, z, \gamma, u_j)$, to $e(g,g)^{r_i q_z(0)}$, and then transmit the translated result to user U_i. When user U_i gathers all the secrets to construct the secret of the root node, he/she can construct $F_r = e(g,g)^{r_i s}$ by using Lagrange interpolation equation in the same way as that in a traditional CP-ABE scheme. Lastly, the user translates F_r to the one that is related to the group as $F = e(g,g)^{\theta_k s}$, and gets the encrypted symmetric key κ by Equation 2.

VI. Security Analysis

In this section, we analyze some security properties, namely data confidentiality, user collusion resistance, controlled collaboration within a group, secret key privacy, secure revocation, and non-resuability of intermediate results.

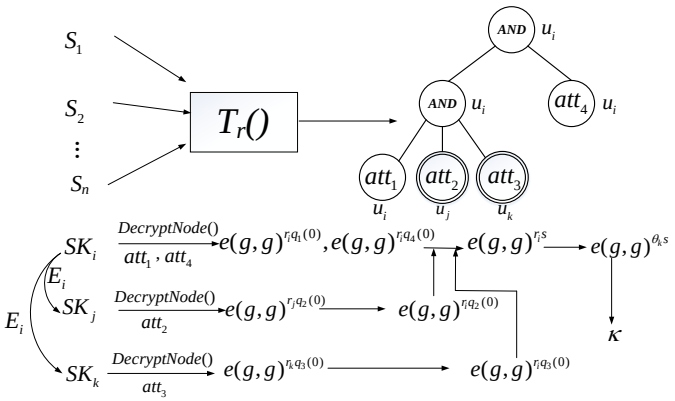


Fig. 3: A Summary of Our Proposed Approach

A. Data Confidentiality

By data confidentiality, we mean that an adversary cannot distinguish two messages in our security game. The adversary will choose to be challenged on an encryption with an access structure $\mathbb{A}^*$ and can ask for security keys associated with any attribute set, as long as the attribute set does not satisfy $\mathbb{A}^*$. The security proof of our approach is similar to that in [21]. We will prove that if there is any vulnerability in our proposed approach, the vulnerabilities can be used to break CP-ABE [11]. The security model of our scheme is similar to that of CP-ABE schemes. Thus, we first introduce the security model of CP-ABE, and then show how to use the vulnerabilities to break CP-ABE.

Security Model: The security model of CP-ABE schemes, such as BSW scheme in [11], is as follows.

- **Setup**. The challenger runs the Setup algorithm and gives public parameters PK to the adversary $\mathcal{A}$.
- **Phase 1**. The adversary $\mathcal{A}$ makes repeated queries for security keys corresponding to attribute sets $S_1, ..., S_{q_1}$.
- **Challenge**. The adversary $\mathcal{A}$ submits two equal length messages M_0 and M_1, and a challenge access structure $\mathbb{A}^*$ such that none of the attribute sets $S_1, ..., S_{q_1}$ satisfy the access structure. The challenger flips a random coin b, and encrypts M_b under $\mathbb{A}^*$. The resulting ciphertext CT is given to the adversary.
- **Phase 2**. Phase 1 is repeated with the restriction that none of the attribute sets $S_{q_1+1}, ..., S_q$ satisfy the access structure $\mathbb{A}^*$ corresponding to the challenge.
- **Guess**. The adversary $\mathcal{A}$ outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $|Pr[b' = b] - 1/2|$.

Definition 1. *A CP-ABE scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.*

To prove data confidentiality, we first define two threat models. One is that the adversary is a set of users who are from the same group, referred to as **Model A**. The other is that the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

adversary is a set of users who are from different groups (none of the users belongs to the same group), referred to as **Model B**. Then, we prove data confidentiality in these models. For the adversary in Model A, we give a detailed proof. For the adversary in Model B, we describe the proof by concentrating on the difference compared to the former proof. Then, we show how data confidentiality is guaranteed in real scenarios where two threat models are combined.

*1) **Data confidentiality in Model A**:* In Model A, the adversary is a group of users who come from the same group and their combined attribute set does not satisfy the access structure $\mathbb{A}^*$. Without loss of generality, we suppose users are in the group k. We denote $S^{(i)} = \{S_1, S_2, \cdots, S_i\}$, where S_j, $(j \in [1, i])$ represents the queried attribute set until S_i is queried. For users in group k, a value $g^{(\alpha+\theta_k)/\beta_1}$ is fixed in their secret keys. When the adversary queries a security key associated with S_i, it means that the group is absorbing new collaborators possessing attribute set S_i, and the group will have attribute set $\gamma = \{S_i\} \bigcup S^{(i-1)} = \{S_1, S_2, \cdots, S_{i-1}, S_i\} = S^{(i)}$. Therefore, after querying about q_1 sets, the attribute set of the adversary is $\gamma = S^{(q_1)} = \{S_1, \cdots, S_{q_1}\}$. Accordingly, in the security game, neither $S^{(q_1)}$ nor $S^{(q)}$ can satisfy the challenge access structure $\mathbb{A}^*$.

Theorem 1. *Suppose there is no polynomial time adversary who can break the security of CP-ABE with non-negligible advantage; then there is no polynomial time adversary who can break our system with non-negligible advantage.*

Proof: Suppose we have an adversary $\mathcal{A}$ with non-negligible advantage against our proposed scheme. Using $\mathcal{A}$, we show how to build an adversary $\mathcal{B}$ that breaks the CP-ABE scheme [11] with non-negligible advantage. The adversary $\mathcal{B}$ plays a similar game with the CP-ABE scheme.

- **Initialization**. The adversary $\mathcal{B}$ takes the public key of CP-ABE, $PK' = \{G, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}\}$, and the corresponding private key (β, g^{α}) is unknown to the adversary.
- **Setup**. The adversary $\mathcal{B}$ selects a random number $t \in Z_p$, and computes the parameters of our approach from PK' as $PK = \{G, g, h_1 = g^{\beta}, f_1 = g^{1/\beta}, h_2 = g^{t\beta}, f_2 = g^{1/t\beta}, e(g,g)^{\alpha}\}$. That is, the adversary $\mathcal{B}$ sets $\beta_1 = \beta$, and $\beta_2 = t \cdot \beta$. Then the public key PK is given to the adversary $\mathcal{A}$.
- **Phase 1**. In this phase, $\mathcal{B}$ answers security key queries of $\mathcal{A}$. Suppose the adversary $\mathcal{B}$ is given a security key query for a set S_i, $1 \le i \le q_1$. In the first query related to S_1, in order to answer the query, $\mathcal{B}$ makes a security key query to CP-ABE challenger with S_1 twice. As a result, $\mathcal{B}$ obtains two different security keys ($SK_1^{(0)}$ and SK_1) corresponding to S_1. In each subsequent query, the adversary $\mathcal{B}$ only queries CP-ABE challenger once. Thus, the responses of CP-ABE challengers are:

$$SK_1^{(0)} = (D = g^{(\alpha+r)/\beta}, \forall a_{1,j} \in S_1 : D_j = g^r H(a_{1,j})^{r_{1,j}}, D'_j = g^{r_{1,j}}),$$

and for all $i \in [1, q_1]$

$$SK_i = (D = g^{(\alpha+r_i)/\beta}, \forall a_{i,j} \in S_i : D_{i,j} = g^{r_i} H(a_{i,j})^{r'_{i,j}}, D'_{i,j} = g^{r'_{i,j}}),$$

where $SK_1^{(0)}$ denotes one of the security keys for S_1, and SK_i denotes the security key for every queried set S_i (including S_1), $i \in [1, q_1]$. $\{a_{i,j}\}$ are the attributes in S_i, and $r, r_i, r_{1,j}, r'_{i,j}$ are random numbers in Z_p. From $SK_1^{(0)}$ and SK_i, $\mathcal{B}$ can obtain $g^{(r-r_i)/\beta}$ by dividing D in $SK_1^{(0)}$ by D in SK_i. $\mathcal{B}$ selects random numbers $t_i, t_{i,j} \in Z_p$, where i denotes that the security key query is associated with S_i, and j denotes the j-th attribute in S_i. Let $r^* = t_i - r_i$ and $r'' = t_{i,j} - r'_{i,j}$. Then $\mathcal{B}$ can derive the security key as

$$SK_i^* = (D = g^{(\alpha+r)/\beta}, D_{i,j} = g^{r^*} H(a_{i,j})^{r''}, D'_{i,j} = g^{r''}, E_i = g^{(r+r^*)/t\beta}),$$

for $1 \le i \le q_1, 1 \le j \le n_i$, where n_i is the number of attributes in S_i. $r + r^* = r - r_i + t_i$, thus $E_i = g^{(r+r^*)/t\beta}$ can be computed by $(g^{(r-r_i)/\beta})^{t^{-1}} \cdot g^{t_i/t\beta}$. Intuitively, we can regard each query of the set S_i as the partitipation of a new collaborator in the collaborating team to decrypt the ciphertext. The parameter r can act the role of θ_k. Thus, for each key query corresponding to S_i, a security key is returned as SK_i^*. After returning SK_i to $\mathcal{A}$, the attribute set of the adversary $\mathcal{A}$ is the sum of all queried attribute sets $\gamma = \{S_1, S_2, \cdots, S_i\}$.
- **Challenge**. When Phase 1 is finished, it outputs an access structure $\mathbb{A}^*$ and two messages $M_0, M_1 \in \mathbb{G}$ in which $\mathcal{A}$ wishes to be challenged. To be noted, the combination of $\mathcal{A}$'s queried attribute sets, i.e. $S^{(q_1)} = \{S_1, S_2, \cdots, S_{q_1}\}$, should not satisfy $\mathbb{A}^*$. $\mathcal{B}$ gives the two messages $M_0, M_1 \in \mathbb{G}$ to the CP-ABE challenger, and the challenger generates the ciphertext.

$$CT = (\mathbb{A}^*, \tilde{C} = M_b \cdot e(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}).$$

Then $\mathcal{B}$ computes the ciphertext for $\mathcal{A}$ from CT as :

$$CT^* = (\mathbb{A}^*, \tilde{C} = M_b \cdot e(g,g)^{\alpha s}, C = h_1^s, \bar{C} = h_2^s, \forall y \in \mathbb{Y} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}, \forall x \in \mathbb{X} : \hat{C}_x = h_2^{q_x(0)}.$$

In CT^*, $\tilde{C}$, C, C_y and C'_y can be easily obtained from CT. As is pointed out in [21], $h_2^{q_x(0)}$ can be computed from h_2^s and other known values. Finally, $\mathcal{B}$ returns the ciphertext CT^* to the adversary $\mathcal{A}$.
- **Phase 2**. $\mathcal{A}$ issues queries which are not issued in Phase 1. The combination of $\mathcal{A}$'s queried attribute sets, i.e. $S^{(q)} = \{S_1, S_2, \cdots, S_{q_1}, S_{q_1+1}, \cdots, S_q\}$ should not satisfy the access structure $\mathbb{A}^*$. $\mathcal{B}$ responds by the same process as he/she does in Phase 1, returning SK_i^* to each query corresponding to S_i, $i \in [q_1 + 1, q]$.
- **Guess**. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$, and then $\mathcal{B}$ concludes its own game by outputting b'. According

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

to the security model, the advantage of the adversary $\mathcal{B}$ against our approach is:

$$Adv_{\mathcal{B}} = |Pr[b = b'] - 1/2| = Adv_{\mathcal{A}}.$$

This means $\mathcal{B}$ has non-negligible advantage against the CP-ABE scheme, which completes the proof of the theorem.

*2) **Data confidentiality in Model B**:* In fact, when considering users in different groups in the threat model, the proof is similar to that of users in the same group. Since most of the part are similar, we just concentrate on the difference in the process of proof. For simplicity, we prove the security of our approach against adversaries from the different group where there are only 2 users. The case can be expanded to $n \geq 2$ easily. We suppose users are U_a and U_b, and they are in groups l and m, respectively.

When we prove data confidentiality against users from the same group, in Phase 1, for each key query corresponding to S_i, a security key is returned as SK_i^*.

$$SK_i^* = (D = g^{(\alpha+r)/\beta}, D_{i,j} = g^{r^*} H(a_{i,j})^{r''}, \\ D'_{i,j} = g^{r''}, E_i = g^{(r+r^*)/t\beta}),$$

Here, the parameter r can act the role of θ_k. Since we assume the adversary asking for queries with regard to the same group, r is fixed for all sets. After the simulator returning $\{SK_1^*, \cdots, SK_{q_1}^*\}$ to $\mathcal{A}$, the attribute set of the adversary $\mathcal{A}$ is the sum of all queried attribute sets $\gamma = \{S_1, S_2, \cdots, S_{q_1}\}$.

However, when the adversary is a set of users from different groups where each user belongs to a different group and neither user belongs to the same group, it is equivalent that the adversary keeps querying keys with regard to different sets associated with different groups each time, as long as each single set does not satisfy the access tree. Phase 1 in this threat model will be:

- **Phase 1.** In this phase, $\mathcal{B}$ answers security key queries of $\mathcal{A}$. Suppose the adversary $\mathcal{B}$ is given a security key query for a set S_a. In the first query related to S_a, in order to answer the query, $\mathcal{B}$ makes a security key query to CP-ABE challenger with S_a twice. As a result, $\mathcal{B}$ obtains two different security keys ($SK_a^{(0)}$ and $SK_a^{(1)}$) corresponding to S_a. Thus, the responses of CP-ABE challengers are:

$$SK_a^{(0)} = (D = g^{(\alpha+r_a^0)/\beta}, \forall a_{1,j} \in S_a : \\ D_j = g^{r_a^0} H(a_{a,j})^{r_{a,j}}, D'_j = g^{r_{a,j}}),$$

$$SK_a^{(1)} = (D = g^{(\alpha+r_a^1)/\beta}, \forall a_{a,j} \in S_a : \\ D_{a,j} = g^{r_a^1} H(a_{a,j})^{r'_{a,j}}, D'_{a,j} = g^{r'_{a,j}}),$$

where $\{a_{a,j}\}$ are the attributes in S_a, and $r_a^1, r_a^0, r_{a,j}, r'_{a,j}$ are random numbers in Z_p. From $SK_a^{(0)}$ and $SK_a^{(1)}$, $\mathcal{B}$ can obtain $g^{(r_a^0 - r_a^1)/\beta}$ by dividing D in $SK_a^{(0)}$ by D in $SK_a^{(1)}$. $\mathcal{B}$ selects random numbers $t_a, t_{a,j} \in Z_p$, where a denotes that the security key query is associated with S_a, and j denotes the j-th attribute in S_a. Let $r^* = t_a - r_a^1$ and $r'' = t_{a,j} - r'_{a,j}$. Then $\mathcal{B}$ can derive the security key as

$$SK_a^* = (D = g^{(\alpha+r_a^0)/\beta}, D_{a,j} = g^{r^*} H(a_{a,j})^{r''}, \\ D'_{a,j} = g^{r''}, E_a = g^{(r_a^0+r^*)/t\beta}),$$

where n_a is the number of attributes in S_a. $r_a^0 + r^* = r_a^0 + t_a - r_a^1$, thus $E_a = g^{(r_a^0+r^*)/t\beta}$ can be computed by $(g^{(r_a^0 - r_a^1)/\beta})^{t^{-1}} \cdot g^{t_a/t\beta}$. Intuitively, the parameter r_a^0 can act the role of θ_l.

For the query made by U_b with regard to S_b, the simulator $\mathcal{B}$ repeats the above procedure and generates

$$SK_b^* = (D = g^{(\alpha+r_b^0)/\beta}, D_{b,j} = g^{r^*} H(a_{b,j})^{r''}, \\ D'_{b,j} = g^{r''}, E_b = g^{(r_b^0+r^*)/t\beta}),$$

where $r^* = t_b - r_b^1$ and $r'' = t_{b,j} - r'_{b,j}$. The simulator $\mathcal{B}$ returns SK_a^* and SK_b^* to the adversary $\mathcal{A}$.

Phase 1 is repeated for $q_1/2$ times, which denotes the adversary $\mathcal{A}$ gets q_1 secret keys, we denote them as $\{S_1, S_2, \cdots, S_{q_1}\}$.

The following procedures of the proof is the same as the procedures in threat models where users are in the same group.

To conclude, the difference of proofs between the two threat models is the D component. For users in the same group, the same $D = g^{(\alpha+r)/\beta}$ is returned in all security key queries, which means group secret θ_k (the parameter r) is fixed. For users from different groups, different $D = g^{(\alpha+r_a^0)/\beta}$, $D = g^{(\alpha+r_b^0)/\beta}$ is returned for every key query, meaning the group secret is changing each time.

*3) **Data confidentiality in the combined model**:* In reality, the threat model is a combination of the two threat models. That is, the adversary is a group of users where some of them are from the same group, and the rest are from different groups. In this case, we can firstly group the users from the same group, and regard them as a single user (e.g. U_a) in the latter threat model. Then, using the security property in two models, we can prove the data confidentiality.

B. User Collusion Resistance

The property of data confidentiality implicitly implies the property of user collusion resistance. The concept of user collusion resistance means that collaboration is only allowed upon translation nodes. In our proof, given the combined attribute set $\gamma = \{S_1, \cdots, S_n\}$ where there are n collaborating users (in other words, n queries are made to the challenger), and γ does not satisfy the access structure $\mathbb{A}^*$ by the tree satisfaction algorithm $\mathcal{T}(\gamma)$, the adversary cannot decrypt the ciphertext. We give an example to help readers understand this property.

We consider the collusion attack launched by malicious users within the same group m. Each attribute set of those users cannot individually satisfy the policy tree. For users in the group m, their attribute sets make up a new combined attribute set γ. Taking the policy tree of Fig. 1 in which only the leaf node denoting the attribute '*Auditor*' is set as a translation node. Assume that Alice has the attribute set {'*Senior*', '*Manager*'}, and Bob has the attribute set {'*Junior*', '*Accountant*'}. Their combined attribute set seems to have sufficient attributes to satisfy the policy tree. However, since the leaf node denoting the attribute '*Accountant*' is not a translation node, they cannot collude to decrypt the ciphertext without the necessary translation value.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

C. Controlled Collaboration within the Same Group

Users are able to collaborate if and only if they satisfy all of the following requirements: 1) They are from the same group; 2) Their combined attribute set γ has sufficient attributes that satisfy the policy tree; 3) The collaboration happens only on translation nodes on which data owner allows the collaboration. The collaborating users who satisfy the requirements 1) and 2), except 3) is considered as colluding users, which will be denied to access by the definition of user collusion resistance. The property of data confidentiality in Model B in Section VI-A2 explicitly guarantees that users from different groups cannot collaborate.

D. Security Key Privacy

It is worth noting that in the decryption process, users do not directly expose their secret keys that are related to attributes. Instead, each user (for example, the user U_j) takes his/her secret key as input in his/her own decryption device and outputs the secret share $e(g,g)^{r_j q_x(0)}$. With the translation value, they then translate the output to $e(g,g)^{r_i q_x(0)}$ for a representative user with identifier u_i by receiving E_i. They only transmit $e(g,g)^{r_i q_x(0)}$ to the representative. The disclosure of E_i and $e(g,g)^{r_i q_x(0)}$ does not harm the privacy of secret keys that are related to attributes.

E. Secure Revocation of the Collaboration

From the perspective of the data owner, if he/she wants to revoke the collaboration between users to preserve the privilege, our scheme can support this procedure by simply removing the translation value, since the collaboration can only be achieved when there exists a corresponding translation value. The data owner can ask the cloud server to delete the component with respect to the translation value, and thereafter the policy tree associated with the ciphertext is updated and collaboration is not allowed anymore. For users who intend to access the updated ciphertext, due to lack of translation values, the collaboration is denied.

From the perspective of users, if one or more users want to leave the collaboration team, they can simply refuse to process the attributes for which they are responsible in the decryption. If there are no any other collaborators who are still in the team and can also process these attributes to satisfy the policy tree, the access privilege will not be obtained.

F. Non-reusability of Intermediate Results

The property of non-reusability of intermediate results can guarantee that each collaboration is only useful to decrypt one ciphertext. Thus, the collaboration to decrypt one ciphertext will not harm the security of other ciphertexts. If a user gets access once, he/she cannot use the same intermediate results thereafter to access other objects. The rationality is given in what follows.

As shown in Fig. 3, in the collaborative decryption, a user whose identifier u_i labels the root node can be selected as the representative to decrypt the ciphertext. Other users, such as user U_j and user U_k, are responsible to decrypt some nodes that are labelled with their identifiers, and they translate the decrypted secrets (such as $e(g,g)^{r_j q_2(0)}$) to those that are related to identifier u_i, such as $e(g,g)^{r_i q_2(0)}$. The translated output can be viewed as if it was computed by user U_i. The representative user U_i gathers all the translated output computed by other users. Together with the output computed by himself/herself, user U_i can get $e(g,g)^{r_i s}$ and further obtain the symmetric key κ. Then, he/she can distribute κ to other users who also want to obtain the plaintext.

From the above process, the intermediate results in the collaboration are the translated output F_z of function $F'_z = DecrytNode(CT, z, \gamma, u_j/u_k)$, such as $e(g,g)^{r_i q_2(0)}, e(g,g)^{r_i q_3(0)}$, and only E_i is exposed to collaborators. These intermediate results are only related to one specific ciphertext that they want to decrypt in the collaboration process, where the purpose is to construct $e(g,g)^{r_i s}$. Here, s is the secret of the root node of the policy tree. For any other ciphertexts, even if the policy tree is the same as the decrypted ciphertext, since the secret of the root node is generated randomly, such as s' for a specific ciphertext, the intermediate results in the last collaboration will not be helpful to construct $e(g,g)^{r_i s'}$. In other words, if the collaborators have collaborated once, the exposition of the intermediate results such as F_z and translation key E_i will not give any advantage to collaborators (such as user U_i) to gain the attributes belonging to other collaborators. After the collaboration, any collaborator still possesses the original attribute set assigned to him/her. To decrypt a ciphertext whose policy tree cannot be satisfied by himself/herself, he/she should find collaborators again.

G. Discussion of Dynamic Groups

In fact, our scheme assumes that the groups to which users belong are relatively fixed for a long time. This assumption is reasonable since the group is built by long-term social relations, like in companies and organizations. Thus, it will not change frequently. When a user U_i is removed from one group A to another group B, we must revoke all his/her secret keys in the previous group A. Thus, one requirement is to update all secret keys related to attributes that U_i has, in case he/she uses his/her previous key to decrypt or collaborate with other users, acting as a member of the previous group A. The problem becomes to revoke all attributes of U_i. We can adopt similar ideas of existing works (e.g. [2, 15]) which address the attribute revocation to revoke U_i's attributes. For U_i's secret key that is related to group secret keys, since U_i's secret key component corresponding to attributes are not valid any more, he/she cannot collaborate with any other users in group A. Thus, the group secret key can remain unchanged.

VII. Performance Analysis

Since our scheme is built on BSW scheme [11], we make a comparison with it to show our performance. For discussion convenience, let n_u denote the average number of attributes of a user, n_c denote the average number of attributes associated with the policy tree of ciphertext, $|tr|$ be the average number of translation nodes in a ciphertext, and $|tr_\gamma|$ be the average

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

necessary translation nodes for a set γ to decrypt ciphertexts, where γ is a combined attribute set which contains attribute sets from multiple users within the same group.

A. Computation Overhead

In terms of the computation overhead, the main operations include exponentiation and pairing, since the cost of addition and multiplication operation can be neglected compared with pairing and exponentiation operation. Let T_p denote the cost of pairing operation on group $\mathbb{G}$ and T_e denote the cost of exponentiation operation. Also, let $n_{c,u}$ be the average number of attributes for a user to decrypt ciphertexts, $n_{c,\gamma}$ be the average number of attributes used to decrypt ciphertext by a set of users with attribute set γ, and $|nl|$ be the the average number of non-leaf nodes when computing the secret of the root node from leaf nodes in access policies. Assume that there are n groups. From Table I, we can observe that the two schemes share similar computation overhead that is not related to translation nodes except that there are n exponentiation operations in terms of the master secret key of n groups. The introduced overhead of translation nodes is linear to the number of translation nodes, which is caused by the additional pairing operation and exponentiation operation at translation nodes. When the number of translation nodes is not very large, which is practical in real world, the added computation overhead is acceptable.

TABLE I: Computation Overhead

Phase	BSW scheme	Ours
$Setup$	$3T_e + T_p$	$(5+n)T_e + T_p$
$KeyGen$	$(2n_u + 2)T_e$	$(2n_u + 3)T_e$
$Encrypt$	$(2n_c + 2)T_e$	$(2n_c + 3 + \|tr\|)T_e$
$Decrypt$	$(2n_{c,u} + 1)T_p$ $+(n_{c,u} + \|nl\|)T_e$	$(2n_{c,\gamma} + \|tr_\gamma\| + 2)T_p$ $+(n_{c,\gamma} + \|nl\|)T_e$

B. Storage and Communication Overhead

We analyze secret key size, ciphertext size and communication overhead in collaboration to show the efficiency of our proposed scheme. Let $|p|$ be the size of element in the groups with the prime order p. The results are shown in Table II. As is shown, secret key size and ciphertext size are similar between two schemes, where the ciphertext size grows linearly as the number of translation nodes increases. In terms of communication cost, only one user U_i, needs to receive $|tr_\gamma|$ translation values and broadcast his/her translation key E_i for a successful decryption. Other users just need to receive E_i, translate their computed output of their attribute sets and transmit them to the user U_i. The main communication overhead lies on only one user. As we mentioned above, $|tr_\gamma|$ will not be very large, thus the communication overhead is affordable.

C. Comparison with the Trivial Solution

In this section, we compare our scheme with the trivial solution (named as TrSo in the figures). The trivial solution is further formulated as follows. As illustrated in Fig. 4, we assume

TABLE II: Storage and Communication Overhead

Sectors	BSW scheme	Ours
Ciphertext Size	$(2n_c + 2)\|p\|$	$(2n_c + 3 + \|tr\|)\|p\|$
Secret Key Size	$(2n_u + 1)\|p\|$	$(2n_u + 2)\|p\|$
Communication Overhead	N/A	$(\|tr_\gamma\| + 1)\|p\|$

that an access policy expressed as "$(\ t\ of\ \{A_1, A_2, \ldots, A_n\})$ *AND* B" and the collaboration is allowed at the node denoted as A_n, implying that the sub-policy rooted at A_n can be satisfied by a collaborator.

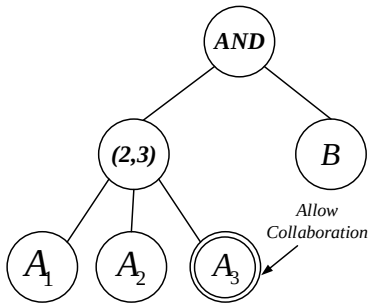


Fig. 4: An Example

We use an access policy shown in Fig. 4 to show how the trivial solution works. By the trivial solution, the policy tree shown in Fig. 4 expressed by our approach is now divided into two sub-policies, as shown in Fig. 5. Sub-policy A denotes the scenario where an independent user should satisfy the policy tree in sub-policy A independently to access data, while sub-policy B denotes how collaborating users should access the data. In the example, one of the users should satisfy the left-side policy tree in sub-policy B, and the other user should satisfy node 'A_3'. The symbol 'AND' means the behavior of collaboration. The two users whose attribute sets satisfy each tree (A_3 can be seen as a tree with only one node) of sub-policy B can collaborate to combine their decrypted results to access data. We depict the two policies, sub-policy A and sub-policy B, by connecting them to an OR gate, and the dot line means the two policies can be seen as sub-policies of a unified tree. Users can access data by either policy.

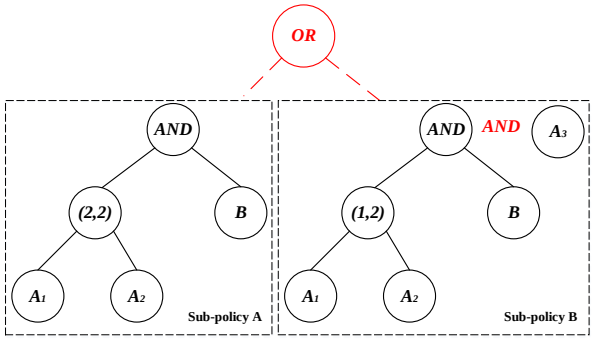


Fig. 5: Two Sub-Policies by the Trivial Solution

As we can see from Fig. 5, almost each node appears twice in the description of the access policies by the trivial solution. Intuitively, sub-policy A means that, in this case,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

the collaboration is not needed, and sub-policy B means that, in this case, the collaboration is required. In the worst case, all the translation nodes are rooted at different parent nodes. By the trivial solution, each translation node will divide the original tree into two sub-policies, each of which corresponds to the scenario where the collaboration is needed or not upon that node. For example, in Fig. 5, one translation node (i.e. A_3) will divide the 'compound policy tree' into two sub-policies, such as sub-policy A and sub-policy B. The divided sub-polices will further be divided due to translation nodes inside the sub-policies. For instance, if B is a parent node of leaf nodes denoting attributes and one of these leaf nodes is set as a translation node, sub-policy A will further be divided into two sub-policies, and so does sub-policy B. Therefore, the expansion of ciphertext will grow exponentially with the increase of the number of translation nodes in the worst case.

We further describe the details how the trivial solution works by the example shown in Fig. 4. Similar to the description in Section V-A, data will be encrypted under sub-policy A and sub-policy B separately. The generated ciphertext will contain two pieces of ciphertext according to each sub-policy. Sub-policy A is designed for independent users who do not need collaboration and sub-policy B is designed for collaborating users. For sub-policy A, the encryption and decryption are just the same as the phases of CP-ABE schemes such as that of [11]. For sub-policy B, in the encryption phase, s is randomly chosen. s_1 and s_2 are assigned to the root node of the right tree and the left tree, respectively, with the restriction that $s_1 = q_r(index(1))$ and $s_2 = q_r(index(2))$. q_r is a polynomial with degree $d = 1$, and $q_r(0) = s$. Then, s can be constructed by using Lagrange polynomial interpolation with s_1 and s_2. For simplicity, we assume that $s = s_1 + s_2$. Then, s_1 and s_2 are distributed to their child nodes as a normal CP-ABE scheme does, taking place of the role of s in a traditional CP-ABE scheme. In the decryption phase, the user U_1 is expected to get the results $e(g,g)^{\alpha s_1}$ and the user U_2 is expected to get $e(g,g)^{\alpha s_2}$. Then, $e(g,g)^{\alpha s}$ can be obtained by multiplying $e(g,g)^{\alpha s_1}$ and $e(g,g)^{\alpha s_2}$. Then, by dividing the component $e(g,g)^{\alpha s}$, the encrypted symmetric key κ is exposed to users. Aiming at dividing users into groups and only allowing collaboration between users from the same group, some modifications on the master secret key α are necessary. If there exists k groups, then the master key is replaced by $\{\alpha + \theta_i, i \in [1,k]\}$ and the public key is replace by $\{e(g,g)^{\alpha+\theta_i}, i \in [1,k]\}$. To encrypt a ciphertext κ, the component $C = \kappa \cdot e(g,g)^{\alpha s}$ will be replaced by $\widetilde{C}^{(i)} = \kappa \cdot e(g,g)^{(\alpha+\theta_i)s}, i \in [1,k]\}$. $\widetilde{C}$ will expand k times for every possible collaborating group. Accordingly, if a user is assigned in the group j, the user's secret key component D of SK will be replaced by $D = g^{(\alpha+\theta_j+r)/\beta}$. The decrypted results will be $e(g,g)^{(\alpha+\theta_j)s_1}$ and $e(g,g)^{(\alpha+\theta_j)s_2}$. Thus, $e(g,g)^{(\alpha+\theta_j)s}$ is obtained and κ is computed with the corresponding $\widetilde{C}^{(j)}$ of CT: $\widetilde{C}^{(j)} = \kappa \cdot e(g,g)^{(\alpha+\theta_j)s}$ ($j \in [1,k]$).

We analyze how the encryption time, decryption time and ciphertext size changes as the number of translation nodes grows. For illustration purpose, we use a policy tree as shown in Fig. 6. The policy tree contains non-leaf nodes representing threshold gate and leaf nodes representing attributes, which is a normal policy tree. Thus, there is no loss of generality by analyzing the property of the example. The only point to be noted is that we set each translation node to be rooted at different parent nodes. If there are more than one translation nodes rooted at the same parent node, the performance is similar to the scenario where there is only one translation node rooted at that parent node. Therefore, we analyze the following three different cases by changing the number of translation nodes in the policy tree. Case 1: There is only one translation node at node A_3. Case 2: There are two translation nodes lying upon A_3 and B_3 respectively. Case 3: There are three translation nodes lying at A_3, B_3 and A_{23}, respectively.

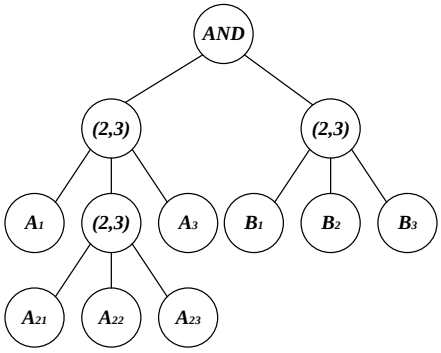


Fig. 6: An Example for Analysis

We conduct the simulation on Ubuntu 14.04 with Intel(R) Core(TM) i7-3610QM CPU @2.30GHz and 8.00 GB RAM. The code uses the Pairing-Based Cryptography library version 0.5.14. We use type A pairing parameters contained in param/a.param. We test the time of one multiplication, one pairing operation and one exponentiation operation respectively. The time of multiplication is only $1/500$ times of the other two operations, and we thus ignore it in our analysis. The time of one pairing operation is about 1.0ms and that of one exponentiation operation is about 1.5ms. To show the superiority of our scheme, we let every ciphertext be associated with only one group to reduce the complexity of the trivial solution. When ciphertexts are related to more groups, our scheme is even far superior.

1) Encryption Time: The comparison of encryption time between the trivial solution and our proposed scheme is shown in Fig.7. Our proposed scheme has an evident advantage. In the encryption phase, the main operation is exponentiation operation of which we count the times. As we have predicted, the encryption time by the trivial solution will cause expansion exponentially, as shown in Fig.7. When the number of translation nodes increases from one to three, the encryption time of the trivial solution is almost 2 times (51 ms), 3 times (97.5 ms) and 6 times (187.5 ms) of that of our proposed scheme (about 30ms). What's more, we only assume that there is only one group related to the ciphertext. If there are more groups, the encryption of the trivial solution will be even much slower. On the contrast, in our proposed scheme, the encryption time remains steady at about 30ms, where only one additional exponentiation operation is added for every translation node. No additional cost will be added if there are more potential

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

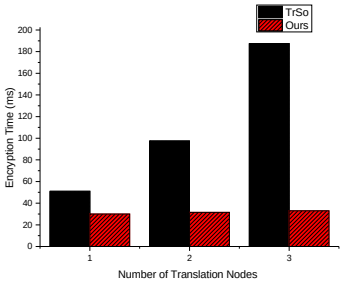


Fig. 7: Encryption Time

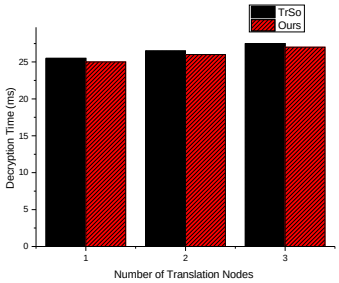


Fig. 8: Decryption Time

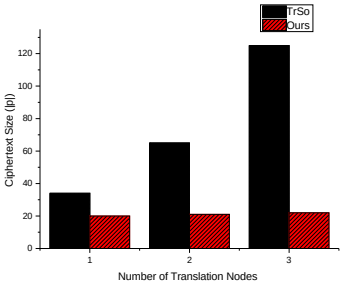


Fig. 9: Ciphertext Size

groups related to the ciphertext. In other words, the ciphertext size is independent of the number of potential groups of data consumers in our proposed scheme.

2) Decryption Time: We calculate the decryption time in the worst case where all translation nodes are involved and the collaborating users have to possess maximum number of attributes to satisfy the access policy. We count the decryption time of all participants in the collaborating team. As shown in Fig. 8, the decryption time of both schemes is very close, and our scheme spends a slight less time (0.5 ms) than the trivial solution. We take the example of Fig. 4 to give the reason of the results. In both schemes, for every leaf node that is not a translation node, the operations are the same which include two pairing operations and the same number of times of exponentiation operations. Regarding the translation node (Node A_3), our proposed scheme requires one pairing operation to translate the value associated with $r_{i'}$ to r_i and one exponentiation operation to pass the value to its parent node. Finally, two more pairing operations are needed at the root node to decrypt the ciphertext. The trivial solution requires two pairing operations to remove the random number r_i and $r_{i'}$ of the two collaborating users (with identifier u_i and identifier $u_{i'}$, respectively), which are embedded in the root secrets of two policy trees in Fig. 5. Finally two exponentiation operations are need to reconstruct the final secret to decrypt the ciphertext, using exponential polynomial interpolation. Therefore, the decryption time of both schemes are very close.

3) Ciphertext Size: The growth of ciphertext size versus the increase of the number of translation nodes shares the same tendency with that of encryption time, as is given in Fig. 9. In the trivial solution, the ciphertext size grows rapidly from $34|p|$ to $125|p|$ because of the expanded policies, while that of our proposed schemes grows only a little bit (from $20|p|$ to $22|p|$) for an additional translation value for each translation node.

VIII. Conclusion

In this paper, we proposed an attribute-based controlled collaborative access control scheme, in which data owners can designate selected users to collaborate for accessing their data at their will. Considering practical scenarios, we let users within the same group to collaborate for data access. More importantly, the data owner can devise the way for chosen users to combine their attribute sets to satisfy the access policy, and at the same time also resist the collusion attack when curious users try to combine their attribute sets in other ways. Technically, we embed translation keys in the secret keys of CP-ABE schemes and modify the secret keys to associate groups to users. The data owner can designate collaboration by setting translation nodes in the policy tree. Our security analysis shows that our proposed scheme effectively supports data confidentiality, user collusion resistance, controlled collaboration within the same group, secret key privacy, secure revocation of the collaboration and non-reusability of intermediate results. The performance is very satisfactory. Thus, our proposed scheme is highly promising to provide fine-grained access control in collaborative settings where data need to be accessed by multiple users.

Acknowledgements

The authors sincerely thank the anonymous referees for their valuable suggestions that have led to the present improved version. This work is supported in part by the National Key R&D Plan of China under Grants No. 2017YFB0801702 and No.2016YFB0800301, the National Natural Science Foundation of China under Grant No. 61671420, and Youth Innovation Promotion Association CAS under Grant No. 2016394.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 2895–2903.

[3] 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*, vol. 24, no. 1, pp. 131–143, 2013.

[4] Y. Wu, Z. Wei, and H. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 778–788, 2013.

[5] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. Wei, and P. Hong, "RAAC: Robust and auditable access control

with multiple attribute authorities for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 953–967, 2017.

[6] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, 2016.

[7] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2062–2074, 2018.

[8] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[9] T. Tassa, "Hierarchical threshold secret sharing," *Journal of Cryptology*, vol. 20, no. 2, pp. 237–264, 2007.

[10] M. Li, X. Huang, J. K. Liu, and L. Xu, "GO-ABE: group-oriented attribute-based encryption," in *Proceedings of the 8th International Conference on Network and System Security (NSS)*. Springer, 2014, pp. 260–270.

[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2007, pp. 321–334.

[12] 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)*, 2003.

[13] E.-j. Goh, H. Shacham, N. Modadugu, and D. Boneh, "SiRiUS: Securing remote untrusted storage," in *Proceedings of the 10th Network and Distributed Systems Symposium Security (NDSS)*, vol. 3, 2003, pp. 131–145.

[14] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography (PKC)*. Springer, 2011, pp. 53–70.

[15] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[16] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography (PKC)*. Springer, 2014, pp. 293–310.

[17] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the 20th USENIX Security Symposium*, vol. 2011, no. 3. USENIX, 2011.

[18] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proceedings of the 34th IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2677–2685.

[19] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2014.

[20] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *European Symposium on Research in Computer Security (ESORICS)*. Springer, 2009, pp. 587–604.

[21] Z. Wan, J. Liu, and R. H. Deng, "HASBE: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2012.

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

[23] J. M. M. Perez, G. M. Perez, and A. F. Gomez-Skarmeta, "SecRBAC: Secure data in the clouds," *IEEE Transactions on Services Computing, Avaiable online*, 2016.

[24] S.-C. Yeh, M.-Y. Su, H.-H. Chen, and C.-Y. Lin, "An efficient and secure approach for a cloud collaborative editing," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1632–1641, 2013.

[25] R. K. Thomas, "Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments," in *Proceedings of the 2nd ACM Workshop on Role-based Access Control*. ACM, 1997, pp. 13–19.

[26] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, and S. Ioannidis, "SAMPAC: socially-aware collaborative multi-party access control," in *Proceedings of the 7th ACM Conference on Data and Application Security and Privacy (CODASPY)*. ACM, 2017, pp. 71–82.

[27] B. Carminati and E. Ferrari, "Privacy-aware collaborative access control in web-based social networks," in *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, vol. 5094. Springer, 2008, p. 81.

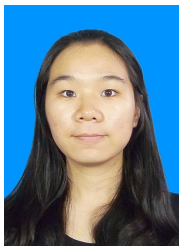
[28] C. Hu, W. Li, X. Cheng, J. Yu, S. Wang, and R. Bie, "A secure and verifiable access control scheme for big data storage in clouds," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2017.

[29] G. R. Blakley, "Safeguarding cryptographic keys," *Proceedings of 1979 AFIPS National Computer Conference*, vol. 48, pp. 313–317, 1979.

[30] L. Harn and F. Miao, "Weighted secret sharing based on the chinese remainder theorem," *International Journal of Network Security*, vol. 16, no. 6, pp. 420–425, 2014.

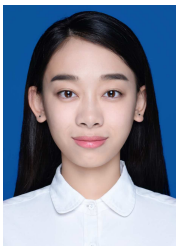
[31] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 20th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2001, pp. 213–229.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2019.2911166, IEEE Transactions on Information Forensics and Security

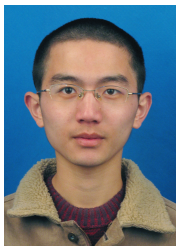


Yingjie Xue received her B.S. degree from the department of Information Security, University of Science and Technology of China (USTC), in 2015 and received her Master degree in Communication and Information System from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2018. Now she is a Ph.D. student in Department of Computer Science in Brown University. Her research interests include Network security and Cryptography.

Kaiping Xue (M'09-SM'15) received his B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an Associate Professor in the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security. He is the corresponding author of this paper.



Na Gai received the B.S. degree from the department of Information Security, University of Science and Technology of China (USTC), in 2018. She is currently a graduated student in Information Security from the Department of Electronic Engineering and Information Science (EEIS), USTC. Her research interests include network security protocol design and analysis.



Jianan Hong received the B.S. degree from the department of Information Security, University of Science and Technology of China (USTC), in 2012 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2018. Now he is a research Engineer in Huawei Shanghai Research Institute, Shanghai. His research interests include secure cloud computing and mobile network security.

David S.L. Wei (SM'07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. He is currently a Professor of Computer and Information Science Department at Fordham University. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 100 technical papers in various archival journals and conference proceedings. He served on the program committee and was a session chair for several reputed international conferences. He was an associate editor of IEEE Transactions of Cloud Computing, a lead guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Mobile Computing and Networking, a lead guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Networking Challenges in Cloud Computing Systems and Applications, a guest editor of IEEE Journal on Selected Areas in Communications for the special issue on Peer-to-Peer Communications and Applications, and a lead guest editor of IEEE Transactions on Cloud Computing for the special issue on Cloud Security. He is currently an Associate Editor of Journal of Circuits, Systems and Computers, and a guest editor of IEEE Transactions on Big Data for the special issue on Trustworthiness in Big Data and Cloud Computing Systems. Currently, His research interests include cloud computing, big data, IoT, and cognitive radio networks.

Peilin Hong received her B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986. Currently, she is a Professor and Advisor for Ph.D. candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published 2 books and over 150 academic papers in several journals and conference proceedings.