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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

PHE: An Efficient Traitor Tracing and Revocation for Encrypted File Syncing-and-Sharing in Cloud

Yan Zhu*, Guohua Gan, Ruiqi Guo, and Dijiang Huang

Abstract—Recently, many more enterprises have moved their data into the cloud by using file syncing and sharing (FSS) service, but bring-your-own-device (BYOD) policies and greatly increasing mobile devices have in fact raised a new challenge for preventing the player/decoder abuse in the FSS service. In this paper, we address this issue using a new system model with anomaly detection, tracing and revoking traitors. To implement this model, we present a new threshold cryptosystem, called Partially-ordered Hierarchical Encryption (PHE), which implements the partial-order key hierarchy, similar to role hierarchy in Hierarchical RBAC, in public-key infrastructure. This cryptosystem provides two security mechanisms, traitor tracing and revocation, to support efficient digital forensics. The security and performance analysis shows that our construction is threshold provably secure and has following features: dynamic joining and revoking users, constant-size ciphertexts and decryption keys, lower overloads for large-scale systems.

Index Terms—Security, Cloud Storage, File Syncing-and-Sharing, Partial Order Key Hierarchy, Traitor Tracing, Revocation.

1 Introduction

In recent years, many cloud storage services, such as Box, Dropbox, MediaFire, SkyDrive, SugarSync, have been available to small-to-medium business, and individual. These cloud-based storage could be particularly attractive for consumers by providing on demand capacity, low-cost service, and long-term archive. Furthermore, cloud services have brought great convenience to people's lives because consumers can access applications and data from the cloud anywhere in the world and via any available device, such as personal computers, tables, and mobile phones. Therefore, many more enterprises and individuals have moved their data, such as personal data and large archive system, into the cloud everyday. The cloud has become a necessity to many of us for individual, enterprise, and government use.

Bring-your-own-device (BYOD) policies and an increasingly mobile devices are changing the requirements for how users want (and need) to access corporate data. The cloud storage service is generally initiated by individual users who store data and download it to sync and collaborate while working on projects. Therefore, more and more cloud-based storage platforms provide **file syncing and sharing (FSS)** services. These two services introduce new features for enterprise file sharing solution for online collaboration and storage:

File sharing: it allows the users to not only access files anywhere, anytime and from a variety of endpoint devices, but also collaboratively edit file together;

File syncing: it is a new online backup mechanism for syncing data across multiple devices, such as a home computer, tablet or smartphone, as well as collaboration and working with teams.

In Fig. 1, we describe a simple framework of FSS service, which consists of end users, player/decoder, FSS over cloud. In this framework, the player/reader/editor is also called **Decoder** that provides easy-to-use interface for accessing the data on cloud. The FSS services must provide the above features in a seamless manner so syncing and sharing happens transparently to the user. In this paper we also work on such a corporate FSS service with online collaboration.

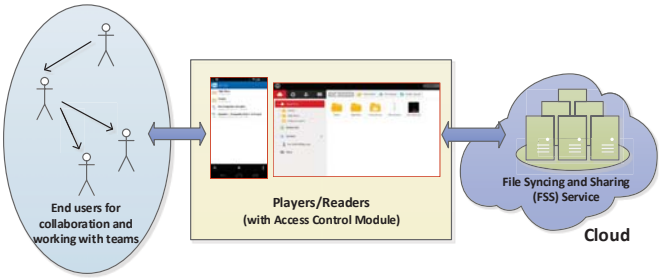


Fig. 1. File syncing and sharing services based on cloud.

Motivation. Security is a problem that must be considered for deploying a file syncing-and-sharing service. Several recent surveys [1] show that 88% potential cloud consumers worry about the privacy of their data, and security is often cited as the top obstacle for cloud adoption. At first, the multi-tenant nature of the cloud is vulnerable to data leaks, threats, and malicious attacks. Therefore, it is important for enterprises to have strong access control policies (such as Role-based Access Control (RBAC) or Attribute-based Access Control (ABAC)) in place to maintain the privacy and confidentiality of data for collaboration with teams. Sometimes cloud providers have access to the data stored in the cloud, and can control access to it by outside entities. When this is the case, the challenge is to maintain the confidentiality of data and limiting privileged user access to it. This can be achieved by encrypting the data before storing it in the cloud, and enforcing legal agreements and contractual obligations with the cloud service provider to ensure protection of data.

On the other hand, the protection of decoders is also becoming increasingly important due to huge potential commercial value of data stored in cloud. Compared with the traditional cryptographic technique, the encryption system used by the decoders should be a group-oriented cryptosystem that supports a large number of users. Considering openness of cloud envi-

- *Y. Zhu, G. Gan, and R. Guo are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China, 100083 E-mail: zhuyan@ustb.edu.cn*
- *Dijiang Huang is with the Arizona State University, Tempe, Arizona, 85287. E-mail: dijiang.huang@asu.cn*

**corresponding author*

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

ronment and complexity of management for a large amount of users, it may also give rise to some new security risks in this group-oriented cryptosystem. For example, the redistribution of traitor's decoder, the forgery of decoder with a colluded decryption key, and the privilege attacks with changing and forging the role/identity key. Therefore, it is very necessary for the FSS service to provide the functionality of digital forensics.

As with more traditional computer forensics investigations, any form of encryption places a large burden on the forensics investigation [2] and increases the complexity of the investigation. Related to encrypted data, a forensic analysis mechanism [3] should be proposed within cloud environment to guide investigations, which is flexible enough to be able to work with future providers offering new services. There have been some work [4], [5], [6] focused on this proposal from two aspects of traitor tracing and revocation based on key fingerprint in suspected decryption box. In the Fig. 2, the tracing algorithm can still identify at least one particular key (whose holder is called traitor) used for building the suspected decryption device. and then the revocation mechanism might ensure the normal running by revoking the right of the seized traitors.

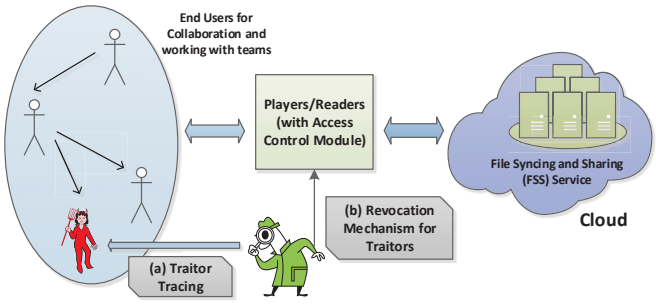


Fig. 2. The FSS service with digital forensics

Unfortunately, there exist some obstacles for migrating the data from enterprises, especially for an amount of existing RBAC systems, into the cloud. One of these obstacles is the security of migrated resources. In order to solve this issue, attribute-based encryption (ABE) [7], [8], [9], [10], [11] has been proposed in the recent years. In an ABE system, an access policy on Boolean function with AND/OR gates and equivalent matching (=) was realized over encrypted data. The decryption can then be done if the policy associated to the ciphertext is satisfied by the attribute set associated to the key. Although ABE is a powerful tool which meets a variety of applications, the current ABE schemes cannot fulfill the requirements for the existing RBAC systems owing to lack of support for partial ordering relations, as well as traitor tracing and revocation mechanism for traitors (rather than revocation for attributes). It is well-known that RBAC is an industry recognized and widely adopted access control model. In this model, role hierarchy (RH) is an important notion, which reflects organization's lines of authority and responsibility. Mathematically, role hierarchies are partial orders. Unfortunately, this kind of partial ordering relation still cannot be implemented in the existing ABEs. Therefore, it is necessary to develop a new RBAC-compatible encryption scheme to support the secure migration from RBAC systems into the cloud.

To construct a cryptosystem compatible with RBAC model, several schemes for hierarchical key management (HKM) have been designed [12], [13]. These existing schemes have following common features:

- Each user has a secret-key sk_i corresponding to a role c_i in RH, where $\{c_1, \cdots, c_n\}$ is a set of roles;
- There exists an efficient way to derive a descendant's key sk_j from the own key sk_i in accordance with the partial order relation $c_j \preceq c_i$ in RH; and
- Key derivation can be implemented under the precondition of the existence of an one-way function.

Existing schemes can effectively derive the keys with the help of partial order structure. However, such kind of derivation process has following problems:

- A role may be assigned to multiple users who share the same secret-key. That means there is no way to distinguish those assigned users; and
- The secret-key derivation is not able to support functions, such as user revocation and traitor tracing, in terms of digital forensics.

To address these problems, it is necessary to design a construction for hierarchical cryptosystems, considering the new features provided by some recently proposed cryptography technologies, such as IBE[7], HIBE[14] and ABE. In such a construction, a user secret-key must be unique and is accompanied by the user identity. In addition, the derivation of secret-key in such a construction should be avoided.

One compelling advantage of our key structure is that it could be seamlessly integrated into the existing RBAC systems. Consequently, an RBAC system can directly use the public role-key to encrypt resources in terms of users' assigned roles, and then the users owned the senior roles can use their privacy-keys to decrypt the encrypted resources. This kind of cryptosystem can be used for secure migrating the resources from existing RBAC systems to cloud. Other potential applications of our solution include email encryption system (EES), privacy preservation for peer-to-peer (P2P) data sharing, and encrypted file system (EFS).

Related Work. For a large-scale group-oriented communication, broadcast encryption was first considered [15] in 1991 and, subsequently, formally defined by Fiat and Naor [16] in 1994. Since then, it has become one attractive topic in cryptography community. In symmetric-key setting, only trusted system designer can broadcast data to the receivers. However, the public-key scheme, first introduced by Boneh *et al.* in 1999 [4], can publish a short public key, which enables anybody to broadcast data, thus overcome the deficiency of symmetric-key setting. Also, Boneh *et al.* have done massive work in the development of group-oriented encryption, e.g., Boneh, Sahai, and Waters [5] propose a fully collusion resistant traitor tracing with ciphertexts of size $O(\sqrt{n})$ and private keys of size $O(1)$ in 2006, where n is the total number of users. However, these work did not take into account the hierarchy structure.

Boneh and Franklin proposed the first fully identity-based encryption (IBE) [7] in 2001, in which the public key can be an arbitrary string such as an email address. Unfortunately, IBE does not support broadcast function unless some members can share the same private-key when they hold the same identity. According to this idea, Boneh *et al.* provided a hierarchical identity-based encryption (HIBE) system to support an organizational hierarchy [17], but this kind of hierarchy must be a tree structure and cannot provide identity-based revocation and tracing due to the global sharing of hierarchical identity/privacy-key for all users. In addition, attribute-based encryption (ABE) is also considered as an effective group communication method [18], but the existing ABE schemes have not yet been able to support the hierarchical structure.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

While it is challenging to enforce digital forensics analysis, Boneh and Waters [19] introduced augmented broadcast encryption that is efficient for constructing traitor-tracing, and trace-and-revoke systems. The scheme is resistant to an arbitrary number of colluders and secure against adaptive adversaries. Attrapadung and Imai [20] proposed a new cryptosystem, called Broadcast ABE (BABE), with direct revocation mechanism. Garg *et al.* [21] presented a trace-and-revoke scheme based on prime order bilinear groups, and provided the first implementations of efficient fully collusion-resilient traitor tracing scheme. Liu *et al.* [6] also presented a blackbox traceable ABE that achieves the traceability in $O(\sqrt{n})$, where n is the number of users in the system.

There have been some cryptosystems constructed on the partial order relation. Akl and Taylor [22] put forward a simple scheme to solve multilevel security problem in 1982. Kim *et al.* [23] proposed a new key management system for multilevel security using various one-way functions in 2005. Chung *et al.* [24] proposed a hierarchy method based on the elliptic curve cryptosystem and one-way hash function to solve dynamic access problems in 2008. Another related field is *hierarchical key management with time control*. For example, Tzeng [25] proposed a time-bound scheme based on Lucas function in 2002, but it is insecure against collusion attacks by Yi and Ye [26]. Another similar schemes based on the tamper-resistant device and the hash function were proposed by Chien [27] in 2004 and Bertino *et al.* [28] in 2008, respectively. Santis *et al.* [29] summarized and provided several provably-secure hierarchical key assignment schemes based on an existing schemes in 2007. Peer *et al.* [30] also propose an interesting method to represent data in multiple resolutions with each resolution secured with a different key in 1014. In all, these work cannot support common access control and digital forensics, but their techniques are worth learning for our construction.

Contributions. In this paper, our objective is to present a secure file syncing-and-sharing service on cloud based on digital forensics mechanisms against the abnormal players. To meet this goal, our core task is to construct a group-oriented cryptosystem for cloud data encryption, which supports the traitor tracing-and-revoking mechanism for digital forensics of the detected abnormal players. We propose a RBAC-compatible cryptosystem, called as Partially-ordered Hierarchical Encryption (PHE), that is more effective than traditional cryptosystem for tracing-and-revoking because it can organize all decryption keys into a hierarchical key structure according to partially ordering relation of role hierarchy in RBAC. In brief, the major contributions of our work are summarized as follows:

- We present a new secure FSS model to provide a forensic analysis framework to guide investigations. This model is generic enough to apply to anomaly detection for the abnormal players hit the FSS service, as well as to trace and revoke the traitors in these players. Moreover, the users are organized in different groups and given decryption keys associated with their groups' roles and partially ordering relations based on role hierarchy in RBAC.
- We provide a practical Partially-ordered Hierarchical Encryption construction, in which the players can encrypt and decrypt the outsourcing data by specifying the RBAC policies to meet high-level security requirements along with the RBAC standard. We also conduct a thorough analysis of the PHE construction for semantic security (against chosen-plaintext attacks) and secure key hierarchy against the collusion attack.
- We provide two kinds of forensics mechanisms, traitor tracing and revocation, based on key hierarchy. The tracing algorithm, including single-key tracing and hierarchical tracing, can be run efficiently in a black-box manner (without knowing how the decoder box was produced) to trace at least one compromised key among those used to create the decoder. Our PHE construction also supports the revocation mechanism not only for the groups but also for the users along with key hierarchy. Moreover, the tracing-and-revoking procedure only requires few computations.

In addition, our PHE scheme provides several new secure features, such as public user label, constant-size user key storage, fast tracing, lower computational costs and communication bandwidths. Our cryptosystem also takes full advantage of RBAC, which provides a well-designed and easy-to-manage approach for accessing cloud resources without user intervention. These advantageous features make our model and construction a promising solution to preserve the integrity and validity of long-term cryptosystems and to prevent the leakage of cloud outsourced data via illegal decoders (or illegal decryption softwares).

Organization. The rest of the paper is organized as follows. Section 2 and 3 describe the research background and the definition of key structure. In Section 4 and 5, we address our PHE scheme and security analysis on RBAC. In Section 6, we describe two security mechanisms, revocation and traitor tracing, for digital forensics. The results of performance evaluation are showed Section 7. Finally, we conclude and discuss the future work in Section 8.

2 System Model and Definitions

2.1 File Sharing and Syncing Service

We first describe a simple and general framework of file syncing and sharing service developed over the cloud computing platforms. As shown in the Fig. 3, this framework consists of three following entities:

FSS service: provides users with the ability to remotely store their data and access the same cloud-based data. Enterprise or business class versions of FSS services provide these capabilities in a secure manner that gives IT oversight and control.

Online player/editor: provides the ability to access this data from any location and any of their devices, including smart-phones and tablets, without having to go through a corporate VPN or firewall (shown in the middle module in the figure).

End users: The FSS service also provide the ability to share information with other users, both inside and outside the organization.

This kind of FSS service could be built on the open-source cloud platforms, such as, OpenStack and CloudStack[1], in which computing, networking and storage resources are integrated and managed as a unified system. These platforms provide a prefect interface with cloud service providers and tenants, but do not provide a direct interface with end users. As a more flexible and convenient way, online player/editor is developed as the bridge between FSS service and end users. They may be built a lot of different ways, such as web service, virtual desktop, and client/server-based applications.

1. http://www.openstack.org and http://cloudstack.apache.org

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

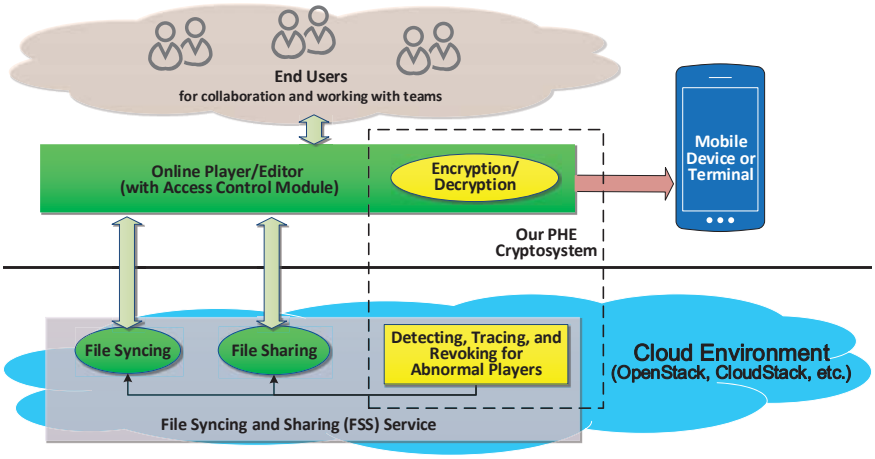


Fig. 3. A new FSS model for player abuse prevention.

2.2 New System Model for FSS

We present a new FSS model for player abuse prevention and enhanced protection against unauthorized access. The proposed model uses the hierarchical role-based access control (H-RBAC) model, which is recognized for its support for simplified administration and scalability of collaboration and working with teams. Moreover, the design of this model is generic enough to support other access control policies, such as discretionary access control and multilevel security.

We show such a cloud-based FSS model in the Fig. 4. This model that addresses and incorporates the afore-mentioned authorization requirements can be built using three types of components:

Anomaly detection: This is used for detecting abnormal players. More exactly, it is responsible for monitoring deployed resources and might allocate or release them to ensure the compliance of enterprise-side existing access control system. The output of this module is some suspected anomaly players.

Tracing traitors: This is responsible for finding out the traitors from the suspected players recognized in the previous step. In some cases this is simple and straightforward, but such a practice procedure sometimes results in solution difficulties if we request that the secrets or keys stored in the player cannot be leaked in the tracing procedure. We call it 'black box tracing'.

Revoking traitors: This is responsible for revoking the authority (or license) of traitors found in the previous step. The simple revocation method (e.g., the license is appeared in Blacklist) may be evaded in the way of license forgery and tampering. Taking into account the difficulty in comparing cryptographic key forgery and license forgery, the key-based revocation would be a more effective and secure manner.

The above model may be developed in a non-cryptographic way, but the cryptographic technique can dramatically increase the difficulty of attackers and enhance the security of system. Therefore, we will introduce a new cryptographic structure and construction into our FSS system model, as follows:

Key Hierarchy: This is a partial-order structure in which all user's keys are organized in a hierarchy according to role-hierarchy in H-RBAC. In this way, we can use the partial order to manage these keys.

PHE Cryptosystem: This is a new cryptosystem that enables the granting, usage tracking, and revoking of authorization. It is built on the key hierarchy described above, so that we call it the partially-ordered hierarchical encryption (PHE).

In summary, two new modules are added into the existing FSS services (as shown in the dashed box of the Fig. 3): one is the monitor module for detecting, tracing, and revoking for abnormal players; another is the cryptographic module for encrypting and decrypting data. In our system model, the PHE cryptosystem might be deployed in the client side (e.g., running on the Client/Server model) or the cloud side (e.g., working on the virtual desktop in cloud) according to the requirement of practical applications. The traitor tracing and revocation mechanisms will be also realized on this model by monitoring and tracing the situation of every decoder.

2.3 Anomaly Detection and Key Hierarchy

Obviously, the anomaly detection is an important module for protecting the outsourcing data against unauthorized access and finding out the suspected players in our system model. Some potential abnormal behaviors are listed as follows:

- To recognize unauthorized access, e.g., anomaly behavior of players against RBAC policies and the license terms (such as a period of time, days, weeks, months, etc.), privilege escalation (PE) [31] attacks, etc.
- To detect non-normal distribution exceeding the preset bounds, e.g., the unregistered devices away from mobile device management (MDM) [32], a user's license (or key) used into other unauthorized users or devices, etc.
- To identify the player abuse outside the permitted range, e.g., the request to access a sensitive resource in the extra working time or region, etc.

The above-mentioned anomaly behaviors can be recognized by using some existing approaches, e.g., audit, pattern matching, risk assessment [33], [34]. Therefore, we turn attention to other modules in our model.

In this paper the most challenging task is to design a new cryptosystem that meets the requirements in our FSS model. In particular, we require that this cryptosystem can support the role hierarchy (RH) in the existing H-RBAC systems. Therefore, we introduce a new hierarchical key structure using the public-key settings. Such a key hierarchy can achieve following functions:

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

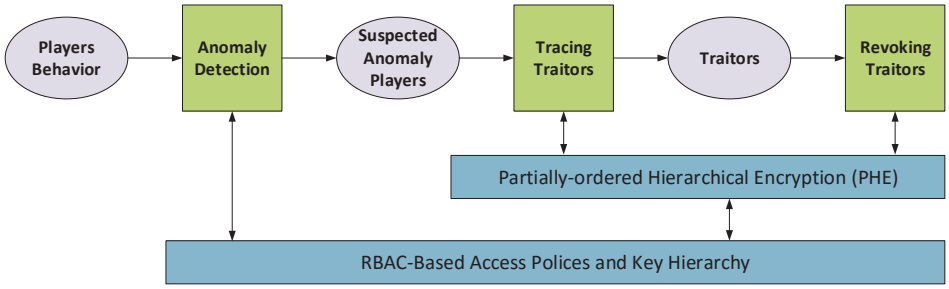


Fig. 4. A new FSS model for player abuse prevention.

- Each role is assigned with an encryption key (called role-key) in H-RBAC, and there exists a derivation function on these role-keys in accordance with RH;
- Each user has a unique identity and private key, which retain his/her role information, but the derivation of secret-key is prohibited; and
- Such a key structure can be used to establish some important security mechanisms, such as encryption, signature, revocation, and traitor tracking.

In order to achieve this goal, we expect to provide an effective method that transforms the RBAC-type access permission into the cryptographic rule. Based on this rule, the data can be encrypted and stored into cloud, then it can be accessed (or decrypted) by the cloud users as they would do in the original RBAC system. We will provide such a framework of RBAC-compliant cryptosystem, where a user can use the existing system with RBAC (such as Windows, Linux, etc.) to store and access cloud resources.

Although some existing schemes [35], [36], [37] with hierarchic structure are built on the secret-key setting, but our techniques also apply in the public-key setting based on the following advantages:

- The role-key could be made public, so everyone uses it to distribute the message/file to all users in this role. It is very useful for some applications that need to send/write (don't need to read) the data for the special role.
- Such a limit does not exist for the size of users/roles and the structure of key hierarchy in the public-key setting, e.g., new role/user is permitted to dynamically join our PHE scheme. But they are predefined and fixed when a symmetric cryptosystem is built.

3 Key Hierarchy for H-RBAC Model

3.1 Partial Order Hierarchy

Assume that the users of a computer (or communication) system are divided into a number of disjoint sets, $c_1, c_2, \cdots, c_n$. The term **security class** (or role in RBAC) is used to designate each of the c_i. In fact, a class can represent these members with the same interest in a company, department, project, or family. Assume further that a binary relation $\preceq$ partially orders among the set $C = \{c_1, c_2, \cdots, c_n\}$ of classes. In partially ordered set (Poset) $\langle C, \preceq \rangle$, the notation $c_j \preceq c_i$ is used to indicate that the class c_i dominates c_j. the meaning of $c_j \preceq c_i$ is that the users in class c_i can have access to the data held by users in c_j. Hence, we have following definition [28], [27]:

Definition 1 (Partial-order hierarchy): The poset $\Omega = \langle C, \preceq \rangle$ is called a (finite) partial-order hierarchy with relation $\preceq$ on a (finite) set C, if for two distinct classes c_i, c_j in C and $c_j \preceq c_i$, c_i is a security class higher than or equal to c_j.

Let $C = \{c_i\}_{i \in \mathbb{N}}$ be a partial-order hierarchy and $|C|$ denotes the number of classes in C. We write $c_j \prec c_i$, if $c_j \preceq c_i$ but $c_j \neq c_i$. For any $c_i \in C$, we write: $\downarrow c_i = \{c_j \in C : c_j \prec c_i\}$ and $\uparrow c_i = \{c_j \in C : c_i \preceq c_j\}$. Note that $c_i \in \uparrow c_i$. We call $\downarrow c_i$ as the **subordinates** of c_i. We say that c_i **directly dominates** c_j, written $c_j \prec_d c_i$, if and only if $c_i \neq c_j$ and $c_j \preceq c_* \preceq c_i$ implies $c_* = c_i$ or $c_* = c_j$. The direct dominations are described as directed edges in Fig. 5. We also say that c_i dominates c_j via n directed edges if there exist $\{c_{i_k}\}_{1 \leq k \leq n-1} \subseteq C$ such that $c_j \prec_d c_{i_{n-1}} \prec_d \cdots \prec_d c_{i_1} \prec_d c_i$, where $c_{i_k} \prec_d c_{i_{k-1}}$ for $2 \leq k \leq n-1$. We define $\Gamma(j,i) = \{v \in C : c_j \preceq v \prec c_i\}$ as an arbitrary path in all pathes from c_i to c_j for $c_j \preceq c_i$, as well as $\Delta(j,i) = \{c_{l_1} \prec_d c_{l_2} : c_{l_1}, c_{l_2} \in \Gamma(j,i)\}$ as the set of direct dominations in this path.

The partial-order hierarchy $\langle C, \preceq \rangle$ can be represented by the directed graph $G = \langle C, E \rangle$, where each class corresponds to a vertex in the graph and there is an edge from c_j to c_i if and only if $c_j \prec_d c_i$. We can compose various different structures including tree, inverted-tree, or lattice into a partial-order hierarchy. For example, Fig. 5 shows a simple example for an enterprise management system, in which more powerful (senior) classes are shown toward the top of the diagram and less powerful (junior) classes toward the bottom.

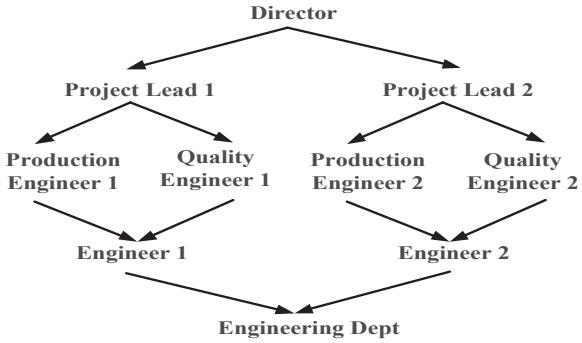


Fig. 5. Example of partial-order hierarchy for an enterprise management system.

3.2 Partial-order Key Hierarchy

Let M_i be a piece of information, or object, that the (authorized) user desires to store in (or broadcast over) the system. The meaning of the subscript i is that the object M is accessible to users in class c_i. The partial order on Ω implies that M_i is also accessible to users in all class c_j such that $c_i \preceq c_j$. So we use the partial order hierarchy to construct a key hierarchy for secure group communications as follows:

Definition 2 (Partial-order Key Hierarchy): Given a hierarchy $\langle C, E \rangle$ on $\Omega = \langle C, \preceq \rangle$, the triple $\Psi = \langle C, E, K \rangle$ is called a Partial-order Key Hierarchy (PKH), if given $C = \{c_0, c_1, \cdots, c_n\}$

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

and $c_i = \{u_{i,1}, u_{i,2}, \cdots, u_{i,n_i}\}$*, there exists a set of keys* $K = \{pk_i, \{sk_{i,j}\}_{j\in[1,n_i]}\}_{i\in[0,n]}$*, where* pk_i *belongs to* c_i *and* $sk_{i,j}$ *belongs to* $u_{i,j}$*, and then the access control of encryption (with* pk_i*) and decryption (with* $sk_{i,j}$*) complies with the partial order relation in* Ω*:*

$$Decrypt(sk_{i,j}, Encrypt(pk_k, M)) = M \quad (1)$$

for $c_k \preceq c_i$*, where* $Decrypt(\cdot)$ *and* $Encrypt(\cdot)$ *are decryption and encryption functions.*

This definition means that each class corresponds to an encryption key pk_k, by which users can encrypt (or store) the messages into the class c_k, and each user $u_{i,j}$ saves a private decryption key $sk_{i,j}$ to decrypt the messages. Moreover, Equation (1) shows $1:n$ relationship between pk_k and $sk_{i,j}$. In Fig. 6, we show such a PKH example in terms of the above-mentioned example for an enterprise management system. In this figure, each circle denotes a class and each class involves some users (denoted as a black dot). According to the definition of PKH, each class holds an encryption key pk_k and each user has a private decryption key $sk_{i,j}$. The inclusion relationship between the circle and the black dot denotes the $1:n$ relationship between pk_k and $sk_{i,j}$.

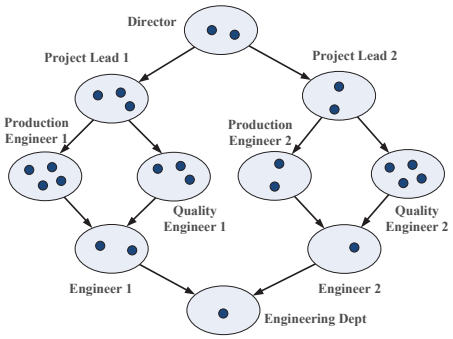


Fig. 6. Example for PKH with the general hierarchy.

In contrast with the previous key hierarchy structure, the partial-order key hierarchy has some different properties:

1) Each user $u_{i,j}$ is assigned a unique decryption key $sk_{i,j}$, by which the sender can choose or identify certain users in the process of encryption, revocation, and tracing. Such a feature is essential for digital forensics.
2) Public-key cryptography can be introduced to ensure the security of the user's decryption key even if the encryption key makes public, *i.e.*, in email encryption systems.
3) The *derivation* function of user decryption keys is forbidden even for the cases of partial-order relations and every probabilistic polynomial-time (PPT) algorithm $Delegate$, we have

$$\Pr[Delegate(sk_{i,j}, c_l) = sk_{l,j'}] \leq \epsilon, \forall c_l \preceq c_i. \quad (2)$$

where, ϵ is small enough. So the users can not use the key derivation function to obtain the new keys or identities.

For ease of use, we expect that a system manager assigns the user key $sk_{i,j} = (lab_{i,j}, dk_{i,j})$ to users, where $lab_{i,j}$ is a public (easily remembered) label and $dk_{i,j}$ is a decryption key. In public-key settings, the sender does not hold any private information and the encryption process is only performed with the help of the public encryption keys $\{pk_i\}$ containing all user labels $\{lab_{i,j}\}$.

PKH supports arbitrary partial-order structures, such that it is closely related to practical scenarios in which a hierarchical structure exists. This kind of key hierarchy is also different from other logical key hierarchies, such as, LKH, LSD, OFT (One-way Function Tree), and so on. More importantly, the number of users in a class is unlimited and dynamically alterable. In order to ensure the security of systems, PKH needs to satisfy the following properties:

- The encryption key does not provide enough information to increase the capabilities of adversaries;
- Each user in a class cannot decrypt data of another classes except it's subordinates, and the user cannot forge other decryption keys;
- The encryption key can be modified to satisfy the users' requirements, but it cannot be interfered from the issued keys of other users;
- To support digital forensics (or traitor tracing), there exists an efficient black-box tracing algorithm to identify all traitors from the illegal decoder.

For the sake of clarity, we list major variables used throughout this paper in Table 1.

TABLE 1
Description of the variables

No.	Variable	Description
1	Ψ	The key hierarchy and $\Psi = \langle C, E, K\rangle$;
2	n, n_i	The number of classes in C and users in c_i;
3	t	The maximal coalition size;
4	s	The security parameter;
5	mk	The master key for the system manager;
6	P	The public parameters and $P = \{\{pp_i\}, \{label_{i,j}\}_{j\in[1,n_i]}\}_{i\in[1,n]}$;
7	pk_i	The encryption key of the i-th group;
8	$sk_{i,j}$	The decryption key of the user $u_{i,j}$ and $sk_{i,j} = (label_{i,j}, dk_{i,j})$;
9	$dk_{i,j}$	The private key of the user $u_{i,j}$;
10	$label_{i,j}$	The public label of the user $u_{i,j}$;
11	R_i	The set of the revoked users in pk_i;
12	M, ek	The plaintext and the session key;
13	$\mathcal{C}_i$	The ciphertext encrypted by pk_i.

4 PHE Cryptosystem

Given a secure key hierarchy $\Psi = \langle C, E, K\rangle$ and the total number n of classes, we can define a (t, n)-Partially-ordered Hierarchical Encryption (PHE), which ensures a content provider to securely transmit a message to a subset of authorized users under the assumption of at most t collusion. More formally, a (t, n)-PHE scheme with a security parameter s is a 6-tuple of probabilistic algorithms (S, J, E, D, R, T) described as follows:

1) **Setup(Ω, s, t):** Takes as input a partial-order hierarchy Ω, a security parameter s and a maximal collusion number t. It outputs a main encryption key pk_0 as the starting point of cryptosystem, a set of public parameters P[2], and a master key mk as the manager secret.
2) **Join(P, mk, c_i or $u_{i,j}$):** Includes two sub-algorithms:
   - **Join_Group(P, mk, c_i):** Takes as input the manager secret mk and a group identifier c_i. It generates an encryption key pk_i and some public parameters pp_i as the description of this class. $P = P \cup \{pp_i\}$ is made public.
   - **Join_User($P, mk, u_{i,j}$):** Takes as input the manager secret mk and a user identifier $u_{i,j}$. It outputs a user key

2. The signature of P can be generated avoid tampering.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

$sk_{i,j} = (lab_{i,j}, dk_{i,j})$. $P = P \cup \{lab_{i,j}\}$ and $sk_{i,j}$ is sent to $u_{i,j}$ securely. [3]

3) **Encrypt(P, pk_i, M):** Encrypts a message M using the encryption key pk_i and outputs a ciphertext $\mathcal{C}_i$.
4) **Decrypt($P, sk_{i,j}, \mathcal{C}_k$):** Decrypts a ciphertext $\mathcal{C}_k$ using a decryption keys $dk_{i,j}$ and outputs the message M, if $u_{i,j} \in c_i$ and $c_i \preceq c_k$.
5) **Revoke(P, pk_i, M, R_i):** Takes as input pk_i, a set of revoked classes (and users) R_i, and a message M. It outputs a ciphertext $\mathcal{C}_i'$ which can be decrypted by the remaining users in $(\uparrow c_i) \setminus R_i$.
6) **Trace$^{\mathcal{D}}$(P, pk, mk):** Suppose an adversary uses k user keys $R = \{sk_{i_1,j_1}, \cdots, sk_{i_k,j_k}\}$ to create a decryption box $\mathcal{D}$. As an oracle algorithm on $\mathcal{D}$, it takes as input pk, mk, and can determine all keys in R.

A tracing algorithm is said to be 'Black Box' if the decoder $\mathcal{D}$ can only be queried as an Oracle but not opened to reveal its internal keys. The scheme is said to be 't-resilient' if there is an effective cryptosystem with the collusion of at most t keys. Note that, the four algorithms (S, J, E, D) are used to realize cryptographic access control under RBAC model, and the two algorithms (R, T) provide two security mechanisms, revocation and traitor tracing, for digital forensics.

4.1 Proposed PHE Scheme

Given a hierarchy $\Omega = \langle C, \preceq \rangle$, a security parameter s, and the maximal coalition size t. Let $\mathbb{G}_q$ be a group of prime order q and $\log_2 q > s$. One can take as $\mathbb{G}_q$ the subgroup of $\mathbb{Z}_p^*$ of order q, where p is a large prime with $q|p-1$. Let $g \in_R \mathbb{Z}_p^*$ be a generator of $\mathbb{G}_q$.

Before describing our PHE scheme, we introduce *Lagrange interpolation polynomial* that is a main technique for our scheme. Given a set of t points $\{(x_0, y_0), (x_1, y_1), \cdots, (x_t, y_t)\}$, where $x_i, y_i \in \mathbb{Z}_q^*$ for $i \in [0, t]$ and all x_i are unique. Then there is a unique polynomial $f(x)$ of degree at most t, that passes through all these $t+1$ points, i.e. $f(x_i) = y_i$ for all $i \in [1, t]$. The polynomial $f(x)$ is defined as

$$f(x) = \sum_{i=0}^{t} y_i \cdot f_i(x) = s + \sum_{i=1}^{t} a_i x^i \pmod q,$$

where $f_i(x) = \prod_{0 \le j \le t, j \ne i} \frac{x - x_j}{x_i - x_j}$, $i \in [0, t]$. When $x = 0$, it is easy to recover the item s of $f(x)$ by using $f(0) = s = \sum_{i=0}^{t} y_i \cdot f_i(0)$, where $f_i(0) = \prod_{j=0, j \ne i}^{t} \frac{x_j}{x_j - x_i}$, $i \in [0, t]$.

Next, we describe the random variates to generate the encryption key pk_i and the user's decryption key dk_{ij}:

- The random variate x_k is used in the encryption key pk_i, where k is the serial number from 1 to t;
- The random variate $x_{i,j}$ is used in the decryption key dk, where i denotes the i-th group or role and j denotes the j-th users in this group or role.

We require that these two variates differ from each other and keep public. We usually use the collision free hash function $hash$ to generate them based on some easy-to-remember strings, e.g., $x_{i,j} = hash(ID_{i,j})$, where $ID_{i,j}$ is the identity of $u_{i,j}$.

Our PHE scheme is described as follows:

1) **Setup(Ω, s, t):** The manager chooses t random integers $a_1, \cdots, a_t \in \mathbb{Z}_q^*$ to construct an initial random polynomial $f(x) = \sum_{i=1}^{t} a_i x^i \pmod q$ with degree t. It therefore randomly chooses t integers $x_1, \cdots, x_t$ to generate $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$. It makes the parameters $P = \{p, q, (x_1, g^{f(x_1)}), \cdots, (x_t, g^{f(x_t)})\}$ public. Without loss of generality, we assume that c_0 be only the senior-most class in Ω. It chooses a random integer $s_0 \in_R \mathbb{Z}_q$ for c_0 as its secret, so that the random polynomial $f(x)$ is replaced by $p_0(x) = s_0 + \sum_{i=1}^{t} a_i x^i \pmod q$. It computes $z_{0,i} = g^{p_0(x_i)} = g^{s_0} \cdot g^{f(x_i)} \pmod p$ from P, and then uses $(x_1, p_0(x_1)), \cdots, (x_t, p_0(x_t))$ to generate an initial encryption key:

$$\begin{aligned} pk_0 &= \langle g, z_{0,0}, (x_1, z_{0,1}), \cdots, (x_t, z_{0,t}), T_0 \rangle \\ &= \langle g, g^{p_0(0)}, (x_1, g^{p_0(x_1)}), \cdots, (x_t, g^{p_0(x_t)}), \emptyset \rangle \end{aligned} \quad (3)$$

where, $T_0 = \emptyset$ denotes a null initial control domain. The system manager keeps $mk = \{s_0, a_1, \cdots, a_t\}$ secret.

2) **Join(P, mk, c_i or $u_{i,j}$):** which includes two forms:
   a) **Join_Group(P, mk, c_i):** To generate pk_i and pp_i of $c_i \in C$, the manager assigns the random $s_i \in \mathbb{Z}_q$ for c_i as its secret. For $\forall c_l \in C$ and $c_i \prec_d c_l$, it computes $t_{i,l} = g^{(s_l - s_i)} \pmod p$ as the public parameter of this relation, and then defines $pp_i = \{t_{i,l}\}_{c_i \prec_d c_l}$ as the set of all relations which directly dominate c_i. Finally, it appends s_i and pp_i into mk and P respectively, i.e., $mk = mk \cup \{s_i\}$ and $P = P \cup pp_i$. The encryption key pk_i in c_i can be computed from the polynomial $p_i(x) = s_i + f(x)$. In terms of mk and P, the manager sets $z_{i,k} = g^{p_i(x_k)}$ for all $k \in [1, t]$ and generates

$$\begin{aligned} pk_i &= \langle g, z_{i,0}, (x_k, z_{i,k})_{k=1}^{t}, T_i \rangle \\ &= \langle g, g^{p_i(0)}, (x_k, g^{p_i(x_k)})_{k=1}^{t}, T_i \rangle, \end{aligned} \quad (4)$$

   where, T_i is a set of all relations in $\uparrow c_i$, i.e., $T_i = \{t_{j,l}\}_{c_j, c_l \in \uparrow c_i, c_j \prec_d c_l}$.
   b) **Join_User($P, mk, u_{i,j}$):** To generate $sk_{i,j}$ of $u_{i,j}$, the manager computes the random polynomial $p_i(x) = s_i + f(x) \pmod q$ by using the secret in mk. It generates a new random integer $x_{i,j} \in_R \mathbb{Z}_q$ and sends $sk_{i,j} = (x_{i,j}, p_i(x_{i,j}))$ to the user via a secret channel, where $lab_{i,j} = x_{i,j}$, $dk_{i,j} = p_i(x_{i,j})$, and $P = P \cup \{lab_{i,j}\}$.
3) **Encrypt(P, pk_i, M):** For a message $M \in \mathbb{Z}_p^*$ [4], the user randomly chooses a random number $r \in_R \mathbb{Z}_q^*$, and then computes the ciphertext by pk_i as follows:

$$\begin{aligned} \mathcal{C}_i &= \langle h, S_i, (x_k, h_{i,k})_{k=1}^{t}, T_i' \rangle \\ &= \langle g^r, M \cdot (z_{i,0})^r, (x_k, (z_{i,k})^r)_{k=1}^{t}, \{(t_{k_1,k_2})^r\}_{t_{k_1,k_2} \in T_i} \rangle. \end{aligned} \quad (5)$$

   where, $h_{i,k} = (z_{i,k})^r \pmod p$, $t'_{k_1,k_2} = (t_{k_1,k_2})^r$, and $T_i' = \{t'_{k_1,k_2}\}_{t_{k_1,k_2} \in T_i} = \{(t_{k_1,k_2})^r\}_{t_{k_1,k_2} \in T_i}$ denotes a control domain which includes all relations in $\uparrow c_i$.
4) **Decrypt($P, sk_{i,j}, \mathcal{C}_l$):** After receiving a cipher-text $\mathcal{C}_l = \langle h, S_l, (x_k, h_{l,k})_{k=1}^{t}, \{t'_{k_1,k_2}\}_{t_{k_1,k_2} \in T_l} \rangle$, the user first extracts $x_{i,j}$ from the private key $sk_{i,j} = \langle x_{i,j}, y_{i,j} \rangle$ and sets $x_0 = x_{i,j}$ to generate the set $\{x_0 = x_{i,j}, x_1, \cdots, x_t\}$. Next, the user makes use of the set to compute the coefficient of Lagrange interpolation polynomial $\lambda_k =$

3. A secure channel is usually used to ensure the security of distribution of secret keys. For example, the SSL/TLS protocol can provide such a communication security over key distribution. In our architecture, the secret key is stored in a certain "security zone" of the Player/editor or a security token (e.g., secure USB Flash Drive).

4. The plaintext M must be converted into an element of $\mathbb{Z}_p^*$, see ElGamal encryption system.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

$\prod_{l=0,l\neq k}^{t} \frac{x_l}{x_l - x_k} \pmod q$ for all $k \in [0,t]$. [5] And then, the user computes the following equation by the value y_{ij} of $sk_{i,j} = \langle x_{i,j}, p_i(x_{i,j})\rangle$ if we hold $u_{i,j} \in c_i$, $c_i \preceq c_l$, and

$$U_{\mathcal{C}_l}(sk_{i,j}) = \frac{h^{p_i(x_{i,j})\cdot\lambda_0}\prod_{k=1}^{t}(h_{l,k})^{\lambda_k}}{(\prod_{c_{k_1}\prec_d c_{k_2}\in\Delta(l,i)} t'_{k_1,k_2})^{\lambda_0}}, \quad (6)$$

where, $\Delta(l,i) = \{c_{k_1} \prec_d c_{k_2} : c_{k_1}, c_{k_2} \in \Gamma(l,i)\}$ denotes the set of direct dominations on an arbitrary path between c_i and c_l. It finally outputs the plaintext $M = S_i / U_{\mathcal{C}_l}(sk_{i,j})$.

Before going further, we briefly show that the encryption scheme is valid. Firstly, in terms of Lagrange interpolation polynomial of $p_l(x)$, we have $p_l(0) = \sum_{k=0}^{t} p_l(x_k)\cdot\lambda_k(0) = \sum_{k=0}^{t} p_l(x_k)\cdot\lambda_k \pmod q$. Next, we have the equation $p_l(x_{i,j}) = s_l + f(x_{i,j}) = s_i - (s_i - s_l) + f(x_{i,j}) = p_i(x_{i,j}) - (s_i - s_l) \pmod q$ for $p_i(x_{i,j}) = s_i + f(x_{i,j})$ according to the definition of $p_i(x)$ and $p_l(x)$. Based on them, we can verify the following decryption equation

$$\begin{aligned} U_{\mathcal{C}_l}(sk_{i,j}) &= \frac{(g^r)^{p_i(x_{i,j})\cdot\lambda_0}\prod_{k=1}^{t} g^{p_l(x_k)\cdot\lambda_k\cdot r}}{(\prod_{c_{k_1}\prec_d c_{k_2}\in\Delta(l,i)} t_{k_1,k_2}^{r})^{\lambda_0}} \\ &= \frac{g^{p_i(x_{i,j})\cdot\lambda_0\cdot r}\prod_{k=1}^{t} g^{p_l(x_k)\cdot\lambda_k\cdot r}}{g^{\sum_{c_{k_1}\prec_d c_{k_2}\in\Delta(l,i)}(s_{k_2}-s_{k_1})\cdot\lambda_0\cdot r}} \\ &= \frac{g^{p_i(x_{i,j})\cdot\lambda_0\cdot r}\prod_{k=1}^{t} g^{p_l(x_k)\cdot\lambda_k\cdot r}}{g^{(s_i-s_l)\cdot\lambda_0\cdot r}} \\ &\overset{x_0=x_{i,j}}{=} g^{p_l(x_0)\cdot\lambda_0\cdot r}\prod_{k=1}^{t} g^{p_l(x_k)\cdot\lambda_k\cdot r} \quad (7) \\ &= g^{\sum_{k=0}^{t} p_l(x_k)\cdot\lambda_k\cdot r} = g^{p_l(0)\cdot r} = z_{l,0}^{r}. \end{aligned}$$

where $s_i - s_l = \sum_{c_{k_1}\prec_d c_{k_2}\in\Delta(l,i)}(s_{k_2} - s_{k_1}) \pmod q$ for an arbitrary path $\Gamma(l,i)$ between c_i and c_l [6].

4.2 Further Discussion

The above initial process of partial-order hierarchy is constructed from bottom (junior-class) to top (senior-class), but the $Setup$ algorithm is still available for the senior-most class. Without loss of generality, we assume that $c_0^{(1)}, c_0^{(2)}, \cdots, c_0^{(l)}$ are l senior-most classes in Ω. Then, it chooses a random integer $s_0^{(i)} \in_R \mathbb{Z}_q$ for $c_0^{(i)}$ as the secret of this class, such that it constructs l random polynomials, $p_0^{(i)}(x) = s_0^{(i)} + \sum_{k=1}^{t} a_k x^k$, where $i \in [1,l]$. Finally, the encryption key is generated:

$$\begin{aligned} pk_0^{(i)} &= \langle g, z_{0,0}^{(i)}, (x_1, z_{0,1}^{(i)}), \cdots, (x_t, z_{0,t}^{(i)}), T_0\rangle \\ &= \langle g, g^{p_0^{(i)}(0)}, (x_1, g^{p_0^{(i)}(x_1)}), \cdots, (x_t, g^{p_0^{(i)}(x_t)}), \emptyset\rangle, \end{aligned}$$

where, $g^{p_0^{(i)}(x_k)} = g^{s_0^{(i)}} g^{f(x_k)} \pmod p$.

In order to share information, the encryption keys pk_n of junior-most classes are usually made public, which is called the main encryption key, e.g., for the enterprise management system shown in Fig. 6, if the encryption key of "Engineering Dept" class is used to send the message, all employees are able to decrypt it by their own private keys. Moreover, the storage ratio of encryption keys is also an important feature considering a number of classes in the large-scale organizations. We, of course, expect that it is as low as possible. Since $p_i(x) = (s_i - s_l) + p_l(x)$, the user can generate pk_i by using a known pk_j and public parameters P for $i \neq j$. For example, the user can compute her/his own encryption key pk_i from a junior-most encryption key pk_n by $\dot{T}_i = \prod_{c_j\prec_d c_l\in\Delta(n,i)} t_{j,l} = g^{\sum_{c_j\prec_d c_l\in\Delta(n,i)}(s_l - s_j)} = g^{s_i - s_n} \pmod p$ and

$$\begin{aligned} pk_i &= \langle g, z_{i,0}, (x_k, z_{i,k})_{k=1}^{t}, T_i\rangle \quad (8) \\ &= \langle g, z_{n,0}\cdot\dot{T}_i, (x_k, z_{n,k}\cdot\dot{T}_i)_{k=1}^{t}, T_i\rangle, \end{aligned}$$

where, $z_{n,k}\cdot\dot{T}_i = g^{p_n(x_k)}\cdot g^{s_i - s_n} = g^{p_i(x_k)} \pmod p$, and T_i is found from P in terms of Equation (4). Therefore, the user only needs to store an encryption key pk_i and a private key $sk_{i,j} = (label_{i,j}, dk_{i,j})$.

The key hierarchy is saved in public parameters P, irrespective of the user private keys, so that the public parameters can be merely modified dynamically to support the change of key hierarchy and the revocation of users. As a group-oriented cryptosystem, the parameter P is usually shared through secure public storage medium, e.g., FTP, Web, Facebook. So the users have access to the current-time P in a only-read way by the ACM of Player/editor as described above.

Finally, it is well-known that public key systems are clumsy to use in transmitting long messages. Instead, we use our PHE cryptosystem to encrypt a random symmetric key ek (called a session key) and then encrypt long message by using the traditional symmetric encryption on ek (denoted as $E(ek, M)$). We describe this encryption process as $Encrypt(P, pk_i, ek)$ and $E(ek, M)$, such that the ciphertext is denoted as $\langle \mathcal{C}_i, E(ek, M)\rangle$. The decryption process is that the random symmetric key ek is first recovered by using our PHE decryption algorithm, and then the long message M is decrypted by the corresponding symmetric decryption algorithm.

5 Security Analysis

Semantic security is a widely-used security notion in a public-key encryption scheme. Informally, it requires that it is infeasible to learn anything about the plaintext from the ciphertext. This security requirement is also fit for PHE scheme. We show that our encryption scheme is semantically secure agaist chosen plaintext attack (IND-CPA) under the Decision Diffie-Hellman (DDH) assumption.

*Theorem 1 (**Semantic Security**): The proposed (t,n)-PHE scheme is semantically secure under chosen plaintext attacks assuming the difficulty of Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_q$.*

The proof of this theorem is provided in see Appendix 2.1. Obviously, semantic security is not enough to satisfy the security requirement of "1:n" encryption scheme. It is important to consider all types of potential attacks when we attempt to design the key hierarchy and broadcast scheme. The security of key hierarchy must assure that the adversary cannot gain any advantage by analyzing public-keys, ciphertexts, and user's private keys. There exist two strategies to attack the PHE scheme:

1) **Privilege Attack**: it focuses on changing the privileges of the granted users or getting the keys of the other users. This attack also involves two ways:
   - **Collusion attack for corrupted classes**, in which the corrupted users in $R = \{u_{i_k,j_k}\}_{k=1}^{t}$ wish to forge a (new or unused) key in $\{c_{i_1}, \cdots, c_{i_t}\}$ (called as the corrupted

5. Given a set of $t+1$ different data points $(x_0, y_0), \cdots, (x_t, y_t)$, the language interpolation polynomial is a linear combination $L(x) = \sum_{j=0}^{t} y_j \lambda_j(x)$ where the coefficient $\lambda_j(x) = \prod_{i=0, i=j}^{t} \frac{x - x_i}{x_j - x_i}$. Here, we set $x = 0$ to define $\lambda_j(0) = \lambda_j$.

6. For the different pathes, we have the same polynomial $p_i(x) = s_i + \sum_{i=1}^{t} a_i x^i$, because $p_i(x) = (s_i - s_{i-1}) + (s_{i-1} - s_{i-2}) + \cdots + (s_1 - s_l) + p_l(x)$ for any path $s_i, s_{i-1}, \cdots, s_1, s_l$.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

classes). The aim of this attack is to avoid tracing and frame the innocent users.

- **Collusion attack for honest classes**, in which the corrupted users in $R = \{u_{i_k,j_k}\}_{k=1}^{t}$ wish to forge a (new or unused) key in $C \setminus \{c_{i_1}, \cdots, c_{i_t}\}$. The aim of this attack is to change the privileges in partial order hierarchy.

2) **Access Attack**: it focuses on gaining the advantage of adversary to break the cryptosystem or extending the range of access by the collusion of corrupted users, especially gaining the advantage to break the revocation algorithm.

We would like to adopt appropriate technologies to prevent the above attacks, but the collusion attack is unavoidable in the way of technology because the traitor has been a granted user before s/he is not found. Thus traitor tracing is an efficient method to frighten the collusion attack. However, we must ensure that the traitors cannot forge an 'unused' key to avoid tracing but leave some 'foregone' clue of evidence to find them. We present such a definition for Secure Key Hierarchy (SKH) as follows:

*Definition 3 (**Secure Key Hierarchy**): A (t,n)-PHE sche-me $(\mathcal{S},\mathcal{J},\mathcal{E},\mathcal{D})$ is said to have a secure key hierarchy $\langle C,E,K\rangle$ satisfying the following conditions:*

1) ***Validity:** for any member $u_{i,j}$ in $c_i \in C$, the session key ek can be efficiently computed from B_l and $sk_{i,j}$, where $c_i \prec c_l$. Then for every pair $(pk_l, sk_{i,j})$ in the range of $\mathcal{G}(1^n)$ and every sequence M_n, $|M_n| \leq poly(n)$,*

$$\Pr\left[\mathcal{D}(sk_{i,j}, \mathcal{E}(pk_l, M_n)) = M_n\right] \geq 1 - \frac{1}{|p(n)|};^{7} \quad (9)$$

2) ***Privilege attack:** for any set $R \subseteq \{u_{i_1,j_1}, \cdots, u_{i_m,j_m}\}$, $|R| \leq t$, it is computationally infeasible to compute $sk_{i,j}$ of a user $u_{i,j} \notin R$ and the (public) encryption key pk. Then for every probabilistic polynomial-time algorithm $\mathcal{A}$, every polynomial $p(\cdot)$, and all sufficiently large n,*

$$\Pr\left[\begin{array}{c}\mathcal{A}(pk, \{sk_{i_l,j_l}\}_{u_{i_l,j_l}\in R}) = sk_{i,j} \\ : sk_{i_l,j_l} \notin \{sk_{i,j}\}_{u_{i_l,j_l}\in R}\end{array}\right] < \frac{1}{|p(n)|}; \quad (10)$$

where, $pk = P \cup \{pk_i\}_{c_i \in C}$.

3) ***Access attack**: for any set $R \subseteq \{u_{i_1,j_1}, \cdots, u_{i_m,j_m}\}$ $|R| \leq t$, it is computationally infeasible to gain the advantage to break the revocation algorithm from the collusion set R and any ciphertexts $\mathcal{C}_l = \mathcal{E}^R_{pk_l}(M_n)$, where $\mathcal{E}^R$ denotes revocation algorithm on R and M_n is a sequence with $|M_n| \leq poly(n)$. Then for every probabilistic polynomial-time algorithm $\mathcal{A}$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \to \{0,1\}^*$ (see [38]), every polynomial $p(\cdot)$, and all sufficiently large n,*

$$\begin{aligned}&\Pr\left[A\left(\begin{array}{c}pk, h(X_n), \mathcal{E}^R_{pk_l}(X_n), \\ \{sk_{i,j}\}_{u_{i,j}\in R}\end{array}\right) = f(X_n)\right] \quad (11)\\ <\ &\Pr\left[A\left(\begin{array}{c}pk, h(X_n), \\ \{sk_{i,j}\}_{u_{i,j}\in R}\end{array}\right) = f(X_n)\right] + \frac{1}{|p(n)|}.\end{aligned}$$

Where, $f(X_n)$ denotes the information that the adversary tries to obtain from the plaintext X_n and $h(X_n)$ denotes a priori partial information about the plaintext.

In this definition, the condition 3) aims at the risk of revocation mechanism and puts forward this security requirement (tighter than Theorem 1), which conforms to the definition of 'semantic security' besides the additional key information $\{sk_{i,j}\}_{u_{i,j}\in R}$ for a set of revoked users R. As is well known, the encryption scheme is semantically secure if and only if it has indistinguishable encryptions (see Theorem 5.2.5 in [38]). So, we replace Equation (11) with the following equation

$$\left|\begin{array}{c}\Pr[\mathcal{A}(pk, \{sk_{i,j}\}_{u_{i,j}\in R}, \mathcal{E}^R_{pk_l}(X_n)) = 1] - \\ \Pr[\mathcal{A}(pk, \{sk_{i,j}\}_{u_{i,j}\in R}, \mathcal{E}^R_{pk_l}(Y_n)) = 1]\end{array}\right| < \frac{1}{|p(n)|}, \quad (12)$$

such that it is easier than ever to prove the security of scheme against access attack.

Next, we turn our attention to the security proof of our scheme. The proof of this theorem is provided by Appendix 2.2-5, in which the security against privilege attack includes two cases: privilege attack for honest classes and one for corrupted classes. Firstly, we prove that our scheme is secure against the privilege attack for honest classes based on the following lemma:

*Lemma 1 (**Privilege attack for honest classes**): The proposed (t,n)-PHE scheme can assure that the adversary cannot change the privilege in key hierarchy by forging a key of honest classes under Computing Discrete Logarithm (CDL) assumption, even if there exist more than t-collusion in key hierarchy.*

The proof of this lemma is presented in Appendix 2.2. Secondly, we prove that our scheme is secure against the privilege attack for corrupted classes based on the following lemma:

*Lemma 2 (**Privilege attack for corrupted classes**): The proposed (t,n)-PHE scheme can assure that the adversary cannot avoid tracing or frame the innocent users by forging a key of corrupted classes under Computing Discrete Logarithm (CDL) assumption, even if there exist at most t-collusion in key hierarchy.*

The proof of this lemma is presented in Appendix 2.3. We say that our scheme is secure against privilege attack in terms of Lemma 1 and 2. Thirdly, we prove that our scheme is secure against the access attack based on the following lemma:

*Lemma 3 (**Access attack**): The proposed (t,n)-PHE sche-me can assure that the adversary cannot get the additional advantage to access the secret of ciphertexts, encrypted by revocation algorithm, under Decisional Diffie-Hellman (DDH) assumption and at most t-collusion in key hierarchy.*

The proof of this lemma is presented in Appendix 2.5. Based on all three of the above lemmas, we prove that our PHE scheme satisfies the following theorem:

*Theorem 2 (**Secure Key Hierarchy**): The proposed (t,n)-PHE scheme has a secure key hierarchy satisfying Definition 3 under Computing Discrete Logarithm (CDL) and Decisional Diffie-Hellman (DDH) assumptions on Lemma 1, 2 and 3.*

The above theorem shows that the PHE scheme is secure under the threshold t of collusion attacks, but t is not the upper limit of threshold of collusion attacks for the whole PHE scheme. A reasonable assumption is that the threshold of attacks is at least t for each class, and the scheme is secure if the number of corrupted users is less than t in each class. This is not a large problem in applications where the large threshold t might be used without deterioration in performance (see Fig. 7).

6 PHE Scheme for Digital Forensics

In this section, we discuss the revocation and traitor tracing mechanisms for the support of digital forensics. It is very

7. $\frac{1}{|p(n)|}$ denotes negligible or negligibly small, which means that the absolute value is asymptotically smaller than any polynomial bound.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

hard for the adversary to directly break a cryptosystem with provable security, but the adversary could make other means to break it. It is well-known that **"the easiest way to capture a fortress is from within"**. Based on the same idea, the collusion attack between the adversary and some corrupted users (called traitors) is such an internal attack for group-oriented cryptosystem. In this attack, the adversary may have access to a set of legitimate user's secret keys to decrypt the ciphertext. In order to withstand such attacks, traitor tracing is introduced in the recent years.

Usually, the traitor tracing algorithm is an effective detection approach to find out the corrupted users from a group of authorized users based on a found pirate decoder. We prefer that the tracing algorithm is only able to access any pirate decoder as a black box and perform the tracing based on the decoder's response on different input ciphertexts. More importantly, the revocation mechanism can be used to revoke the traitors' decryption keys used in the pirate decoder, so that the management can quickly eliminate negative influence of traitors. We show this process in Fig. 2. In this section, we will present a public revocation algorithm and a black-box tracing algorithm to improve their performances.

6.1 Traitor Tracing based on Key Hierarchy

The traitor tracing is an efficient mechanism to support digital forensics in the existing group-oriented cryptosystems. Some tracing schemes have also been proposed via the polynomial interpolation method in the recent years. We here propose a new traitor tracing scheme for our partial-order key hierarchy on the basis of these existing schemes. This algorithm only needs to know the public label $label_{i,j}$ of users rather than their private keys. Note that, traitor tracing, as a way of digital forensics, has a precondition where the adversary cannot forge an 'unused' key to avoid tracing. We will prove that this attack is infeasible for our scheme. We now turn our attention to the tracing algorithm from the following two aspects:

6.1.1 Single-key Tracing

The single-key tracing algorithm focuses on finding the traitors of collusion one by one. It is very important of Lemma 2 in Appendix A.3 to tell us that at most t users cannot forge a new unused key in the corrupted class, such that we can find all traitors only if we search all used keys in this class. For such a collusion attack, we can use the revocation algorithm to construct a ciphertext, revoked by the suspicious key, into the illegal decoder. If the decoder does not work, this revocation key includes at least a traitor. Otherwise, we search other users in this subset. Finally, we can find all traitors. To improve the performance, we can check out t suspicious keys at the same time. Hence, the searching complexity is $O(m/t)$, where m is the total number of users in a security calls or a group of users.

Many tracing algorithms [39] have noticed that a certain linear combination of $sk_1, \cdots, sk_m$ is also a 'new' private key, but in this case the adversary is not confined to the original decryption algorithm to build a decryption box. In such a case, this 'single-key' is not a new key but a linear combination of some keys. For such a decoder, we can construct an encryption key, which includes t user keys, and search all combinations among the keys in this subset. In this case, the searching complexity is $O(\binom{t}{m})$.

6.1.2 Hierarchical Tracing

The hierarchical tracing algorithm is a more efficient method to find the traitors in terms of partial-order key hierarchy. According to Lemma 1 in Appendix, our proposed scheme is a t-resilient encryption based on CDH assumption in the honest classes, showing that the traitors cannot collude to forge a new key outside the corrupted classes. This property gives us an advantage for constructing the tracing algorithm.

In contrast to single-key tracing, we can first go through each class c_i in a key hierarchy Ψ to locate the suspicious classes of the traitors, and then use single-key tracing algorithm to find the actual traitors in every class. In terms of this idea, given an illegal decoder, we present a black-box traitor tracing algorithm based on the key hierarchy, which involves two steps: *subtree searching* and *subset traversing*. We describe in the details of these two algorithms as follows:

V1. *Subtree searching*: Given a key hierarchy Ψ, we start from $c_i \leftarrow c_n$ (the junior-most class) in C and run the following processes from bottom to top:

S1. Randomly selects t new different values $\langle x_1, x_2, \cdots, x_t \rangle$ and constructs an enabling block:

$$\mathcal{C}_i = \langle g^r, ek \cdot g^{rp_i(0)}, (x_k, g^{rp_i(x_k)})_{k=1}^{t}, \{t_{j,l}^r\}_{t_{j,l} \in T_i} \rangle.$$

S2. Sends $\langle \mathcal{C}_i, E(ek, M) \rangle$ to the decoder.

S3. If the decoder can return correctly the message M, we consider c_i as a suspicious class and run V1 by $c_i \leftarrow c_j$ for $\forall c_j \prec_d c_i$, otherwise, repeat V1 by a sibling node of c_i.

V2. *Subset traversing*: Let $\langle c_1', c_2', \cdots, c_k' \rangle$ be the set of suspicious subset by $V1$, for each c_i' in this set, we run the following processes:

T1. Chooses any m user's labels in c_i' at random, $\{x_{i,1}, \cdots, x_{i,m}\}$, $m \leq t$, and then randomly selects $t - m$ unused shares, $\langle v_1, v_2, \cdots, v_{t-m} \rangle$, and constructs an enabling block:

$$\mathcal{C}_i' = \left\langle \begin{array}{c} g^r, ek \cdot g^{rp_i(0)}, (x_{i,j}, g^{rp_i(x_{i,j})})_{j=1}^{m}, \\ (v_k, g^{rp_i(v_k)})_{k=1}^{t-m}, \emptyset \end{array} \right\rangle.$$

T2. Sends $\langle \mathcal{C}_i', E(ek, M) \rangle$ to the pirate decoder.

T3. If the decoder does not output correctly M, we consider the set of label, $\{x_{i,1}, \cdots, x_{i,m}\}$, as a set of traitors and decrease the number of key of this set to run T1. Otherwise, repeats T1 until no more users.

Note that, the decoder can decrypt correctly M if and if only the set of points $\{(x_{i,1}, p_i(x_{i,1})), \cdots, (x_{i,m}, p_i(x_{i,m})\}$ does not cause a collision with the point hidden in the decoder's description key. Otherwise, there must been at least a traitor only if the decoder does not output correctly M in the step T1. In additional, there does not exist any 'unexpected' or forgery value $x_{i,j}$ in the traitor's decryption key only if the total of traitors is less than the threshold t in the same role or class according to equation (10) and (11) in Section 5.

Therefore, our tracing algorithm improves computation complexities and searching times as a result that key hierarchy divides the users into a large number of classes in the key hierarchy. Especially, in the worst case, the complexity of subtree searching is $O(\log n)$ time queries, where n is the number of classes. We also provide more detailed statement of tracing performance in Section 7.3.

6.2 Public Revocation Algorithm

A basic property desired in group-based cryptosystems is that the members should be dynamically joined or revoked in an ef-

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

ficient way. Also, the revocation is an important mechanism for computer forensic to avoid the traitors to further damage the systems. It is obvious that our scheme provides the unlimited number of users dynamically added to a class due to the reason that a new user $sk_{i,j}$ is independent of that of the existing users. Similarly, we provide an efficient approach to revoke the undesired classes and users in our scheme as follows:

The revocation necessarily implies a modification of the keys in a cryptosystem, but we expect that the users can be revoked without modifying the preexisting user decryption keys and the decryption algorithm. Hence, we focus on the modification of public encryption key to realize the revocation in the following algorithm.

Given $\Psi = \langle C, E, K\rangle$ and a revoked set $R_i = \{c_{k_1}, \cdots, c_{k_l}, u_{i_1,j_1}, \cdots, u_{i_m,j_m}\}$, we propose an efficient scheme to revoke all classes and users in R_i, where $\forall c_{k_e} \in \uparrow c_i$ for $e \in [1, m]$ and $|u_{i_k,j_k}| \leq t$. Namely, this scheme can find an efficient encryption key pk_i', which covers the authorized users $\uparrow c_i \setminus R_i$, as follows:

- Revoke classes: We easily realize the revocation of classes by replacing the control domain $\{t_{j,l}^r\}_{c_j,c_l \in \uparrow c_i, c_j \prec_d c_l}$ into a modified control domain $\{t_{j,l}^r\}_{c_j,c_l \in S_i^{R_i}, c_j \prec_d c_l}$ in the Equation (5), where

$$S_i^{R_i} = \{c_j \in \uparrow c_i : c_j \notin R_i', \exists \Gamma(i,j) \subseteq (\uparrow c_i \setminus R_i')\},$$

  and $R_i' = \{c_{k_1}, \cdots, c_{k_l}\} \subseteq R_i$. The $S_i^{R_i}$ denotes all reachable classes from c_i after cutting some nodes R_i' from $\uparrow c_i$ in Ψ. To all appearances, this way does not limit the number of revoked classes.
- Revoke users: Suppose that $R_i'' = \{u_{i_1,j_1}, \cdots, u_{i_m,j_m}\} \subseteq R_i$ is a set of revoked users and $m \leq t$, where all u_{i_k,j_k} may be in different classes. We can revoke these users by their public-labels as follows: Let $lab_{i_k,j_k} = x_{i_k,j_k}$ be the public label of u_{i_k,j_k} for $k = 1, \cdots, m$. To send a message M to the remaining users, instead of the assigned shares $(x_1, g^{p_i(x_1)}), \cdots, (x_t, g^{p_i(x_t)})$ in pk_i, the modified encryption key pk_i' fixes the first k shares as $\{(x_{i_1,j_1}, p_i(x_{i_1,j_1})), \cdots, (x_{i_m,j_m}, p_i(x_{i_m,j_m}))\}$ in terms of mk and remains (or randomly chooses) the rest $t - m$ shares $\{(x_{m+1}, p_i(x_{m+1})), \cdots, (x_t, p_i(x_t))\}$ in order to generate the encryption key

$$pk_i' = \left\langle \begin{array}{c} g, g^{p_i(0)}, (x_{i_k,j_k}, g^{p_i(x_{i_k,j_k})})_{k=1}^m, \\ (x_k, p_i(x_k))_{k=m+1}^t, T_i \end{array} \right\rangle, \quad (13)$$

  where, $p_i(x_{i_k,j_k}) = s_i + \sum_{j=1}^t a_j x_{i_k,j_k}^j \pmod q$ from mk. It is easy to prove that the users in R_i'' can not decrypt the ciphertexts which is encrypted by pk_i', but other users can decrypt them.

  We make easily pk_i' if we hold the master key mk, otherwise, pk_i' can not be computed because $p_i(x_{i_k,j_k})$ is unknown. In order to solve this problem, we propose an efficient public revocation method to compute $(x_{i_k,j_k}, g^{p_i(x_{i_k,j_k})})$ as follows:

  (1) To get $g^{a_1}, \cdots, g^{a_t}$ from $(x_k, g^{p_i(x_k)})_{k=1}^t$ in pk_i: Let $\{(0, s_i), (x_1, y_2), \cdots, (x_t, y_t)\}$ be a group of solutions of equation and $y_k = p_i(x_k)$, such that there exists a unique polynomial $h(x) = s_i + f(x) = s_i + \sum_{k=1}^t a_k x^k$ and $v_k = y_k - s_i = \sum_{j=1}^t a_j (x_k)^j$, where $k \in [1, t]$. These equations can be denoted as the following matrix:

$$M \cdot A = \begin{pmatrix} x_1 & x_1^2 & \cdots & x_1^t \\ x_2 & x_2^2 & \cdots & x_2^t \\ \vdots & \vdots & \cdots & \vdots \\ x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_t \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_t \end{pmatrix}$$

  Then we compute $M^{-1} = (u_{i,j})_{i,j=1}^t$ and have

$$a_k = \sum_{j=1}^t u_{k,j} v_j = \sum_{j=1}^t u_{k,j} y_j - (\sum_{j=1}^t u_{k,j}) s_i. \quad (14)$$

  Such that, for $k = 1, \cdots, t$, we have

$$\begin{aligned} g^{a_k} &= g^{\sum_{j=1}^t u_{k,j} y_j} / g^{s_i \sum_{j=1}^t u_{k,j}} \\ &= \prod_{j=1}^t z_{i,j}^{u_{k,j}} / z_{i,0}^{\sum_{j=1}^t u_{k,j}}, \end{aligned} \quad (15)$$

  where, $z_{i,j}$ is obtained from pk_i for $j = 0, \cdots, t$.

  (2) To compute $(x_{i_k,j_k}, g^{p_i(x_{i_k,j_k})})_{k=1}^m$ from $g^{a_1}, \cdots, g^{a_t}$: we can use a base $z_{i,0} = g^{p_i(0)}$, $\{g^{a_1}, \cdots, g^{a_t}\}$ to compute

$$\begin{aligned} g^{p_i(x_{i_k,j_k})} &= g^{p_i(0)} \cdot g^{\sum_{j=1}^t a_j (x_{i_k,j_k})^j} \\ &= z_{i,0} \cdot \prod_{j=1}^t (g^{a_j})^{(x_{i_k,j_k})^j}. \end{aligned} \quad (16)$$

  Hence, we obtain $(x_{i_k,j_k}, g^{p_i(x_{i_k,j_k})})_{k=1}^m$.

  The remaining process is the same as the original encryption scheme. Hence, this way can revoke at most t users at the same time.

By combining the above two ways, we present a revocation algorithm $Revoke^{\mathcal{E}}(P, pk_i, ek, R_i)$ for c_i to deduce pk_i into a new pk_i' in terms of R_i, as follows: let $R_i = \{c_{k_1}, \cdots, c_{k_l}, u_{i_1,j_1}, \cdots, u_{i_m,j_m}\}$, we have

$$\mathcal{C}_i' = \left\langle \begin{array}{c} g^r, ek \cdot g^{r p_i(0)}, (x_{i_k,j_k}, g^{r p_i(x_{i_k,j_k})})_{k=1}^m \\ (x_k, g^{r p_i(x_k)})_{k=m+1}^t, \{t_{j,l}^r\}_{c_j,c_l \in S_i^{R_i}, c_j \prec_d c_l} \end{array} \right\rangle, \quad (17)$$

where, $S_i^{R_i} = \{c_j \in \uparrow c_i : c_j \notin R_i', \exists \Gamma(i,j) \subseteq (\uparrow c_i \setminus R_i')\}$.

For a permanent revocation, let $\{x_{i_1,j_1}, \cdots, x_{i_m,j_m}\}$ be the set of long-term revoked users with $m \leq t$, we only need to replace $(x_1, g^{f(x_1)}), \cdots, (x_t, g^{f(x_t)}) \in P$ with $(x_k, g^{f(x_k)})_{k=1}^{t-m}, (x_{i_k,j_k}, g^{f(x_{i_k,j_k})})_{k=1}^m$, and to indicate the number m of revoked users in public parameters P. All users can revoke these users by using the modified public parameters P. Note that, the total number of revoked users, for permanent and dynamic revocations, is not greater than t. When the number of permanently revoked users is large enough (e.g. $\geq t/2$), we need to renew all keys in this cryptosystem.

7 Performance Evaluation

We evaluate the performances of the proposed schemes based on the following factors: bandwidth, user's storage, computation costs and the number of keys. For the sake of clarity, we list some major variables as follows: for secure key hierarchy $\Psi = \langle C, E, K\rangle$, let n is the number of classes, m is the average number of users in each class, $\bar{n}$ is the number of partial order relations, l is the number of roots (uppermost classes), and t is the value of threshold. Without loss of generality, we define $h = O(\log n)$ as the approximate height of graph.

Our solution also provides the scalability for practical applications. As a general guideline, the system parameters can be

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

TABLE 2
The parameters under different scales

	Small size	Medium size	Large size
n	10-50	50-100	100-200
t	16-32	32-64	64-128
m	10-20	20-50	50-100
h	1-5	5-7	7-8
size	100-1,000	1,000-5,000	5,000-20,000

chosen to support the scalability with respect to the mumbler of roles, the number of users, the height of role hierarchy, limits on the collusion, etc. Table 2 shows that our PHE scheme can provide the efficient key management for large-scale systems.

7.1 Computation Costs

The basic operations of our scheme include the computation of modular exponentiation and multiplication in $\mathbb{Z}_p^*$ and the algebraic operations in $\mathbb{G}_q$, where $q|p-1$. Thus, we define that t_m is the cost of modular multiplication or division in $\mathbb{Z}_p^*$, t_e is the cost of modular exponentiation in $\mathbb{Z}_p^*$ [8], t'_a is the cost of algebraic operations in $\mathbb{Z}_q$, where algebraic operations involves addition, multiplication, and division. The second column in Table 3 shows the computational costs of various variables, in which $h = O(\log n)$ denotes the approximate length of path of $\prod_{c_{k_1} \prec_d c_{k_2} \in \Delta(i,l)} \bar{t}_{k_1,k_2}$ and $\dot{T}_i$ in Equation (6) and (8), respectively. In conclusion, it is quite clear that our scheme has the lower computation and storage costs.

TABLE 3
Computation and storage costs in the worst case

	Computation costs	Storage costs
pk_0	$((t+1)t_e + 3t^2t'_a)l + \bar{n}t_e$	$(\bar{n}+t+2)l_p + tl_q$
mk	0	$(n+t)l_q$
pk_i	$(h+t)t_m$	0
$sk_{i,j}$	$3t \cdot t'_a$	$2l_q$
$Encrypt$	$(\bar{n}+t+2)t_e + t_m$	$(\bar{n}+t+2)l_p + tl_q$
$Decrypt$	$(h+t+1)\cdot t_m +$ $(t+2)t_e + 2t(t+1)t'_a$	0

7.2 Storage Costs

Storage overload is an important requirement for large-scale information systems. In our scheme, the storage overloads are directly relevant to the size of $\mathbb{Z}_q^*$ and $\mathbb{Z}_p^*$. Hence, we define l_q and l_p as the length of data in $\mathbb{Z}_q^*$ and $\mathbb{Z}_p^*$, respectively. The second column in Table 3 lists the storage overloads of keys and ciphertexts (in Encrypt). Note that, the encryption key pk_i can be computed from P and pk_0, so we don't need to save it. We know that the overload is not static but changes with respect to the size of key hierarchy h, the number of groups n, the number of revoked users t. However, our scheme has a remarkable advantage: constant length of ciphertexts and decryption keys only if the average number of groups/roles $\bar{n}$ in $\Psi = \langle C, E, K \rangle$ and the number of points t in the Lagrange interpolation polynomial are fixed. The third column in Table 3 shows the storage costs of various variables. It is easy to find that the length of variables is related to the size of n, $\bar{n}$ and t. In particular, the overloads are independent on the number of users, such that new users can join dynamically, i.e., without the modification of user decryption keys, ciphertext size and user encryption keys.

7.3 Tracing and Revoking Costs

A preliminary analysis of tracing performance has been introduced for the single-key tracing and the hierarchical tracing in section 6.1. Clearly, the hierarchy-based tracing requires at most $h = O(\log n)$ times subtree searching and at most $\frac{m}{t}$ times subset traversing, such that the whole cost requires $O(h + \frac{m}{t})$ iterations of broadcast encryption. In the worst case, it is $O(h + \binom{t}{m})$ for the case of pirate box with modifying decryption algorithm over multiply traitors' keys. In contrast with a traditional single-key tracing, it requires $O(\frac{nm}{t})$ times encryption computation, where nm is the total number of users. Hence, the hierarchy-based tracing is more efficient than traditional methods.

According to the fast algorithm for the inverse of $t \times t$ Vandermonde matrix [41] with $O(t^2)$ time, the revocation algorithm does not increase the encryption costs. Firstly, the sender must obtain the base set $\{g^{a_1}, \cdots, g^{a_t}\}$ from the public-label of revoked users by Equation (15). Fortunately, we can store these parameters beforehand or make use of the fast Vandermonde inversion algorithm. Secondly, at most t times Equation (16) are computed by using $(t \cdot t_e + (t+1)t_m)$ costs. The remain process is equal to the encryption algorithm. Such that, the total cost of revoking algorithm is approximately $O(t^2)$ modular exponentiations. Hence, the revocation overheads for traitors are acceptable.

7.4 Performance of PHE Cryptosystem

We have implemented our scheme in C++ and experiments were run on a small cloud based on open-source Openstack platform with 6 servers. Based on this cloud platform, a file syncing and sharing service is simulated on the OpenStack Swift that is an object storage system for storing petabytes of data. A simple client-side storage explorer is implemented for cloud storage build on OpenStack Swift. Using GMP and PBC libraries, we have implemented the PHE cryptosystem based on our solution in this paper. We will evaluate the performance of PHE cryptosystem for large-size users through some simple experiments.

One interesting aspect of our PHE scheme is that it provides the scalability for the interoperability among a large number of organizations, as well as the lower communication overloads to a large number of users. We illustrate these properties with an example of secure mailing lists. Suppose the PHE system is distributed globally to a large group of users by using online service. The system maintains a list of users and groups(classes), in which the user public labels $label_{i,j}$ and the encryption keys pk_i are made public via the Web service. Thus, a sender can chooses conveniently some users and groups to decrypt an email by users' public labels and public parameters.

Without loss of generality, we assume p and q are two secure primes with $|q| = 160$ bits and $|p| = 512$ bits, such that $q|p-1$ and $s = \frac{1}{2}|q| = 80$, where s is security parameter [9]. We have $t_q = 20$ bytes and $t_p = 64$ bytes in Table (3). For a system with $10,000$ users, we assume to divide these users into one hundred groups ($n = 100$), each group contains one hundred

8. According to fast exponentiation with pre-computation [40], the computation of modular exponentiation g^n for $n < N$ can be obtained using $O(\log N / \log \log N)$ multiplications.

9. $|q| = 160$ and $|p| = 512 + 64i$, $i \in [0, 8]$ for NIST DSA. Security parameter is dependent on $|q|$ rather than $|p|$.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

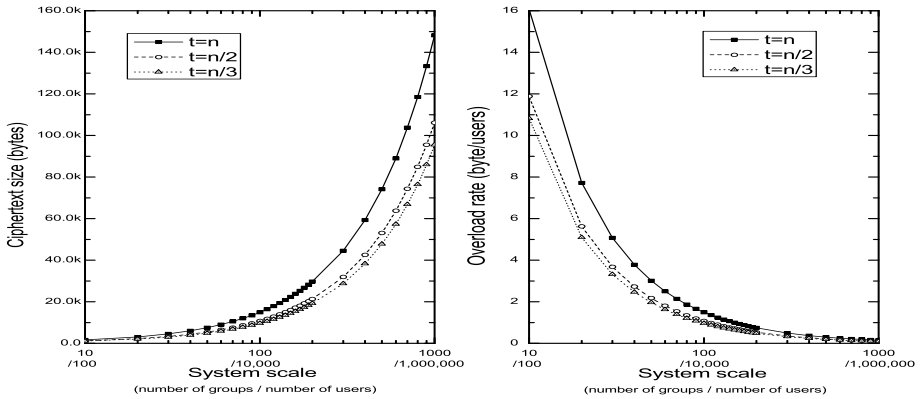


Fig. 7. The ciphertext size (left) and the overload (right) under the different system scales.

users ($m = 50$), and $t = 50$. Then, each of users needs $40 = 2 * 20$ bytes for the private-key. The encryption block, at most $102 * 64 + 50 * 20 = 7.4K$ bytes, is appended into the header of each email. In the left of Fig. 7, we show the change rate of ciphertext size for various revocation thresholds ($t = n, \frac{n}{2}, \frac{n}{3}$) under the different number of users (from 100 to 1,000,000) and the different number of groups (from 10 to 1000), where the number of users in each group is equal to the number of groups. The overload rate ($\frac{ciphertext-size}{User-size}$) is showed in the right of Fig. 7 under the same situations.

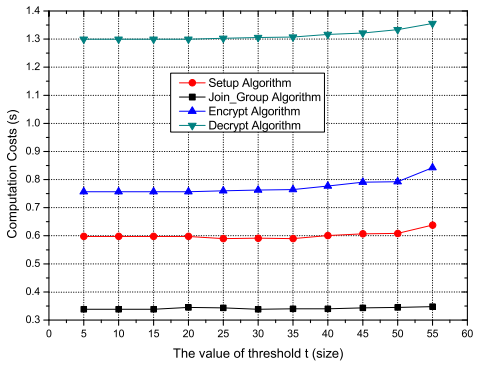


Fig. 8. The performance of our PHE cryptosystem under the different thresholds.

We next provide the result of experiments to illustrate the performance of our PHE cryptosystem. Our experiments were implemented by a small simulated RBAC system with 10 classes and about 10 users per class, which had the same structure as that shown in Fig. 5. At first, we tested the time consumption of each functions in PHE under the different value of thresholds (from 5 to 55). In Fig. 8, we show the result of experiments of four functions, Setup, Join_Group, Encrypt, and Decrypt. The time overheads of Join_User is not being shown here because their values are too small (considering that the operations in $\mathbb{Z}_q$ are too fast). From this figure, it is easy to see that the time consumption of all functions grows large with the increase of threshold value, but the change is not obvious. The reason may be that the algebraic operations held very small proportion of the whole Java code and they can be implemented rapidly. Also, we tested the costs of traitor tracing and revocation on our small RBAC system. The average time of traitor tracing was less than 10 second for finding at least one traitor (for a key hierarchy of depth 5), and the time of revocation was close to that of encryption. These results are the similar to our theoretical analysis.

8 Conclusion

In this paper, we focus on protection the privacy of outsourcing data and preventing player abuse in file syncing and sharing services in the cloud. We highlight the development of a group-oriented cryptosystem with digital forensics, especially for tracing and revoking methods that can ensure the security of player/editor. Based on this cryptosystem, we present a new secure service model to provide a forensic analysis framework to guide investigations. In our future work, we are planning to introduce a comprehensive anomaly detection, using audit, pattern matching, and risk assessment, for identifying the suspected players.

Acknowledgments

The authors are indebted to anonymous reviewers for their valuable suggestions. This work was supported by the National 973 Program (Grant No. 2013CB329601) and the National Natural Science Foundation of China (Grant No. 61472032).

References

[1] F. R. Institute, "Personal data in the cloud: A global survey of consumer attitudes," http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu/personal-data-in-the-cloud.pdf, 2010.

[2] D. Quick and K. R. Choo, "Google drive: Forensic analysis of data remnants," *J. Network and Computer Applications*, vol. 40, pp. 179–193, 2014.

[3] H. Chung, J. Park, S. Lee, and C. Kang, "Digital forensic investigation of cloud storage services," *Digital Investigation*, vol. 9, no. 2, pp. 81–95, 2012.

[4] D. Boneh and M. K. Franklin, "An efficient public key traitor tracing scheme," in *CRYPTO*, 1999, pp. 338–353.

[5] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT*, 2006, pp. 573–592.

[6] Z. Liu, Z. Cao, and D. S. Wong, "Traceable CP-ABE: how to trace decryption devices found in the wild," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 1, pp. 55–68, 2015.

[7] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001, pp. 213–229.

[8] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005, pp. 457–473.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM Conference on CCS*, 2006, pp. 89–98.

[10] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *ACM Conference on Computer and Communications Security*, 2007, pp. 195–203.

[11] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "Generic constructions for chosen-ciphertext secure attribute based encryption," in *Public Key Cryptography*, 2011, pp. 71–89.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

[12] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, 2009.

[13] M. Blanton and K. B. Frikken, "Efficient multi-dimensional key management in broadcast services," in *ESORICS*, 2010, pp. 424–440.

[14] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology (EUROCRYPT'2005)*, vol. 3494 of LNCS, 2005, pp. 440–456.

[15] S. Berkovits, "How to broadcast a secret," in *Advances in Cryptology (EUROCRYPT'91)*, vol. 547 of LNCS. springer-verlag, 1991, pp. 536–541.

[16] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology (CRYPTO'93)*, vol. 773 of LNCS. springer-verlag, 1994, pp. 480–491.

[17] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology (EUROCRYPT'05)*, vol. 3494 of LNCS, http://eprint.iacr.org/2005/015, 2005, pp. 440–456.

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

[19] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," in *ACM Conference on Computer and Communications Security*, 2006, pp. 211–220.

[20] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, 2009, pp. 248–265.

[21] S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters, "Building efficient fully collusion-resilient traitor tracing and revocation schemes," in *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, 2010, pp. 121–130.

[22] S. Akl and P. Taylor, "Cryptographic solution to a multilevel security problem," in *Advances in Cryptology (CRYPTO'82)*, 1982, pp. 237–249.

[23] H. Kim, B. Park1, J. Ha, B. Lee, and D. Park, "New key management systems for multilevel security," in *ICCSA 2005*, vol. 3481 of LNCS, 2005, pp. 245–253.

[24] Y. Chung, H. Lee, F. Lai, and T. Chen, "Access control in user hierarchy based on elliptic curve cryptosystem," *Information Sciences*, vol. 178, pp. 230–243, 2008.

[25] W. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, no. 1, pp. 182–188, 2002.

[26] X. Yi and Y. Ye, "Security of tzeng's time-bound key assignment scheme for access control in a hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 1054–1055, 2003.

[27] H. Chien, "Efficient time-bound hierarchical key assignment scheme," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1301–1304, 2004.

[28] E. Bertino, N. Shang, and S. Wagstaff, "An efficient time-bound hierarchical key management scheme for secure broadcasting," *IEEE Trans. on Dependable and Secure Computing*, vol. 5, no. 2, pp. 65–70, 2008.

[29] A. D. Santis, A. L. Ferrara, and B. Masucci, "Efficient provably-secure hierarchical key assignment schemes," in *MFCS*, 2007, pp. 371–382.

[30] C. D. Peer, D. Engel, and S. B. Wicker, "Hierarchical key management for multi-resolution load data representation," in *2014 IEEE International Conference on Smart Grid Communications, SmartGridComm 2014, Venice, Italy, November 3-6, 2014*, 2014, pp. 926–932.

[31] N. Provos, M. Friedl, and P. Honeyman, "Preventing privilege escalation," in *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003*, 2003.

[32] L. S. Liu, R. Moulic, and D. G. Shea, "Cloud service portal for mobile device management," in *IEEE 7th International Conference on e-Business Engineering, ICEBE 2010, Shanghai, China, November 10-12, 2010*, 2010, pp. 474–478.

[33] Y. Chen and B. Malin, "Detection of anomalous insiders in collaborative environments via relational analysis of access logs," in *Proceedings of the First ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '11. New York, NY, USA: ACM, 2011, pp. 63–74.

[34] Y. Chen, S. Nyemba, and B. Malin, "Detecting anomalous insiders in collaborative information systems," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 3, pp. 332–344, May 2012.

[35] T. Asano, "Reducing receiver's storage in cs, sd and lsd broadcast encryption schemes," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 88, no. 1, pp. 203–210, 2005.

[36] D. Halevy and A. S. A, "The lsd broadcast encryption scheme," in *Advances in Cryptology (Crypto'2002)*, vol. 2442 of LNCS. springer-verlag, 2002, pp. 47–60.

[37] K. Ogawa, G. Hanaoka, and H. Imai, "Content and key management to trace traitors in broadcasting services," in *Security and Trust Management - 11th International Workshop, STM 2015, Vienna, Austria, September 21-22, 2015, Proceedings*, 2015, pp. 236–252.

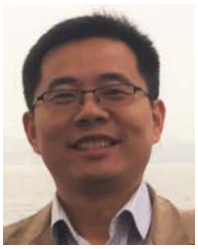
[38] O. Goldreich, *Foundations of Cryptography Volume II, Basic Application*. Cambridge University Press, 2004.

[39] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *Public Key Cryptography*, 2001, pp. 207–224.

[40] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson, "Fast exponentiation with precomputation (extended abstract)," in *EUROCRYPT*, 1992, pp. 200–207.

[41] J. Demmel and P. Koev, "The accurate and efficient solution of a totally positive generalized vandermonde linear system," *SIAM J. Matrix Anal. Appl*, vol. 27, pp. 142–152, 2005.

Yan Zhu is currently a professor in the School of Computer and Communication Engineering at the University of Science and Technology Beijing, China. He was an associate professor at Peking University in China from 2007 to 2012. He was a visiting associate professor in the Arizona State University From 2008 to 2009, and a visiting research investigator of the University of Michigan-Dearborn in 2012. His research interests include cryptography, secure group computation, secure multi-party computing and network security. He is a member of the IEEE.



GuoHua Gan received the MS degree in computer science from Harbin Engineering University, China, in 2005. He is currently a Ph.D Student in the School of Computer and Communication Engineering at the University of Science and Technology Beijing, China. His research interests include cryptography, secure computation, and network security.

Ruiqi Guo received the BS degree from the Taiyuan University of Technology, China, in 2014. She is a Master student in the School of Computer and Communication Engineering at University of Science and Technology Beijing, China from 2014. Her research interests include cryptography, network security and software engineering.

Dijiang Huang received the BS degree from the Beijing University of Posts and Telecommunications, China, in 1995 and the MS and PhD degrees from the University of Missouri-Kansas City in 2001 and 2004, respectively. He is an associate professor at the School of Computing Informatics and Decision System Engineering, Arizona State University. His current research interests include computer networking, security, and privacy. His research is supported by the US National Science Foundation, the US Office of Naval Research (ONR), the US Army Research Office, NATO, and the Consortium of Embedded Systems. He is a recipient of the ONR Young Investigator Program Award. He is a senior member of the IEEE.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

PHE: An Efficient Tracing and Revoking for Traitors in Cloud File Syncing-and-Sharing (Appendix)

Abstract—Recently, many more enterprises have moved their data into the cloud by using file syncing and sharing (FSS) service, but bring-your-own-device (BYOD) policies and greatly increasing mobile devices have in fact raised a new challenge for preventing the player/decoder abuse in the FSS service. In this paper, we address this issue using a new system model with anomaly detection, tracing and revoking traitors. To implement this model, we present a new threshold cryptosystem, called Partially-ordered Hierarchical Encryption (PHE), which implements the partial-order key hierarchy, similar to role hierarchy in Hierarchical RBAC, in public-key infrastructure. This cryptosystem provides two security mechanisms, traitor tracing and revocation, to support efficient digital forensics. The security and performance analysis shows that our construction is threshold provably secure and has following features: dynamic joining and revoking users, constant-size ciphertexts and decryption keys, lower overloads for large-scale systems.

Index Terms—Security, Cloud Storage, Partial Order Key Hierarchy, Traitor Tracing, Revocation, Player Security.

1 Security Model of PHE Scheme

We define the security of PHE scheme in terms of a family of security games between a challenger and an adversary. The partial-order hierarchy Ω and system parameters P are fixed, and the adversary is allowed to depend on them. The users can be divided into two categories: the honest users and the corrupted users, so that a set of corrupted users $\mathcal{R}$ is built. The responsive classes is called as honest classes C_1 or corrupted classes C_2, in which the corrupted users can access all encrypted messages. Sometimes, there exist many honest and corrupted users in the same class. We first define a general model against collusion attacks:

1) Initial: The challenger $\mathcal{B}$ constructs an arbitrary partial-order hierarchy Ω, and then runs $Setup(\Omega, s, t)$ to generate the partial-order key hierarchy Ψ and initial public parameters P, and sends them to the adversary $\mathcal{A}$.
2) Learning: $\mathcal{A}$ adaptively issues n times queries $q_1, \cdots, q_n$ to learn the information of Ψ, where q_i is one of the following:
   - Honest class/user query ($u_{i,j} \notin \mathcal{R}$): using $Join(P, mk, c_i$ or $u_{i,j})$, $\mathcal{B}$ generates a class/user label (pp_i, pk_i, $lab_{i,j}$) and sends $lab_{i,j}$ to $\mathcal{A}$.
   - Corrupted class/user query ($u_{i,j} \in \mathcal{R}$): $\mathcal{B}$ generates a class (pp_i, pk_i) with the corrupted users, or a user label $lab_{i,j}$ and a decryption key $dk_{i,j}$, and returns $(lab_{i,j}, dk_{i,j})$ to $\mathcal{A}$.

   $\mathcal{A}$ ends up with a key hierarchy Ψ (include P, pk_i) and a collusion set $\{sk_{i,j}\}_{u_{i,j} \in \mathcal{R}}$. Note that the decryption query is unnecessary because $\mathcal{A}$ can use the corrupted key to generate it.
3) Challenge: $\mathcal{A}$ chooses two equal length plaintexts M_0, $M_1 \in \mathbb{M}$ and appoints a classes c_i on which it wishes to be challenged. $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and sends the challenge ciphertext $\mathcal{C}_i = Encrypt(P, pk_i, M_b)$ or $Revoke(P, pk_i, M_b, R_i)$ to $\mathcal{A}$. where, $\mathcal{R}_i$ denotes all corrupted users in $\uparrow r_i$.
4) Guess: $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{A}$ wins if $b = b'$, and otherwise it loses.

There are several important variants for this game:

- In a game for chosen plaintext attack (CPA), the adversary $\mathcal{A}$ may not issue the corrupted user queries and decryption queries during the learning phase.
- In a game for user's private key attack, the challenger $\mathcal{B}$ may not issue the challenge ciphertext during the challenger phase. The adversary $\mathcal{A}$ returns a forged private-key in polynomial time during the guess phase.
- In a game for unauthorized access attack, by which user can exceed its authority, we hold the above game. [1]

We denote by $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}(t, n)$ the advantage of adversary $\mathcal{A}$ in winning the game:

$$\begin{aligned}\mathrm{Adv}_{\mathcal{E},\mathcal{A}}(t,n) &= \frac{1}{2}\left|\Pr[\mathcal{A}_{\mathcal{E}}(\mathcal{C}_i) = b] - \Pr[\mathcal{A}_{\mathcal{E}}(\mathcal{C}_i) \neq b]\right| \\ &= \left|\Pr[\mathcal{A}_{\mathcal{E}}(C_i) = b] - \frac{1}{2}\right|\end{aligned}$$

We say that a PHE is (t, n)-secure if for all setup parameter P and all probabilistic polynomial-time adversaries $\mathcal{A}$, the function $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}(t, n)$ is a negligible function of s.

1. This game may be more strict than the other two games.

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

2 Provable Security of PHE Scheme Based on Secure Key Hierarchy

2.1 Proof of semantical security

Proof: Suppose that our scheme is not semantically secure under chosen plaintext attack. For a secure key hierarchy Ψ and the public key $pk_i = \langle g, z_{i,0}, (x_1, z_{i,1}), \cdots, (x_t, z_{i,t})\rangle$, there exists an adversary $\mathcal{A}$ that produces two message $M_0, M_1 \in G_q$. Then this adversary, given the encryption $\mathcal{C} = \mathcal{E}(M_b)$ as one of these messages, can tell with non-negligible advantage ϵ which of the two messages was encrypted, that is, $\Pr(A_{\mathcal{E}}(\mathcal{E}(M_b)) = b) \geq 1/2 + \epsilon$, where $b \in \{0, 1\}$.

We show that such an adversary $\mathcal{A}$ can be constructed a probabilistic polynomial time algorithm $\mathcal{B}$ that can decide DDH with a non-negligible advantage ϵ. Given an input $Query = (g_1, g_2, u_1, u_2)$, we perform the following algorithm $\mathcal{B}(Query)$ to determine whether it is chosen from R or D:

1) Initial: $\mathcal{B}$ chooses random number $a_1, \cdots, a_t \in_R G_q$ and generates a polynomial $f(x) = \sum_{i=1}^{t} a_i x^i$. Set $g = g_1$, and $(i, g_1^{f(i)})$ for $i = 1, \cdots, t$. Secondly, $\mathcal{B}$ constructs any a hierarchy Ω and outputs the public parameters $P = \{p, g, (i, g_1^{f(i)})_{i=1}^{t}\}$.
2) Learning: $\mathcal{B}$ chooses some random integers $r_{j,l}$ for each relation $c_j \prec_d c_l \in \Omega$, and then generates the element $t_{j,l} = g_1^{r_{j,l}}$ of control domain. Let $g = g_1$, $z_{0,0} = g_2$, and $(i, z_{0,i} = g_2 \cdot g_1^{f(i)})$ for $i = 1, \cdots, t$. Let $T_i = \{t_{j,l}\}_{\forall c_j \prec_d c_l, c_j, c_l \in \uparrow c_i}$, $r_i = \sum_{c_j \prec_d c_l \in \Delta(i,0)} r_{j,l} \pmod q$, $\mathcal{B}$ sends the encryption keys

$$\begin{aligned} pk_i &= \langle g, z_{i,0}, (1, z_{i,1}), \cdots, (t, z_{i,t}), T_i\rangle \\ &= \langle g_1, g_2 \cdot g_1^{r_i}, (i, g_2 \cdot g_1^{r_i + f(i)})_{i=1}^{t}, T_i\rangle \end{aligned}$$

to the adversary $\mathcal{A}$. And then $\mathcal{A}$ may use it to generate any number of ciphertexts for any $c_i \in C$, but $\mathcal{B}$ does not provide the decryption query. Finally, the adversary returns $M_0, M_1 \in G_q$.
3) Challenge: $\mathcal{B}$ picks a random $b \in \{0, 1\}$ and $i \in [0, n]$, let $r_i = \sum_{c_j \prec_d c_l \in \Delta(i,0)} r_{j,l}$ and constructs the ciphertext

$$\mathcal{C}_i = \left\langle \begin{array}{c} u_1, M_b \cdot u_2 \cdot u_1^{r_i}, (1, u_2 \cdot u_1^{r_i + f(1)}), \cdots, \\ (t, u_2 \cdot u_1^{r_i + f(t)}), \{u_1^{r_{j,l}}\}_{c_j, c_l \in \uparrow c_i, c_j \prec_d c_l} \end{array} \right\rangle, \quad (1)$$

then sends $\mathcal{C}_i$ to the adversary $\mathcal{A}$.
4) Output: The adversary returns $b' \in \{0, 1\}$ as the guess. If $b = b'$, $\mathcal{B}$ outputs 1 (denoted D), otherwise outputs 0 (denoted R).

Let $v = \log_{g_1} g_2$, such that $p_0(x) = v + f(x)$ and $p_i(x) = v + \sum_{c_j \prec_d c_l \in \Delta(i,0)} r_{j,l} + f(x)$. If the tuple $\langle g_1, g_2, u_1, u_2\rangle$ is chosen from D, we have $v = log_{u_1} u_2$, $z_{i,0} = u_2 \cdot u_1^r = u_1^{v+r}$,$z_{i,j} = u_2 \cdot u_1^{r+f(j)} = u_1^{p_i(j)} (1 \leq j \leq t)$, and then the ciphertext $\mathcal{C}_i$ is an encryption of M_b. Thus,

$$\begin{aligned} &\Pr[\mathcal{B}(g_1, g_2, u_1, u_2) = 1 | (g_1, g_2, u_1, u_2) \in D] \\ &= \Pr(b = b' | \mathcal{C}_i \in D) \\ &= \Pr(\mathcal{A}_{\mathcal{E}}(\mathcal{C}_i) = b) \geq 1/2 + \epsilon. \end{aligned} \quad (2)$$

If the tuple is from R, then the $\mathcal{C}_i$ is the encryption of $M_b \cdot u_2/u_1^v$ and the ciphertext is the encryption of a random message. Hence,

$$\begin{aligned} &\Pr[\mathcal{B}(g_1, g_2, u_1, u_2) = 1 | (g_1, g_2, u_1, u_2) \in R] \\ &= \Pr(b = b' | \mathcal{C}_i \in R) = 1/2. \end{aligned} \quad (3)$$

This implies the algorithm B can decide DDH with a non-negligible success probability $\epsilon/2$,

$$\begin{aligned} &\Pr[\mathcal{B}(Query) = 1 : Query = (g_1, g_2, u_1, u_2)] \\ &= \Pr[\mathcal{B}(Query) = 1 | Query \in D] \Pr[Query \in D] \\ &\quad + \Pr[\mathcal{B}(Query) = 1 | Query \in R] \Pr[Query \in R] \\ &\geq \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon}{2}. \end{aligned} \quad (4)$$

which would contradict the assumption. □

2.2 Proof of Privilege attack 1

Lemma 1: [Privilege attack for honest classes] The proposed (t, n)-PHE scheme can assure that the adversary cannot change the privilege in key hierarchy by forging a key of honest classes under Computing Discrete Logarithm (CDL) assumption, even if there exist more than t-collusion in key hierarchy.

Proof: The theorem is proved by the reduction of absurdity. We formally describe the theorem as follows: Suppose that an adversary $\mathcal{A}$ can guess a new private key $sk_{i,j}$ from the encryption key $pk = P \cup \{pk_i\}$ and at most t secret-keys $\{sk_{i_1,j_1}, \cdots, sk_{i_m,j_m}\}$ for every polynomial $p(\cdot)$, and all sufficiently large n,

$$\Pr\left[\begin{array}{c} A(pk, \{sk_{i_l,j_l}\}_{l=1}^{m}) = sk_{i,j} \\ : i \notin \{i_k\}_{k=1}^{t}, sk_{i,j'} \notin \{sk_{i_k,j_k}\}_{k=1}^{m} \end{array} \right] \geq \frac{1}{|p(n)|}. \quad (5)$$

If $\tilde{g}$ be a generator of G_q and $y = \tilde{g}^x \pmod p$, then we construct an algorithm $\mathcal{B}$ which can use the adversary $\mathcal{A}$ to compute $x = \log_{\tilde{g}} y$, as follows:

1) Initial: The algorithm $\mathcal{B}$ chooses random number $a_1, \cdots, a_t \in_R G_q$ and generates a polynomial $f(x) = \sum_{i=1}^{t} a_i x^i \pmod q$. Set $g = \tilde{g}$ and $(i, \tilde{g}^{f(i)})$ for $i = 1, \cdots, t$. Then, $\mathcal{B}$ generates any a key hierarchy $\Omega = \langle C, E\rangle$ and some random integers $r_i \in \mathbb{Z}_q$ for $c_i \in C$, where $i \in [0, n]$ and n is the number of classes in C. Thus, the public parameters is $P = \{p, q, (i, \tilde{g}^{f(i)})_{i=1}^{t}\}$.
2) Learning: The adversary $\mathcal{A}$ issues m times $Join$ queries, but before each invocation $\mathcal{A}$ specifies whether the corresponding user is honest or corrupted, such that all classes are divided into two categories: the honest classes C_1 and the corrupted classes C_2. $\mathcal{B}$ responds the queries as follows:

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

- Honest user query ($u_{i,j} \notin \mathcal{R}$): B sets $C_1 = C_1 \cup \{c_i\}$ and $z_{i,0} = y^{r_i} \pmod p$ in pk_i, and then chooses a unused random integers $x_{i,j}$ as $lab_{i,j}$ for $\mathcal{A}$, but $\mathcal{B}$ does not assign the $dk_{i,j}$ for $\mathcal{A}$. In fact, $\mathcal{B}$ cannot compute $dk_{i,j}$.
- Corrupted user query ($u_{i,j} \in \mathcal{R}$): B sets $C_2 = C_2 \cup \{c_i\}$ and $z_{i,0} = \tilde{g}^{r_i} \pmod p$ in pk_i. $\mathcal{B}$ assigns $sk_{i,j} = \langle x_{i,j}, r_i + f(x_{i,j})\rangle$ for $\mathcal{A}$, where, $x_{i,j}$ is also an unused random integers.

After all classes are queried, $\mathcal{B}$ defines for $\forall c_i \in C$,

$$z_{i,0} = \begin{cases} \tilde{g}^{r_i} & c_i \in C_1 \\ y^{r_i} & c_i \in C_2 \end{cases}, \tag{6}$$

and $t_{j,l} = z_{j,0} \backslash z_{l,0}$ and $c_j \prec_d c_l$. Thus, the public parameter is $P = \{p, q, (i, \tilde{g}^{f(i)})_{i=1}^t, \{t_{j,l}\}_{\forall c_j \prec_d c_l}\}$. B assigns the main public-key for A as follows:

$$\begin{aligned} pk_i &= \langle g, z_{i,0}, (1, z_{i,1}), \cdots, (t, z_{i,t}), T_i\rangle \\ &= \begin{cases} \langle \tilde{g}, \tilde{g}^{r_i}, (k, \tilde{g}^{r_i + f(k)})_{k=1}^t, T_i\rangle, & c_i \in C_1 \\ \langle \tilde{g}, y^{r_i}, (k, y^{r_i}\tilde{g}^{f(k)})_{k=1}^t, T_i\rangle, & c_i \in C_2 \end{cases} \end{aligned}$$

where, $T_i = \{t_{j,l}\}_{c_j, c_l \in \uparrow c_i, c_j \prec_d c_l}$. Note that the adversary $\mathcal{A}$ can obtain any ciphertexts itself after it get pk_i and decrypt it by $sk_{i,j}$.

3) Challenge: $\mathcal{B}$ requires that the adversary $\mathcal{A}$ outputs a private key in $c_{i'} \in C_1$, then $\mathcal{A}$ returns a new key $sk_{i',j'} = (v, u)$ with non-negligible probability in a polynomial-time.
4) Output: $\mathcal{B}$ firstly verifies $U_{pk_i}(sk_{i',j'}) = z_{i',0}$ according to Equation (6) in Section 3. If it holds, $\mathcal{B}$ computes $f(v)$ and outputs

$$x = \frac{u - f(v)}{r_i} = \frac{u - \sum_{k=1}^t a_k v^k}{r_i} \pmod q. \tag{7}$$

Otherwise, $\mathcal{B}$ repeats the above process from step (3).

According to the above construction, we have $p_i(x) = r_i \cdot \lambda_i + f(x)$, where λ_i is a character function

$$\lambda_i = \begin{cases} 1 & c_i \in C_1 \\ x & c_i \in C_2 \end{cases}. \tag{8}$$

$\mathcal{B}$ can compute the decryption key $dk_{i,j}$ in C_1, But $\mathcal{B}$ cannot compute the decryption key $dk_{i,j}$ in the corrupted classes C_2 since x is unknown. Furthermore, if $\mathcal{A}$ returns $sk_{i',y} = (v, u)$, then $p_i(v) = r_i \cdot \lambda_i + f(v) = r_i \cdot x + f(v) = u$, so that we have Equation (7). Thus, according to the assumption that $\mathcal{A}$ can find a private key in the honest classes, $\mathcal{B}$ can compute the discrete logarithm in polynomial-time with the help of $\mathcal{A}$, namely,

$$\begin{aligned} &\Pr[\mathcal{B}(\tilde{g}, y = \tilde{g}^x) = x] \\ &= \Pr\left[\begin{array}{c} \mathcal{A}(pk, \{sk_{i_l,j_l}\}_{l=1}^m) = sk_{i',j'} \\ : i \in \{i_k\}_{k=1}^t, sk_{i',j'} \notin \{sk_{i_k,j_k}\}_{k=1}^m \end{array}\right] \geq \frac{1}{|p(n)|}. \end{aligned}$$

but we know that it is infeasible to compute discrete logarithm problem in polynomial-time: $\Pr[\mathcal{B}(\tilde{g}, y = \tilde{g}^x) = x] < \frac{1}{|p(n)|}$. Consequently, the contradiction leads to the fact that the attack is infeasible. The proof of theorem is completed. □

2.3 Proof of Privilege attack 2

Lemma 2: [Privilege attack for corrupted classes] The proposed (t, n)-PHE scheme can assure that the adversary cannot avoid tracing or frame the innocent users by forging a key of corrupted classes under Computing Discrete Logarithm (CDL) assumption, even if there exist at most t-collusion in key hierarchy.

Proof: Let $\tilde{g}$ be a generator of G_q and a random number $y = \tilde{g}^s \pmod p$, where the secret s is unknown. Suppose that an adversary $\mathcal{A}$ can forge a new private-key $sk_{i,j'}$ from the public-key and at most t private-keys $\{sk_{i,j}\}_{j=1}^t$ with non-negligible advantage $\frac{1}{|p(n)|}$, that is,

$$\Pr\left[\begin{array}{c} \mathcal{A}(pk, \{sk_{i_k,j_k}\}_{k=1}^t) = sk_{i',j'} \\ : i \in \{i_k\}_{k=1}^t, sk_{i',j'} \notin \{sk_{i_k,j_k}\}_{k=1}^t \end{array}\right] < \frac{1}{|p(n)|}. \tag{9}$$

Then we shall show how to construct an algorithm $\mathcal{B}$ to compute $s = \log_{\tilde{g}} y$ by the adversary $\mathcal{A}$ as follows:

1) Initial: The algorithm $\mathcal{B}$ generates any a hierarchy $\Omega = \langle C, E\rangle$ and some random integers $r_i \in \mathbb{Z}_q$ for $c_i \in C$, where $i \in [0, n]$ and n is the number of classes in C. Let $z_{i,0} = y^{r_i} = \tilde{g}^{sr_i} \pmod p$ for $\forall c_i \in C$, $\mathcal{B}$ computes and storages $t_{j,l} = \frac{z_{j,0}}{z_{l,0}} = y^{r_j - r_l} \pmod p$ for all relations $c_j \prec_d c_l$.
2) Learning: The adversary $\mathcal{A}$ issues m times $Join$ queries, but before each invocation $\mathcal{A}$ specifies whether the corresponding user is honest or corrupted. Then, $\mathcal{B}$ divides all classes into two categories: the honest classes C_1 and the corrupted classes C_2, where the number $|C_2|$ of the corrupted classes is not great than t, i.e., $|C_2| \leq t$. $\mathcal{B}$ responds the queries as follows:
   - Honest user query ($u_{i,j} \notin \mathcal{R}$): $\mathcal{B}$ chooses a unused random integers $x_{i,j}$ as $lab_{i,j}$ for $\mathcal{A}$, but $\mathcal{B}$ does not assign the $dk_{i,j}$ for $\mathcal{A}$.
   - Corrupted user query ($u_{i,j} \in \mathcal{R}$): $\mathcal{B}$ assigns two random integers v_i, u_i as $sk_{i,j} = \langle x_{i,j}, dk_{i,j}\rangle = \langle v_i, u_i\rangle$, where, v_i is also an unused random integers. Note that, $\mathcal{B}$ can assign at most t user keys for $\mathcal{A}$ in all corrupted classes.

Let $\{(v_1, u_1, r_{i_1}), (v_2, u_2, r_{i_2}), \cdots, (v_t, u_t, r_{i_t})\}$ be a group of solutions of equation and r_{i_k} corresponds to $c_{i_k} \in C_2$, such that there exists a unique polynomial $h_{i_k}(v_k) = r_{i_k} \cdot s + f(v_k) = r_{i_k} \cdot s + \sum_{i=1}^t a_i v_k^i$, where $k = [1, t]$. These equations can be denoted as the following matrix:

$$\begin{aligned} M \cdot A &= \begin{pmatrix} v_1 & v_1^2 & \cdots & v_1^t \\ v_2 & v_2^2 & \cdots & v_2^t \\ \vdots & \vdots & \vdots & \vdots \\ v_t & v_t^2 & \cdots & v_t^t \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_t \end{pmatrix} \\ &= \begin{pmatrix} u_1 - r_{i_1} \cdot s \\ u_2 - r_{i_2} \cdot s \\ \vdots \\ u_t - r_{i_t} \cdot s \end{pmatrix} \end{aligned} \tag{10}$$

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

Then B computes $M^{-1} = (z_{i,j})_{i,j=1}^{t}$ and has

$$\begin{aligned} a_i &= \sum_{k=1}^{t} z_{i,k}(u_k - r_{i_k} \cdot s) \\ &= \sum_{k=1}^{t} z_{i,k}u_k - (\sum_{k=1}^{t} z_{i,k}r_{i_k})s. \end{aligned} \quad (11)$$

Hence,

$$\tilde{g}^{a_i} = \frac{\tilde{g}^{\sum_{k=1}^{t} z_{i,k}u_k}}{y^{\sum_{k=1}^{t} z_{i,k}r_{i,k}}} = \prod_{k=1}^{t}(\frac{\tilde{g}^{u_k}}{y^{r_{i,k}}})^{z_{i,k}}. \quad (12)$$

Finally, $\mathcal{B}$ is able to obtain a base $\{\tilde{g}^{a_1}, \cdots, \tilde{g}^{a_t}\}$ to compute $\tilde{g}^{f(x_i)} = \prod_{j=1}^{t}(\tilde{g}^{a_j})^{x_i^j} \pmod p$, where x_i is a unused integer and $i = [1, t]$. Thus, the public parameters are defined as $P = \{p, q, \tilde{g}, (x_i, \tilde{g}^{f(x_i)})_{i=1}^{t}, \{t_{j,l}\}_{\forall c_j \prec_d c_l}\}$. The encryption key is

$$pk_i = \{\tilde{g}, y^{r_i}, (x_k, y^{r_i} \cdot \tilde{g}^{f(x_k)})_{k=1}^{t}, \{t_{j,l}\}_{\forall c_j \prec_d c_l}\},$$

where, $\tilde{g}^{h_i(x_k)} = y^{r_i} \cdot \prod_{j=1}^{t}(\tilde{g}^{a_j})^{x_k^j} \pmod p$.

3) Challenge: $\mathcal{B}$ requires that the adversary $\mathcal{A}$ outputs a new private key in C_2, then $\mathcal{A}$ returns a key $sk_{i',j'} = (v, w)$ with non-negligible probability in a polynomial-time, where $c_{i'} \in C_2$ and $v \neq v_i$ in $\{v_1, v_2, \cdots, v_t\}$.
4) Output: $\mathcal{B}$ checks the validation of (v, w) by using $\tilde{g}^{h(v)} = y^{r_{i'}} \cdot \prod_{j=1}^{t}(\tilde{g}^{a_j})^{v^j} = \tilde{g}^{w}$. If it holds, $\mathcal{B}$ sets $(v, w, r_{i'}), (v_1, u_1, r_{i_1}), \cdots, (v_t, u_t, r_{i_t})$ as the following equations:

$$M' \cdot A' = \begin{pmatrix} r_{i'} & v & \cdots & v^t \\ r_{i_1} & v_1 & \cdots & v_1^t \\ \vdots & \vdots & \cdots & \vdots \\ r_{i_t} & v_t & \vdots & v_t^t \end{pmatrix} \cdot \begin{pmatrix} s \\ a_1 \\ \vdots \\ a_t \end{pmatrix} = \begin{pmatrix} w \\ u_1 \\ \vdots \\ u_t \end{pmatrix}$$

It is not hard to prove that M is a matrix of order $t+1$, such that $\mathcal{B}$ can compute s as the final result by using $M^{-1} = \{z'_{i,j}\}_{i,j=1}^{t}$, i.e., $s = z'_{0,0}w + \sum_{k=1}^{t} z'_{0,k}u_k \pmod q$.

However, $s = \log_{\tilde{g}} y$ is the discrete logarithm of y. Consequently, the contradiction leads to the fact that

$$\begin{aligned} &\Pr[\mathcal{B}(\tilde{g}, y = \tilde{g}^s) = s] \\ &= \Pr[\mathcal{A}(P/pk_i, \{(v_k, u_k)\}_{k=1}^{t}) = (v, w)] \\ &= \Pr\left[\begin{array}{c} \mathcal{A}(pk, \{sk_{i_k,j_k}\}_{k=1}^{t}) = sk_{i',j'} \\ : i \in \{i_k\}_{k=1}^{t}, sk_{i',j'} \notin \{sk_{i_k,j_k}\}_{k=1}^{t} \end{array}\right] < \frac{1}{|p(n)|}. \end{aligned}$$

holds. Thus, the theorem follows. □

2.4 Summary of Privilege attack

Theorem 1: The proposed (t, n)-PHE scheme can assure that anyone cannot forge a new key in key hierarchy by forging key under the Computing Discrete Logarithm (CDL) assumption, even if there exist at most t-collusion in key hierarchy.

Proof: This theorem includes two aspects: the forging key in the different subsets and one in the same subset. According to Lemma (1,2), we have this following equation for the collusion set P, which includes at most t collusion in each $c_i \in C$,

$$\Pr\left[\begin{array}{c} \mathcal{A}(pk, \{sk_{i_k,j_k}\}_{k=1}^{t}) = sk_{i',j'} \\ : sk_{i',j'} \notin \{sk_{i_k,j_k}\}_{k=1}^{t} \end{array}\right] < \frac{1}{|p(n)|}.$$

where, $pk = \{pk_i\}_{c_i \in C}$. Thus, this theorem holds. □

2.5 Proof of Access attack

Lemma 3: [Access attack] The proposed (t, n)-PHE scheme can assure that the adversary cannot get the additional advantage to access the secret of ciphertexts, encrypted by revocation algorithm, under Decisional Diffie-Hellman (DDH) assumption and at most t-collusion in key hierarchy.

Proof: Suppose that, for a key hierarchy Ψ, the encryption key $pk = \{pk_i\}_{\forall c_i \in C}$, and the user's keys $sk_{i,j}$ in a collusion set R with at most t collusion, there exists an adversary $\mathcal{A}$ that produces two message $M_0, M_1 \in G_q$. Then $\mathcal{A}$, given the encryption $\mathcal{C} = \mathcal{E}^R(M_b)$ as one of these messages, can tell with non-negligible advantage ϵ, which of the two messages it was given, that is, $\Pr(\mathcal{A}_{\mathcal{E}}(\mathcal{C}) = b) > 1/2 + \epsilon$, i.e., $\mathcal{A}$ has the advantage of

$$\left|\begin{array}{c} \Pr[\mathcal{A}(pk, \{sk_{i,j}\}_{u_{i,j} \in P}, \mathcal{E}_{pk_l}^R(M_0)) = 1] - \\ \Pr[\mathcal{A}(pk, \{sk_{i,j}\}_{u_{i,j} \in P}, \mathcal{E}_{pk_l}^R(M_1)) = 1] \end{array}\right| > 2\epsilon. \quad (13)$$

We show that such an adversary $\mathcal{A}$ can be used to construct a probabilistic polynomial time algorithm $\mathcal{B}$ that can decide DDH with a non-negligible advantage ϵ. Given an input $query = \langle g_1, g_2, u_1, u_2 \rangle$, we perform the following algorithm $\mathcal{B}$ to determine if it is chosen from R or D:

1) Initial: $\mathcal{B}$ generates a key hierarchy $\Omega = \langle C, E \rangle$ and some random integer $r_i \in \mathbb{Z}_q$ for $c_i \in C$, where $i \in [0, n]$ and $n = |C|$. Secondly, according to Lemma (2), $\mathcal{B}$ chooses t number pairs $(x_1, y_1), \cdots, (x_t, y_t)$ at random. Let $v = \log_{g_1} g_2$, but it is unknown for $\mathcal{B}$. $\mathcal{B}$ solves the equations $\{(0, g_2 = g_1^v), (x_1, g_1^{y_1}) \cdots, (x_t, g_1^{y_t})\}$ to get a base of polynomial, $\{g_1^{a_1}, \cdots, g_1^{a_t}\}$, where $a_1, \cdots, a_t \in_R G_q$ is the coefficients of polynomial $h(x) = v + f(x) = v + \sum_{j=1}^{t} a_j \cdot x^j$ and

$$g_1^{y_k} = g_1^{h(x_k)} = g_1^{v} \cdot g_1^{\sum_{j=1}^{t} a_j \cdot x_k^j}, \quad (14)$$

as follows: Let $M \cdot A =$

$$\begin{pmatrix} x_1 & x_1^2 & \cdots & x_1^t \\ x_2 & x_2^2 & \cdots & x_2^t \\ \vdots & \vdots & \cdots & \vdots \\ x_t & x_t^2 & \cdots & x_t^t \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_t \end{pmatrix} = \begin{pmatrix} y_1 - v \\ y_2 - v \\ \vdots \\ y_t - v \end{pmatrix},$$

then we compute $M^{-1} = (u_{i,j})_{i,j=1}^{t}$ and have

$$a_k = \sum_{j=1}^{t} u_{k,j}(y_j - v) = \sum_{j=1}^{t} u_{k,j}y_j - (\sum_{j=1}^{t} u_{k,j})v. \quad (15)$$

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/TCC.2016.2573299, IEEE Transactions on Cloud Computing

So, for $k = 1, \cdots, t$, we have

$$g_1^{a_k} = g_1^{\sum_{j=1}^{t} u_{k,j} y_j} / g_2^{\sum_{j=1}^{t} u_{k,j}}. \quad (16)$$

Let $g_1^{f(x)} = \prod_{i=1}^{t} (g_1^{a_i})^{x^i} \pmod p$, $\mathcal{B}$ inputs the public parameters $P = \{p, q, (x'_k, g_1^{f(x'_k)})_{k=1}^{t}\}$, where x'_k is unused random integer for $k = [1, t]$.

2) Learning: Let $z_{i,0} = g_1^{r_i} g_2$, $\mathcal{B}$ computes and stores $t_{j,l} = \frac{z_{j,0}}{z_{l,0}} = g^{r_j - r_l}$ for all relations $c_j \prec_d c_l$. $\mathcal{B}$ assigns the encryption key for $\mathcal{A}$ in terms of $\{g_1^{a_1}, \cdots, g_1^{a_t}\}$ as follows:

$$\begin{aligned} pk_i &= \langle g, z_{i,0}, (x'_1, z_{i,1}), \cdots, (x'_t, z_{i,t}), T_i \rangle \\ &= \left\langle \begin{array}{c} g_1, g_2 \cdot g_1^{r_i}, (x'_1, g_2 \cdot g_1^{r_i} \cdot g_1^{f(x'_1)}), \cdots, \\ (x'_t, g_2 \cdot g_1^{r_i} \cdot g_1^{f(x'_t)}), T_i \end{array} \right\rangle, \end{aligned}$$

where, $T_i = \{t_{j,l}\}_{c_j, c_l \in \uparrow c_i, c_j \prec_d c_l}$. $\mathcal{A}$ runs m times the queries of user's key, in which $\mathcal{A}$ specifies whether the corresponding user is honest or corrupted. Following the i-th query, $\mathcal{B}$ operates as follows:

- Honest user query ($u_{i,j} \notin R$): $\mathcal{B}$ assigns a random integer x''_i as the label of user, i.e., $label_{i,j} = x''_i$. But $\mathcal{B}$ does not assign the user's decryption key $dk_{i,j}$ for $\mathcal{A}$.
- Corrupted user query ($u_{i,j} \in R$): $\mathcal{B}$ chooses a pair (x_i, y_i) from $(x_1, y_1), \cdots, (x_t, y_t)$, and then $\mathcal{B}$ assigns $sk_{i,j} = \langle x_k, r_i + y_k \rangle$ for $\mathcal{A}$, where (x_i, y_i) is generated in initial stage. Note that, $sk_{i,j}$ can be in arbitrary c_i, moreover, $\mathcal{B}$ can assign at most t user keys for $\mathcal{A}$.

Note that the adversary $\mathcal{A}$ may use the encryption key pk_i to generate any number of ciphertexts for an arbitrary $c_i \in C$. Further, $\mathcal{A}$ decrypts them if there are the corrupted keys $sk_{j,l}$ with $c_j \in \uparrow c_i$. Finally, the adversary returns $M_0, M_1 \in G_q$.

3) Challenge: $\mathcal{B}$ picks a random $b \in \{0, 1\}$ and a class $c_i \in C$, then constructs the following ciphertext: according to Equation (15), $\mathcal{B}$ solves the equations $\{(0, u_2 = u_1^v), (x_1, u_1^{y_1}) \cdots, (x_t, u_1^{y_t})\}$ to get the set $\{u_1^{a_1}, \cdots, u_1^{a_t}\}$, where

$$u_1^{a_k} = u_1^{\sum_{j=1}^{t} u_{k,j} y_j} / u_2^{\sum_{j=1}^{t} u_{k,j}}, \quad (17)$$

for $k \in [1, t]$. So, let $u_1^{f(x)} = \prod_{i=1}^{t} (u_1^{a_i})^{x^i}$, and outputs the ciphertext

$$\begin{aligned} \mathcal{C}_i &= \left\langle \begin{array}{c} u_1, M_b \cdot u_2 \cdot u_1^{r_i}, (x_1, u_2 \cdot u_1^{r_i} \cdot u_1^{f(x_1)}), \\ \cdots, (x_t, u_2 \cdot u_1^{r_i} \cdot u_1^{f(x_t)}), \{\hat{t}_{l,j}\} \end{array} \right\rangle. \\ \hat{t}_{l,j} &= u_1^{(r_l - r_j)}, \forall c_l, c_j \in \uparrow c_i, c_l \prec_d c_j, \end{aligned}$$

then sends $\mathcal{C}_i$ to the adversary $\mathcal{A}$.

4) Output: $\mathcal{A}$ returns $b' \in \{0, 1\}$ as the guess. If $b = b'$ output 1 (D), otherwise output 0 (R).

Let $v = \log_{g_1} g_2$, such that $p_i(x) = r_i + v + f(x)$. We can compute the corrupted key $sk_{i,j} = \langle x_i, r_i + y_i \rangle$ for $\mathcal{A}$. If the tuple $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from D, we have $v = log_{u_1} u_2$, $z_{i,0} = u_2 \cdot u_1^{r_i} = u_1^{r_i + v}$, $z_{i,j} = u_2 \cdot u_1^{r_i} \cdot u_1^{h(x_j)} = u_1^{p_i(x_j)}$ for any $1 \le i \le t$, and then the ciphertext $\mathcal{C}_i$ is an available ciphertext of M_b. Thus, we have $\Pr[\mathcal{B}(Query) = 1 | Query \in D] = \Pr[\mathcal{A}_{\mathcal{E}}(\mathcal{E}(M_b)) = b] > 1/2 + \epsilon$. If the tuple is from R, then the $\mathcal{C}_i$ is the encryption of $M_b \cdot u_2 / u_1^v$, so that the ciphertext is considered as the encryption of a random message. Hence, $\Pr[\mathcal{B}(Query) = 1 | Query \in R] = \Pr(b = b' | \mathcal{C}_i \in R) = 1/2$.

This implies the algorithm $\mathcal{B}$ can decide DDH with a non-negligible success probability $\epsilon/2$:

$$\begin{aligned} &\Pr[\mathcal{B}(Query) = 1 : Query = (g_1, g_2, u_1, u_2)] \\ &= \Pr[\mathcal{B}(Query) = 1 | Query \in D] \Pr[Query \in D] \\ &\quad + \Pr[\mathcal{B}(Query) = 1 | Query \in R] \Pr[Query \in R] \\ &= \frac{1}{2} \cdot \Pr[\mathcal{A}_{\mathcal{E}}(\mathcal{E}(M_b)) = b] + \frac{1}{2} \cdot \frac{1}{2} \ge \frac{1}{2} + \frac{\epsilon}{2}. \end{aligned}$$

which would contradict assumption. □