Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing

Shaohua Tang, Xiaoyu Li, Xinyi Huang, Yang Xiang, Lingling Xu

Abstract—Access control is an indispensable security component of cloud computing, and hierarchical access control is of particular interest since in practice one is entitled to different access privileges. This paper presents a hierarchical key assignment scheme based on linear-geometry as the solution of flexible and fine-grained hierarchical access control in cloud computing. In our scheme, the encryption key of each class in the hierarchy is associated with a private vector and a public vector, and the inner product of the private vector of an ancestor class and the public vector of its descendant class can be used to derive the encryption key of that descendant class. The proposed scheme belongs to direct access schemes on hierarchical access control, namely each class at a higher level in the hierarchy can directly derive the encryption key of its descendant class without the need of iterative computation. In addition to this basic hierarchical key derivation, we also give a dynamic key management mechanism to efficiently address potential changes in the hierarchy. Our scheme only needs light computations over finite field and provides strong key indistinguishability under the assumption of pseudorandom functions. Furthermore, the simulation shows that our scheme has an optimized trade-off between computation consumption and storage space.

Index Terms—Access control, hierarchical key assignment, linear geometry, pseudorandom function, strong key indistinguishability.

1 Introduction

Cloud computing has attracted great attention of both academia and information technology industry. Built on the architecture of parallel and distributed computing, this new computing paradigm possesses numerous advantages including low cost, high-efficiency, flexibility and scalability. From the bottom layer to the top, cloud computing can be divided into three delivery models, namely infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [2]. Nowadays, there are many kinds of cloud service systems providing services via the Internet, such as Amazon's EC2, IBM's Blue Cloud, Google App Engine, Microsoft's Azure, etc. With the envisioned advances of cloud service systems, enterprise users can access to the resource on the cloud anytime and anywhere, without the need to manage the underlying handware/software systems.

Without any doubt, *DATA* is an extremely important asset for all organizations, especially for enterprise users. But this deserves special attention in the era of cloud computing. Different from the conventional local storage, cloud computing stores data on remotely located servers via the Internet. The data owner needs to upload his/her own data into the cloud server, and authorized users can retrieve the data from the cloud. In cloud computing, it is the semi-trusted Cloud Service Provider (CSP) that manages the cloud and all data on the cloud [31], and as a result, data confidentiality is at the top of the list of concerns on cloud computing [27]. While encryption can provide data confidentiality, classic encryption methods alone cannot meet another requirement in many applications of cloud storage, namely flexible and fine-grained data access control [32].

With the development of group-oriented applications, access control is confronted with the requirement of different and flexible access privileges. Take a multinational corporation as an example, in order to realize efficient data sharing and mobile office, the corporation can outsource all files and data to the cloud. Access control in such a situation has a hierarchical structure. As shown in Fig. 1, the access to the cloud data requires that (1) ordinary staff can only access the data of their own department, (2) department managers need to access more data than the ordinary staff, and (3) the chief executive officer (CEO) has the supreme seniority and can access all data. Apparently, it would be both time and effort costly to realize such a hierarchical data access with only classic encryption methods.

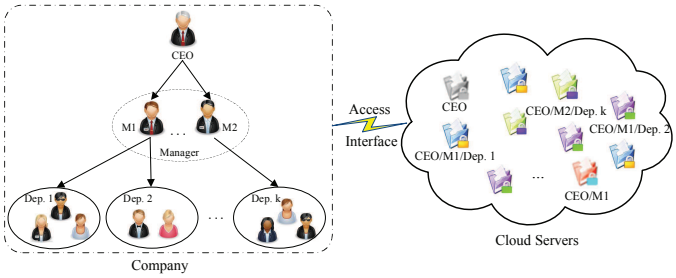

Fig. 1. Hierarchical data access in cloud computing

At the first glance, attribute-based encryption (ABE) [23]

- *S. Tang, X. Li and L. Xu are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, P.R. China, 510640. E-mail: shtang@ieee.org, leexiaoyu1987@gmail.com, csllxu@scut.edu.cn.*
- *X. Huang is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, P.R. China, 350108. E-mail: xyhuang@fjnu.edu.cn.*
- *Y. Xiang is with the School of Information Technology, Deakin University, Australia. E-mail: yang@deakin.edu.au.*

received August 3, 2014; revised July xx, 2015.

appears to be a good solution to data confidentiality in cloud computing with flexible and fine-grained data access control. ABE provides flexibility, scalability and fine-grained access control and there are many schemes available, such as those in [16], [29], [30]. However, existing ABE schemes have a high overhead because of the complexity of realizing the access structure and costly bilinear map operations. In addition, attribute revocation is a subtle issue in ABE and requires extra computation and communication costs to deal with. With this in mind, we study the issue of flexible and fine-grained data access control in cloud computing with a different primitive: Hierarchical Key Assignment.

1.1 Hierarchical Key Assignment

A hierarchical key assignment scheme (HKAS), first proposed by Akl and Taylor in [1], can be used to solve the hierarchical multi-group communication problem. From the aspect of access control, HKAS allows authorized users to have different access privileges in multi-group communication. According to different access privileges, users are organized in a hierarchy and divided into different disjoint groups called security classes (or classes for short).

Generally, HKAS is modeled as follows: a central authority (CA) assigns a piece of private information and an encryption key to each class in the hierarchy. The encryption key of each class, which is derived from the class private information and other public information, is used to protect the data of that class (e.g., the secret key in symmetric-key cryptographic algorithms). Moreover, the private information and encryption key of a class-A are sufficient to derive the encryption key of each descendant class of class-A. In other words, anyone from class-A can access the data as those from class-A's descendant classes. HKAS has been found useful in various situations, such as mobile ad-hoc networks [15], cloud computing [13], [18] and wireless sensor networks [34].

The disadvantage of the first HKAS [1] is that the size of public parameters grows rapidly as the number of classes increases. Subsequently, a lot of HKAS were proposed with various features, such as better performance [22], [25], supporting dynamic changes of classes [3], [26], [36], and more general access control policies [20], [24].

There are some designs of HKAS based on elliptic curve cryptosystems (ECC) to solve the hierarchical access control problem, such as [6], [10], [17], [21]. However, Das et al. [11] pointed out that the scheme in [10] is insecure against the exterior root finding attack, namely an outside attacker can derive the secret key by using the root finding algorithm when a new class is inserted into the user hierarchy. Lin and Hsu [21] showed that Jeng-Wang scheme [17] is vulnerable to the compromising attack, which means any adversary can derive the encryption key from the ancestor class when it is removed from the hierarchical structure.

By utilizing the one-way hash function and symmetric key cryptosystem, Chen and Huang proposed an efficient HKAS for dynamic access control in [8]. Their scheme has a better performance than Akl-Taylor scheme [1]. In practice, users may be assigned to a certain class according to the time period. Thus, time-dependent HKAS is useful. The first time-bound HKAS was proposed by Tzeng in [28]. However, his scheme is very complicated, ineffective and insecure against the collusive attack [33]. Other time-bound HKAS were proposed in [4], [13].

1.2 Motivation

A number of existing HKASs are indirect access schemes [9], [12], [15], [22], [25]. Namely, in order to obtain the encryption key of a descendant class, the ancestor needs to iteratively compute all encryption keys on the path from itself to the descendant class. There are direct access schemes on hierarchical access control which can derive the encryption key of any descendant class by only one computation, such as those in [7], [10], [21], [26]. Unfortunately, the schemes in [7], [10], [26] are insecure as shown in [11], [14], [35]. Compared with the schemes based on hash function or symmetric-key encryption/decryption, HKAS based on ECC and polynomial interpolating in [21] requires a high overhead on computation and storage. Therefore, it is meaningful and important to construct a simple, secure and effective HKAS to satisfy the requirements of direct data access in cloud computing.

1.3 Our Contribution

In this work, we propose a HKAS based on linear geometry as a solution to hierarchical access in cloud computing. The essential principle of our approach is that a private vector and a public vector are associated with the encryption key of each class. Furthermore, we adopt the orthogonality concept of vector space in linear geometry to make sure that two independent classes have nothing to do with each other. The essential ideas are:

- For every class in the system, its private vector is non-orthogonal to the corresponding public vector, and the value of their inner product is equal to the encryption key of this class;
- The private vector of an ancestor class is non-orthogonal to the public vector of its descendant classes. The value of their inner product can be considered as an indirect key between the two classes, and the ancestor class can derive the encryption key of the descendant class via this value;
- For each descendant class, its private vector is orthogonal to the public vector of ancestor class;
- If the relationship between two classes is neither descendant nor ancestor, the private vector of one class is orthogonal to the public vector of the other.

The most distinguishing feature of our proposed scheme is that it only needs to compute the values of pseudorandom function (PRF) and vector multiplication, which leads to a very light computation overhead. Additionally, the proposed scheme can achieve strong key indistinguishability security (S-KI-security) with a formal security proof.

1.4 Organization

The rest of this paper is organized as follows. In Section 2, we introduce some preliminaries used in our construction. The proposed HKAS for solving hierarchical data access in cloud computing is presented in Section 3. In

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/TC.2015.2479609, IEEE Transactions on Computers

addition to the basic key derivation, we also give solutions for dynamic key management due to the changes in the hierarchy. Section 4 presents the security analysis of the proposed scheme. Section 5 analyzes the performance of the proposed scheme and provides the simulation result. We conclude this paper in Section 6.

2 PRELIMINARIES

In this section, we first review the system description of hierarchical key assignment in cloud computing. Then, we present the existing security model of hierarchical key assignment. After that, we give the definition of pseudorandom function used in our proposed scheme.

2.1 System Description

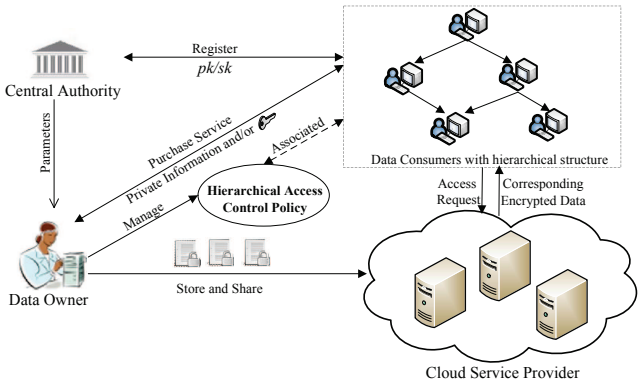

Fig. 2. System Model

As shown in Fig. 2, a hierarchical access control system in cloud computing consists of four kinds of entities: CA, data owner, data consumer and CSP. Notice that, the access control policy is managed by the data owner, not the CSP. It keeps pace with the hierarchical structure of the data consumers. The shared data which is encrypted by the data owner stores in the cloud servers with a hierarchical label ID. If a data consumer sends an access request to CSP with a hierarchical label ID , CSP finds the corresponding encrypted data for the data consumer. In our proposed scheme, we neglect the operation of shared data and only consider the hierarchical key management. We use $(V, \preceq)$ to denote the hierarchical structure of data consumers, where V is a finite set of classes and '$\preceq$' is a partial order on V. Obviously, $(V, \preceq)$ is a partially ordered set (or poset for short). The class may represent an individual data consumer or a group of data consumers in a community, and any two classes are disjoint. Therefore, we use the terminology "class" instead of data consumer in the following. The notation $V_i \preceq V_j$ means the data consumers in V_j can access the data that can be accessed by the data consumers in V_i, *i.e.*, the security clearance of V_j is higher than or equal to that of V_i. If $V_i \preceq V_j$ and there does not exist $V_k \in V$ such that $V_i \preceq V_k \preceq V_j$, then we say that V_j is an immediate predecessor of V_i, denoted by $V_i \prec V_j$ or $V_j \succ V_i$. Here, V_i is also called the immediate successor of V_j. From the perspective of graph, any poset $(V, \preceq)$ can be represented by an access graph $G = (V, E)$, where the vertices in G coincide with the classes in V and there is an edge from $V^{'}$ to $V^{''}$ if and only if $V^{'} \succ V^{''}$. Clearly, G is a directed acyclic graph.

Let Γ be a set of access graphs corresponding to partially order hierarchies. The problem that we focus on is how to assign encryption key to each class in a poset and make sure that the class can efficiently compute the encryption key of each descendant class. The method to solve this challenge is called HKAS, which is defined as follows.

Definition 1. *A hierarchical key assignment scheme for Γ is a pair of algorithms $(\mathbf{Gen}, \mathbf{Der})$ satisfying the following conditions:*

- $\mathbf{Gen}(1^\kappa, G)$ *is a probabilistic polynomial-time algorithm that takes as input a security parameter 1^κ and a graph $G = (V, E) \in \Gamma$, and outputs:*
  - *S_i: private information for each class $V_i \in V$;*
  - *k_i: encryption key for each class $V_i \in V$;*
  - *pub: public information;*
- $\mathbf{Der}(G, V_i, V_j, S_i, pub)$ *is a deterministic polynomial-time algorithm that takes as input a graph G, two classes $V_i, V_j \in V$, V_i's private information S_i and the public information pub, and outputs the encryption key k_j assigned to class V_j if $V_j \preceq V_i$, or a rejection symbol $\perp$ otherwise.*

We denote (S, k, pub) as the output of $\mathbf{Gen}(1^\kappa, G)$, where S and k can be regarded as the sets of private information and encryption key, respectively.

2.2 Security Model

Although various HKASs are proposed, many of them do not have any formal security analysis. The formal model of HKAS was first given in [3]. Atallah et al. proposed two different notions of security for HKAS: key recovery security (KR-security) and key indistinguishability security (KI-security). In this paper, we only discuss the KI-security of HKAS, since KI-security implies KR-security. As shown in [3], [12], the security models with respect to both static and dynamic adversaries are polynomially equivalent. Thus, we focus on the security model with respect to a static adversary. Given an access graph $G = (V, E)$, a static adversary $\mathcal{A}_{stat}$ firstly chooses a class $V_i \in V$ to attack. Using $\mathbf{Gen}$ algorithm on G, we define an algorithm $Corrupt$ which can provide private information S_j to the adversary. We denote $corr$ as the output of $Corrupt$. Once receiving private information S_j, the adversary can compute the encryption key k_j of class V_j. But all the private information S_j and encryption key k_j should not be able to compute the encryption key k_i. In the challenge phase, the adversary is given either the encryption key k_i or a random string of the same length, and its goal is to distinguish these two cases.

However, some information of the cryptographic key in a scheme might leak through cryptanalysis or misuse. Therefore, Freire et al. [12] proposed S-KI-security of HKAS. In their security model, the adversary can get the encryption key of class V_j, where V_j is an ancestor class of V_i. We define an oracle Key to provide this function to the adversary. Let key_{G,V_i} denote the output of Key. The formal definition of S-KI-security is given as follows:

Definition 2. *Let Γ be a set of access graphs corresponding to partially order hierarchies, and let $(\mathbf{Gen}, \mathbf{Der})$ be a HKAS for Γ. Consider the following experiment:*

Experiment $\mathrm{Exp}^{\mathrm{S-KI-ST}}_{\mathcal{A}_{\mathrm{stat}},\mathrm{G}}(1^\kappa, \mathrm{G})$
$u \leftarrow \mathcal{A}_{stat}(1^\kappa, G)$
$(S, k, pub) \leftarrow \mathbf{Gen}(1^\kappa, G)$
$corr \leftarrow Corrupt_{V_i}(S), \quad key_{G,V_i} \leftarrow Key(G, k, V_i)$
$\beta \xleftarrow{R} \{0,1\}$
if $\beta = 1$ *then* $T = k_i$, *else* $T \leftarrow \{0,1\}^{length(k_i)}$
$d \leftarrow \mathcal{A}_{stat}(1^\kappa, G, pub, corr, key_{G,V_i}, T)$
return d

The advantage of $\mathcal{A}_{stat}$ is defined as:

$$Adv^{S-KI}_{\mathcal{A}_{stat}}(1^\kappa, G) = 2|\Pr[Exp^{S-KI-ST}_{\mathcal{A}_{stat},G}(1^\kappa, G) = \beta] - 1/2|.$$

The scheme is said to be secure in the sense of strong key indistinguishability with respect to each static adversary, if $Adv^{S-KI}_{\mathcal{A}_{stat}}(1^\kappa, G)$ is negligible for each graph $G \in \Gamma$ and each class $V_i \in G$.

Similar to group key assignment schemes, HKAS must be resistant to collusive attack. It means that any two or more lower classes, by colluding with each other, cannot derive the encryption key of higher class. Sometimes, HKAS should guarantee the forward security and backward security upon the needs of practical application.

- Forward security: the user U cannot access the future communication data of the class V_i when U leaves the class V_i [19].
- Backward security: the user U cannot access the previous communication data of the class V_j when U is added to the class V_j [19].

2.3 Pseudorandom Function

On the construction of HKAS, pseudorandom function is a very useful tool. Roughly speaking, pseudorandom function is a family of functions, which can be efficiently computed given a short and random key. Meanwhile, the input/output behavior of the function is computationally indistinguishable from that of a random function [5].

Definition 3. *Let $F : K \times D \to RG$ be a family of functions, where K is the set of keys of F, D and RG are the domain and range of F, respectively. For $k \in K$, $F_k(x) = F(k, x)$ represents an instance of F. Let $Rand = \{g | g : D \to RG\}$ denote a set of all functions from D to RG. Assuming that $\mathcal{A}_F$ is an adversary with polynomial time t that can access to an oracle $\mathcal{D}$ for a function $g : D \to RG$, where g can be chosen randomly from $Rand$ or F. Consider the following two experiments:*

Experiment $\mathrm{Exp}^{\mathrm{PRF-1}}_{\mathcal{A}_\mathrm{F}}(1^\kappa, \mathrm{t})$
$k \xleftarrow{R} K$
$d \leftarrow \mathcal{D}^{F_k}_{\mathcal{A}_F}(1^\kappa, t)$
return d

Experiment $\mathrm{Exp}^{\mathrm{PRF-0}}_{\mathcal{A}_\mathrm{F}}(1^\kappa, \mathrm{t})$
$g \xleftarrow{R} Rand$
$d \leftarrow \mathcal{D}^{g}_{\mathcal{A}_F}(1^\kappa, t)$
return d

The advantage of $\mathcal{A}_F$ is defined as: $Adv^{PRF}_{\mathcal{A}_F}(1^\kappa, t) = |\Pr[Exp^{PRF-1}_{\mathcal{A}_F}(1^\kappa, t) = 1] - \Pr[Exp^{PRF-0}_{\mathcal{A}_F}(1^\kappa, t) = 1]|$.
F is pseudorandom if $Adv^{PRF}_{\mathcal{A}_F}(1^\kappa, t)$ is negligible for every efficient adversary $\mathcal{A}_F$.

In our proposed scheme, we will use a special PRF where $K = D = RG = F_q$ and F_q is a finite field.

3 The Proposed Scheme

In this section, we propose a HKAS based on linear geometry. The novelty of our scheme is that we use a specially designed matrix to describe the hierarchy of classes. With this approach, any ancestor class can directly compute the encryption keys of its descendant classes. In our proposed scheme, data owner firstly interacts with CA, and obtains system parameters. Then, the data owner generates a finite field F_q and a PRF F. All computations in our scheme are over F_q. The public information of our proposed scheme consists of a matrix $A = [A_1^T, \cdots, A_n^T]$ and a random element on F_q. Besides the encryption key of class V_i, the data owner also generates two vectors Y_i and Z_i as the private information of class V_i.

Definition 4. *Let V_i, $V_j \in V (i \neq j)$, the indirect key $k_{i,j}$ between V_i and V_j is defined by*

$$k_{i,j} = \begin{cases} (k_{j,j} - w_{i,1}k_{i,i})w_{i,2}^{-1}, \ if\ V_j \preceq V_i; \\ 0, \qquad\qquad otherwise; \end{cases} \tag{1}$$

where $k_{i,i}$ and $k_{j,j}$ are the encryption key of V_i and V_j respectively, $w_{i,1}$ and $w_{i,2}$, which is defined in Section 3.1, can be computed by V_i from the private information Y_i.

Before presenting our scheme, we introduce the following rules. These rules utilize vector orthonormality of linear geometry between the public vectors and the private vectors to process the access rights of any two classes in our scheme.
Rule 1. For each V_i, the inner product between the private vector X_i and its public vector A_i is equal to the encryption class of V_i, *i.e.,*

$$X_i \times A_i^T = k_{i,i}, \ for\ i = 1, \cdots, n. \tag{2}$$

Rule 2. If V_i is an ancestor class of V_j, then the inner product between the private vector X_i of V_i and the public vector A_j of V_j is equal to the indirect encryption class $k_{i,j}$, *i.e.,*

$$X_i \times A_j^T = k_{i,j}, \ if\ V_j \preceq V_i. \tag{3}$$

Rule 3. If V_i is not an ancestor class of V_j, then the private vector X_i of V_i is orthogonal to the public vector A_j of V_j, *i.e.,*

$$X_i \times A_j^T = 0, \ if\ V_j \npreceq V_i. \tag{4}$$

Let Γ be the set of access graphs and n be the number of classes, the details of our scheme are given as follows.

3.1 Key Generation

Let $G = (V, E) \in \Gamma$ be an access graph, where $V = \{V_1, \cdots, V_n\}$. The data owner executes the process of key generation for G by the following steps.
Step 1. The data owner randomly chooses non-zero vectors $Y_i = (y_{i,1}, y_{i,2})$ and $Z_i = (z_{i,1}, z_{i,2})$ for class V_i as its private information.
Step 2. The data owner maps all Y_i to private vectors W_i by applying the PRF F as follows: the data owner randomly chooses a map parameter $r \in F_q$, then computes $w_{i,1} = F_{y_{i,1}}(r)$, $w_{i,2} = F_{y_{i,2}}(r)$, $for\ i = 1, \cdots, n$. If $w_{i,2} = 0$, the data owner repeats this step by re-selecting a new r and recomputing $w_{i,1}$ and $w_{i,2}$. At last, the data owner maps each Z_i to a n-dimensional vector X_i. For $i = 1, 2$, computes $x_{i,1} = z_{i,1}$, $x_{i,2} = z_{i,2}$, and lets $x_{i,3} = \cdots = x_{i,n} = 0$. For

$i = 3, \cdots, n$, the data owner computes $x_{i,1} = z_{i,1}$, $x_{i,i} = z_{i,2}$, and lets $x_{i,j} = 0$, where $j \neq 1, i$.

After this step, the data owner can get a set of n-dimensional vectors: $X_1 = (x_{1,1}, x_{1,2}, 0, \cdots, 0)$, $X_2 = (x_{2,1}, x_{2,2}, 0, \cdots, 0)$, $\cdots$, $X_n = (x_{n,1}, 0, \cdots, 0, x_{n,n})$. Denote the matrix X by

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & 0 & 0 & 0 \\ x_{2,1} & x_{2,2} & 0 & 0 & 0 \\ x_{3,1} & 0 & x_{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & 0 & 0 & \cdots & x_{n,n} \end{bmatrix}$$

Step 3. The data owner tests whether $X_1, X_2, \cdots, X_n$ are linearly independent or not. If they are linearly independent, go to **Step 4**. Otherwise, re-selects $Z_1, Z_2, \cdots, Z_n$ until $X_1, X_2, \cdots, X_n$ are linearly independent.

Since the matrix X is sparse as a triangle matrix, the data owner can easily compute that $|X| = (x_{1,1}x_{2,2} - x_{1,2}x_{2,1})x_{3,3,} \cdots x_{n,n}$. As long as $x_{1,1}x_{2,2} \neq x_{1,2}x_{2,1}$ and $x_{j,j} \neq 0$, we can conclude that $|X| \neq 0$. Therefore, it is very easy to get a set of linearly independent vectors $X_1, \cdots, X_n$.

Step 4. The data owner chooses encryption key for each class and determines the public matrix A.

- **Step 4.1.** For each class $V_i \in G$, the data owner randomly chooses a non-zero element from F_q as its encryption key $k_{i,i}$. Then, the data owner computes the indirect key $k_{i,j}$ by Eq.(1).
- **Step 4.2.** The data owner can construct a system of equations about the public matrix A.

By combining **Definition 4 and Rules 1, 2 and 3, the data owner** can obtain $X_i \times A_j^T = k_{i,j},\ for\ i = 1, \cdots, n,\ and\ j = 1, \cdots, n$. Denote $K_j = (k_{j,1}, k_{j,2}, \cdots, k_{j,n})$ and $K = [K_1, K_2, \cdots, K_n]^T$, then the above equations can be rewritten as

$$X \times A = K. \tag{5}$$

- **Step 4.3.** Solve the Eq.(5), and obtain the solution $A = X^{-1} \times K$.

Step 5. The data owner sends Y_i, Z_i and $k_{i,i}$ to the corresponding class V_i via a secure channel, and transmits the public matrix A and the selected mapping parameter r to the CSP via open channels.

In fact, each class can determine whether the received encryption key is correct after receiving its private information X_i and interacting with the CSP. That is because the encryption key of class V_i is computed by $k_{i,i} = X_i \times A_j^T$.

3.2 Key Derivation

If a class V_i wants to access the encrypted data held by class V_j satisfying $V_j \preceq V_i$, then V_i can derive the encryption key of V_j as follows:

Step 1. V_i computes the indirect key $k_{i,j}$ by **Rule 2**.

Step 2. Through the equation $k_{j,j} = w_{i,1}k_{i,i} + w_{i,2}k_{i,j}$, V_i derives V_j's encryption key $k_{j,j}$.

According to **Rule 3**, we know that $k_{i,j} = 0$ if V_i is a descendant class of V_j, or there is no descendant or ancestor relationship between V_i and V_j. Therefore, in this case the encryption key of V_j could not be derived by V_i.

3.3 Dynamic Key Management

In this section, we discuss the issue of dynamic key management, such as inserting a new class, removing an existing class and adding/deleting relationships in the hierarchy.

Inserting a new class. Assume that there exists a relationship $V_i \prec V_j$ between the class V_i and V_j. A new class, denoted by V_k, is inserted into the hierarchy such that $V_i \prec V_k \prec V_j$.

First, the data owner randomly chooses two new non-zero vectors $Y_k = (y_{k,1}, y_{k,2})$, $Z_k = (z_{k,1}, z_{k,2})$ and k_k for class V_k, and transmits these to class V_k via a secure channel. Then, the data owner chooses new encryption keys for all descendant classes of V_k, and computes the new public matrix A as described in **Step 3 and 4**, **Section 3.1**. After that, each class calculates its own encryption key and encryption keys of all descendant classes by its private information.

Note that the above procedure fully involves V_i and all descendant classes of V_i. Each of them is assigned a new encryption key to guarantee the backward security. Encryption keys of other classes remain unchanged, although the dimensions of X_i, A and K have increased.

Removing an existing class. Suppose there are $n+1$ classes in a system with a hierarchical structure. A class, denoted by V_l, needs to be removed from this system. The data owner needs to update the encryption keys of all V_l's descendant classes to guarantee the forward security. The process is as follows. The data owner first re-selects the encryption keys of all descendant classes of V_l, then computes a new public matrix A as described in **Section 3.1**. Once finding that the public matrix A is changed, all descendant classes of V_l derive their new encryption keys from the new matrix A. These new encryption keys can guarantee the forward security. As in the original scheme, each class can derive the encryption key of its descendant class from new public parameters.

Adding/deleting relationships. Let us suppose that there is no relationship between V_i and V_j in a hierarchy. Now a new relationship between V_i and V_j (such as $V_i \prec V_j$) is added into the hierarchy. In this case, the data owner first updates the encryption keys of V_i and its descendant classes to guarantee the backward security. Meanwhile, the data owner also updates the indirect keys of V_j and its ancestor classes. Then the data owner computes a new public matrix A and distributes it among all classes in the hierarchy. After that, V_i and its descendant classes can derive their new encryption keys from the new public matrix A.

Assume that the relationship between V_i and V_j is $V_i \preceq V_j$ and there is no overlap between their descendant classes. In this case, the data owner needs to update the encryption keys of V_i and its descendant classes to guarantee the forward security when deleting the relationship of $V_i \preceq V_j$. The detail of this implementation is similar to that of removing an existing class in the hierarchy.

A note in our scheme is that communications are not authenticated, since we focus on hierarchical key assignment. This drawback exists in almost all HKASs, though a simple method to achieve communication authentication is using digital signatures.

4 Security Analysis and Discussions

In this section, we will discuss the security of our proposed scheme.

We first show the security of our scheme can reach S-KI-secrutiy under the assumption of pseudorandom function. We provide the concrete proof, which consists of a theorem and a lemma in the online supplemental material.

As shown in our scheme, the private information is randomly chosen by the data owner and independent with the encryption key. Therefore, even the ancestor class cannot derive the private information of its descendant class.

The potential attack from the descendant classes is the collusive attack. These collusive classes wish to obtain the encryption key of their ancestor class through the public information and their own private information. Suppose V_m is a class at a higher level of the system and $V_{m_1}, \cdots, V_{m_c}$ are V_m's descendant classes, where c is a positive integer. From the scheme, class V_{m_i} can get the following two equations via (1) and (2) when attacking V_m

$$k_{m_i,m_i} = k_{m,m} w_{m,1} + (x_{m,1} a_{m_i,1} + x_{m,m} a_{m_i,m}) w_{A,2} \quad (6)$$

$$k_{m,m} = x_{m,1} a_{m,1} + x_{m,m} a_{m,m} \quad (7)$$

Since $V_{m_1}, \cdots, V_{m_c}$ are colluding with each other, they can obtain a system of equations as follows:

$$\begin{cases} k_{m,m} w_{m,1} + (x_{m,1} a_{m_1,1} + x_{m,m} a_{m_1,m}) w_{m,2} = k_{m_1,m_1} \\ \vdots \\ k_{m,m} w_{m,1} + (x_{m,1} a_{m_c,1} + x_{m,m} a_{m_c,m}) w_{m,2} = k_{m_c,m_c} \end{cases}$$

where $k_{m,m}$, $x_{m,1}$, $x_{m,m}$, $w_{m,1}$ and $w_{m,2}$ are unknown to $V_{m_1}, \cdots, V_{m_c}$.

Let $u_1 = k_{m,m} w_{m,1}$, $u_2 = x_{m,1} w_{m,2}$, and $u_3 = x_{m,m} w_{m,2}$. Then, the above system of equations becomes a system of non-homogeneous linear equations. If $c \geq 3$, then u_1, u_2 and u_3 can be solved. But for the following system of equations

$$\begin{cases} u_1 = k_{m,m} w_{m,1} \\ u_2 = x_{m,1} w_{m,2} \\ u_3 = x_{m,m} w_{m,2} \\ k_{m,m} = x_{m,1} a_{m,1} + x_{m,m} a_{m,m} \end{cases},$$

$V_{m_1}, \cdots, V_{m_c}$ cannot derive a unique solution for $k_{m,m}$. Hence, our scheme is secure against collusive attacks.

5 Performance Analysis and Simulations

This section shows the performance of our scheme by theoretical analysis and simulation results.

5.1 Performance Analysis

As many other cryptography primitives, the efficiency of a HKAS is evaluated by storage, computation and rekeying overhead. Suppose that the size of an element in the finite field F_q is L bits, the number of classes in the system is n, and the average number of descendants of each class is c.

Storage Overhead. In our scheme, each class needs to store its private information: two-dimensional vectors Y_i and Z_i. Hence, the storage cost of each class is $4L$. The data owner must store the private data of all security classes $Y_1, \cdots, Y_n$ and $Z_1, \cdots, Z_n$, which leads to a storage cost of $4nL$.

Computation Overhead. Let H denote the computational cost of PRF F. The notation A and M denote the computational cost of addition and multiplication over F_q, respectively. The average computational cost of inversion over F_q is denoted by R. In order to obtain its encryption key, each class computes 2 multiplications and 1 additions over F_q. When deriving one of its descendant classes, each class needs to compute the value of F with 2 times. Moreover, the other computation of this class is 4 multiplications and 2 additions over F_q. Therefore, the total computation overhead of each class is $2H + (4c+2)M + (2c+1)A$.

For the data owner, the computation includes three parts: 1) calculate the value of PRF F in **Step 2**, which requires $2n$ times of F computation; 2) calculate all $k_{i,j}$ in **Step 4.1**, which requires cn additions, $2cn$ multiplications and n inversions over F_q at average; 3) solve public matrix A in **Step 4.3**, which requires no more than $(3n^2 + 2n + 2)$ additions, $(3n^2 + 2n + 6)$ multiplications and n inversions over F_q. Hence, the amount of computation overhead by the data owner is $2nH + ((3n^2 + (2c+2)n + 6)M + ((3n^2 + (c+2)n + 2))A + 2nR$.

Rekeying Overhead. When there is a dynamical change in the system, the data owner needs to compute new public parameters and publish the new public matrix A. The size of rekey messages transmitted by the data owner is $n^{'2}L$ bits, where $n^{'}$ is the number of classes after the dynamical change.

From the construction of our scheme, the relationship between the size of public matrix A and the number of classes is square. This may be a shortcoming of our scheme, especially on the rekeying. We compare our scheme with some other typical HKASs. Table 1 shows the comparison results. In this table, the private information is measured per class in the access graph $G = (V, E)$, and the public parameter is measured for the whole system.

5.2 Implementation and Experimental Results

We evaluate the performance of our proposed scheme by the public cryptographic library MIRACL and C/C++ programming language. In our experiment, we set $|L| = 160$ and use a cryptographic hash function to substitute the pseudorandom function. Concretely, we use $F_k(x) = SHA\text{-}1(x)\ XOR\ x$, for $\forall x \in F_q$. We take the average time by repeating the experiment 10 times and do not take into account the time for constructing the access hierarchy. Since $|L| = 160$, the size of the public parameter is $20 \times n^2$ bytes, and the size of private data for each class is 80 bytes.

Fig. 3 shows the average time for the data owner in key generation. The platform for the data owner is a DELL T5610 workstation equipped with 2.5GHz Intel Quad-Core Processer and 8GB Memory, and running Windows 7 Professional 64-bit operating system. In this figure, the parameter c denotes the average number of descendants of each class. But it does not mean that each class in the hierarchical structure has c descendant classes. We can see that it takes the data owner more time when n and c increase.

The platform for the CSP is the same as the one of the data owner in our experiment. It stores the public information and encrypted data, and responds to each class's access requests.

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/TC.2015.2479609, IEEE Transactions on Computers

TABLE 1
Comparison with Other Schemes

Scheme	Private Information	Public Information	Key derivation	Dynamic	Type of Security	Security Assumption
Atallah et al. [3]	L	$L\|E\|+\rho\|V\|$	$(C_{SE-DEC}+C_{PRF})h$	Yes	KI	CPA-Secure+PRF
Lin et al. [21]	L	$(3\|V\|+\sum_{i=2}^{\|V\|}k_i+4)L$	$C_H+C_F+C_{ECC-DEC}$	Yes	N/A	CPA-Secure+OWHF
A. D. Santis et al. [25]	L	$(\|E\|+2\|V\|)\rho$	$(h+2)C_{SE-DEC}$	Yes	KI	CPA-Secure
E. Freire et al. [12]	L	$2L$	$(h+1)C_{PRF}$	No	S-KI	PRF
Our Scheme	$4L$	$(\|V\|^2+1)L$	$4M+2A$	Yes	S-KI	PRF

Notation in Table 1 :
$|E|$: number of edges in the access graph G;
$|V|$: number of classes in the access graph G;
h: path length between class V_i and V_j when class V_i wants to derive the encryption key of class V_j;
ρ: size of ciphertext in a symmetric key encryption scheme;
C_{SE-DES}: decryption time of a symmetric key encryption scheme;
C_{PRF}: time of calculating the PRF;
$C_{ECC-DES}$: decryption time of an Elliptic Curve based public key encryption scheme;

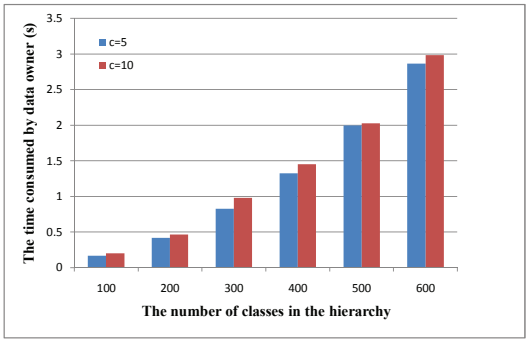

Fig. 3. Time Consumed by Data Owner in Key Generation

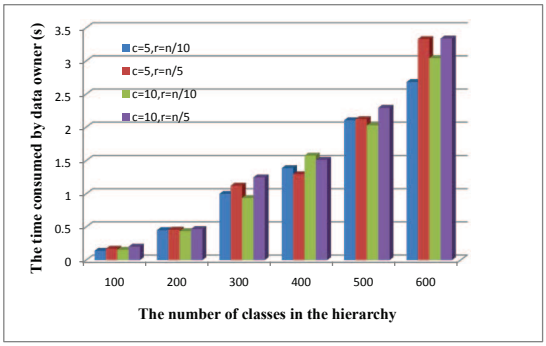

Fig. 5. Time Consumed by Data Owner in Dynamic Management

In our experiment, the platform for each class is a HP XW4600 workstation with 2.33GHz Intel Dual-Core Processer and 2GB Memory, and running Windows 7 Professional 64-bit operating system. The average time for obtaining the encryption key of any descendant class is 0.0019 ms which is very small. Assume that a class has already known all its descendant classes. If $c = 5$, the average time for obtaining the encryption key of all descendant classes is 0.0508ms. In the case $c = 10$, the above time is 0.0689ms. Fig. 4 shows the average time of each class in deriving encryption key of all descendant classes by testing that the indirect key is equal to zero or not for all classes in the hierarchy.

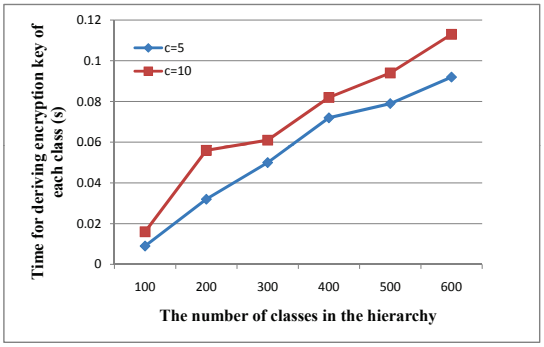

Fig. 4. Deriving the Encryption Key of Each Class

In the simulation of dynamic key management, we adopt a random value r to represent the total number of changed classes to keep the experiment simple. The number of newly joined classes and deleted classes are randomly selected in $[0, r/2]$, respectively. Let r minus the sum of these two values be the number of classes whose relationships are changed. Fig. 5 shows the average time consumed by the data owner for updating the public parameter. Comparing Fig 3 with Fig 5, we find that the time consumed by the data owner in Fig 5 is slightly more than that in Fig. 3. As described in our scheme, the major time cost for the data owner is re-computing the encryption key and indirect key in K when a class dynamically changes its access structure. Then, we need to traverse a graph whose root is the changed class and find the ancestor class of this changed class.

6 Conclusion

We presented a new design of hierarchical key assignment based on linear geometry. In our design, a private vector and a public vector are associated with the encryption key of each class in the hierarchy. Our scheme provides hierarchical access because class encryption key can be derived from the inner product of the public vector of that class and the private vector of its ancestor class. We also provided efficient key management solutions to address potential changes in the hierarchy. A formal proof shows

that the proposed scheme has the property of strong key indistinguishability under the assumption of pseudorandom functions. Simulations show that our scheme provides an optimized trade-off between computation consumption and storage space, though the size of the public parameter in our scheme is slightly larger than others. Since the computing method is very simple, we believe the proposed scheme can serve as an efficient solution of flexible and fine-grained access control in cloud computing.

ACKNOWLEDGMENTS

This work was supported by the 973 Program (No. 2014CB360501), the National Natural Science Foundation of China (No. U1135004 and 61170080), Guangdong Provincial Natural Science Foundation (No. 2014A030308006), and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011). X. Huang's work was partially supported by the NSFC (No. U1405255 and 61472083), and L. Xu's work was partially supported by the NSFC (No. 61202466).

REFERENCES

[1] S. G. Akl and P. D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, 1983.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[3] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Transactions on Information and System Security*, 12(3), 2009.

[4] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci. Provably-secure time-bound hierarchical key assignment schemes. *Journal of Cryptology*, 25(2):243–270, 2012.

[5] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *Proceedings of 37th Annual Symposium on Foundations of Computer Science(FOCS'1996)*, pages 514–523. IEEE, 1996.

[6] E. Bertino, N. Shang, and S. S. Wagstaff. An efficient time-bound hierarchical key management scheme for secure broadcasting. *IEEE Transactions on Dependable and Secure Computing*, 5(2):65–70, 2008.

[7] T.-S. Chen and Y.-F. Chung. Hierarchical access control based on Chinese remainder theorem and symmetric algorithm. *Computers & Security*, 21(6):565–570, 2002.

[8] T.-S. Chen and J.-Y. Huang. A novel key management scheme for dynamic access control in a user hierarchy. *Applied Mathematics and Computation*, 162(1):339–351, 2005.

[9] Y.-R. Chen, C.-K. Chu, W.-G. Tzeng, and J. Zhou. CloudHKA: A cryptographic approach for hierarchical access control in cloud computing. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security(ACNS'2013)*, volume 7954 of *Lecture Notes in Computer Science*, pages 37–52. Springer Heidelberg, 2013.

[10] Y. F. Chung, H. H. Lee, F. Lai, and T. S. Chen. Access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences*, 178(1):230–243, 2008.

[11] A. K. Das, N. R. Paul, and L. Tripathy. Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. *Information Sciences*, 209:80–92, 2012.

[12] E. Freire, K. Paterson, and B. Poettering. Simple, efficient and strongly KI-Secure hierarchical key assignment schemes. In *Proceedings of the Cryptographers' Track at the RSA Conference 2013(CT-RSA'2013)*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer Heidelberg, 2013.

[13] B.-Z. He, C.-M. Chen, T.-Y. Wu, and H.-M. Sun. An efficient solution for hierarchical access control problem in cloud environment. *Mathematical Problems in Engineering*, 2014, Article ID 569397:8 pages, 2014.

[14] C.-L. Hsu and T.-S. Wu. Cryptanalyses and improvements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy. *Computers & Security*, 22(5):453–456, 2003.

[15] D. Huang and D. Medhi. A secure group key management scheme for hierarchical mobile ad hoc networks. *Ad Hoc Networks*, 6(4):560–577, 2008.

[16] J.-Y. Huang, C.-K. Chiang, and I.-E. Liao. An efficient attribute-based encryption and access control scheme for cloud storage environment. In *Proceedings of 8th International Conference on Grid and Pervasive Computing(GPC'2013)*, volume 7861 of *Lecture Notes in Computer Science*, pages 453–463. Springer Heidelberg, 2013.

[17] F.-G. Jeng and C.-M. Wang. An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem. *Journal of Systems and Software*, 79(8):1161–1167, 2006.

[18] Y.-W. Kao, K.-Y. Huang, H.-Z. Gu, and S.-M. Yuan. uCloud: a user-centric key management scheme for cloud data protection. *IET Information Security*, 7(2):144–154, 2013.

[19] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proceedings of the 7th ACM Conference on Computer and Communications Security(CCS'2000)*, pages 235–244. ACM, 2000.

[20] I.-C. Lin, M.-S. Hwang, and C.-C. Chang. A new key assignment scheme for enforcing complicated access control policies in hierarchy. *Future Generation Computer Systems*, 19(4):457–462, 2003.

[21] Y.-L. Lin and C.-L. Hsu. Secure key management scheme for dynamic hierarchical access control based on ECC. *Journal of Systems and Software*, 84(4):679–685, 2011.

[22] H. Ragab Hassen, H. Bettahar, A. Bouadbdallah, and Y. Challal. An efficient key management scheme for content access control for linear hierarchies. *Computer Networks*, 56(8):2107–2118, 2012.

[23] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology-EUROCRYPT'2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer Heidelberg, 2005.

[24] A. D. Santis, A. L. Ferrara, and B. Masucci. Cryptographic key assignment schemes for any access control policy. *Information Processing Letters*, 92(4):199–205, 2004.

[25] A. D. Santis, A. L. Ferrara, and B. Masucci. Efficient provably-secure hierarchical key assignment schemes. *Theoretical Computer Science*, 412(41):5684–5699, 2011.

[26] V. R. Shen and T.-S. Chen. A novel key management scheme based on discrete logarithms and polynomial interpolations. *Computers & Security*, 21(2):164–171, 2002.

[27] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1 – 11, 2011.

[28] W.-G. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):182–188, 2002.

[29] 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*, 7(2):743–754, 2012.

[30] G. Wang, Q. Liu, and J. Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th ACM Conference on Computer and Communications Security(CCS'2010)*, pages 735–737. ACM, 2010.

[31] K. Yang and X. Jia. Data storage auditing service in cloud computing: challenges, methods and opportunities. *World Wide Web*, 15(4):409–428, 2012.

[32] K. Yang and X. Jia. Expressive, efficient and revocable data access control for multi-authority cloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1735–1744, 2014.

[33] X. Yi and Y. Ye. Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1054–1055, 2003.

[34] Y. Zhang, X. Li, J. Liu, J. Yang, and B. Cui. A secure hierarchical key management scheme in wireless sensor network. *International Journal of Distributed Sensor Networks*, 2012, Article ID 547471:8 pages, 2012.

[35] S. Zhong and T. Lin. A comment on the Chen-Chung scheme for hierarchical access control. *Computers & Security*, 22(5):450–452, 2003.

[36] X. Zou, Y. Karandikar, and E. Bertino. A dynamic key management solution to access hierarchy. *International Journal of Network Management*, 17(6):437–450, 2007.