2nd IEEE International Conference on Cloud Computing Technology and Science

Fine-grained Data Access Control Systems with User Accountability in Cloud Computing

Jin Li[1], Gansen Zhao[2,5], Xiaofeng Chen[3], Dongqing Xie[1]
Chunming Rong[4], Wenjun Li[5], Lianzhang Tang[1], Yong Tang[2]
[1] School of Computer Science and Educational Software
Guangzhou University, China
[2]School of Computer Science, South China Normal University, China
[3]Key Laboratory of Computer Networks and Information Security
Xidian University, China
[4]Dept. of Electrical Engineering and Computer Science
Stavenger University, Norway
[5]School of Software, Sun Yat-sen University, China

***Abstract*—Cloud computing is an emerging computing paradigm in which IT resources and capacities are provided as services over the Internet. Promising as it is, this paradigm also brings forth new challenges for data security and access control when users outsource sensitive data for sharing on cloud servers, which are likely outside of the same trust domain of data owners. To maintain the confidentiality of sensitive user data against untrusted servers, existing work usually apply cryptographic methods by disclosing data decryption keys only to authorized users. However, in doing so, these solutions inevitably introduce heavy computation overhead on the data owner for key distribution and data management when fine-grained data access control is desired, and thus do not scale well.**

In this paper, we present a way to implement scalable and fine-grained access control systems based on attribute-based encryption (ABE). For the purpose of secure access control in cloud computing, the prevention of illegal key sharing among colluding users is missing from the existing access control systems based on ABE. This paper addresses this challenging open issue by defining and enforcing access policies based on data attributes and implementing user accountability by using traitor tracing. Furthermore, both the user grant and revocation are efficiently supported by using the broadcast encryption technique.

Extensive analysis shows that the proposed scheme is highly efficient and provably secure under existing security models.

Keywords: Fine-grained access control, Accountability, Attribute-based encryption

I. INTRODUCTION

Cloud computing is a promising next-generation computing paradigm which primarily relies on technologies such as virtualization, utility computing, Service Oriented Architecture, and so forth. Cloud computing is capable to provides seemly unlimited "virtualized" resources to users as services across the Internet while hiding platform and implementation details from users. The services are invoked by users in the on-demand fashion and "bill like utility". With this paradigm computation/storage intensive tasks can be performed even by resource-constrained users through being delegated to resource-abundant cloud servers. As compared to setting up their own infrastructures, cloud users can tremendously save their capital expenditures via the usage of cloud computing, not to mention other benefits such as reliability and high scalability of the system. With the emergence of commercial cloud computing platforms such as Amazon's EC2 and S3 [1], Google App Engine [2], and Microsoft Azure [3], cloud computing has been more a reality than just a concept.

As in any existing application and system, security and privacy play an extremely important role for the success of cloud computing, and certainly raise new challenges among the many others that cloud computing is confronted with. Although it is believed that security in cloud computing is generally improved than the security in traditional systems in the sense that more resources can be devoted to solving security issues, unauthorized access to sensitive data is one of the most critical concerns from cloud computing customers.

As prevention of unauthorized access can be generally achieved by encrypting sensitive contents before uploading them to cloud servers, there still exist many challenge issues for its implementation in practical systems. In particular, differentiated content access is frequently required in the sense that users with different roles (or paying different prices) should be granted different level of access privileges.

Data access control has been evolving in the past thirty years and various techniques [4] have been developed to effectively implement fine-grained access control, which allows flexibility in specifying differential access rights of individual users. Traditional access control architectures usually assume that data owners and storage servers are in the same trusted domain, where the storage servers are fully trusted as an omniscient reference monitor responsible for defining and enforcing access control policies. This assumption however no longer holds in cloud computing since the data owners and cloud storage servers are very likely to be in two different trust domains. On the one hand, cloud servers are not entitled to access the outsourced data content for data confidentiality;

*Correspondence should be made to Jin Li, Gansen Zhao, or Yong Tang.

978-0-7695-4302-4/10 $26.00 © 2010 IEEE
DOI 10.1109/CloudCom.2010.44

on the other hand, the data resources themselves are not physically under the full control of the owner any more. For the purpose of helping the data owner impose fine-grained access control of data stored on untrusted cloud servers, a feasible solution would be encrypting data through certain cryptographic primitive(s). The data owner only discloses decryption keys to authorized users. Unauthorized users, including cloud servers, are not able to decrypt since they do not have the data decryption keys. This general method actually has been widely adopted by existing works which aim at security of data stored on untrusted servers. One critical issue of this branch of approaches is how to achieve the desired security goals without introducing high complexity of key management and data encryption. Existing work resolve this issue either by introducing a per file access control list (ACL) for fine-grained access control, or by categorizing files into several filegroups for efficiency. As the system scales, however, the ACL-based scheme would introduce an extremely high complexity which could be proportional to the number of system users. The filegroup-based scheme, on the other hand, is just able to provide coarse-grained access control of data.

Recently, the notion of attribute-based encryption (ABE), which was proposed by Sahai and Waters [19], has attracted much attention in the research community to design flexible and scalable access control systems. For the first time, ABE enables public key based one-to-many encryption. Therefore, it is envisioned as a highly promising public key primitive for realizing scalable and fine-grained access control systems, where different yet flexible access rights can be assigned to individual users. To address complex and general access policy, two kinds of ABE have been proposed [14]: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE, access policy is assigned in attribute private key, whereas, in CP-ABE, the access policy is specified in the ciphertext.

Although ABE provides secure and fine-grained access control, before its deployment in cloud computing, two critical security aspects have to be addressed, which are, 1) efficient user revocation; 2) user accountability. In access control systems, when a user's access privilege is to be revoked, traditional revocation techniques, such as [11], can be used. However, for scalability purpose, it is necessary to enable efficient revocation operation. Furthermore, another important issue to be addressed is user accountability . In another word, the problem of key abuse, *i.e.*, illegal key sharing among users, should be prevented. This problem is extremely important as the attribute private keys directly imply users' privileges to the protected resources . Dishonest users may share their attribute private keys with other users, who do not have these privileges. They can just directly give away part of their original or transformed keys such that nobody can tell who have distributed these keys. Consequently, it renders the system useless. Though Li et al. [21] have addressed this problem, they only solved the user accountability for a special application, that is, anonymous ABE. Thus, it is still one critical issue that has to be carefully addressed before the access control system can be used in cloud computing.

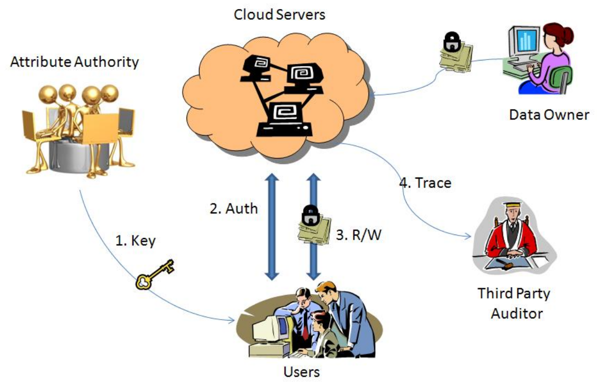

Fig. 1: Architecture of the access control system

Contributions The contributions of this paper in addressing the secure fine-grained access control problem in Cloud Computing is multi-fold.

Firstly, for each file, we achieve to define and enforce access policies based on attributes in the system. The access structures of files is defined so that a user is able to access a file if and only if the user's attributes satisfy the file's access structure. Such a design also brings about the efficiency benefit, as compared to existing work, in that, 1) the complexity of data file encryption is only related to the number of access policy associated to the file, which is independent to the number of users in the system, and 2) file creation/deletion and new user grant operations only affect current file/user without involving system-wide data file update or re-keying.

Secondly, we achieve user accountability in fine-grained data access control systems. With the ABE technique to realize the fine-grained access control, the user accountability is implemented by using the traitor tracing technique. Clearly, to securely deploy an ABE-based access control system, it is imperative to guarantee that the key issued to each user cannot be shared. Such key abuse problem exists in ABE-based access control schemes [20] as their attribute keys are never designed to be linked to any user specific information except the commonly shared attributes. Extensive analysis shows that our proposed scheme is highly efficient and provably secure under existing security models.

Finally, we deploy the cloud servers to implement user revocation. For the other entities, including the data owner and users, only small computational cost is involved. In another word, the data owner outsources most of the computational cost of revocation to the server as the servers have powerful computation ability.

Organization In Section II, the definitions and models for access control problem in cloud computing are presented. In Section III, a construction of access control scheme is described. The performance analysis is given in Section IV. Finally, Section V concludes this paper.

II. Models and Basic Techniques

A. System Model

In this paper, we consider a cloud data system consisting of data owners, data users, Cloud Servers, and a third Party Auditor. A data owner stores his sensitive data on Cloud Servers. Users are issued attributes. To access the remote stored data files shared by the data owner, users need to download the data files from the Cloud Servers. For simplicity, we assume that the only access privilege for users is data file reading. Cloud Servers are always online and operated by Cloud Service Provider (CSP). They are assumed to have abundant storage capacity and computation power. The Third Party Auditor is also an always online party which audits every file access event. In addition, we also assume that the data owner can store data files besides running his own code on Cloud Servers to manage his data files. The architecture is shown in Fig. 1.

B. Security Models

In this work, we consider Honest but Curious Cloud Servers. That is to say, Cloud Servers will follow the proposed protocol, but try to find out as much secret information as possible based on their inputs. More specifically, we assume Cloud Servers are more interested in contents of data files stored. Secure communication channels between users and Cloud Servers are required. Users would try to access data files either within or out of the scope of their access privilege. Two kinds of attackers are considered in this system, which are, 1) external attackers, including the server, revoked users and other unauthorized users; 2) internal attackers who share their privileges with other users without such ones.

C. Design Goals

In this paper, we address the problem of enabling efficient fine-grained access control of data files stored on Cloud Servers, while achieving user accountability and efficient revocation. Specifically, we allow data owners to enforce an access structure on every file, which precisely designates the set of files that the user is allowed to access. Cloud Servers are also prevented from learning the plaintexts of data files. In addition, the proposed scheme should be able to achieve basic security goals, e.g., user grant/revocation, as any general one-to-many communication system would require. All these security goals should be achieved efficiently in the sense that the system is scalable.

III. Basic Schemes

A. Straightforward Approach

Before introducing our construction, we first propose a straightforward approach that achieves a fine-grained access control with user accountability, which also aims at providing an overview of how accountable access control scheme works. A broadcast encryption scheme is demanded in this paper. Assume $\mathsf{BE} = (\mathit{KeyGen}_{BE}, \mathit{Enc}_{BE}, \mathit{Dec}_{BE})$ is a broadcast encryption scheme providing revocation-scheme security against a coalition of all revoked users [11]. More specially, KeyGen_{BE} is the key generation algorithm that is used to generate a long-lived key for the user, Enc_{BE} is the encryption algorithm that is used to encrypt files to a privileged user group G. Only user in the group G can decrypt and get the message from the ciphertext. The group G can be dynamically changing, as users can be added to or removed from G. If a user is removed from G, his privilege of decryption is also canceled. Dec_{BE} is the decryption algorithm that is used to decrypt the ciphertext if with a non-revoked secret key at the time the message was encrypted.

We also need a secure symmetric encryption scheme in the following constructions. Assume SE=(KeyGen_{SE}, Enc_{SE}, Dec_{SE}) is a symmetric encryption scheme, where KeyGen_{SE} is the setup algorithm with a predefined security parameter λ, Enc_{SE} and Dec_{SE} are the encryption and decryption algorithms, respectively.

For efficiency, hybrid encryption method is applied here. More specially, the data files are encrypted with symmetric keys, which are encapsulated with the broadcast encryption. The straightforward approach of fine-grained access control goes as follows:

- To upload a file F, the data owner chooses a random secret key K for the symmetric encryption scheme SE and encrypts the file as $C = \mathit{Enc}_{SE}(K, F)$. Then, the data owner chooses the authorized user group U_i who can access the file by computing $C = \mathit{Enc}_{BE}(U_i, K)$. Actually, the same symmetric encryption key K can be used to encrypt all the files with the same access policy, where these files can be regarded as belonging to the same file collection.
- Assume a user in U_i has been issued a long-lived secret key sk_i of the broadcast encryption scheme. This long-lived key is issued by the data owner at the setup of the system to each authorized user, including the cloud servers. This user can decrypt and access the file by computing $K = \mathit{Dec}_{BE}(sk_i, C)$ with sk_i. After decrypting to get K, he can decrypt to retrieve the file as $F = \mathit{Dec}_{SE}(K, C)$.

The user grant and revocation can be implemented by using the broadcast encryption BE. This straightforward approach apparently provides fine-grained access control over the encrypted files while achieving user accountability using the technique of broadcast encryption. More specially, user accountability can be achieved by utilizing the traitor tracing algorithm [11] in broadcast encryption to detect the dishonest user who shares the key in the illegal decoder box. However, this approach has serious efficiency disadvantages for scalable system. In concrete, it would introduce large storage and computation complexities because the size attached for the access control or the computation cost in the decryption is linearly increasing with the number of authorized users and files, which greatly degrade the usability.

B. Access Control System based on Attribute-based Encryption

Attribute-based encryption (ABE), which was proposed by Sahai and Waters [19], is envisioned as a highly promising primitive for realizing scalable and fine-grained access control systems. The length of public parameters and ciphertext is only related with the number of attributes, instead of the number of users as in most of the other techniques. Thus, ABE enables public key based one-to-many encryption, where differential yet flexible access rights can be assigned to individual users. There are two methods in cryptography to realize the fine-grained access control based on ABE: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE) [14]. In KP-ABE, the access policy is bounded with the attribute private keys issued to users, which specifies which type of files the user is able to access. In a CP-ABE system, on the contrary, the access policy over attributes is specified in each file, and each user's key is associated with a set of attributes. Therefore, CP-ABE is different from KP-ABE in the sense that, in CP-ABE, the data owner is able to assign certain access policy for each file. When a file is being encrypted, it will be associated with an access structure over a predefined set of attributes. The user will only be able to access the underlying file if his attributes match the access structure specified in the ciphertext. While in KP-ABE, the access policy is bounded with each user, not with each file. Thus, we explore how to utilize CP-ABE to manage the files stored in cloud computing .

We show how to deploy the CP-ABE in the construction of scalable access control system in cloud computing.

- Initially, the data owner wishes to allow users only with specific privileges to access the data files stored in cloud servers. In the ABE-based access control system, these privileges can be categorized via the users' attributes. For each file F, the data owner defines an access policy W and chooses a random secret key K. The data owner encrypts F with the symmetric encryption as $C = \mathit{Enc}_{SE}(K, F)$. For the same reason as above, the same symmetric encryption key can be used to encrypt all the files with the same access policy, where these files can be regarded as belonging to the same file collection. Finally, the key K is encapsulated with the encryption algorithm of CP-ABE under access policy W.
- When a user is allowed to access the files, the data owner assigns a set of attributes L to the user as his privileges by deploying the key generation algorithm in CP-ABE scheme.
- The user with the attribute private key on L can decrypt and get K if his attribute list L matches the access policy W. With K, he can decrypt and retrieve the file F as $\mathit{Dec}_{SE}(K, C)$.

Because the access policy for each file is defined through attributes, this system enables fine-grained access control based on the application of CP-ABE. However, another critical issue, that is, user accountability, cannot be achieved because of the ABE. More specially, each user in the ABE is issued attribute private keys based on their attributes, which directly imply users' privileges to the protected resources. However, the dishonest users may illegally distribute and share their attribute private keys with other users and thus abuse the system without being detected. The reason is that no identity information is bounded with the attribute private key in the existing ABE schemes. Furthermore, it is very inefficient to realize the user revocation in ABE: Whenever there is a user to be revoked, the attribute private keys of the other authorized users have to be updated by the data owner [19].

IV. ABE-BASED FINE-GRAINED ACCESS CONTROL WITH USER ACCOUNTABILITY

Based on the above observations, there are still two challenges in realizing fine-grained access control in cloud computing, *i.e.*, user accountability and efficient revocation. In this section, we propose a fine-grained access control scheme enabling user accountability. The intuition of our proposed scheme is to enable owners to identify guilty users given misbehaving secret keys. For this purpose, the proposed scheme embeds the unique identity of a user into each secret generated, and ensures that any third party, e.g., the law authority, is able to disclose the user identity of any given secret key. In addition, our proposed scheme enables the owner to efficiently revoke any user with the help of the cloud servers. The details of improved ABE-based fine-grained access control system are presented as follows.

A. System Description

System Setup Assume that the universe of attributes in the system is denoted by $U = \{\omega_1, \omega_2, \cdots, \omega_n\}$. Note that each attribute is a multi-value set. For example, the attribute "year" in a system can be defined as a multi-value set $\{2000, 2001, \cdots, 2010\}$. Then, choose a security parameter 1^λ and run the algorithm $\mathit{Setup}(1^\lambda, U)$ as shown in Fig.2 to obtain the public parameter $para$ and the master key sk. The public parameter is then published on cloud servers so that every user can download it, while the master key is kept by the attribute authority as secret information.

For data owner, he computes the key for the broadcast encryption scheme BE as defined in Section III-A. The data owner also chooses a random r and computes the broadcast encryption of r as $C' = \mathit{Enc}_{BE}(G, r)$ for the authorized user group G in this system, where C' is published on cloud servers.

New File Creation Whenever the data owner creates and uploads a file F to the cloud servers, he first defines an access policy for this file, which is represented by $W = [W_1, W_2, \cdots, W_n]$. Each W_i are chosen from the multiple values of attribute ω_i. For example, W_i could be an attribute like "year=2009" while the attribute "year" has multiple values. If a certain attribute, say ω_j, is not needed in the access policy, the owner just sets $W_j = *$ in W. Then, the owner randomly chooses a symmetric key K from the key space and encrypts the file F with K using standard symmetric key algorithm such as AES. Later on, he runs the algorithm $\mathit{Encrypt}(K, W)$ and obtains the ciphertext C which is the encryption of the symmetric key K with respect to the access policy W. Finally, the data owner uploads the encrypted file as well as the ciphertext C to the cloud servers.

New User Grant Assuming a new user with identity

Setup(1^{λ},U) Define a bilinear group $\mathbb{G}_1$ of prime order p with a generator g, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, with the properties of *Bilinearity* and *Non-degeneracy*, and a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. Then output the pubic parameter $para = (g, g_1, g_2, g_3, u_1, u_2, \cdots, u_n)$, and the master key $sk = g_2^{\alpha}$, where $g_1, g_2, g_3, u_1, \cdots, u_n \leftarrow_R \mathbb{G}_1$, $\alpha \leftarrow_R \mathbb{Z}_p$, $g_1 = g^{\alpha}$ and $n = |U|$.

Encrypt(K,W) Assuming $W = [W_1, W_2, \cdots, W_n]$, the secret K is encrypted as $(C_0, C_1, C_2, \{T_i\}_{W_i=*}, E)$ where $C_0 = K\hat{e}(g_1, g_2)^s$, $C_1 = g^s$, $C_2 = (\prod_{W_i \neq *} u_i^{H(W_i)} \cdot g_3)^s$, T_i=$\{u_i^s\}_{W_i=*}$, and $E = u_0^s$ by choosing a random $s \in \mathbb{Z}_p$.

KeyGen(ID,L,sk) Assume the attribute authority wants to assign a set of attributes $L = [L_1, L_2, \cdots, L_n]$ to a user with identity ID, he computes the corresponding decryption key as $(d_0, d_1, H(ID), L)$ where $r \leftarrow_R \mathbb{Z}_p$, d_0=g_2^{α} $(u_0^{H(ID)} u_1^{H(L_1)} \cdots u_n^{H(L_n)} g_3)^r$, and $d_1 = g^r$.

Decrypt(C,para) Upon receiving the ciphertext $C = (C', C_0, C_1, C_2, \{T_i\}_{W_i=*}, E)$, the user computes $C_2' = C_2 \prod_{W_i=*} T_i^{H(L_i)} E^{d_2}$ and get the symmetric key $K = C_0\hat{e}(d_1, C_2')/\hat{e}(d_0, C_1)$ with the decryption key $(d_0, d_1, H(ID), L)$.

Fig. 2: Algorithm Description

ID wants to join the system, he needs to be issued a private key on his attributes from the attribute authority. In concrete, the attribute authority assigns a set of attributes L by running the algorithm of *KeyGen*(ID, L, sk) in Fig.2. After receiving the attribute private key, the user can verify its correctness by using the bilinear pairing.

If the user is allowed to access data owned by some data owner, a long-lived secret key of the broadcast encryption scheme will be computed and sent to the user. The data owner also needs to update and encrypt the same r stored on the Cloud Servers by computing *Enc*$_{BE}(G', r)$ such that the new user set $G' = G \cup ID$.

File Access Suppose a user wants to access and retrieve files of his interest, he proceeds as follows: He first downloads the ciphertext C' of the broadcast encryption and decrypts to get r. Because the key r currently used is encrypted in the way that only the server and the set of currently authorized users can get decrypt C' and get r, the request $H(r, FIDs)$ is valid only if the user is not revoked. Then, he computes $H(r, FIDs)$ as a message authentication code (MAC), where $FIDs$ are the identities of the files to retrieve.

After receiving $FIDs$ and their MAC, the cloud servers verify its correctness by using r. If it is valid, then, the cloud servers send the requested encrypted files to the user. If the user has the privilege to access the encrypted file, he can decrypt to get the symmetric key K as depicted in the *Decrypt* algorithm of Fig.2. Then, he applies $Dec_{SE}(K, C)$ on the encrypted data C and retrieves the file F.

User Revocation Whenever there is a user ID' to be revoked, the data owner picks a new r' and stores it encrypted on the Cloud Servers by computing *Enc*$_{BE}(G, r')$ such that only user set G, including the non-revoked users and Cloud Servers can decrypt it. Additionally, the original encrypted value *Enc*$_{BE}(G \cup ID', r)$ with respect to $G \cup ID'$ will be deleted. This broadcast encryption is utilized here to prevent the user who was authorized but currently revoked in the system.

Trace Suppose an illegal device is found to be used to access the files stored in Cloud Servers. Let (d_0, d_1, d_2, d_3) be a valid decryption key in this device. Then, the third party auditor computes the hash values of identities for all authorized users. If there is a match between one hash value of identity and d_2, the data owner reveals the identity as the dishonest user and revoke this user. The validity of the tracing algorithm is based on the fact that the user cannot change his identity in his secret key inserted in the attribute private key.

B. Further Enhancements

1) How to Achieve Black-box Traitor Tracing: In the above construction, the identity of traitor will be detected based on the assumption that the decryption key is well-formed. More specifically, the dishonest user could release an obfuscated program (or simply a decryption box) which successfully decrypts the ciphertexts and yet does not contain the decryption key in any canonical form. We further show how to modify it to allow the traitor tracing in the black-box model by using the similar idea proposed in [13]. In particular, just by observing the input/output behavior of the given decryption box, a judge should be able to find out the dishonest user. The tracing algorithm will hone in on the ciphertexts that the key cannot decrypt in an attempt to catch a dishonest user. We revise the generic technique on the transformation from well-formed decryption key assumption to black-box model. The main idea of the revised technique can be sketched as follows:

By choosing appropriate m and k, the structure of attributes in a user key is formed as the attribute list L and m "parallel" repetitions where each consists of k dummy attributes randomly chosen by the data owner. Thus, the user's set of attributes will loosely look like $(L, A_1, \cdots, A_m)$, where L is the user's real attributes, while each A_i consists k identity-related dummy attributes. A ciphertext will have a similar attribute structure, which we can loosely write as $(W, A_1', \cdots, A_m')$ and each A_i' will also consist of k attributes for

$1 \leq i \leq m$. The access policy in this new technique can be stated as follows: A user can decrypt a ciphertext if and only if (L matches W) AND ($A_1 \cap A'_1$ contains at least γ attributes) AND $\cdots$ AND ($A_m \cap A'_m$ contains at least γ attributes).

To enforce this policy, our construction will make use of technique in the KP-ABE scheme of Goyal et. al. [14]. By appropriately choosing the number of dummy attributes k (in the decryption key and ciphertext) and the threshold γ as in [19], we can guarantee that a randomly encrypted ciphertext can be decrypted with high probability. The selection of parameter and analysis is the same with [19]. The only difference here is that the dummy attributes are chosen by the data owner, not in an oblivious transfer technique as in [19]. In this way, the data owner knows the dummy attributes and he can choose some special key policy to create a ciphertext such that the user cannot decrypt if the attribute private key embedded in the illegal device is from the suspicious user.

The tracing algorithm proceeds as follows based on the above observation. Suppose an illegal user has created a illegal device with his attribute private key. Assume that the third party auditor finds that this device can decrypt some ciphertext under access policy W. Then, the third party auditor extracts the attribute list $(L_{i_1}, L_{i_2}, \cdots, L_{i_k})$ out of W. The values in other positions except $\{i_1, i_2, \cdots, i_k\}$ in W are $*$. The third party auditor checks the issuing record of attribute private key and determines the suspicious users set S, who have the attributes $(L_{i_1}, L_{i_2}, \cdots, L_{i_k})$. There are two ways to pinpoint the exact identity from S:

- If the size of set S is not huge, then, the third party auditor just encrypts a message with respect to access policy W for each user in S until the identity is found. More specially, a carefully chosen dummy attribute set is chosen for each user in S such that only the selected suspicious user cannot decrypt. It can be easily seen that the user is able to decrypt the ciphertext only when he is not the suspicious user tracing.
- If $|S|$ is too huge, the tracing algorithm works in the following way: First, the third party auditor tries an attribute value L_j from the position j where $W_j = *$. Then, it encrypts a message as the normal encryption algorithm with respect to W' such that all positions are set to be $*$, except the positions of $\{i_1, i_2, \cdots, i_k, j\}$ are set to be $L' = L \cup L_j$. The ciphertext is sent to the pirate device. If the ciphertext can be decrypted correctly, the third party auditor knows one of the users with L' shares his attribute private key. The suspicious user set is of course not greater than $|S|$. The third party auditor continues the above procedure until the suspicious set $|S|$ is not too huge. Finally, the technique for small $|S|$ can be applied and the identity in the pirate device can be pinpointed.

Actually, based on the tracing algorithm, the scheme is secure against collusion attack, in which users with different attributes can collude to generate an illegal decoder. The tracing algorithm still works and at least one of the illegal users will be detected from the pirate device. Our definition and construction of tracing requires that the adversary produces a perfect pirate decoder device, namely a decoder that correctly decrypts all well-formed ciphertexts [11]. In reality, the pirate has a decoder that may work only a fraction of the time. When interact with such a decoder, just repeat the tracing algorithm for each suspicious identity such that the error-rate is lower than some predefined number.

V. Security and Performance Analysis

A. Security Analysis

Recall that two kinds of attackers are considered, that is, 1) security against external attackers: The security definition for this kind of attacker is captured by confidentiality. It means that, given access to a set of encrypted data, the external attackers, including the server, are not able to learn any partial information about the underlying files; 2) security against internal attackers: The security definition that is used to capture the internal attackers is user accountability. It means that if a user shares his decryption key, he will be traced. In another word, he cannot output and share a new decryption key for a different identity in the decryption keys issued from the data owner.

Before giving the security proof of the above construction, we present two assumptions, *i.e.*, CDH assumption and DBDH assumption, which our proof is based on.

CDH Assumption. The Computational Diffie-Hellman assumption is that, given g, g^x, $g^y \in \mathbb{G}_1$ for unknown random $x, y \in \mathbb{Z}_p^*$, it is hard to compute g^{xy}.

DBDH Assumption. The Decision Bilinear Diffie-Hellman assumption is that, given g, g^x, g^y, $g^z \in \mathbb{G}_1$ for unknown random $x, y, z \in \mathbb{Z}_p^*$, $T \in \mathbb{G}_2$, it is hard to decide if $T = \hat{e}(g, g)^{xyz}$.

Intuition of the Security Proof In this system, each user is issued an attribute private key on $L \| ID$, where L denotes an attribute list and ID denotes the user's identity, respectively. When a message is to be encrypted under a access policy W, a ciphertext is created with the underlying policy $W' = W \| *$. Users with attributes L matching the ciphertext policy can decrypt the ciphertext, even if their identities are different. This holds because the second part in W' is $*$, namely, "don't care" (This technique is used to keep the one-to-many property in ABE, even though different identities have been embedded in the attribute private keys for the same attributes). Because the anonymity of the attribute in the ciphertext, the adversary cannot tell the difference of the normal encryption algorithm and traitor tracing algorithm. For the external attackers, they may change the attribute list L in the attribute private key or pretend to be an authorized user to decrypt ciphertext. For internal attackers, they may change the identity L embedded in their private keys to escape detecting. To prove the security against the external attackers, we will classify three kinds of categories and prove the security against them respectively. To prove the security against these internal attackers, we will prove that the user cannot change his attribute-based private key on $L \| ID$ to $L^* \| ID^*$, where $(L, ID) \neq (L^*, ID^*)$. Next, we prove the security of the construction in more details.

1. Security against external attackers. We need to prove that, given access to a set of encrypted data, the external attackers are not able to learn any partial information about the underlying files. Recall that there are three kinds of external attackers here: 1) Cloud servers; 2) Revoked users; 3) Users with unmatched attributes.

- For revoked users, it is easy to prove the security because they cannot get new r' encrypted by the broadcast encryption with respect to eligible users, thus, they cannot generate valid message authentication code of the file IDs;
- For cloud servers, they have not attribute private key on any attribute set, that is, they cannot get any information of the secret key K used in the symmetric encryption. Therefore, they are not able to decrypt and get any information of the underlying files if the symmetric encryption is secure;
- For users whose attributes do not match the access policy, they cannot get the underlying secret key used in the symmetric encryption. As a result, they cannot decrypt to get the files though they have r';
- If they collude, based on the security of the original ABE, it already has been designed to prevent from such attack.

Based on the above analysis, the key point of the security proof is the underlying fine-grained ABE. We only need to prove that the ABE will not leak information of the underlying message. The formal definition has been given in [7], [10], [14], which is called indistinguishability against chosen message attack. More specially, after setup of the system and uploading of the data, the adversary submits some queries to retrieve the attribute private keys. Finally, $\mathcal{A}$ tries to distinguish if a ciphertext is an encryption to which one of two adaptively chosen messages under some chosen access policy.

Another stronger attack is the chosen ciphertext attack, in which the adversary is allowed to query decryption oracle additionally besides the attribute private key queries. As there are many generic techniques [12], [18] to transform schemes under chosen message attack to chosen ciphertext attack, we just prove the security against chosen message attack for simplicity.

2. Security against internal attackers. To trace the identity who shares the decryption key, security requirement of user accountability is defined additionally. To guarantee user accountability, it is required that even if a user has a decryption key, he cannot output and share a new decryption key for a different identity. The user accountability is defined through the following game of key unforgeability. Notice that the security model of indistinguishability cannot guarantee that the unforgeability of attribute private key. In the above security model, the user is unable to decrypt a ciphertext if his attributes does not match the access policy. However, it does not consider whether the user can change the identity in his decryption key. More specially, we should make sure that any user cannot decrypt ciphertext with the same attribute list while replacing the identity in the attribute private key with another one to avoid tracing. Therefore, to achieve user accountability, it should be ensured that the user can not change the identity inserted in his attribute private key.

The formal definition for key unforgeability is based on the following game involving an adversary $\mathcal{A}$:

- The simulator chooses a sufficiently large security parameter, and runs Setup to get a master secret key and public key. He retains secret key and gives public key to $\mathcal{A}$;
- $\mathcal{A}$ can perform a polynomially bounded number of queries to key generation oracle for private key on a pair of identity and attribute list;
- Finally, $\mathcal{A}$ outputs a decryption key for the challenge identity on some attribute list.

The simulator runs a sanity check on the forged secret key to ensure that it is well-formed. It aborts if the check fails. $\mathcal{A}$ wins the game if this identity has not been submitted to the key generation oracle. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins the game.

The security proof can be derived from [21], where two security aspects have been included, that is, key unforgeability and confidentiality. The key unforgeability and confidentiablity are based on the CDH assumption and DBDH assumption, respectively. For more details, the reader is referred to [21].

TABLE I: Computation Overhead of Data Owner.

New File Creation	$1\ Pair$+n_1-$MultExp + (n - n_1 + 2)Exp.$
New User Grant	n-$MultExp + 3Exp$+1 $Mult_{\mathbb{G}}$+ $n\ Hash$
User Revocation	$1\ Enc_{BE}.$

TABLE II: Computation Cost of File Access

User	n_1-$MultExp + 2$-$MultExp + 1\ Dec_{SE} + 1\ Dec_{BE}$
Server	$1\ Dec_{BE}$+1 $Hash$

B. Performance Analysis

We begin by estimating the cost in terms of basic cryptographic operations, as notated in Table II. Suppose there are n attributes defined by the data owner. Under this setting, we focus on the computational overhead of data owner and users. We now assess the performance of the proposed accountable access control scheme given in Section IV.

We define the notations of cryptographic operations that will be used in the following analysis:

- $Hash$: hash function into the group $\mathbb{Z}_p^*$.
- $Mult_{\mathbb{G}}$: one multiplication in group $\mathbb{G}$.
- $Exp_{\mathbb{G}}(\ell)$: one exponentiation g^s, where $g_1 \in \mathbb{G}_1$, $|s| = \ell$.
- m-$MultExp$: m-term exponentiations $\prod_{i=1}^{m} g_1^{s_i}$.
- $Pair$: pairing computation of $e(g_i, h_i)$, where $g_i \in \mathbb{G}_1$, $h_i \in \mathbb{H}$.
- m-$MultPair$: m-term pairings $\prod_{i=1}^{m} e(g_i, h_i)$.

1) Data Owner Computation Overhead: Recall that we use n to denote the number of attributes defined by the data owner. We also use n_1 to denote the number of wildcard in the access policy for a file. On the data owner side, there are four algorithms he should be involved in the system, including the New File Creation, New User Grant and User Revocation. To create a new file, the data owner need to run the algorithm *Encrypt*, as well as a broadcast encryption for authorized users in the system. Actually, he only need to compute the broadcast encryption once for all of his files if no user revocation happens. Otherwise, to revoke a user, the data owner only need to compute one broadcast encryption with respect to the updated authorized user group. The corresponding computation cost is shown in Table I.

2) Computation Overhead of Users and Cloud Server: Recall that we use n to denote the number of attributes defined by the data owner. We also use n_1 to denote the number of wildcard in the access policy for a file. On the data owner side, there are four algorithms he should be involved in the system, including the New File Creation, New User Grant and User Revocation. The corresponding computation cost is shown in Table II. For the cloud server, in the file access algorithm, he should decrypt the broadcast ciphertext to get r and compute the message authentication code. The efficiency analysis is given in Table II.

VI. Conclusion

This paper constructs an ABE based cryptography scheme for implementing fine-grained access control for clod computing. The constructed scheme enables user accoutability, which can be used to prevent illegal key usages.

The contribution is as follows. Firstly, we identify the need for fine-grained access control in cloud computing. We also demonstrate the challenges that existing technologies may be confronted with when tackling the need for fine-grained access control. Secondly, we propose a construction of an attributed based encryption scheme to implement fine-grained access control in cloud computing. Thirdly, we achieve user accountability by inserting user specifi information into users' attribute private keys. The incoperated user information in users's attribute private keys could be viewed as another default attribute. In this way, users will not share their decryption keys as users cannot change the user specific information in the attribute private keys. Fourthly, we perform a comprehensive security analysis with respect to data confidentiality and fine-grained access control. The analysis suggests that proposed scheme has implemented data confidentiality and fine-grained access control. Lastly, we conduct a performance analysis, which shows that the construction is efficient and practical.

This work is not without its limitations. The granularity is limited by the size of the attribute set that is associated with the encryption. When finer granularity is required, re-encryption is needed to associated with a bigger attribute set. The re-encryption inevitably incurs complex computation as most encryption schemes do.

References

[1] Amazon Web Services (AWS), Online at http://aws. amazon.com.

[2] Google App Engine, Online at http://code.google.com/appengine/.

[3] Microsoft Azure, http://www.microsoft.com/azure/.

[4] 104th United States Congress, Health Insurance Portability and Accountability Act of 1996 (HIPPA), Online at http://aspe.hhs.gov/ admnsimp/pl104191.htm, 1996, last access: July 16, 2009.

[5] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. *Identity-Based Encryption Gone Wild*. ICALP'06, LNCS 4052, pp. 300-311, Springer, 2006.

[6] Man Ho Au, Qiong Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong, and Guomin Yang. *Traceable and Retrievable Identity-Based Encryption*. ACNS'08, LNCS 5037, pp. 94-110, Springer, 2008

[7] John Bethencourt, Amit Sahai, and Brent Waters. *Ciphertext-Policy Attribute-Based Encryption*. IEEE Symposium on Security and Privacy'07, pp. 321-334, IEEE, 2007.

[8] Melissa Chase. *Multi-Authority Attribute Based Encryption*. TCC'07, LNCS 4392, pp. 515-534, Springer, 2007.

[9] M. Chase and S. S. M. Chow. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In ACM CCS, pages 121-130, 2009.

[10] Ling Cheung and Calvin Newport. *Provably Secure Ciphertext Policy ABE*. In CCS'07, Proceedings of the 14th ACM conference on Computer and communications security, pages 456-465, ACM, 2007.

[11] Benny Chor, Amos Fiat, and Moni Naor. *Tracing Traitor*. CRYPTO'94, LNCS 839, pp. 257-270, Springer, 1994.

[12] Eiichiro Fujisaki and Tatsuaki Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes*. CRYPTO'99, LNCS 1666, pp. 537-554, Springer, 1999.

[13] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. *Black-Box Accountable Authority Identity-Based Encryption*. CCS'08, pp. 427-436, ACM, 2008.

[14] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. *Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data*. CCS'06, pp. 89-98, ACM, 2006.

[15] Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. *Attribute-based Publishing with Hidden Credentials and Hidden Policies*. In Proc. of Network and Distributed System Security Symposium (NDSS), pp. 179-192, 2007.

[16] Jin Li, Kui Ren, Bo Zhu, and Zhiguo Wan, *Privacy-aware Attribute-based Encryption with User Accountability*. 12nd Information Security Conference (ISC), Springer, 2009.

[17] Rafail Ostrovsky, Amit Sahai, and Brent Waters. *Attribute-based Encryption with Non-Monotonic Access Structures*. CCS'07, pp. 195-203, ACM, 2007.

[18] Amit Sahai. *Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen Ciphertext Security*. IEEE Symp. on Foundations of Computer Science, 1999.

[19] Amit Sahai and Brent Waters. *Fuzzy Identity-Based Encryption*. EUROCRYPT'05, LNCS 3494, pp. 457-473, Springer, 2005.

[20] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, IEEE INFOCOM, 2010.

[21] Jin Li, Kui Ren, and Kwangjo Kim. *A2BE: Accountable Attribute-Based Encryption for Abuse Free Access Control*, available at http://eprint.iacr.org/2009/118.