P2E: Privacy-preserving and Effective Cloud Data Sharing Service

Xin Dong[†], Jiadi Yu[†], Yuan Luo[†], Yingying Chen[‡], Guangtao Xue[†], and Minglu Li[†]
[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, P.R.China
Email: {xindong, jiadiyu, yuanluo, gt_xue, mlli}@sjtu.edu.cn
[‡]Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, USA
Email: yingying.chen@stevens.edu

***Abstract*—Data sharing in the cloud, fueled by favorable cloud technology trends, has emerging as a promising pattern in regard to enabling data more accessible to users in a convenient manner. To achieve data sharing, enterprises and customers in increasing numbers keep their data stored into cloud server. In this paper, we focus on seeking a solution that allows secure and effective access to the cloud data. We propose an effective and flexible privacy-preserving data policy, P2E, utilizing ciphertext policy attribute-based encryption (CP-ABE) and combining it with technique of identity-based encryption (IBE). In addition to ensuring strong data sharing security, the policy succeeds in preserving the privacy of cloud users. Security analysis indicates that the proposed policy is security and enforces fine-grained access control and full collusion resistance simultaneously. Furthermore, our performance analysis and experimental results show that P2E is as light as possible.**

I. Introduction

Cloud computing [1] represents one of the most currently emerging technologies, in which a cloud service provider (CSP) offers efficient data computing and storage to a global client base. Storing data into the cloud offers great convenience in the sense that users do not need care about the direct deployment and management of hardware infrastructure. As promising as it is, cloud computing is much more powerful than personal computing, but it brings new security challenges to users' data. Since users no longer have physical possession of their outsourced data, data outsourcing is relinquishing users' control over their data. As a result, the privacy of users and the security of data face various threats.

Therefore, the data owner requires high security and confidentiality of the data when outsourcing it in the cloud. However, traditional cryptographic primitives can not be directly employed to achieve data security. Recently, there has been a plethora of work on privacy and security in the content of ensuring sharing of remotely stored data under different systems and security models [2], [3], [4]. Those works mainly focus on how to preserve the user's privacy and realize the desired security goal without bringing a high complexity on the user decrypted stage. To solve this issue, researchers either utilize key-policy attribute-based encryption (KP-ABE) for secure access control, or employ hierarchical identity-based encryption (HIBE) for data security. The KP-ABE-based schemes [2], [3], however, reveal some users access attributes to the cloud, and then these can not fully preserve the user's privacy and are also not fully collusion resistant. On the other hand, the HIBE-based schemes [4], introduce too many keys (each user has a mass of keys) and cannot manage efficiently. Therefore, the challenge to achieve goals of both privacy-preserving and effective cloud data sharing service still remains open.

To realize an effective and privacy-preserving data sharing service in cloud computing, the following requirements should be achieved. Firstly, the data owner should be able to decide whether a user can access to his cloud data or not. Secondly, the privacy of users should be protected against the cloud. Finally, the accessing users may access the sharing data using connected terminals with low computing ability, such as smartphone and tablet. To date, these important fields in cloud sharing remains elusive.

In this paper, we address these issues and propose an effective and flexible privacy-preserving data sharing scheme, **P2E**. To preserve privacy and guarantee the data confidentiality against the cloud, we employ a cryptographic primitive, named ciphertext policy attribute-based encryption (CP-ABE), combined it with the technique of identity-based encryption (IBE). Compared with KP-ABE-based schemes, P2E introduces user public key which is tight with user secret key to realize fully collusion secure and privacy preserving. Meanwhile, P2E does not increase user keys so as to reduce the key management issues compared with HIBE-based schemes. P2E describes each data file with a set of meaningful attributes and assigns an access structure to these attributes for each user that reflects the scope of data files the user is allowed to access. When combined with each user's public key, the secret key of the same attribute for different users differs. To enforce these access structures, we define a public/secret key pair for each attribute. Data files are encrypted by public key components and access matrices converted from the access structure. User secret keys are defined to reflect their access privileges so that a user is able to decrypt a ciphertext only if he has the matched attributes to satisfy the ciphertext. Specifically, the main contributions of this paper can be summarized as following: 1) We propose an effective privacy-preserving encryption scheme P2E that simultaneously achieves full privacy-preserving, collusion resistance and data confidentiality for cloud data sharing service; 2) We prove that P2E is secure and P2E also simultaneously enforces fine-grainedness, backward secrecy and access privilege confidentiality for data sharing in

This work is supported in part by the National Key Basic Research and Development Plan of China under Grant 2013AA01A601, the National Natural Science Foundation of China under Grant 61170237, and the Doctoral Program of Higher Education of China (No. 20120073120034)

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

cloud computing; 3) The performance analysis indicates that P2E incurs only a small overhead compared to the existing works. Meanwhile, the experiment results demonstrate P2E is as light as possible.

The rest of the paper is organized as follows. Section II discusses related works. In Section III, we introduce the system model and adversary model. Section IV provides details of P2E. We analyze the security and performance of P2E in Section V and VI respectively. Finally, Section VII concludes the whole paper.

II. Related Work

The concept of IBE is proposed by Shamir [5], and the first fully functional IBE schemes are described by Boneh [6] and Cocks [7]. The IBE scheme is a public key cryptosystem (PKC) where the public key for a unique user is an arbitrary string like a user ID. There is a trusted third party called the private key generator (PKG) to calculate the corresponding private key. An attribute-based encryption (ABE) system is actually a simplified IBE system, with only one attribute in the system. In an ABE scheme [8], the sender encrypts the message with a set of attributes and specifies a number d. The recipient who has at least d attributes of the given attributes can decrypt the encrypted message. There are two classes of ABE named key-policy attribute-based encryption (KP-ABE) and ciphertext policy attribute-based encryption (CP-ABE). In KP-ABE [9], the access structure is used to encrypt the secret key, while the attributes are used to describe the ciphertext. In a CP-ABE scheme [10], the access structure is used to encrypt the ciphertext and the secret key is generated based on an attribute set. Thus, the roles of the secret key and the ciphertext in CP-ABE are opposite to what they are in KP-ABE.

Yu *et al.* [3] proposed a scheme that exploits KP-ABE and combines it with techniques of proxy re-encryption and lazy re-encryption. In Yu's scheme, the data owner can delegate most of the computation tasks of user revocation to the cloud server without disclosing any data to the untrusted cloud. However, delegation may leak user attributes and parts of secret keys to the cloud. In addition, the related ciphertext must be re-encrypted and informed to non-revoked users. We also discover the proxy re-encryption techniques applied in CP-ABE in the work of Wang *et al.* [2] and Yu *et al.* [11]. However, the user privilege is not protected from the cloud. Li *et al.* [12] provided a secure cloud storage scheme for health records in cloud computing. They used Chase and Chow's multi-authority ABE scheme to divide users into different domains. This scheme is just one case and is not general in cloud computing. Wang *et al.* [4] proposed a hierarchical fine-grained access control scheme that relies on Hierarchical IBE [13] and CP-ABE. The architecture of this scheme is arranged in a hierarchical way with a root master and several domain masters to generate keys for users. However, there is a mass of keys for each entity and the system is very complicated.

III. Problem Statement

A. System model

Our system model, as shown in Fig.1, necessitates four parties in network: **Data owner**: who has data stored in the cloud and depends on the cloud for data maintenance. Data owner can be enterprises or individual customers. **Data consumers**: who access data shared by the data owner, download data of interest from the cloud and then decrypt with their secret keys. For brevity, we refer to data consumers as users from now on. **Cloud server (CS)**: which provides high-quality services utilizing a number of servers with significant storage space and computation power. **Private key generator (PKG)**:which is a trusted third party that computes corresponding private keys for users [6]. In the system, we introduce a trusted third party called PKG. The PKG is only tasked with the issue of delivering public keys to the data owner and corresponding private keys to the users. Thus, other parties have no incentives to reveal the data to the PKG. So that, the system can keep the data private away from PKG and ***does not move any other security issues towards the PKG***. Here, a data owner stores his data into a set of cloud servers which are running in a cooperated and distributed manner. Data owner and consumers can be offline, whereas the cloud server have to be online all the time.

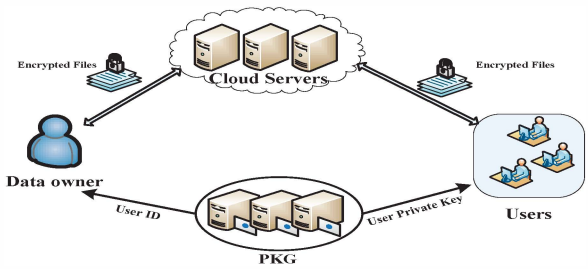

Fig. 1. System model

B. Adversary model

The adversary model considers most threats toward cloud data confidentiality. In our system model, for this initial work we assume that the cloud server is semi-trusted (also known as passive). Namely, it behaves properly most of time, but for some benefits the cloud server might try to find out as much secret information as possible. In fact, there are three types of threats: Both inner threats (CSP and users who might obtain the unauthorized data) and outer threats (external adversaries beyond the domain of this system, e.g., unauthorized attackers) might be presented; Attacks can either be active (unauthorized users who may inject malicious files into the cloud) or be passive (unauthorized users eavesdropping on conversations between users and the cloud); For the purpose of harvesting file contents, CSP and users may collude and try to access unauthorized data. We also have the following ***security requirements***: 1) **Fine-grained access control**: Each user should only access the data he is allowed and should not access the data he is not authorized to; 2) **Collusion resistance**: As described above, any user cannot collude and share his secret key with other users/cloud to access any data he is not allowed; 3) **Privacy preserving**: The cloud should not obtain any other users privacy information, e.g., user access privilege, which cannot be disclosed to the cloud.

In addition, each user can download a public/private key pair from the PKG. Attackers can easily obtain the user ID (public key) somewhere. Note that, in the system model, the communication channels between users and CS are secured under existing protocols, such as SSL.

IV. The Design of P2E

In this section, we first introduce the formats of users access policies in P2E, and then give a overview of P2E. After that, we present the detail algorithms of P2E and provide a example of P2E in practice.

A. Formats of access policy

Access policy can be expressed by an access tree $\mathbb{A}$ with attributes at leaves and logic gates e.g., $AND(\wedge)$, $OR(\vee)$ as intermediate nodes represented in ABE. Any access tree $\mathbb{A}$ can be converted to a Linear Secret Sharing Scheme(LSSS) matrix M [14]. In LSSS, every piece is a vector over some finite field, and every set in the access structure reconstructs the secret using a linear combination of the coordinates of its pieces. In P2E, a message $\mathcal{M}$ is encrypted with a LSSS access structure (M, ρ) where ρ is a permutation function that maps rows of M to attributes in $\mathbb{A}$. The user who only has the secret keys for a subset of rows M_x of M such that (1,0,...0) is in the span of these rows can decrypt the message correctly.

B. Overview of P2E

P2E is based on a bilinear map. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two cyclic groups of prime order q, and g_1 is the generator of group $\mathcal{G}_1$. A bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ satisfies the following properties:

1) Bilinearity: for all $u, v \in \mathcal{G}_1$ and $a, b \in \mathbb{Z}_q$, where $\mathbb{Z}_q = \{0, 1, 2, ...q-1\}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2) Computability: for any $u, v \in \mathcal{G}_1$, there is a polynomial time algorithm to compute $e(u, v) \in \mathcal{G}_2$.
3) Non-degeneracy: $e(g_1, g_1) \neq 1$.

In P2E, each file is described by a set of attributes. Determined by the set of attributes that a file has, an access tree is assigned to the file. The access tree is converted to a LSSS matrix. Each user has a set of attributes given by the data owner and owns a unique ID regarded as his public key. For each attribute the user has, a key for the user ID is created. At the initialization stage, the data owner chooses two random exponents for every attribute and publishes system public key. For each user, the data owner generates a key $sk_{i,u}$ of each attribute $i \in I_u$ for the user. The data owner encrypts message $\mathcal{M}$ with LSSS matrix M and system public key PK, and then uploads the ciphertext C to the cloud. The cloud server distributes secret keys and LSSS matrix to each user U_u. Users decrypt ciphertext C using these information, if the attribute set matches with file attributes. Otherwise, the ciphertext C will be undecryptable.

C. The detail of P2E

Based on the system model, we provide P2E which consists of four polynomial time algorithms in detail.

1) System Initialization: A data owner chooses a large prime q, two groups $\mathcal{G}_1, \mathcal{G}_2$ of order q, a map $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ and a hash function $H : \{0,1\}^* \rightarrow \mathcal{G}_1$ which maps a user ID to a element of $\mathcal{G}_1$. Then the data owner defines a set of attributes W for sharing data files and selects two random exponents $\alpha_i, \beta_i \in \mathbb{Z}_q$ for each attribute in W. So the secret key SK for the system is

$$SK = \{\alpha_i, \beta_i, i \in W\}.$$

The public key PK for the system is published:

$$PK = \{e(g_1, g_1)^{\alpha_i}, g_1^{\beta_i}, i \in W\}.$$

2) Encryption: The data owner defines a set of attributes $I \in W$ for each data file. As described in Section IV-A , the formats of access policy can be represented as a $n \times l$ LSSS matrix M with a function ρ mapping its rows to attributes. The data owner processes the message $\mathcal{M}$ as follows:

- randomly select a seed $s \in \mathbb{Z}_q$ and a random vector $v \in \mathbb{Z}_q^l$ with the first entry as s. Let $\lambda_x = M_x \cdot v$ where M_x is row x of M.
- randomly select a vector $\omega \in \mathbb{Z}_q^l$ with the first entry as 0 and a seed $r_x \in \mathbb{Z}_q$. Let $\omega_x = M_x \cdot \omega$.
- encrypt the message $\mathcal{M}$ with (M, ρ) as follows:

$$C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x}, C_{2,x} = g_1^{r_x},$$

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x}, C_0 = Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M}) \forall x,$$

where $\rho(x)$ is a permutation function mapping M_x to attribute i, and $Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$ is a symmetric encryption under key $e(g_1, g_1)^s$.

Finally, the data owner uploads the encryption file $C = \{\forall x, \{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}; (M, \rho)\}$ to the cloud servers.

3) Key generation and distribution: The data owner obtains user ID (ID_u) from PKG and assigns a set of attributes I_u for user U_u. Then the owner calculates the secret key component $sk_{i,u}$ for ID_u of attribute i belonging to user U_u:

$$sk_{i,u} = g_1^{\alpha_i} H(ID_u)^{\beta_i}.$$

The secret key for user U_u is $SK_u = \{sk_{i,u}, i \in I_u\}$. SK_u is encrypted by the user public key (ID_u) and delivered to the user via the cloud server.

4) Decryption: User U_u receives a ciphertext $C = \{\forall x, \{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}; (M, \rho)\}$ and $H(ID_u)$ from the cloud and selects constants $c_x \in \mathbb{Z}_q$ such that $\Sigma_x c_x M_x = (1, 0, ..., 0)$. The secret key of U_u is $\{sk_{i,u}, i \in I_u\}$. Then U_u calculates:

$$\begin{aligned}
& \Pi_x \{C_{1,x} \cdot e(H(ID_u), C_{3,x}) / e(sk_{\rho(x),u}, C_{2,x})\}^{c_x} \\
= \; & \Pi_x \{e(g_1, g_1)^{\lambda_x} \cdot e(g_1^{\alpha_{\rho(x)}}, g_1^{r_x}) \cdot e(H(ID_u)^{\beta_{\rho(x)}}, {g_1}^{r_x}) \cdot \\
& e(H(ID_u), g_1^{\omega_x}) / e(g_1^{\alpha_{\rho(x)}} H(ID_u)^{\beta_{\rho(x)}}, {g_1}^{r_x})\}^{c_x} \\
= \; & e(g_1, g_1)^{\Sigma_x \lambda_x c_x} \cdot e(H(ID_u), g_1)^{\Sigma_x \omega_x c_x} \\
= \; & e(g_1, g_1)^{v \cdot \Sigma_x c_x M_x} \cdot e(H(ID_u), g_1)^{\omega \cdot \Sigma_x c_x M_x} \\
= \; & e(g_1, g_1)^s.
\end{aligned}$$

User U_u can obtain the message $\mathcal{M}$=$Dec^{sym}_{e(g_1,g_1)^s}(C_0)$.

D. How P2E works in practice

Considering the situation of a healthcare case, a medical center stores millions of healthcare records in the cloud. We consider a access structure of one record $\mathcal{M}$ as Fig.2. The attribute set of this record is $I = \{i_1(Diabetes), i_2(Chinese), i_3(White), i_4(American)\}$. The LSSS

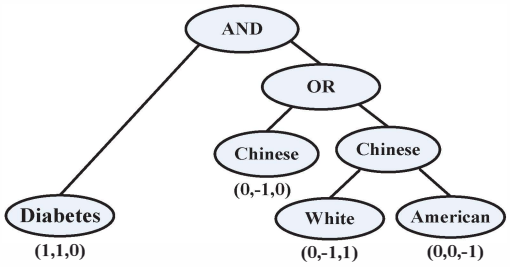

Fig. 2. Access tree of one healthcare record for the medical center

matrix M can be generated using the algorithm in [15]. Thus, the matrix M for Fig 2. is

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} \quad (1)$$

Let the permutation function ρ be denoted as,

x	1	2	3	4
$\rho(x)$	i_1	i_2	i_3	i_4

The medical center encrypts this record $\mathcal{M}$ with (M, ρ) and system public key PK, and then sends the ciphertext $C = \{\{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}_{x\in\{1,2,3,4\}}; (M, \rho)\}$ with the hash function H to the cloud server. So a doctor who looks up records relating to "diabetes and Chinese" is able to access this record. However, if looking up "diabetes and American", he will be unable to access this record. Next, we will check these access policies.

Supposing there is a doctor U_2 (where we assume his ID is 2 for brevity) who looks up patients with "diabetes and white from American". He then is given attributes i_1, i_3 and i_4, so $I_2 = \{i_1, i_3, i_4\}$. Next he will be given the secret key $sk_2 = \{sk_{1,2}, sk_{3,2}, sk_{4,2}\}$ which is assigned by the medical center through the cloud servers. The doctor obtains $C = \{\{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}_{x\in\{1,2,3,4\}}; (M, \rho)\}$ and $H(ID_2)$ from the cloud servers. The doctor first looks for the common attributes associated with the record through the permutation function ρ, and gets common attributes i_1, i_3 and i_4. Then he finds corresponding vectors of attribute i_1, i_3 and i_4 are $(1, 1, 0), (0, -1, 1)$ and $(0, 0, -1)$ respectively in the LSSS matrix M. According to the decryption algorithm in section IV-C, the doctor finds the linear combination of rows 1,3 and 4 to $(1, 0, 0)$ as follows:

$$(1, 1, 0) + (0, -1, 1) + (0, 0, -1) = (1, 0, 0).$$

Then the doctor can use the decryption algorithm to calculate $e(g_1, g_1)^s$. The record $\mathcal{M}$ can be recovered once $e(g_1, g_1)^s$ is calculated.

Supposing there is a doctor who looks up records relating to "diabetes and American". The common attributes are i_1, i_4. Corresponding vectors of these attributes are $(1, 1, 0)$ and $(0, 0, -1)$. We observe that there is no linear combination of rows 1 and 4 of matrix M to $(1, 0, 0)$. Thus, the doctor can not calculate $e(g_1, g_1)^s$. Further on, he can not recover the record $\mathcal{M}$.

V. Security Analysis

In P2E, we assign flexible and different access privileges for each user to achieve fine-grained access control. Meanwhile, P2E achieves fully collusion resistance which is important when several users collude and share their secret keys to access the unauthorized data. P2E can also achieve user access privilege confidentiality. In this section, we first provide an intuitive security argument about P2E. Then, the security requirements in Section III-B are focused on.

A. Security

We recall that in P2E the message $\mathcal{M}$ is encrypted in the form of $C_0 = Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$. Obviously, the adversary $\mathcal{A}$ must construct $e(g_1, g_1)^s$ to decrypt ciphertext C_0. To recover $e(g_1, g_1)^s$, the adversary $\mathcal{A}$ needs some matched secret keys. Although $\mathcal{A}$ can obtain some public parameters, he is unaware of the value of random seed s. $\mathcal{A}$ cannot construct $e(g_1, g_1)^s$ directly. To obtain $e(g_1, g_1)^s$, the adversary has to use secret keys requested for A_i which does not contain $I_{\mathcal{A}}$, yielding

$$\begin{aligned} & \Pi_x\{C_{1,x} \cdot e(H(ID_{\mathcal{A}}), C_{3,x})/e(sk_{\rho(x),\mathcal{A}}, C_{2,x})\}^{c_x} \\ = \; & \Pi_x\{e(g_1, g_1)^{\lambda_x} \cdot e(g_1^{\alpha_{\rho(x)}}, g_1^{r_x}) \cdot e(H(ID_{\mathcal{A}})^{\beta_{\rho(x)}}, g_1{}^{r_x}) \cdot \\ & e(H(ID_{\mathcal{A}}), g_1^{\omega_x})/e(g_1^{\alpha_{\rho(x)}} H(ID_{\mathcal{A}})^{\beta_{\rho(x)}}, g_1{}^{r_x})\}^{c_x} \\ = \; & e(g_1, g_1)^{\Sigma_x \lambda_x c_x} e(H(ID_{\mathcal{A}}), g_1)^{\Sigma_x \omega_x c_x}. \end{aligned}$$

The adversary can recover a target group element with the form $e(g_1, g_1)^{\lambda_x} e(H(ID), g_1)^{\omega_x}$ at each node 'x'. This group element contains a secret share λ_x of the secret s in the exponent. However, each share λ_x is "blinded" by a shared ω_x. To obtain $e(g_1, g_1)^s$ from the above process, the value of $\Sigma_x \lambda_x c_x$ has to be s and $\Sigma_x \omega_x c_x$ must be 0. However, there is no existing $c_x \in \mathbb{Z}_q$ that satisfies $\Sigma_{x\in X'} c_x M_x = (1, 0, ..., 0)$ for a set of rows X' in matrix M if the adversary does not have a matched set of attributes. So $\Sigma_x \omega_x c_x$ cannot to be 0, i.e., the terms of $e(H(ID), g_1)$ will not be omitted. Thus, $e(g_1, g_1)^s$ cannot be constructed and the adversary cannot compromise the ciphertext. In addition, due to lack of space, the semantical security proof of P2E is emigrated.

B. Security requirements

1) Fine-grained access control: In P2E, each user receives a flexible access structure from the data owner. Each user U_u has been assigned a set of attributes for the data owner. Suppose a file has an attribute $i \in I$, so it has a corresponding row r_b in the LSSS matrix. However, if the user U_u does not have the attribute i, he can not receive the secret key $sk_{i,u}$ for attribute i. In addition, in the decryption stage, as U_u cannot find the corresponding c_x of row r_x to satisfy $\Sigma_x c_x M_x = (1, 0, ..., 0)$, the decryption procedure will fail. Therefore, a user who does not have the attribute i cannot calculate $e(g_1, g_1)^s$. Thus, the user cannot decrypt the unauthorized message. P2E only discloses decryption keys to authorized users, thus unauthorized users and the cloud server cannot decrypt. For this reason, P2E can help the data owner to realize fine-grained access control of the cloud data.

2) Fully collusion secure: The existing schemes achieve the security requirements except full collusion resistant. For example, Yu's scheme can achieve secure and fine-grained data access control but suffered from the following full collusion problem. In their scheme, each user has the corresponding key for each attribute which does not vary amongst different users for the same attribute. Thus, if the conjunction of two users'

attribute sets contains the encrypted file attributes, the file content will be leaked. In addition, their scheme is vulnerable to collusion attacks when the number of colluded users is greater than the degree of the polynomial which used to generate secret keys. However, P2E is fully collusion secure.

Theorem *5.1: Two or more users with different identities cannot construct $e(g_1, g_1)^s$, even if they collude and combine their keys.*

Proof: In P2E, the user ID is "tied" together with the given attributes so that users cannot combine the attributes of others in decryption. In encryption algorithm, the file $\mathcal{M}$ is blinded with $e(g_1, g_1)^s$. The value s is split into the vector λ_x and value 0 is split into the vector ω_x. The user who wants to obtain the file $\mathcal{M}$ must recover $e(g_1, g_1)^s$ by pairing keys for attributes and ID pairs. To achieve this, the user must introduce the term $e(H(ID), g_1)^{\omega_x}$. If the user has the matched set of keys, this term will be canceled in the decryption process. Otherwise, the term can not be canceled. If two or more users with different IDs attempt to collude, the terms $e(H(ID), g_1)^{\omega_x}$ will not cancel each other because the terms for each user are different. Therefore, P2E is fully collusion secure. Hence, Theorem 5.1 holds true. ■

3) User access privilege confidentiality: P2E does not disclose any attribute of a user attribute set to the cloud servers. The cloud has no clue about users' secret keys and does not posses any $sk_{i,u}$. The cloud cannot derive any user's access privilege information so that users' privacy are protected against the cloud. In contrast, Yu's scheme discloses leaf nodes information to the cloud. Only the interior nodes are unknown to the cloud. In addition, the cloud also knows part of the user's secret keys. Thus the more legitimate users are revoked, the more secret keys the cloud knows. This cannot achieve fully privacy-preserving policy in cloud computing.

VI. Performance Analysis

In this section, the performance of P2E is analyzed by comparing with other data sharing schemes that rely on KP-ABE like. We first evaluate the computation and communication overhead, and then give the detailed about the ciphertext size in P2E.

A. Computation complexity

We analyse the computation overhead of P2E according to the encryption and decryption algorithms. In P2E, the main computation operations involved in encryption and decryption algorithms are pairing (calculate $e(g_1, g_1)$) and scalar multiplication. We recall that the ciphertext of P2E is $C = \{\forall x, \{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}\}$. Pairing is the most expensive operation. For each different file, however, data owner and users only need to calculate $e(g_1, g_1)$ once in the beginning. Since both P2E and KP-ABE-based schemes have the same numbers of pairing operation, we do not involve in pairing operation overhead when computation complexity of P2E compares with the KP-ABE-based schemes. In the computation complexity analysis, we only take into account scalar multiplication operation. During encrypting, all encryption operations are at the data owner side. The data owner needs to do two scalar multiplications to calculate $C_{1,x}$, one scalar multiplication for $C_{2,x}(C_{2,x} = g_1^{r_x})$, and one for $C_{3,x}$ for each row x in LSSS matrix. Therefore, the data owner needs at most $4|I|$ scalar multiplications. The computation complexity of data owner converting the access structure to a LSSS matrix is $O(|I|)$ where $|I|$ is the number of attributes about the access structure. Thus, the computation complexity for encryption is $O(|I|)$. In the decryption stage, the decryption operation is similar only for users. To recover ciphertext, the user needs at most another $|I|$ scalar multiplications to calculate $\Pi_x\{e(g_1, g_1)^{\lambda_x} e(H(ID), g_1)^{\omega_x}\}$, so the time complexity is also $O(|I|)$. The computation complexity of P2E and KP-ABE-based schemes is given in the Table I. From Table I, we notice that the computation complexity of encryption performed by the data owner in P2E is the same with KP-ABE-based schemes. In addition, the number of N in KP-ABE-based schemes is bigger than $|I|$ most of the time, so P2E consumes less computational cost in decryption stage.

TABLE I. Computation complexity required in KP-ABE-based scheme and P2E

Scheme	Encryption(Data owner)	Decryption(User)
KP-ABE-based	$O(\|I\|)$	$O(max(\|I\|, N))$
P2E	$O(\|I\|)$	$O(\|I\|)$

TABLE II. Communication costs in KP-ABE-based scheme and P2E

Scheme	Communication costs
KP-ABE-based	$\|I\| + 2l\bullet g\|I\| + (\|I\| + 1)l\bullet g\|\mathcal{G}_1\| + l\bullet g\|\mathcal{G}_2\|$+Data
P2E	$\|I\|^2 + l\bullet g\|I\| + (2\|I\| + 1)l\bullet g\|\mathcal{G}_1\| + (\|I\| + 1)l\bullet g\|\mathcal{G}_2\|$+Data

We also conduct a thorough experimental evaluation about the time cost of P2E. The whole experiment system is implemented by Python language on a Windows 7 machine with Core 2 Duo CPU running at 2.0GHz. All results are the average of 100 trials. First we test the speed of encryption. In P2E, the calculation of C_0 is based on $C_{1,x}, C_{2,x}, C_{3,x}$. Fig.3 plots the overhead to calculate $C_{1,x}, C_{2,x}, C_{3,x}$ versus the number of attributes $|W|$. From Fig.3, we can see the encryption cost increases linearly with the attributes $|W|$. This is consistent with the above computation analysis ($O(|I|)$). Fig.6 plots the performance of two common symmetric encryptions, i.e, 128-bit RC4 and 128-bit AES CBC, which are needed to calculate $C_0 = Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$. From Fig.6, we can see that it is effective to compute C_0, e.g. the time to encrypt a 10MB file using 128-bit RC4 and 128-bit AES CBC approaches to 35 milliseconds and 105 milliseconds respectively, which is an ideal result. The overhead of key generation and decryption is shown in Fig.4 and Fig.5. In these tests, we assume all attributes should be involved in the key generation and decryption. Fig.4 plots the overhead to calculate key $(\{g_1^{\alpha_i} H(ID_u)^{\beta_i}\}_{i\in W})$ versus the number of attributes $|W|$. As we can see, its overhead also grows linearly with $|W|$. Fig.5 plots the speed to recover $(g_1, g_1)^s$. We find that decryption cost grows linearly with $|W|$ and it is cheaper than encryption. The reason is because decryption takes less power operations. The results of our experiments show P2E is light weighted and efficient to be applied in practice.

B. Communication cost

In P2E, the communication cost is mainly attributable to the encrypted data transmission. After encryption, the following information is sent by the data owner along with the encrypted data to the cloud: Value of matrix M which requires $|I|^2$ bits, value of permutation function ρ requir-

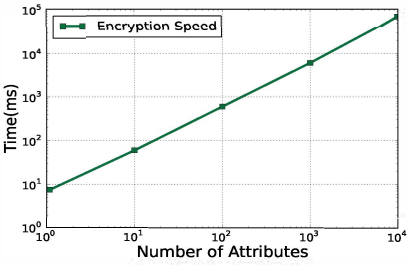

Fig. 3. The overhead of encryption algorithm versus the number of attributes $|W|$.

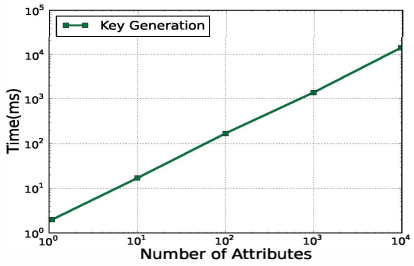

Fig. 4. The overhead of key generation algorithm versus the number of attributes $|W|$.

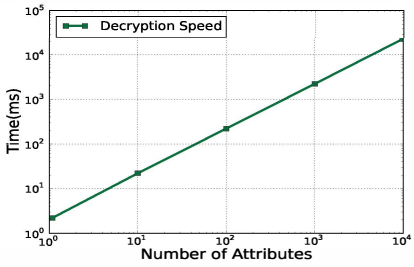

Fig. 5. The overhead of decryption algorithm versus the number of attributes $|W|$.

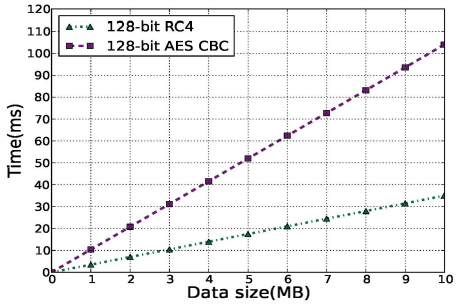

Fig. 6. The cost of symmetric encryption algorithms.

TABLE III. CIPHERTEXT SIZE IN KP-ABE-BASED SCHEME AND P2E

Scheme	Ciphertext size
KP-ABE-based	$\|I\| + \|I\|l\bullet g\|\mathcal{G}_1\| + l\bullet g\|\mathcal{G}_2\|$+Data
P2E	$2\|I\|l\bullet g\|\mathcal{G}_1\| + \|I\|l\bullet g\|\mathcal{G}_2\|$+Data

ing $log|I|$ bits, value of $C_0, C_{1,x}, C_{2,x}$ and $C_{3,x}$ for every x, $log|\mathcal{G}_2| + |I|log|\mathcal{G}_2| + 2|I|log|\mathcal{G}_1|$, and value of $H(ID)$ which requires $log|\mathcal{G}_1|$ bits. Thus, the communication cost is given by $|I|^2$+$log|I|$+$(2|I|+1)log|\mathcal{G}_1|$+$(|I|+1)log|\mathcal{G}_2|$+Data. Table II shows the communication expenses comparison between P2E and KP-ABE-based schemes. We can see that P2E communication cost is a little more. However, in practice, a file is described by just a few attributes, i.e, $|I|$ is small in general cases. For example, in Section IV-D, a record just is described by diabetes, Chinese, White and American, i.e, $|I| = 4$. In addition, even though the order of cyclic group $\mathcal{G}$ is large, $log|\mathcal{G}|$ bits is far less than the file size (Data). For example, the order of $\mathcal{G}$ is equal to 10^{10}, $log|\mathcal{G}|$ bits is just near to 30 bits. Therefore, the main communication costs will depend on the file size.

C. Ciphertext size

As described in Section IV-C, the ciphertext is composed of four parts: $C_0, C_{1,x}, C_{2,x}, C_{3,x}$, so the size of the ciphertext is $(|I| + 1)log|\mathcal{G}_2| + 2|I|log|\mathcal{G}_1| + Data$. We compare the ciphertext size of P2E with other KP-ABE-based schemes in Table III. As discussed in Section VI-B, $|I|$ and $log|\mathcal{G}|$ are far less than the file size (Data), so the difference between P2E and KP-ABE-based scheme is also negligible.

VII. CONCLUSION

In this paper, we present a privacy-preserving and secure data sharing scheme P2E in cloud computing by exploiting CP-ABE and combining it with technique of IBE. P2E ensures fine-grained data access control and security against collusion of users with the cloud. Moreover, P2E does not disclose any attribute of users to the cloud so that keep the privacy of the users away from the cloud. Security analysis show that P2E is secure. In addition, we evaluate the performance of P2E about computation complexity, communication cost and ciphertext size. The result shows that P2E is low overhead and highly efficient. Following the current research, we will implement the proposed privacy-preserving and effective cloud data sharing service in a real CSP platform for future work.

REFERENCES

[1] A. Michael *et al.*, "Above the clouds: A berkeley view of cloud computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[2] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 735–737.

[3] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *International Conference on Computer Communications*. IEEE, 2010, pp. 1–9.

[4] G. Wang, Q. Liu, and J. Wu, "Achieving fine-grained access control for secure data sharing on cloud servers," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 12, pp. 1443–1464, 2011.

[5] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in cryptology*. Springer, 1985, pp. 47–53.

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

[7] C. Cocks, "An identity based encryption scheme based on quadratic residues," *Cryptography and Coding*, pp. 360–363, 2001.

[8] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *Advances in Cryptology–EUROCRYPT 2005*, pp. 557–557, 2005.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 89–98.

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

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

[12] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," *Security and Privacy in Communication Networks*, pp. 89–106, 2010.

[13] J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in *Advances in Cryptology EUROCRYPT*, 2002, pp. 466–481.

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

[15] Z. Liu and Z. Cao, "On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption," Cryptology ePrint Archive, Report 2010/374, 2010. http://eprint. iacr. org/2010/374, Tech. Rep., 2010.