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.2017.2751471, IEEE Transactions on Cloud Computing

A Scalable Attribute-Based Access Control Scheme with Flexible Delegation cum Sharing of Access Privileges for Cloud Storage

Rohit Ahuja and Sraban Kumar Mohanty, Member, IEEE

Abstract—Nowadays cloud servers have become the primary choice to store and share data with multiple users across the globe. The major challenge in sharing data using cloud servers is to protect data against untrusted cloud service provider and illegitimate users. Attribute-Based Encryption (ABE) has emerged as a useful cryptographic technique to securely share data with legitimate recipients in fine-grained manner. Several solutions employing ABE have been proposed to securely share data using cloud servers. However, most of the solutions are data owner-centric and focus on providing data owner complete control on his outsourced data. The existing solutions in cloud computing fail to provide shared access privileges among users and to enable cloud users to delegate their access privileges in a flexible manner. In order to simultaneously achieve the notion of fine-grained access control, scalability and to provide cloud users shared access privileges and flexibility on delegation of their access privileges, we propose a scalable attribute-based access control scheme for cloud storage. The scheme extends the ciphertext policy attribute-based encryption to achieve flexible delegation of access privileges and shared access privileges along with scalability and fine-grained access control. The scheme achieves scalability by employing hierarchical structure of users. Furthermore, we formally prove the security of our proposed scheme based on security of the ciphertext-policy attribute-based encryption. We also implement the algorithm to show its scalability and efficiency.

Index Terms—Cloud computing, delegation, access privileges, data security, attribute-based encryption

1 Introduction

Cloud computing is a promising model to provide unlimited computation and storage capacity to its users, anytime from anywhere. Cloud computing systems benefit enterprises by providing scalable and durable resources. In addition, it reduces the total cost of ownership by transforming the business from capital expenses to operational expenses model [1]. Although cloud computing offers great convenience to its users by providing scalable and durable resources at a reduced rate but due to security and privacy issues, users are reluctant of migrating to the cloud [2], [3]. Enterprise outsources its valuable data on cloud servers and wants Cloud Service Provider (CSP) to store data, perform critical business operations on the required datasets. In addition, CSP should also entertain data requesters with the required information. Data is a crucial asset for any business enterprise, disclosure of which to CSP or illegitimate recipients may cause great loss. Hence, among all the possible security threats in cloud, most prominent and discussed issue, which has really obstruct its acceptance among potential cloud users is data privacy and confidentiality [4].

Furthermore, with the migration of enterprise confidential data on cloud servers, data confidentiality is not the only security requirement. The strongly desired feature is to provide associated employees access to the outsourced data based on their assigned roles and responsibilities, i.e., fine-grained access control. In organizations, certain roles can be mutually addressed by a group of users. Rather than trivially generating the copies of access privileges for every individual in the group, it is preferable to subdivide the access privileges among group members, i.e., to provide them Shared Access Privileges (SAP) to inclusively access the information. Firstly, it facilitates access control solution by dividing trust among users of a group. Next, it saves the key issuing authority from unnecessary computation [5]. Nowadays, in enterprises the concept of collaboration have gained importance because it benefits the overall firm with the development of trust and distributing workload among employees, which catalyzes the productivity and innovation [6], [7]. The concept of collaboration can be brought into practice by allowing employees to delegate their roles among associates, i.e., many to many delegation [8]. Based on the number of delegators and delegatees many to many delegation can be further categorized as: group delegation and many to one delegation. In group delegation, an individual delegator delegates his role to the appointed group of delegatees [9], [10]. In many to one delegation, group of delegators delegates their role to an appointed delegatee. The imperative concern with every delegation is that, every employee should be allowed to delegate his access privileges for a determined period of time, i.e., control delegation.

Traditional techniques to share confidential data outsourced to untrusted storage is to store encrypted data on untrusted servers and the decryption keys are disclosed

R. Ahuja and S. K. Mohanty are with the Department of CSE, Indian Institute of Information Technology Design and Manufacturing, Jabalpur, MP, 482005 India e-mail: {r.ahuja, sraban}@iiitdmj.ac.in

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

to authorized users only [11]. This solution gives rise to several problems, such as efficient key-management problem to distribute decryption keys among authorized users. The key-issuing authority will be overburdened due to the generation of keys with the growing number of users. To revoke an authorized user, related data need to be re-encrypted and new keys need to be generated and distributed among rest of the authorized users. Hence, data owner needs to be online always to encrypt or re-encrypt data and distribute keys to legitimate users [11]. Attribute-Based Encryption (ABE) schemes are widely accepted to provide access control solutions for cloud computing [12], [13]. Unlike the traditional schemes, both ciphertext and decryption key are associated with set of attributes or access policy. User can decrypt the ciphertext if set of attributes satisfies the access policy. Since, introduction of ABE, many variants of ABE have been proposed such as, Key-Policy Attribute-Based Encryption (KP-ABE) and ciphertext-policy Attribute-Based Encryption (CP-ABE). In KP-ABE, message is encrypted corresponding to a set of attributes and decryption key is associated with an access policy. In CP-ABE message is encrypted corresponding to an access policy and ciphertext is associated with a set of attributes.

Another branch of ABE research is the multi-authority ABE [14], [15], [16], [17], [18], [19], where the attributes are managed and issued by multiple authorities. These ABE systems do not consider the issue of flexible delegation of access privileges and shared access privileges.

Several schemes employing ABE techniques have been proposed for secure sharing of data in cloud storage [11], [20], [21], [22]. A scheme employing KP-ABE was proposed to enable data owner to selectively share data using cloud servers [20]. This scheme overburdens both data owner and recipients due to key management problem. Hence the scheme is not scalable. ABE schemes are extended to achieve scalability by taking the advantage of hierarchical structure of an enterprise [11], [23]. In mobile cloud environment, ABE Schemes are employed to achieve data privacy and restrict unauthorized access [24]. However, these schemes are far away from providing cloud users flexibility to delegate their access privileges and shared access privileges among group of users to jointly address the role.

Our Contribution: In this paper, we propose a scalable attribute-based access control scheme for cloud storage. To the best of our knowledge this paper is the first to define and solve the challenging issue of flexible delegation of access privileges, i.e., group delegation, many to one delegation, many to many delegation, control delegation and shared access privileges along with scalability, fine-grained access control, access privileges update and efficient user revocation by employing CP-ABE scheme. Our proposed scheme incorporates CP-ABE with a hierarchical structure to achieve scalability by decentralizing the key issuing authority at different levels of hierarchy, i.e., lower level users obtain secret keys from the users higher in the hierarchy. We formally prove that the security of our proposed scheme is based on the security of CP-ABE scheme, which has been proven to be secure under the generic bilinear group model and the random oracle model [13]. Shared access privileges and hierarchical structure may give rise to cheating and collusion attacks respectively. We also prove that our scheme is resistant against such attacks. We analyze the performance of our scheme in terms of computation, storage overhead and compare it with other competing schemes. Further, we implement our scheme to evaluate its performance and experimental results illustrate that our scheme has satisfying performance.

Organization of the paper: The rest of the paper is organized as follows. Section 2 discusses background study of ABE and existing solutions in cloud computing employing ABE schemes. Section 3 discusses the system model and assumptions. Section 4 focuses on the proposed scheme and the features of our scheme is discussed in section 5. Security analysis of our scheme is given in Section 6. The computational, storage overhead of our scheme and empirical analysis are discussed in section 7. Lastly, in Section 8 we conclude the paper by providing a brief summary of our contributions.

2 Related Work

This section reviews Attribute Based Encryption (ABE) schemes and analyzes ABE based access control solutions for cloud computing.

2.1 Attribute Based Encryption

The concept of ABE was initiated with Identity-Based Encryption (IBE) [25], [26]. In IBE sender encrypts a message corresponding to the identity of an individual recipient, such as $alice@gmail.com$ for a recipient $Alice$. Alice receives his key from the key-issuing authority corresponding to this id. There is a single key issuing authority in IBE due to which the scheme lacks scalability. Hierarchical Identity-Based Encryption (HIBE) overcomes this limitation by decentralizing the key issuing authority at different levels of hierarchy [27]. Every user in the hierarchy obtains key from the user higher in the hierarchy. In IBE and HIBE both message and user's access privileges are associated with an unique id due to which the scheme lacks one to many encryption. Fuzzy Identity-Based Encryption (FIBE) overcomes this limitation by associating both user's access privileges and ciphertext with identities, i.e., set of attributes [28]. For example, user $Alice$ is granted access privileges corresponding to an identity X(set of attributes) and message is encrypted corresponding to an identity $X^{'}$(set of attributes). Alice can access the message if and only if $|X \cap X^{'}| = d$. $|X \cap X^{'}|$ is the only supported threshold gate by FIBE scheme and d is the threshold value, which is fixed at the initiation of the scheme. However, the scheme supports single threshold gate, due to which it lacks expressibility for identities accessing the message. Hence, it has limited applicability in access control of data [12].

To resolve the issue of expressibility, ABE schemes were proposed [12], [13]. The primary elements in ABE schemes

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

are: set of attributes and access policy. Access policy is the tree structure defined over attributes. In ABE schemes, user's key or ciphertext is associated with one of the primary element. User is able to access the message, if there is a match between the access policy and the set of attributes.

In ABE the ongoing research is broadly categorized as single-authority and multi-authority schemes. The pioneering multi-authority ABE scheme was introduced by employing the concept of a trusted central authority. In this scheme, the central authority has access to all the encrypted message due to which the scheme lacks security of encryption and user's privacy [14]. To improve the security of encryption and user's privacy, a multi-authority scheme removes the concept of trusted central authority and restricts users to collude their access privileges [15]. ABE schemes are also decentralized to achieve the concept of multi-authority and allows any party to become an authority without coordination with other parties [17]. This scheme also restricts malicious user to collude their access privileges to jointly access the message. A multi-authority CP-ABE scheme traces the identity of malicious user who degrades the security and privacy of users by compromising his decryption privileges for personal gain [18]. However, these schemes are proceeding in a different direction to the proposed scheme.

ABE schemes are further classified on the basis of association of a set of attributes or an access policy with user's key or ciphertext into two major categories: KP-ABE and CP-ABE. In KP-ABE, key is associated with an access policy and ciphertext is associated with a set of attributes. In this scheme, data owner selects a set of attributes and encrypts the message with chosen set before outsourcing the corresponding ciphertext to the third party storage. When a new user Alice joins the system, the key issuing authority selects attributes and defines an access policy based on those attributes. After that, key issuing authority grants Alice a decryption key associated with the access policy. Alice gets access to the desired message if and only if the set of attributes with the ciphertext, satisfies the access policy with her key. The major disadvantage of this scheme is that the data owner loses control on his outsourced data, despite of describing some expressive attributes for data. Data owner needs to rely on key-issuer to control the access of outsourced data [13]. Hence, it has limited applicability in access control of outsourced data.

CP-ABE is introduced with a notion to resolve the limitation of KP-ABE. In CP-ABE scheme, data is associated with an access policy and user's key is associated with a set of attributes. In CP-ABE, data owner selects attributes and defines an access policy based on that attributes. When a new user Alice joins the system, key-issuing authority selects a set of attributes corresponding to the Alice's identity and grants her decryption key corresponding to the chosen set of attributes. Alice is allowed to access the message if the set of attributes associated with her key satisfies the access policy associated with ciphertext.

CP-ABE is very close to the requirement of modern enterprise, but the straightforward application of CP-ABE is infeasible in cloud environments due to the following shortcomings:

1) Lack of scalability: As there is a single key issuing authority responsible for generating and distributing access privileges among all the users of the system.
2) Do not provide users flexibility to delegate their access privileges, i.e., Group Delegation, Many to One delegation, Many to Many and Control delegation.
3) Do not support shared access privileges for users.

2.2 Access control solutions for cloud computing

The major challenge in providing access control solutions for cloud computing is fine-grained access control, scalability and efficient user revocation. Traditional scheme to revoke a user Alice from further accessing data, requires data owner to re-encrypt the entire data related to Alice. In addition, new keys need to be generated and distributed among rest of the users. This method overburdens data owner due to the heavy computation involve in re-encryption of data and generation of new keys [11], [20].
Several efforts have been made by employing ABE schemes to resolve these challenges [11], [20]. A scheme employing KP-ABE was proposed to achieve fine-grained access control [20]. The major challenge in incorporating KP-ABE is that the data owner loses control on his outsourced data. Data owner needs to rely on key-issuer to control the access of his outsourced data. To overcome this challenge, data owner is solely responsible for generating and distributing the access privileges among recipients. To efficiently handle user revocation, scheme delegates the computation tasks involved in re-encryption of data to CSP and update corresponding access privileges of all the users except the revoked ones efficiently. This scheme has several shortcomings. Firstly, as the number of data recipients increases, data owner will be overburdened due to the workload involved in key-generation and distribution. Next, this scheme is difficult to be applied in many application scenarios, such as health care on cloud. When a medical practitioner needs to gather health information of patients suffering from cancer, he has to request secret keys from every patient. This will overburden practitioner due to the key-management problem.

HASBE scheme was proposed to achieve flexible, scalable and fine-grained access control in cloud computing by employing ABE scheme [29]. To achieve scalability this scheme took the advantage of hierarchical structure of an organization and shifts the workload of key generation and distribution from single higher level authority to multiple lower level domain authorities [11]. In addition, this scheme also supports efficient revocation of user from further accessing data. A flexible system privilege revocation scheme addresses the concept of restricting user from further sharing or accessing data using cloud servers in a flexible manner [23]. To efficiently address this concept, this scheme outsources the computation involved in re-encryption of data to cloud servers and updation of

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

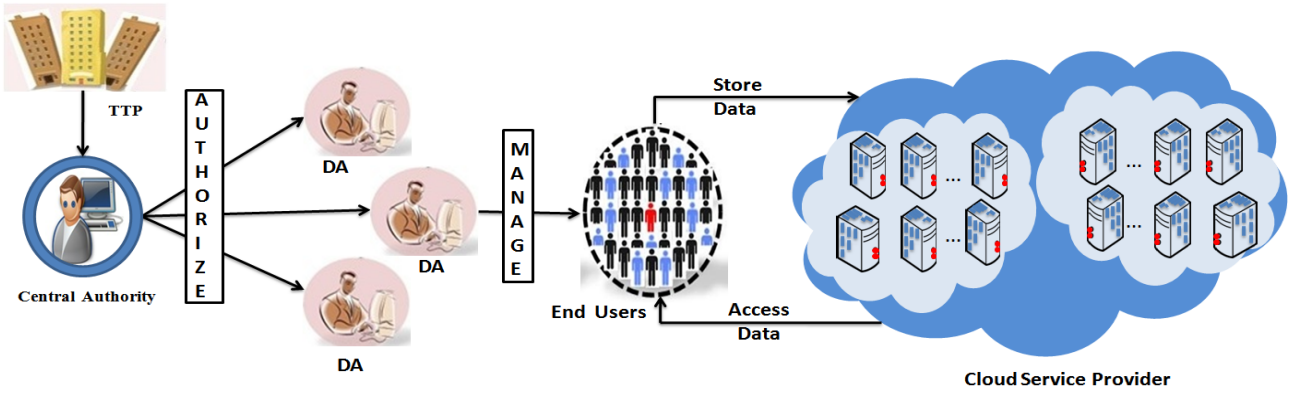


Fig. 1. System Model

keys to individual users.

Precise and Fuzzy Identity-Based Encryption (PFIBE) scheme was proposed by combining CP-ABE and HIBE for secure sharing of data on cloud servers [22]. This scheme achieves scalability, fine-grained access control and full-key delegation. Full-key delegation allows the higher level authority to define the semantics of access privileges while providing it to lower level authority. HABE scheme was also proposed in cloud computing by combining CP-ABE, HIBE to achieve fine-grained access control, scalability and full-key delegation [30]. In addition, this scheme achieves scalable user revocation by outsourcing the computation involved in re-encryption of data to cloud servers. These schemes use disjunctive normal form policy and assumes all attributes in one conjunctive clause are administrated by the same domain master. Thus the same attribute may be administrated by multiple domain masters according to specific policies, which is difficult to implement in practice [11].

A CP-ABE based scheme enables user to query CSP for the desired data, while CSP executes the query and returns appropriate data as a response to the user [31]. This scheme preserves the privacy of data owner and consumer by concealing their identities. However, the subject of shared access privileges and flexible delegation of access privileges remain unnoticed, which is the foremost requirement in enterprises.

Nowadays, researches regarding multi-authority ABE scheme are also gaining importance in cloud environment [16], [19]. A multi-authority scheme outsources the heavy computation involved in decryption to cloud servers so that the users with low computing capability devices, such as smart phones can also access data [19]. To conveniently share personal health record on cloud, a multi-authority scheme focuses on key-management issues for data owners and consumer along with patient privacy and scalability [16]. A fully anonymous multi-authority scheme was proposed to provide user's identity privacy along with data privacy [32]. However, these schemes are progressing in the different direction as of ours.

3 SYSTEM MODEL AND ASSUMPTIONS

3.1 System Model

The cloud computing system under consideration consists of following five parties: Trusted Third Party (TTP), Central Authority (CA), number of Domain Authorities (DA), users and Cloud Service Provider (CSP). Users are further classified into data-owners and data-consumers. The hierarchical formation of the system under consideration is illustrated in Fig. 1. CA is the top level authority, such as the head of the federated enterprise, while DA is the head of lower level organization such as the associated company in a federated enterprise. End users are the employees associated in every domain. All the users in the enterprise are data owners/consumers, i.e., every individual in the enterprise can share data with others and receive from anyone using cloud servers.

The TTP acts as a center of trust and responsible for managing CA, who in turn manages the activities of DA. DA manages the end-users in their individual domain. To share data files, users in the enterprise encrypt files and store it on cloud, while to access them, users download the desired files from cloud and then try to decrypt them.

In our system, all the entities in the enterprise, except end users are responsible for managing the lower lever authority in the hierarchy. End users need not be online always. End users come online, when they want to share or retrieve data. TTP, CSP, CA and DA are always online. This assumption is similar to various former researches in cloud security [11], [20].

3.2 Trust Model:

Every entity depicted in Fig. 1, possesses public and private key pair. Public key is known to all the users of the system, while private key is kept secret with the entity. TTP authorizes CA, which in turn authorizes DA. End-users are granted Exclusive Access Privileges (EAP) corresponding to their individual role, while Shared Access Privileges (SAP) are granted to participants of a group in a domain corresponding to the role they jointly addresses by administrating DA. Participants belonging to a group are

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

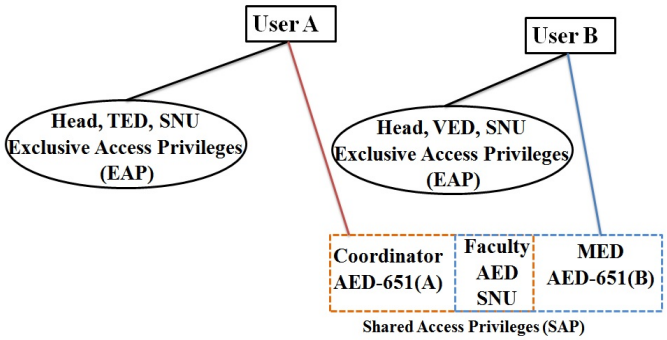


Fig. 2. Key Structure showing EAP and SAP corresponding to user's exclusive and inclusive access privileges respectively

authorized to use SAP inclusively only.

CSP is assumed to be untrusted, i.e., CSP may collude with malicious end users to access the files for personal gain. DA acts as a trusted entity for users in his domain but may collude with malicious end-users outside his domain to harvest the file contents. Malicious DAs may collude to generate the access privileges beyond the scope of their individual domain. Malicious end users may try to collude their EAP to access the files, which exclusively they cannot. Moreover, dishonest participants of a group may try to incorporate their access privileges in an unauthorized manner. For instance, let us suppose group of k users, such that $G_K = \{U_1, U_2, ..., U_k\}$ are granted SAP to inclusively access a message M. Malicious group of n users $G_n : \{G_n \subset G_k, G_n \neq \phi\}$ may do following attacks: (i) Collusion Attack: G_n dishonest participants collude with illegitimate participants, i.e., participants outside the group to access message M (ii) Cheating Attack: G_n dishonest participants may cheat to group $G_x : \{G_x = G_k - G_n\}$ participants by trying to access M for which only G_k participants are inclusively permitted. Similar attacks may be essayed by nefarious delegatees in group Delegation.

4 Our Proposed Scheme

4.1 Preliminaries

Definition 1 (Bilinear Maps [13]). *Let G_1, G_2 be two multiplicative cyclic group with prime order p. Let g be the generator of the group G_1. Then $\hat{e} : G_1 X G_1 \rightarrow G_2$ be the bilinear map, if it satisfies the following properties.*

1) *Bilinearity: for all $P, Q \in G_1$ and a,b $\in Z_p, \hat{e}(P^a, Q^b) = e(P, Q)^{ab}$*
2) *Non-degeneracy: $e(g, g) \neq 1$*
3) *Computable: There exist a polynomial time algorithm to compute $\hat{e}(P, Q) \in G_2$, for any $P, Q \in G_1$.*

Key Structure: The key-structure in proposed scheme is an extension to the key-structure in CP-ABE. Similar to CP-ABE, key is a collection of attribute elements. Attributes present in the key corresponds to the data access privileges of key-owner. Data access privileges of user's key is classified into two categories: Exclusive Access Privileges (EAP) and Shared Access Privileges (SAP). EAP corresponds to a role an individual solely addresses and SAP corresponds to a role, which an individual jointly addresses with other users. Consider the example, let us suppose in university "SNU", four specialization are active under Mechanical Engineering Department (MED), i.e, Thermal Engineering Department (TED), Vehicle Engineering Department (VED), Acoustical engineering department (AED) and Manufacturing Engineering Department. User A is the head of TED and User B is the head of VED. In addition, both the users serve as a faculty in AED for the course code AED-651. AED-651 course is subdivided into AED-651(A) and AED-651(B) and instructed by user A and B respectively. Moreover, both the users jointly act as a coordinator for MED. As depicted in Fig. 2, Key-structure of user-A has EAP corresponding to head of TED, i.e., {Head, TED, SNU} and B has EAP corresponding to head of VED, i.e., {Head, VED, SNU}. To jointly address the responsibility of coordinator, user A and B has SAP for coordinator of MED, i.e., {MED, Coordinator, SNU} and as instructor of AED, i.e., {Faculty, SNU, AED, AED-651(A), AED-651(B)}. Hence the attributes for SAP are subdivided between User A and B. In SAP, user A and B are also provided exclusive privileges for the couse code AED-651(A) and AED-651(B), i.e., {Faculty, SNU, AED, AED-651(A)} and {Faculty, SNU, AED, AED-651(B)} respectively.

Access Policy: In our scheme, policy is the tree structure T representing an access structure. The tree structure is similar to CP-ABE scheme. Each leaf node of the tree is associated with attributes and each non-leaf node represents a threshold gate. The threshold gate of a non-leaf node is described by its children and threshold value. For example, let num_x be the number of childrens of node x and threshold value is K_x, such that $0 \leq K_x \leq num_x$. Every children of non leaf node x is labeled from 1 to num_x. For every leaf node $K_x = 1$.
To conveniently deal with the operation on access tree, several functions are defined, such as $parent(x)$, $index(x)$ to return the parent and index of a node x and $att(x)$ returns the attribute associated with x, where x is a leaf node
$T_x(S)$ is computed recursively, which is described as follows:-
If x is a leaf node

$$T_x(S) = \begin{cases} 1 & \text{if } att(x) \in S. \\ 0 & \text{otherwise.} \end{cases}$$

If x is a non-leaf node and $x^{'}$ is a child node of x

$$T_x(S) = \begin{cases} 1 & \text{if } K_x \text{ number of } T_{x'}(S) = 1. \\ 0 & \text{otherwise.} \end{cases}$$

4.2 Functionalities of Our Proposed Scheme:

The proposed scheme extends CP-ABE to handle the hierarchical structure of users in an organization, i.e., Central Authority, Domain Authority and users, as illustrated in Fig. 1. Now we discuss the main operations of our proposed scheme
System Setup: To generate the system public parameters PK and master key MK, TTP calls Algorithm 1. PK is made public to all the users and MK is kept secret with the TTP.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

Algorithm 1 Setup: Generates the public parameter PK and master key MK [13].

Input: Group $(G_1,\ g)$, G_1 is a bilinear group of prime order p,with generator g.
Output: System public parameters PK and master key MK of TTP.

1: Select two random numbers $\alpha,\ \beta \in Z_p$.
2: Generate the system public parameters PK and master key MK as:
$PK = (G_1,\ g,\ h = g^{\beta},\ f = g^{1/\beta},\ e(g,g)^{\alpha},\ H : \{0,1\}^* \rightarrow G_1)$
$MK = (g^{\alpha},\ \beta)$

Central Authority Grant: Central authority is associated with a unique ID and set of attributes S ={a_1 , a_2,...,a_n}. Whenever CA requests to join the system, TTP verifies the authenticity of CA then calls Algorithm 2 to generate the master key for the CA.

Algorithm 2 GenCA: Central Authority key generation [13].

Input: Master key MK of TTP and the set of attributes S of CA.
Output: Master key MK_{CA} of Central Authority.

1: Select unique random number $r^{CA} \in Z_p$ for CA and different random numbers $\{r_i^{CA} \in Z_p : \forall a_i \in S\}$.
2: Generate the master key of central authority as:
$MK_{CA} = \{D^{CA} = g^{(\alpha + r^{CA})/\beta},\ \forall a_i \in S : D_i^{CA} = g^{r^{CA}}.H(a_i)^{r_i^{CA}},\ D_i^{'CA} = g^{r_i^{CA}}\}$

Furthermore, after getting the master key, CA authorizes the Domain Authority.
Domain Authority Grant: When a new domain authority DA_j joins the organization, CA verifies the authenticity of domain authority and then provides a unique ID and set of attributes $S_{DA}^j : S_{DA}^j \subset S$ to DA DA_j, where S is the key structure of CA. Then CA calls Algorithm 3 to generate the master key for DA.

Algorithm 3 GenDA:Domain Authority Key Generation [13].

Input: Master key MK_{CA} of Central Authority and the set of attributes S_{DA}^j corresponding to DA DA_j.
Output: Master key MK_{DA_j} of Domain Authority.

1: Select unique random number $r_j^{DA} \in Z_p$ for DA DA_j, different random numbers $\{r_k^{DA} \in Z_p : \forall a_k \in S_{DA}^j\}$.
2: The elements of master key of DA is generated as:
$D^{DA} = D^{CA}.f^{r_j^{DA}},\ D_k^{DA} = D_k^{CA}.g^{r_k^{DA}}H(a_k)^{r_k^{DA}},\ D_k^{'DA} = D_k^{'CA}.g^{r_k^{DA}} : \forall a_k \in S_j^{DA})$
3: Generate the master key of domain authority as:
$MK_{DA_j} = (D^{DA}, \{D_k^{DA}, D_k^{'DA}\}_{\forall a_k \in S_{DA}^j})$

DA is responsible for administrating the users in its domain. In addition, DA also maintains the state of users in its domain, i.e., expiry date of users and their privileges.

User Grant: DA generates secret keys for users in its own domain. Firstly DA verifies user U_l in his domain and further based on his roles and responsibilities, assigns him set of attributes S_U^l for exclusive access privileges. Let us suppose group of l users in a domain jointly address a role and are inclusively authorized to access set of files corresponding to that role. DA defines set of attributes S_S for group of l users, who jointly addresses the role and provides S_S^l to user U_l corresponding to shared access privileges. DA_j also assigns every user a unique ID. Then DA_j calls Algorithm 4 to generate the secret key for the users in his domain. If U_l is not jointly addressing any role with other users then, $S_S^l = \phi$

Algorithm 4 GenUsr: User key generation

Input: Master key MK_{DA} of DA associated with set of attributes S_{DA} and set of attributes, S_U^l and S_S^l.
Output: Secret key for user U_l, i.e., SK_{U_l}.

1: Select a random number $r_l^U \in Z_p$ for user U_l, $\bar{r}$ for group of users, who jointly addresses the role and different random numbers $\{r_t \in Z_p : \forall a_t \in S_U^l,\ S_S^l\}$.
2: The secret key elements for user U_l is generated as:
Key components corresponding to EAP:
$D^U = D^{DA}.f^{r_l^U},\ D_t^U = D_t^{DA}.g^{r_l}H(a_t)^{r_t},\ D_t^{'U} = D_t^{'DA}.g^{r_t} : \forall a_t \in S_U^l$
Key components corresponding to SAP:
$D^S = D^{DA}f^{\bar{r}},\ D_t^S = D_t^{DA}.g^{\bar{r}}H(a_t)^{r_t},\ D_t^{'S} = D_t^{'DA}.g^{r_t} : \forall a_t \in S_S^l$
3: Generate the secret key SK_{U_l} for user U_l as:
$SK_{U_l} = (D^U, \{D_t^U, D_t^{'U}\}_{\forall a_t \in S_U^l},\ D^S, \{D_t^S,\ D_t^{'S}\}_{\forall a_t \in S_S^l})$

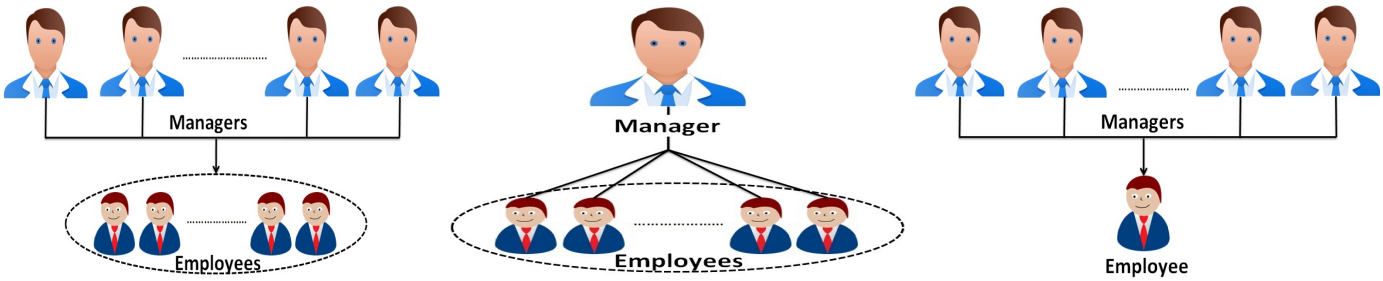
Delegation: Group of delegators delegate their roles to group of delegatees, e.g., as depicted in Fig. 3(a), group of managers (delegators) delegate their roles to group of employees (delegatees). Hence, delegators need to delegate their access privileges to the group of delegatees. Let us suppose, group of M delegators with keys $(SK_{U_1}, ..., SK_{U_M})$ associated with set of attributes $(S_1,\ S_2, ...,\ S_M)$ respectively delegate their access privileges to group of N delegatees. Group of M delegators proceeds as follows:

- Each group member $U_l, \forall l \in \{1, ..., M\}$ selects specific set of attributes $\tilde{S}_l, \tilde{S}_l \subset S_l$ to delegate it to $U_d, \forall d \in \{1, ..., N\}$ delegatees.
- Every individual U_l in the group of M delegators, encrypts the tuple T $((ID_d)_{\forall d},\ ID_l,\ \tilde{S}_l,\ \delta_{U_l}((ID_d)_{\forall d},\ ID_l,\ \tilde{S}_l))$ using public key of the DA, where (ID_d) is the unique ID of d^{th} delegatee, ID_l is the unique ID of the l^{th} delegator and $\delta_{U_l}(X)$ is the u_l^{th} delegator's signature on 'X'.

On receiving the tuple T, DA verifies $(\delta_{U_l}(ID_d)_{\forall d},\ ID_l,\ \tilde{S}_l)$. Then, DA verifies whether the claimed set $\tilde{S}_l \subset S_l,\ \forall$ and then calls Algorithm 5.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing



(a) Many to Many Delegation (b) Many to One Delegation (c) Group Delegation

Fig. 3. Flexible Delegation

Algorithm 5 Delegation: To generate access privileges for delegation.

Input: Master key of DA, i.e., MK_{DA} associated with set of attributes S_{DA}.
Output: Secret key for delegatees.

1: Let M be the number of delegators, $l \in \{1, ..., M\}$ denotes a delegator and $d \in \{1, ..., N\}$ denotes a delegatee.
2: **if** $M > 1$ **then**
3: **switch** N **do**
4: **case** $N > 1$:{Many to Many delegation}
5: Generate set of attributes S_{Del} : $S_{Del} = \bigcup_{l=1}^{M} S_l$, where $S_{Del} \subset S_{DA}$.
6: Generate N sets of attributes S_d : $S_d \subset S_{Del}$, $\forall d \in \{1, 2, ..., N\}$, such that $\bigcup_{d=1}^{N} S_d = S_{Del}$.
7: Select unique random number $\tilde{r}$ for group of N delegatees and different random numbers $r_l \in Z_p$, $\forall a_l \in S_d : \forall d \in \{1, 2, ..., N\}$
8: The secret key elements for an individual delegatee d is generated as: $D^d = D^{DA}.f^{\tilde{r}}$, $D_l^d = D^{DA}.g^{\tilde{r}}H(a_l)^{r_l}$, $D_l^{'d} = D_l^{'DA}.g^{r_l} : \forall a_l \in S_d$
9: The secret key of N delegatees is generated as: $(SK_d = D^d, D_l^d, D_l^{'d})_{\forall d}$
10: **case** $N = 1$:{Many to One delegation}
11: Generate set of attributes S_{Del} : $S_{Del} = \bigcup_{l=1}^{w} \tilde{S}_l$, where $S_{Del} \subset S_{DA}$
12: Select unique random number $\tilde{r} \in Z_p$ for delegatee and different random numbers $\{r_l \in Z_p : \forall a_l \in S_{Del}\}$.
13: The secret key elements for the delegatee is: $D = D^{DA}.f^{\tilde{r}}$, $D_l = D_l^{DA}.g^{\tilde{r}}H(a_l)^{r_l}$, $D_l^{'} = D_l^{'DA}.g^{r_l} : \forall a_l \in S_{Del}$
14: The secret key for the delegatee is defined as: $SK = (D, \{D_l, D_l^{'}\}_{\forall a_l \in S_{Del}})$
15: **else if** $M = 1 \& N > 1$ {Group delegation} **then**
16: Select set of attribute $\tilde{S} : \tilde{S} \subset S_U$ and generate N sets of attributes $\tilde{S}_d, \forall d \in \{1, 2, ..., N\}$, where $\tilde{S}_d \subset \tilde{S} : \bigcup_{d=1}^{N} \tilde{S}_d = \tilde{S}$.
17: Select unique random number $\hat{r} \in Z_p$ for group of N delegatees and different random numbers $(r_k \in Z_p, \forall a_k \in \tilde{S}_d)_{\forall d}$
18: The secret key elements of an individual delegatee d is generated as: $D^d = D^U.f^{\hat{r}}$, $D_k^d = D_k^U.g^{r_k}.H(a_k)^{r_k}$, $D_k^{'d} = D_k^{'U}.g^{r_k} : \forall a_k \in \tilde{S}_d$
19: Generate secret keys for m delegatees as : $(SK_d = D^d, \{D_k^d, D_k^{'d}\}_{\forall a_k \in \tilde{S}_d})_{\forall d}$
20: **end if**

Control Delegation: To delegate the access privileges for a determined time period, firstly delegator encrypts the tuple $T : (ID_u,\ Time\_span,\ \delta_u(Id_u,\ Time\_span))$, where Id_u is the unique identity of delegator, $Time\_span$ is the time-period for which the delegator wants to delegate his access privileges to delegatee and $\delta_u(X)$ is the signature of delegator u on 'X', using public key of the DA. DA calls Algorithm 6 to generate the needful parameters, which enables the delegator to delegate his access privileges for a determined time period.

Algorithm 6 DlgtCtrl: Generate key elements corresponding to required expiry date for delegating user.

Input: Tuple T $(Id_u, Time\_span, \delta_u(Id_u,\ Time\_span))$
Output: Key elements $D_t,\ D_t^{'}$ corresponding to $Time\_span$.

1: DA decrypts the tuple T, using his private key.
2: DA verifies $\delta_u(Id_u, Time\_Span)$ if correct proceeds.
3: DA identifies expiry date of delegator's access privileges, i.e., $Id_{Expiry-date}$.
4: **if** $(Time\_Span \leq Id_{Expiry-date})$ **then**
5: Generate key elements $D_t,\ D_t^{'}$.
6: DA encrypts the tuple $T^{'}$ $(D_t,\ D_t^{'},\ \delta_{DA}(D_t,\ D_t^{'}))$ using public key of delegator and forwards $T^{'}$ to him, here $\delta_{DA}(X)$ is the DA's signature on "X".
7: **end if**

Delegator on receiving T$^{'}$, decrypts it and verifies $\delta_{DA}(D_t, D_t^{'})$ if correct, uses it further for control delegation.

New File Creation: To share data files with intended data recipients using cloud services, data owner assigns a unique Id to the data file F, i.e., F_{Id} and select a symmetric data encryption key K $\in K^{'}$, where $K^{'}$ is the key space and encrypt the file F with K. Then, define a tree access structure T for the file and encrypt F with T using Algorithm 7 which returns ciphertext CT and store. Finally, the encrypted data file is stored on the cloud in the format as depicted in Fig. 4

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

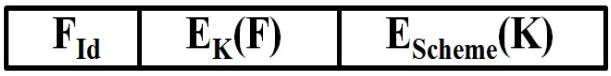


Fig. 4. Format of Data File F On Cloud, where F_{Id} denotes the unique Id of file F, E_K(F) denotes encrypted file F using symmetric encryption key k and $E_{Scheme}(K)$ denotes the encryption of k using our proposed scheme

Algorithm 7 Encrypt: Encrypts the message and convert it to illegible text [13].

Input: System public parameters PK, tree access structure T and message M.
Output: Ciphertext CT.

1: Select a polynomial q_x for each node x in the tree structure. Polynomial for each node is selected in a top down manner, starting from the root node R of the tree T. Degree of polynomial is set to be one less than the threshold value K_x(number of child nodes), i.e., $d_x = K_x - 1$. Thus, the degree of all leaf nodes in tree T is set to be 0.
2: Select $s \in Z_p$ and sets $q_R(0) = s$ further to completely define polynomial q_R, it randomly selects d_R. For other nodes x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and randomly selects d_x to completely define q_x.
3: Generate the ciphertext for message M as:
$CT = (T, \tilde{C} = M.e(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C_y^{'} = H(att(y))^{q_y(0)})$ here Y is the set of leaf nodes of tree T and forwards CT to CSP

File Access: To access the file, user requests cloud server. The cloud server gives response to user by providing the corresponding ciphertext, i.e., CT. User calls Algorithm 8 to obtain the symmetric key K and then decrypts ciphertext using K. This algorithm is derived from the Decryption algorithm of [13]. Algorithm 8 employs Lagrange coefficient $\triangle_{i,S}$ for $i \in Z_p$ and $S \subset Z_p$ of polynomial interpolation, which is defined as $\triangle_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$

Algorithm 8 Decrypt: To access the message M corresponding to ciphertext CT

Input: (G_U, G_{SK_U}), G_U is the group of n users and G_{SK_U} is group of n secret keys of users associated with set of attributes $S_1, S_2, ..., S_n$ respectively trying to access the files. Ciphertext CT is associated with tree access structure T.
Output: Message M

1: **for** all leaf node x of T **do**
2: $i = att(x)$
3: **if** $\exists S_j : i \in S_j$, where $j \in \{1, 2, ..., n\}$ **then**
4: $F_x = \frac{e(D_i,\ C_x)}{e(D_i^{'},\ C_x^{'})} = e(g,g)^{rq_x(0)}$
5: **else**
6: $F_x = \perp$
7: **end if**
8: **end for**
{$p(x)$ = $parent(x)$, $S_{p(x)}$: $k_{p(x)}$ sized set of child nodes x for non leaf nodes $p(x)$, such that $F_x \neq \perp$}
9: **for** all non leaf node $p(x)$ of T **do**
10: **if** no $S_{p(x)}$ set exist **then**
11: $F_{p(x)} = \perp$
12: **else**
13: $F_{p(x)} = \prod_{x \in S_{p(x)}} F_x^{\triangle_{i,s'_{p(x)}}(0)} = e(g,g)^{(r.q_{p(x)}(0))}$
{where $i = index(x), \{S'_{p(x)} = index(x) : x \in S_{p(x)}\}$}
//This step provides F_R for root node R
14: **end if**
15: **end for**
16: Message M is computed as $M = (\tilde{C}/e(C,D)/F_R)$

Access Privilege update: To update the access privileges of a user, DA call Algorithm *ApUpdate*.

$ApUpdate(MK_{DA}, S_{DA})$: This algorithm takes as input master key of domain authority MK_{DA} associated with set of attributes S_{DA} and set of attributes $S_{up}^l : S_{up}^l \subset S_{DA}$ corresponding to the amendment in the role of user U_l. DA selects parameters $r_l \in Z_p$ (as described in Algorithm 4) corresponding to user U_l in his domain and different random numbers $\{r_t \in Z_p : \forall a_t \in S_{up}^l\}$. U_l decrypts T, using his private key and verifies $\delta_{DA}(D_t^U, D_t^{'U})$ if correct, uses it further for accessing data.

User revocation: To revoke the access privileges from a user with a notion to restrict him from further accessing data, data owner calls the Algorithm *RvkUsr*.

$RvkUsr(E^{'})$: This algorithm takes as input expiry date $E^{'}$: $E^{'} > E$, E is the expiry date of user to be revoked from further decrypting ciphertext CT. Generate ciphertext components $C_{E'} = g^{q_y(0)}$, $C_{E'}^{'} = H(E^{'})^{q_y(0)}$ for his data, stored on cloud in encrypted form (as described in Algorithm 7). Data owner encrypts tuple $T(C_{E'}, C_{E'}^{'}, \delta_{do}(C_{E'},\ C_{E'}^{'}))$ using public key of CSP.
CSP decrypts T and verifies $\delta_{do}(C_{E'},\ C_{E'}^{'})$, here $\delta_{do}(X)$ is the data owner's signature on X. If correct, replaces it with existing expiry date components corresponding to CT. Similar technique has been employed in [11].

File Deletion: To delete the encrypted file on cloud, data owner makes a request for deletion of file to CSP. Every file is associated with a unique ID, i.e., Fid. Data owner to delete the file, encrypts the Tuple $T(Fid, \delta_{do}(Fid))$ with the public key of CSP and forwards the tuple to him, where $\delta_{do}(Fid)$ is the data owner's signature on Fid. Afterward CSP decrypts the tuple T and verifies $\delta_{do}(Fid)$, then CSP deletes the file associated with Fid.
Fig. 5, depicts the working of our proposed scheme, TTP executes the $Setup$ phase to generate public key PK and master key MK. PK is employed to encrypt message M corresponding to access policy T using encryption phase. This phase outputs ciphertext CT which is forwarded to Cloud Service Provider (CSP). Every user of our system

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

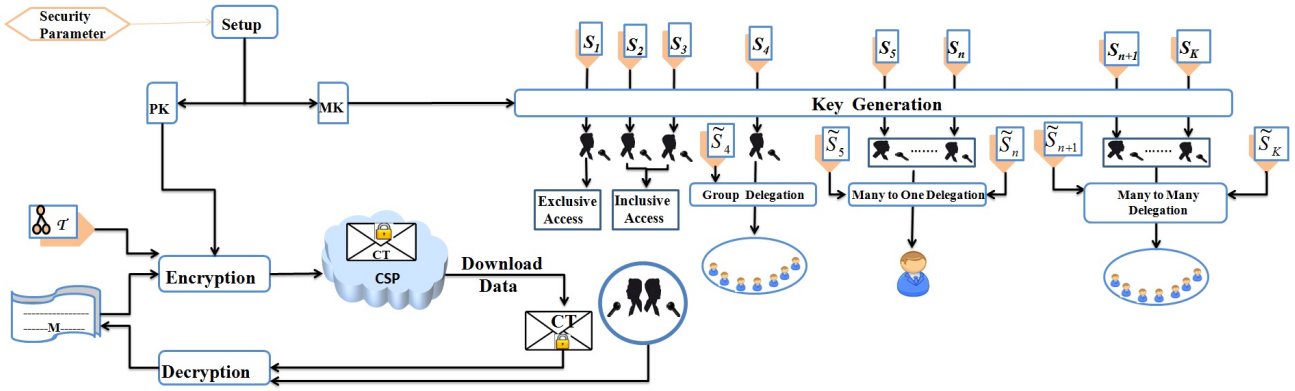


Fig. 5. Working of Proposed Scheme

is granted key corresponding to the set of attributes S using key generation phase. In our proposed scheme, key generation corresponds to $GenCA$, $GenDA$ and $GenUsr$ to generate key for CA, DA and users respectively. Our proposed scheme enables users to delegate their access privileges in an flexible manner, i.e., Group Delegation, Many to One Delegation, Many to Many Delegation.

5 Features of Our Proposed Scheme

This subsection discusses the security features of our proposed scheme.

1) Scalability: Our proposed scheme employs CP-ABE with hierarchical structure to reduce the workload involve in generation and distribution of keys for all the entities of organization. The scheme decentralizes the key-issuing authority at different levels of hierarchy. TTP provides the key only to CA, which in turn generates the key for DA. DA provides the key to end users. Hence, the workload of key-generation is divided among different levels of hierarchy.
2) Fine-Grained Access Control: Our proposed scheme allows data owner to define and enforce flexible access structure for data files similar to CP-ABE. Specifically, the access structure is defined as a policy over attributes, associated with data file as the scheme described in [13].
3) Flexible delegation of access privileges: In our scheme both domain authorities and end users can delegate their access privileges to a group or to an individual voluntarily, either for specified or unspecified time period.
4) Access Privileges: Our Proposed scheme supports two different access privileges for users:
   a. Exclusive Access Privileges (EAP): User gets EAP corresponding to his individual role. User is not allowed to combine his EAP with other users to inclusively access the files, which the user cannot access exclusively.
   b. Shared Access Privileges (SAP): User gets SAP corresponding to the role, he/she jointly addresses with other users. User can not access the files exclusively which is aimed to be accessed inclusively.
5) Access Privilege update & user revocation: To efficiently upgrade user's key on necessity basis, DA maintains state information of users. DA generates new expiry-date and other required attributes for user key to upgrade the validity or data access privileges. This scheme requires DA to maintain state information of user, which enables DA to avoid generation and distribution of new keys every time a user key needs update.
6) Expressiveness: In our scheme, users key is associated with set of attributes corresponding to the role an individual addresses, i.e., solely and jointly with other users. Thus it is more closer to traditional role based access control scheme [13]. Hence, it is more natural to apply our scheme to provide access control solutions for cloud computing environments.

6 Security Analysis of Our Proposed Scheme

6.1 Resistant to Collusion attack

To access Message M, ciphertext is decrypted using Algorithm 8 as

$$M = \tilde{C}/e(C, D)/e(g, g)^{rs} \quad (1)$$

where $\tilde{C} = Me(g, g)^{\alpha s}$. In $e(g, g)^{rs}$, r is the unique random value associated with user key. The recovery of M from $\tilde{C}$ rely upon retrieving $e(g, g)^{rs}$, which can be obtained only if the user's key consists of enough attributes to satisfy access tree T. Each user's key components are randomized using unique random number. Malicious entities collude their keys to inclusively access the message, which exclusively they cannot. When malicious users collude their keys, the colluded key components are randomized with different random values, which restricts it to go through the polynomial interpolation to obtain $e(g, g)^{rs}$. Thus, our scheme is resistant against collusion attack.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

6.2 Resistant to Cheating Attack

Similar to CP-ABE, in our proposed scheme ciphertext CT is associated with an access policy T and keys are identified corresponding to the set of attributes. Leaf nodes of T are attributes and non leaf nodes are threshold gates. To encrypt message M corresponding to T, data owner initially selects a secret random number s. Afterwards, s is distributed among all the nodes of tree T from root node to leaf nodes using Algorithm 7. Thus, every node of tree T is associated with a secret random number. User can access M, if and only if his key consist of enough attributes to satisfy the threshold associated with root node of tree T and obtain $e(g,g)^{r^u.s}$, which can be further used to access M using equation 1. In SAP, the keys of all the participants within the group is randomized using same random number r, which allows them to pool their attributes together. In addition, each participant of a group is granted key corresponding to the set of attributes, such that all the participants of a group are only allowed to inclusively satisfy T and obtain $e(g,g)^{rs}$, which can be employed to obtain message M using equation 1.

Let us consider, group G_k of k users, such that $G_k = \{U_1, U_2, ..., U_K\}$ are provided SAP to inclusively decrypt M. Group of dishonest participants G_n, such that $\{G_n \subset G_k, G_n \neq \phi\}$ cannot decrypt ciphertext, because the polynomial interpolation function cannot construct $e(g,g)^{rs}$, due to the unavailability of required set of attributes with keys of G_x participants, where $G_x = \{G_k \setminus G_n\}$. Therefore, dishonest participants are restricted to decrypt M, which can only be inclusively accessed by all the participants of a group. Thus, our proposed scheme is resistant against cheating attack.

6.3 Security proof

Our proposed scheme is extended from CP-ABE, with a hierarchical structure using delegation algorithm as described in the CP-ABE scheme and by providing shared access privileges to users. We prove the security of our proposed scheme directly based on the security of CP-ABE. We show that if there are any vulnerabilities in the proposed scheme, these vulnerabilities can be used to break CP-ABE, similar to the security model described in [11]. Thus, our proposed scheme is expected to have the same security property as CP-ABE, which has been proven to be secure under the generic bilinear group model and the random oracle model [13]. A generic security model to be defined below describes interactions between an adversary and an encryption algorithm like our proposed scheme or CP-ABE. Identical to the model used in CP-ABE, the security model allows the adversary to query for any private keys that cannot be used to decrypt the challenge ciphertext. In CP-ABE and our proposed scheme the ciphertexts are associated with access structures and the private keys are identified with attributes. Thus, the security model requires that the adversary chooses to be challenged on an encryption to an access structure T^* and can ask for any private key S such that S does not satisfy T^*.

6.3.1 Formal Security Model

Prior to discuss the formal proof for our proposed scheme, we first describe the formal security model of CP-ABE scheme. In this model, the adversary will choose to be challenged on an encryption to an access structure T^* and can ask for any private key S such that S does not satisfy T. The formal security model is defined as follows between an adversary and a challenger:

- **SetUp:** The challenger runs the SetUp Algorithm and gives the public parameters, PK to the adversary.
- **Phase 1:** The adversary makes repeated private key queries corresponding to sets of attributes S_1, S_2, ..., S_q. The challenger responds to the adversary by executing the algorithm GenCA to grant the key MK_{CA} corresponding to set of attributes S_i. Similarly, the adversary can also make private key queries for domain authority (MK_{DA_i}) or end-users (SK_{u_i}) with the private key of administrating authority. The challenger responds them by executing algorithm $GenDA$ or $GenUsr$.
- **Challenge:** The adversary A submits two equal length messages M_0 and M_1 such that $|M_0| = |M_1|$ (both the messages are of equal length). In addition A also provides access structure T^* such that none of the sets S_1, S_2 ,..., S_q from phase 1 satisfies the access structure T^*. The challenger flips a coin $b : b \in \{0,1\}$, and encrypts M_b under the access structure T^*. The ciphertext CT^* corresponding to access structure T^* is given to the adversary.
- **Phase 2:** Phase 1 is repeated with the restriction that none of the sets S_{q+1} ,..., S_{q1} satisfies the access structure T^* corresponding to the challenge ciphertext CT^*.
- **Guess:** The adversary outputs a guess b' of b. The advantage of the adversary A in this game is defined as $Pr[b' = b] - \frac{1}{2}$.

Definition 2. *A Ciphertext-Policy ABE scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game [13].*

6.3.2 Formal Proof

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

Proof: Suppose a group of probabilistic polynomial-time adversaries G_A^* has non negligible advantage ϵ against our proposed scheme. Using G_A^*, we show how to build an adversary A that breaks the CP-ABE scheme with nonnegligible advantage.

- **Initialization:** In our scheme group of users are granted Shared Access Privileges (SAP) and in CP-ABE user is granted Exclusive Access Privileges (EAP). We consider a group of adversaries G_A^* consisting of SAP with non-negligible advantage against our proposed scheme. Using G_A^*, we show how to build an adversary A consisting of EAP against CP-ABE scheme.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

TABLE 1: Comparison of computational complexity of our scheme where, N: # of attributes in key, M: # of sets in key, Y: # of leaf nodes in access tree T, X: Translating nodes of T, I: # of attributes associated with data Files, n: # of users in delegatee group, V: # of attributes provided to the group of delegatees

Operations	[20]	[11]	[31]	[23]	Proposed Scheme
SetUp	O(Y)	O(1)	O(1)	O(1)	O(1)
Central Authority Grant	-	O($2N$+M)	-	O($2N$+M)	O(N)
Domain Authority	O(Y)	O($2N$+M)	O($3N$)	O($3N$+M)	O(N)
User Grant	O(Y)	O($2N$+M)	O($3N$)	O($3N$+M)	O(N)
Group Delegation	-	-	-	-	O(n+V)
Many to One Delegation	-	-	-	-	O(N)
Many to Many Delegation	-	-	-	-	O(n+V)
Control Delegation	-	-	-	-	O(1)
File Creation	O(I)	O($2Y$+X)	O($2Y$)	O($3Y$+X)	O(Y)
File Access	O(Y)	O($2Y$)	O(Y)	O(Y+X)	O(Y)
Access privilege update	-	O(M)	-	O(N)	O(N)
User revocation	O(Y)	O(1)	-	-	O(1)
File Deletion	O(1)	O(1)	-	O(1)	O(1)

- **SetUp:** The adversary A takes the public key of $CP-ABE$ from CP-ABE challenger. CP-ABE challenger, randomly selects $\alpha,\ \beta \in Z_p$ and generates the public parameter PK and master key MK, where $PK = \{G, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}\}$ and $MK = \{\beta, g^{\alpha}\}$. A forwards Public Key (PK) to G_A^*.
- **Phase 1:** In this phase, A answers private key query. Suppose G_A^* issues private key query for a set of attributes S to adversary A, such that S cannot satisfy T^*. To answer the query, A makes private key query for the set S to $CP-ABE$ challenger, which generates the key
$SK^A = \{D^A = g^{(\alpha+r)/\beta}, \forall a_j \in S :$
$D_j^A = g^r.H(a_j)^{r_j}, D_j^{'A} = g^{r_j}\}$
and forwards it to A. Adversary A in turn generates the SAP for G_A^* corresponding to set S as follows:
(i) Generates m sets of attributes $S_d : S_d \subset S, \forall d \in \{1, 2, ..., m\}$.
(ii) Selects unique random number $\hat{r} \in Z_p$ for G_A^* and different random numbers $r_k \in Z_p, \forall a_k \in S_d : d \in \{1, 2, ..., m\}$.
(iii) Generates secret keys for G_A^* as:
$\{SK_d = (D^d = D^A.f^{\hat{r}}, \forall a_k \in S_d : D_k^d = D_k^A.g^{\hat{r}}H(a_k)^{r_k}, D_k^{'d} = D_k^{'A}.g^{r_k}) : \forall d \in (1, 2, ..., m)\}$.
- **Challenge:** After the completion of Phase 1, G_A^* outputs an access structure T^* and two messages $M_0, M_1 \in G_1 : |M_0| = |M_1|$, on which G_A^* wishes to be challenged. A forwards M_0, M_1 and T^* to the CP-ABE challenger, which selects $\gamma \in \{0, 1\}$ and forwards challenged ciphertext CT^* to A
$CT^* = \{T^*, \tilde{C} = M_{\gamma} \cdot Z, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C_y^{'} = H(att(y))^{q_y(0)})\}$
Finally A returns the ciphertext CT^* to G_A^*.
- **Phase 2:** G_A^* issues private key queries and A replies as in phase 1, with a condition that none of the key satisfies access structure T^*.
- **Guess:** Finally G_A^* outputs a guess $\gamma^{'} \in \{0, 1\}$ and then A concludes its game by outputting $\gamma^{'}$. According to the formal security model, the advantage to the group of adversaries A against CP-ABE is

$$Adv_A = |Pr[\gamma = \gamma^{'}] - \frac{1}{2}| = Adv_{G_A^*}$$

This means that A has nonnegligible advantage against the CP-ABE scheme, which completes the proof of the theorem.

7 PERFORMANCE ANALYSIS

This section analyzes the theoretical as well as the empirical analysis of our proposed scheme.

7.1 Performance Analysis:

The computational complexity in terms of the number of exponentiation and pairing operations of our proposed scheme is computed. The comparison of the computational complexity with competing schemes [11], [20], [23], [31] is shown in Table 1. All the features require linear amount of exponentiation operations (for sake of brevity the details are omitted). Our scheme does not increase the complexity of operations as compare to competing schemes.

7.2 Storage Analysis:

The storage overhead is computed for the following entities: the number of public key (PK) components available in the system, master key (MK) of TTP, CA, DA, as well as the secret key (SK) of individual user, the number of secret key components required by data owner and the size of ciphertext (CT) for single data content. Table 2 illustrates the comparison of storage overhead with the competing schemes [11], [20], [23], [31] (for sake of brevity the details are omitted). Our proposed scheme does not increase the storage overhead as compare to the other competing schemes.

7.3 Implementation:

To implement our scheme, we use Type A curves (in the Pairing-Based Cryptography (PBC) library [34]) providing groups in which bilinear map e : $G_1 \times G_1 \rightarrow G_2$ is defined. These curves provide good computational efficiency for pairing computation. Implementation is conducted using the PBC library version pbc-0.5.14 [34] on a desktop with Intel core 2 quad 2.83-GHz CPU, 4 GB RAM, running Ubuntu-10.04. The implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over 512-bit finite field. The experimental results depicts time required in pairing and exponentiation operations in G_1 and G_2. The comparatively negligible hash operations are ignored in the experiments.

Key Grant: Time required in key-generation for various entities, i.e., CA, DA and users, using Algorithm 2, 3 and

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

TABLE 2: Analysis of storage overhead of our proposed scheme where, I: # of attributes associated with data file, N: # of attributes in key, M: # of sets in key, Y: # of leaf nodes in access tree T, X: Translating nodes of T

Entity	[20]	[11]	[31]	[23]	Proposed Scheme
Master Key	$Y+1$	3	2	6	2
Public Key	$Y+1$	6	3	3	4
Cloud Storage (CT)	$2\|I\|+1$	$2\|Y\|+X+3+T$	$2\|Y\|+3+T$	$2\|Y\|+X+4+T$	$2\|Y\|+2+T$
CA Master Key	-	$2\|N\|+M+1$	-	$2\|N\|+M+1$	$2\|N\|+1$
DA Master Key	-	$2\|N\|+M+1$	-	$3\|N\|+M+1$	$2\|N\|+1$
User Secret Key	Y	$2\|N\|+M+1$	$3\|N\|+1$	$3\|N\|+M+1$	$2\|N\|+1$
Data Owner	-	-	1	1	-

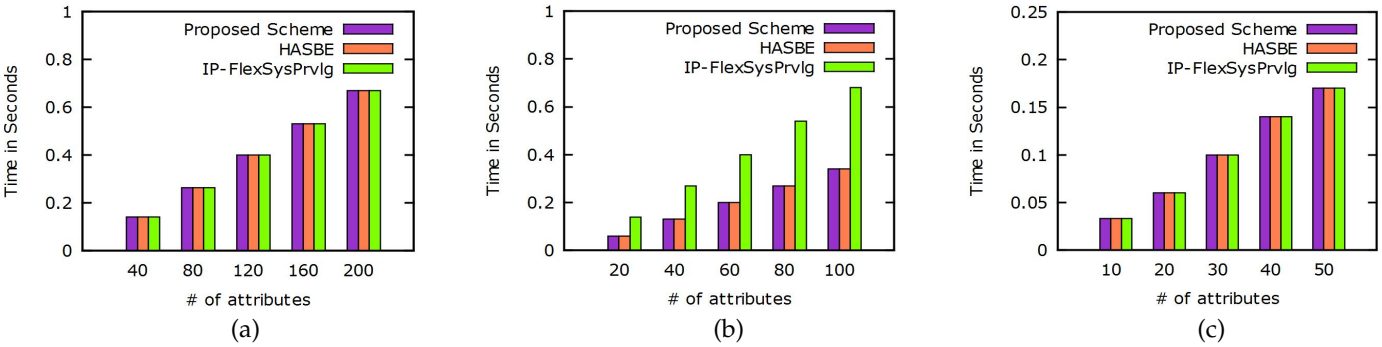


Fig. 6. Experiments on key-generation(a) Time required for Central Authority Grant (b) Time required for Domain Authority Grant (c) Time required for User Grant

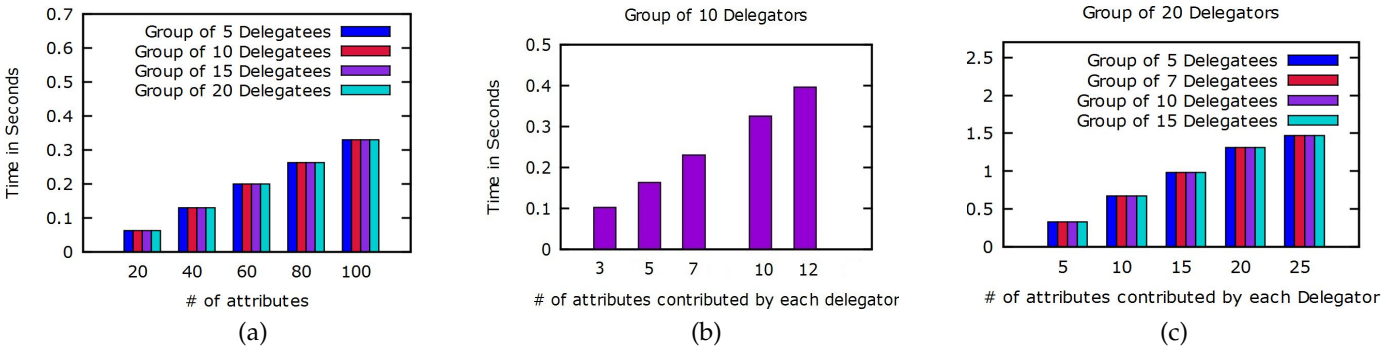


Fig. 7. Experiments on Delegation (a) Time required in Group Delegation phase (An individual delegator delegates his access privileges to group of 5, 10, 15 and 20 delegatees) (b) Time required in Many to One Delegation phase (Group of 10 delegators delegate their access privileges to an individual delegatee) (c) Time required in Many to Many Delegation phase (Group of 20 delegators delegate their access privileges to group of 5, 7, 10 and 15 delegatees)

4 respectively depends on the number of attributes present in the key. Fig. 6(a), illustrates the time required for key generation of CA corresponding to 40, 80, 120, 160, and 200 attributes. Fig. 6(b) depicts the time required for key generation of DA corresponding to 20, 40, 60, 80, and 100 attributes, while Fig. 6(c) illustrates the time required for key generation of end-users corresponding to 20, 40, 60, 80 and 100 attributes.

Group Delegation: Time required in group delegation, depends on the number of delegatees in the group and number of attributes provided to that group. Fig. 6(b), depicts the time required for generation of keys for group of delegatees by varying the number of attributes provided to that group, i.e., 20, 40, 60, 80 and 100 by delegator. For group of 5 delegatees, time for key generation increases linearly with respect to the number of attributes provided to the group. Similar observations are depicted for group of 10, 15 and 20 delegatees. For each delegatee single exponentiation operation is performed, while several exponentiation operations are performed for single attribute. Hence the overall time depends on the number of attributes as compared to the number of delegatees in the group.

Many to One Delegation: Time required in many to one delegation, depends on the number of attributes in the key of delegatee, which in turn rely upon the number of attributes provided to delegatee by each delegator. Fig. 7(a), depicts the time required by group of 10 delegators, where each delegator provides 2, 5, 10, 15 and 20 attributes for delegatee. Hence the total number of attributes delegated by the group to the delegatee is 20, 50, 100, 150 and 200. It is clearly illustrated that the time increases with respect to the number of attributes. The time required increases with the increase in number of attributes provided by group of delegators.

Many to Many Delegation: Time required in many to many delegation, depends on the number of delegatees in the group and number of attributes provided to the group of delegatees, which in turn rely on the number

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

of attributes provided by each delegator to the group. For experimentation purpose, we assume there are 20 delegators and each delegator provides 5, 10, 15 and 20 attributes for group of delegatees. Thus the total number of attributes provided to the group of delegatees by group of delegators is 100, 200, 300 and 400. Fig. 7(b), depicts the time required for generation of keys for group of delegatees. For group of 5 delegatees, time in key generation increases linearly with respect to the number of attributes provided to the group. Similar observations are depicted for group of 10, 15 and 20 delegatees.

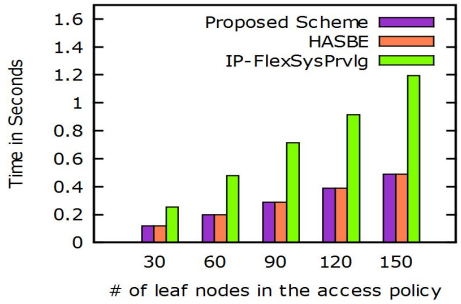


Fig. 8. Time required in Encryption Phase

Encryption/Decryption: In our proposed scheme files are encrypted corresponding to the access policy. Non-leaf nodes of the access policy are threshold gates, while leaf nodes are attributes. Time required to encrypt file corresponding to access policy depends upon the number of leaf nodes of access tree. Fig. 9(a), depicts the time required for encryption by varying the number of leaf nodes of access policy, i.e., 30, 60, 90, 120, and 150 attributes. Similarly, time required for decryption depends upon the number of leaf nodes of access policy. Fig, 9(b) illustrates the time required to decrypt the file by varying the leaf nodes, i.e., 30, 60, 90, 120, and 150.

7.4 Observation

The experiments are performed for our proposed scheme, HASBE [11] and $IP - FlexSysPrvlg$ [23]. After carefully observing the experimental results, we inferred the following conclusions. Time required for key generation for all the entities CA, DA and end users in all the schemes grow linearly with the increase in the number of attributes in the key structure.

For group delegation and many to many delegation, time increases linearly with the increase in number of attributes delegated to the group of delegatees and number of users in delegatee group. For each delegatee single exponentiation operation is performed, while several exponentiation operations are performed for single attribute. Hence, the overall time depends upon the number of attributes as compared to the number of delegatees in the group.

To encrypt a file, time required in all the competing schemes depend upon the number of leaf nodes.

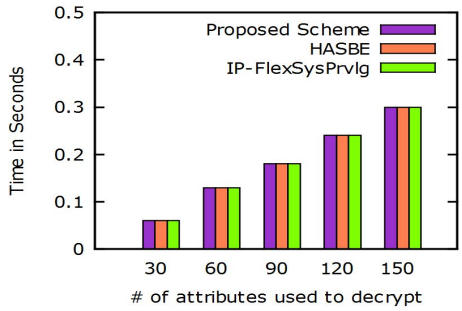


Fig. 9. Time required in Decryption phase

Furthermore, time required for decryption remains same for all the schemes. Time required to encrypt or decrypt the file grows linearly with the increase in number of leaf nodes of access tree.

The experimental results show that the hidden constant in the asymptotic notation is very small. Our proposed scheme is comparable against other competing schemes in terms of time, whereas our scheme provides more features. It is observed that the experimental results substantiate our theoretical analysis.

8 Conclusion

In this paper, we introduced scalable attribute-based access control scheme for cloud computing environments. The scheme extends CP-ABE in a hierarchical structure of users to simultaneously achieve scalability and fine-grained access control. This scheme provides users flexibility to delegate their access privileges, i.e., group delegation, many to one delegation, many to many and control delegation. In addition, scheme enables to provide group of users shared access privileges to jointly address the role. The scheme provides efficient solution for access privilege update and user revocation as well. Security of our proposed scheme is based on the security of CP-ABE scheme. This implies that this scheme is best suited for cloud applications. One promising extension of our scheme is to further extend it for multi-authority attribute based encryption in cloud environments.

Acknowledgment

The authors would like to thank Ben Lynn for making Pairing Based Cryptography (PBC) library, http://crypto.stanford.edu/pbc/ as free.

References

[1] R. Buyya, J. Broberg, A.M. Goscinsk, "Cloud Computing: Principles and paradigms," John Wiley & Sons, 2010.
[2] Cloud Security Alliance (CSA)[Online]: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Top_Ten_v1.pdf. Accessed 13 January 2014.
[3] C. Rong, S.T. Nguyen, M.G. Jaatun,"Beyond lightning: A survey on security challenges in cloud computing," Computers & Electrical Engineering, Vol. 39, pp. 47-54, 2013.
[4] L. Wei, G. Yang, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, A.V. Vasilakos, "Security and Privacy for Storage and Computation in Cloud Computing," Information Sciences, Vol. 258, pp. 371-386, 2014.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

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.2017.2751471, IEEE Transactions on Cloud Computing

[5] R.F. Olimid, "Secret sharing-based group key establishment," Ph.D. thesis, University Of Bucharest. https://www.iacr.org/phds/. Accessed 26 July 2016.

[6] D.W. Hsiao, A. Trappey, L. Ma, P.-S Ho, "An integrated platform of collaborative project management and silicon intellectual property management for IC design industry," Information Sciences, Vol 179, pp. 2576-2590, 2009.

[7] H. Pateman, S. Cahoon, S.L. Chen, "The Role and Value of Collaboration in the Logistics Industry: An Empirical Study in Australia," The Asian Journal of Shipping and Logistics, Vol 32,pp. 33-40, 2016.

[8] B.C. Liang, "Managing and Leading for Science Professionals," Academic Press, San Diego, 2014

[9] T.J. Norman, C. Reed, "Group delegation and responsibility. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems," part 1, Newyork, USA, pp. 491-498, 2002.

[10] T.J. Norman, C. Reed, "A logic of delegation," Artificial Intelligence, Vol 174,pp. 51-71, 2010.

[11] Z. Wan, J. Liu, R.H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," IEEE Transactions on Information Forensics and Security, Vol. 7,pp. 743-754, 2012.

[12] V. Goyal, O. Pandey, A. Sahai, B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," In: Proceedings of ACM conference on Computer and communications security (ACM CCS), Alexandria,VA, pp. 89-98, 2007.

[13] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption. In: Proceedings IEEE Symposium on Security and Privacy, Oakland, CA, pp. 321-334, 2007.

[14] M. Chase, "Multi-authority attribute based encryption," In: Proceedings Theory of cryptography conference, Vol. 4392 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 515-534, 2007.

[15] M. Chase, S.S.M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS09), ACM, pp. 121130, 2009.

[16] M.Li, S. Yu, Y. Zheng, K. Ren, W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," IEEE transactions on parallel and distributed systems, Vol 24, no 1,PP. 131-143, 2013.

[17] A. Lewko, B. Waters, "Attribute-sets: Decentralizing attribute-based encryption," In: Proceedings of the 30^{th} Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vol. 6632 of Lecture Notes in Computer Science, Springer Berlin, Heidelberg, pp. 568-588, 2011.

[18] J. Li, Q. Huang, X. Chen, S.S.M. Chow, D.S. Wong, D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, pp. 386-390, 2011.

[19] K. Yang, X. Jia, K. Ren, B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," In: INFOCOM Proceedings IEEE, pp. 85-93, 2013.

[20] S. Yu, C. Wang, K. Ren, W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," In: INFOCOM, Proceedings IEEE, pp. 1-9, 2010.

[21] R. Ahuja, Y. Yang, K. Shen, "A scalable attribute-set-based access control with both sharing and full-fledged delegation of access privileges in cloud computing," Computers & Electrical Engineering, vol 57, pp. 241-256, 2017.

[22] G. Wang, Q. Liu, J. Wu, "Achieving fine-grained access control for secure data sharing on cloud servers," Concurrency and Computation: Practice and Experience, vol 23, pp. 1443-1464, 2011.

[23] R. Ahuja, S.K. Mohanty, K. Sakurai, "An Identity Preserving Access Control Scheme with Flexible System Privilege Revocation in Cloud Computing," In: Proceedings of Eleventh Asia Joint Conference on Information Security, Japan, Fukuoka, pp. 39-47, 2016

[24] Y. Xie, and H. Wen, and B. Wu, and Y. Jiang, and J. Meng, "A Modified Hierarchical Attribute-based Encryption Access Control Method for Mobile Cloud Computing," IEEE Transactions on Cloud Computing, vol PP, pp. 1-1, 2017.

[25] D. Boneh, M.K., Franklin, "Identity-based encryption from the weil pairing," In: Proceedings Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Vol. 2139 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 213-229, 2001.

[26] S. Chatterjee, P. Sarkar, "Identity-based encryption," Springer Science & Business Media, 2011.

[27] J. Horwitz, B. Lynn, "Toward Hierarchical Identity-Based Encryption," In: Proceedings Advances in Cryptology- EUROCRYPT, International Conference on the Theory and Applications of Cryptographic Techniques, Vol. 2332 of Lecture Notes in Computer Science, Springer Berlin, Heidelberg, pp. 466-481, 2002.

[28] A. Sahai, B. Waters, "Fuzzy Identity-based encryption," In: Proceedings of Advances in Cryptology - EUROCRYPT, Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin, Heidelberg, pp. 457-473, 2005.

[29] R. Bobba, H. Khurana, M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," In: Proceedings Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, pp. 587-604, 2009.

[30] G. Wang, Q. Liu, J. Wu, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," computers & security, vol 30, pp. 320-331, 2011.

[31] D. Koo, J. Hur, H. Yoon, "Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage," Computers & Electrical Engineering, vol. 39, pp. 34-46, 2013.

[32] T. Jung, X. Li, Z. Wan, M. Wan, "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption," IEEE Transactions On Information Forensics and Security, vol. 10, pp. 190-199, 2015.

[33] W. Teng, G. Yang, Y. Xiang, T. Zhang, D. Wang, "Attribute-based Access Control with Constant-size Ciphertext in Cloud Computing," IEEE Transactions on Cloud Computing, vol. PP, pp. 1-1, 2016.

[34] The pairing-based Cryptography library. http://crypto.stanford.edu/pbc/

Rohit Ahuja is pursuing the Ph.D. degree in computer science and engineering discipline in Indian Institute of Information Technology, Design and Manufacturing Jabalpur, MP, India. He was a visiting research scholar in Sakurai Lab, Kyushu University, Fukuoka, Japan. His research area is security and privacy in cloud computing.

Sraban Kumar Mohanty is currently working as an Assistant Professor in computer science and engineering discipline in Indian Institute of Information Technology, Design and Manufacturing Jabalpur, MP, India. He received Ph.D. degree in computer science and engineering from Indian Institute of Technology Guwahati, Assam, India in 2010. His research interest is cryptography and its application.

2168-7161 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.