An Efficient Access Control Optimizing Technique Based on Local Agency in Cryptographic Cloud Storage

Shidong Zhu(✉), Liu Jiang, and Zhenliu Zhou

College of Information, Shenyang Institute of Engineering, Shenyang, China
{sdzhu, jiangliu, zlzhou}@sie.edu.cn

Abstract. With analyzing cloud storage data security requirements, in this paper we focus on data privacy protection in cloud storage, and proposed a measure to optimize the efficiency of access control in cryptographic cloud storage. For the main users that use cloud storage services are enterprise and the community users, they have the characteristics in common that they manage their data access rights by the mode of hierarchical classification. Because of this, combined ciphertext policy attribute-based encryption (CP-ABE) algorithm and hierarchical identity-based encryption (HIBE) algorithm, we proposed to identify users by both precise identity and attribute in the process of making data access control strategy, and use hierarchy when generate keys. The advantage of this is that it can effectively protect the data privacy in cloud storage, and support precise identity and attribute access control, and fine-grained access. Furthermore, for the purpose of reducing cost of access cloud storage, we proposed an efficiency access control optimizing technique based on local agency, which can replace users to complete the ciphertext access control related operations, and cache frequently accessed data, effectively reduce the impact of using the ciphertext access control mechanisms. Experiments show that the scheme can reduce additional cost of cloud storage data protection, and suitable for using in actual scenes of cloud storage.

Keywords: Cloud storage · Ciphertext access control · Ciphertext policy Attribute-Based encryption · Data protection · Local proxy

1 Introduction

Data security is the primary factor for usage and development of cloud storage [1, 2]. Shi Xiaohong, the vice president of inc.360, said that data security is the core issue of cloud storage [3]. From the user's perspective, the ideal state is under user-controllable data protection for the need of cloud storage security. Most users, especially business users hope to obtain protection for cloud storage as local storage, when the data is stored, transmitted, shared, used and destructed in cloud [4]. Once data leak occurs, it may cause incalculable damage, because user data may contain business decisions, the core technology, trade secrets and personal privacy and other sensitive content. Therefore, the security focus of user data in cloud storage is protection of data leakage. The characteristics of cloud storage are open and shared, so the user data is vulnerable

DOI: 10.1007/978-3-319-27051-7_8

to attacks and threats from networks. In addition, cloud service providers (CSP) and its staff may also leak user data actively or passively.

The most direct way to prevent leak of cloud data is to encrypt data by the user and then distribute the decryption key to control the access authority. However, it is not efficient, and greatly increases time cost to use cloud storage. A main idea for data confidentiality protection is to establish Cryptographic Cloud Storage [5] mechanism to protect the security of cloud storage sharing data with the access control policies managed by user own. In this way, even if the illegal users obtain the shared data, they can't get the plaintext.

With the target of user self-management on important data, Many researchers have proposed data protection scheme based on ciphertext access control, that is data are encrypted using symmetric algorithm by data owner, and publish both data and access policies to cloud storage, then distribute decryption keys through secure channel. Data users first fetch ciphertext from cloud storage, and decrypt ciphertext with their own private key, and obtain decryption key. This method has large amount of computation burden, and efficiency of key distribution is low.

With the development of attribute-based encryption technology, researchers have proposed ciphertext access control technology based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [6, 7]. As access structure of CP-ABE is deployed in the ciphertext, data owners have more initiative, may decide access structure by their own. Chase resolved the bottleneck that attribute management and key distribution were managed just by a single Attribute Authority (AA) [8], and proposed to allow multiple independent AA to manage and distribute key properties. Waters presented the first access structure with full expression of CP-ABE [9].

However, using CP-ABE technology to protect data security in cloud storage still faces many problems. One is the large time cost of encryption, decryption and revocation of CP-ABE, the other is that access granularity of key distribution strategy of CP-ABE does not match the needs of most cloud storage scenarios. In this paper, we focus on solving the access efficiency and fine-grained access, and proposed a scheme of ciphertext access control based on hierarchical CP-ABE encryption algorithm, which can support fine-grained access control, and using local proxy technology to optimize the efficiency of cloud storage access.

2 Ciphertext Access Control Based on Hierarchical CP-ABE

2.1 Analysis of Cloud Storage Application Scenarios

Attribute-Based Encryption (ABE) belongs public key cryptography system, and supports fine-grained access control. In the ABE system, the user is described by a set of attributes, ciphertext are associated with attribute-based access structure, user can decrypt ciphertext only when attribute are satisfied access structure. Goyal et al. [10] divided ABE into Key-Policy Attribute-Based Encryption (KP-ABE) and CP-ABE. In access control based on KP-ABE, data users specify access control policies, this is suitable for query type applications [11]. Access structure of CP-ABE is placed in ciphertext, so that the data owner has larger initiative to decide access structure for encryption. In CP-ABE scheme, users are described by attribute set, data owner give a access tree structure, if and only if user's attribute set are satisfied access tree structure, decryption key is able to be recovered, and get plaintext.

Cloud storage users generally can be divided into three types: individual users, enterprise users and community users. The individual user refers to dependent the individual person who use cloud storage service independently, its service form are like network disk service. Individual users are less data and data shared requires, and can implement data security protection by using simple ciphertext access control mechanisms. Enterprise users are those belonging to a particular enterprise or group, which is characterized by its attributes generally associated with hierarchical structure, they usually have large data sharing requires with large amounts of data. Community users refer to the user who have a certain relationship with each other, but geographically dispersed, and have data shared requires. Consider the following scenario: teachers of ten universities in one city will use the same cloud storage service, the staff of Information Institute of A school and Information College of B school develop a research project named X. Its access control list is shown in Table 1.

Table 1. Access control list

File description	Access structure
Core Technical Documents	X Project team members, information collage dean of A, ID in {905066,307083}
Common Technical Documents	X Project team members, staff with Senior title of A information College
Teaching Materials of A School	all staff of A information College

The attribute information in this scenario relate to the organization, department, position, title, project team, precise identity, etc., and have universal representation in cloud storage applications, its characteristics are as follow:

1. Data access architecture requirements need high flexibility in size, with coarse-grained division of attributes, such as organization, department, etc., also contain precise distribution of attribute, such as employee number and position.
2. There is great difference between access structure of different files, such as the access right of teaching materials of A school may be set to: {A school} AND {Information College}; the core technical documentation will be set to: ({A School} AND {X project team members}) OR ({B School} AND {X project team members}) OR ({{A School} AND {Information College} AND {Dean}}) OR {ID: 905066} OR {ID: 307083}.
3. The right distribution of community users or enterprise users are usually associated with much attributes, and usually correspond with department structures and administrative levels.

Therefore, we need to design an access control scheme that meet demands below:

1. Flexible access control policy. Access control tree should support access control of precise identity or simple attribute access.
2. Access control structures can be represented as hierarchical structure like enterprise administrative level or departmental structure.
3. The efficiency of key distribution should be adapted to users commonly used time cost of encryption and decryption in cloud storage.

Based on HIBE system, we added precise identify attribute in access structure of CP-ABE system, and introduce domain management in access control structure of CP-ABE for generate keys and use hierarchy key distribution, as described below.

2.2 Key Generation Model Based on Hierarchical CP-ABE

Figure 1 shows the key generation model based on hierarchy CP-ABE algorithm.

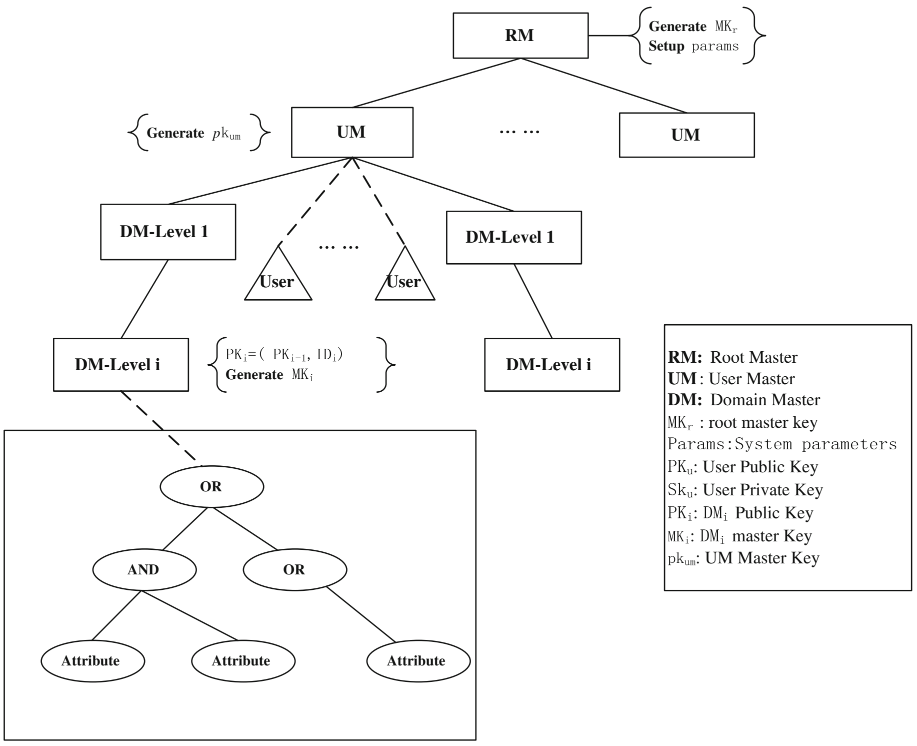

Fig. 1. Key generation model based on hierarchy CP-ABE algorithm

This model is a tree structure, consisting of a root master (RM) node and a plurality of domains, RM generates and distributes system parameters and domain keys, user master (UM) manage users, domain master (DM) is used to generate keys and attributes for the under hierarchy. RM, UM and DM node can be implemented by Local Proxy (LP) or trusted third party (TTP). Each DM and attribute has a unique ID signs, and each user has a unique ID identifier and a series of attribute flags. Ciphertext access structure comprises ID set of data user (DU) and attribute-based access structure tree T. When the data owner (DO) release access strategy, firstly, DM judge whether the ID of DU is in its precise ID set, if True, DM authorize to decrypt ciphertext without attributes judgment; otherwise, DM will analyze the attribute set in the key of DU whether satisfy the access control policies *T*. Access control tree *T* is used to represent an access control structure. Each non-leaf node *x* of *T* represents an k_x-of-num_x threshold operation: numx indicates the number of child nodes, k_x represents the threshold, $0 < k_x \le num_x$. k_x=1 indicates OR operation, $k_x = num_x$ indicates an AND operation. Each leaf node *x* represents an attribute $att(x)$. Access control tree can describe attribute-based access control policy, to judge whether attribute set *S* satisfy an access control tree *T* is as follows:

1. Let *r* be the root node of *T*, use T_r represents *T*, then let T_x be the subtree that root node is *x*, if attribute *S* satisfy T_x, denoted $T_x\ (S) = 1$.
2. Let $x = r$, calculate $T_x\ (S)$: For non-leaf node *x*, let all child nodes of x be $xc_1, \ldots, xc_{numx}$, calculate $T_{xci}(S)(i \in [1, num_x])$, $T_x\ (S)$ returns 1 if and only if at least k_x number leaf nodes return 1; for the leaf node, $T_x\ (S)$ returns 1 if and only if $att\ (x) \in S$.

2.3 Key Generation Algorithm Based on Hierarchical CP-ABE

Key generation algorithm based on hierarchical CP-ABE is as follow:

Setup: RM generate master key MK_r for UM, and generate master key PK_r for UM. The process is: Select $MK_r \in \mathbb{Z}^*_q$, MK_r is master key, output system parameter $Params{=}{<}q,\mathbb{G}_1,\mathbb{G}_2,\hat{e},n,p_0,Q_0,H_1,H_2{>}$,($q,\mathbb{G}_1,\mathbb{G}_2,\hat{e}$) is output, *n* is positive integer, p_0 is generator, $Q_0{=}MK_r\,p \in \mathbb{G}_1, H_1{:}\{0,1\}^* \rightarrow \mathbb{G}_1$和$H_2{:}\mathbb{G}_2 \rightarrow \{0,1\}$, MK_r keep secret.

- **GenSK:** UM generate private key for user *U* using Params and MK_r. The process is: $SK_u{=}$（$Q\text{-}tuple_u{+}mk_uP_u$）, $P_u{=}H_1(PK_u) \in \mathbb{G}_1$.
- **GenDM:** DM manage under domain node and access control *T*, each DM has a unique sign. DM_i own a public key PK_i and master key MK_i, $PK_i{=}(PK_{i\text{-}1},ID_i)$, and generate master key for DM_{i+1}. The process is:
  Let DM_i be the parent node of DM_{i+1}, $MK_{i+1}{=}(mk_{i+1},Sk_{i+1},Q\text{-}tuple_{i+1},H_A)$;$mk_{i+1}$ is random element in
  $$\mathbb{Z}^*_q;Sk_{i+1}{=}Sk_i{+}mk_iP_{i+1} \in \mathbb{G}_1,P_{i+1}{=}H_1(PK_{i+1}) \in \mathbb{G}_1;\ Q\text{-}tuple_{i+1}{=}(Q\text{-}$$
  $tuple_{i+1},Q_{i+1})$, and $Q_{i+1}{=}mk_{i+1}P_0 \in \mathbb{G}_1$; $H_1{:}\{0,1\}^*$ is random oracle model.
- **GenUser:** DM_i judges whether user *U* belongs its own management domain, When the condition is satisfied, DM_i generates identity key SK_{iu} of *U* and attribute key SAK_{iu}. The process is:
  $$SK_{iu}{=}(Q\text{-}tuple_{i\text{-}1},mk_imk_up_0);SAK_{iu}{=}Sk_i{+}mk_imk_up_a;$$
  $mk_u{=}H_A(PK_u) \in \mathbb{Z}^*_q$, $P_a{=}H_1(PK_a) \in \mathbb{G}_1$

- **Encrypt:** Let DO is the data owner, T_{do} be the access control tree of DO. Let DM_i located on the i layer to manage attribute-based access control tree T_i, for the purpose of encrypt K, DO output ciphertext $CK=$ （$\mathbb{R}, T_{do}, CF$）, its input parameters are precise ID set $\mathbb{R}=\{ID_{u1},...,ID_{um}\}$, attribute access control tree T_{do}, and all user public key in $\mathbb{R}$ and all attributes key in T_i. The paramters:

$$P_{ui}=H_1(PK_{ui}) \in \mathbb{G}_1;\ U_{ui}=rP_{ui};\ V=K \oplus H_2(\hat{e}(Q_0, rn_A P_u));\ CF=[u_0, Uu_1, ..., U_{um}, V].$$

- **Decrypt:** Given ciphertext CK, if the precise ID set of U belongs $\mathbb{R}$, then the key K can be recovered by using system parameters params and user private key SK_u. Given ciphertext CK, if the user's attributes satisfies access structure T, that means U has at least one attribute key in access control tree T, then the plaintext will be recovered by using identity key Sk_{iu} and the system parameters *params*, and user attribute secret key $\{Sk_{iu}, a \mid a \in T\}$.

2.4 Ciphertext Access Control Scheme Based on Hierarchical CP-ABE

Basic process: Data will be stored in the cloud storage after encryption using symmetric encryption algorithms such as AES, which can ensure data storage security. The encryption key will be uploaded to cloud storage after encryption using CP-ABE algorithm, the data users which satisfy the access control can decrypt data by their own. Figure 2 shows the ciphertext access control process based on Hierarchical CP-ABE.

LP (or TTP) run Setup in RM, and generate system parameters *Params*, then output master key PK_r.

1. DM run GenDM and generate master key for its next DM using *Params* and its own master key. UM run GenSK and generate attribute key SK_u for the users of its domain, and pass it to each user of the domain through secure channel.
2. When sharing data, data owner DO first use symmetric encryption algorithm E to encrypt data F, and then encrypt the key K using CP-ABE algorithm, get key ciphertext $CK=$ Encypt($\mathbb{R}, T_{do}, CF$), both ciphertext $EK(F)$ and key ciphertext CK will be published to cloud storage. The prrcise ID set $\mathbb{R}=\{ID_{u1},...,ID_{um}\}$, access control tree T_{do}, and all user public key CF in $\mathbb{R}$ will be the parameters when decryption.
3. All users can get ciphertext from cloud storage. When data users 的 decrypt ciphertext, the first step is to judge whether the user ID belong to the corresponding domain, and if true, DM will generate user identity private key SK_{iu} and user attributes SAK_{iu}. DM first determine whether DU has precise ID that matches identity set of DM, if true, DU can obtain the decryption key by decryption CK; otherwise, continue to judge whether the attribute satisfy the access control policies T. When the above conditions are met, DU can decrypt the data using Decrypt algorithm.

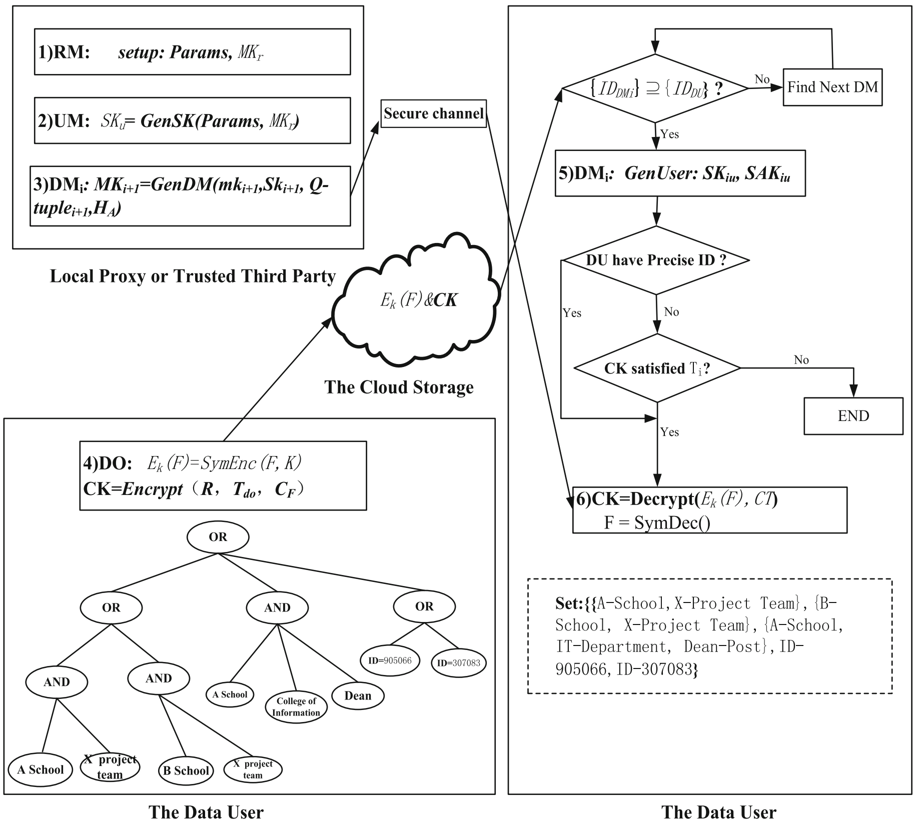
1)RM: setup: Params, MK_r
2)UM: SK_u= GenSK(Params, MK_r)
3)DM$_i$: MK_{i+1}=GenDM(mk_{i+1},Sk_{i+1}, Q-tuple$_{i+1}$,H_A)
Local Proxy or Trusted Third Party
Secure channel
$E_k(F)$&CK
The Cloud Storage
4)DO: $E_k(F)$=SymEnc (F, K)
CK=Encrypt（R，T_{do}，C_F）
OR
OR
AND
OR
AND
AND
A School
College of Information
Dean
ID=905066
ID=307083
A School
X project team
B School
X project team
The Data User
{ID_{DMi}} ⊇ {ID_{DU}} ?
No
Find Next DM
Yes
5)DM$_i$: GenUser: SK_{iu}, SAK_{iu}
DU have Precise ID ?
Yes
No
CK satisfied T_i?
No
END
Yes
6)CK=Decrypt($E_k(F)$, CT)
F = SymDec()
Set:{{A-School, X-Project Team}, {B-School, X-Project Team}, {A-School, IT-Department, Dean-Post}, ID-905066, ID-307083}
The Data User

Fig. 2. Ciphertext access control process based on hierarchical CP-ABE

The safety of this scheme depends on the safety of CP-ABE algorithm and HIBE algorithm, due to space limited, we did not prove it in this paper.

3 Access Control Optimization Technology Based on Local Proxy

3.1 Time Cost of Ciphertext Access Control Scheme Based on Hierarchical CP-ABE

The mainly time cost of ciphertext access control based on hierarchical CP-ABE is as follow:

1. System initialization and key generation time cost. Setup algorithm of RM node is bilinear operation, which will be the largest calculation time cost, and followed by power operation which include once in Setup algorithm, twice in GenDM, once in GenSK and GenUser algorithm.

2. Encryption and decryption time cost. Encryption time cost include symmetric encryption algorithm and key encryption time cost using CP-ABE algorithm. As CP-ABE is asymmetric algorithm, encryption efficiency is very low, the symmetric algorithm time cost can be negligible. Decryption algorithm include decryption time cost of key ciphertext CK and decryption of ciphertext, in fact time cost of decryption is much smaller than encryption.
3. Key distribution time cost. This means the time cost that UM and DM publish all user's private keys, identity of a keys, attribute keys, this CP-ABE based keys distribution need transform access control policy to access control tree T. In general, access policies conversion is simply pretreatment, time cost is relatively small.
4. Right revocation time cost. Revocation operation is re-encryption operation of file F and key k, its process is: data owner retrieve the affected ciphertext $E_k(F)$ and *CK* and decryption, then use a new key *k'* re-encrypt the file *F* and figure out ciphertext *Ek'*(*F*). After that, build a new access control structure *T'*, and re-encrypt *k'*. The new ciphertext *Ek'*(*F*) and *CK'* will be updated to cloud storage, and outdated ciphertext data will be deleted. The main time cost in revocation operation is data re-encryption.

The time cost of system initialization and key generation have the maximum cost for the reason of including bilinear operation and power operation. To improve the efficiency of the system, using local proxy can greatly improve the efficiency of cloud storage access, the advantages are as follows:

- Take full advantage of existing computing resources and storage resources of enterprise or community. The local proxy can be established using their existing equipment, so can save costs and avoid equipment idle.
- Protect data sharing security. Local proxy can be considered fully credible, and can achieve compulsive access control policy or implement ciphertext access control for protection of data sharing security.
- Enhance cloud storage access efficiency. Local proxy can be used to complete the operation of system initialization and key generation that take larger time cost in ciphertext access control, and cache frequently used data, reduce the frequency of cloud storage access, and improve the efficiency of cloud storage access.
- Protect sensitive data. Corporate data involves sensitive content and need to be encrypted before upload to cloud, furthermore, part of the data can be stored in local storage to avoid critical data disclosure when published to the cloud storage.

To sum up, using local proxy is an effective way to optimize the efficiency of cloud storage access.

3.2 Ciphertext Access Control Scheme Based on Local Proxy

Ciphertex access control scheme based on hierarchical CP-ABE use hierarchical structure and domain management about users and attributes. Therefore, we propose to use multi-local proxy in the scheme design, one hand, this can avoid bottleneck of single access proxy, the other is suitable the actual application scenarios of enterprise geographically dispersed. We assume that each local proxy implement writing operation in its own exclusive space, but data can be read by other agents. The main functions of local proxy design are as follows:

- Cloud storage access. Users interact with local proxy which replace users to implement encryption and decryption operations, and interact with cloud storage service, moreover, local proxy are also responsible for uploading encrypted files and download data, users no longer directly access cloud storage.
- Ciphertext access control. In our scheme, initialization operation in RM, user management in UM, and key generation and distribution in DM can be implemented by local proxy, and local proxy can achieve the Ciphertex access control policy based on hierarchical CP-ABE. This can effectively reduce the impact of inefficiency.
- Local data cache. Local proxy cache can set buffer which can cache high-frequency data, and reduce the frequency of cloud storage access.

Local proxy is mainly composed of three parts: storage interface, data processing services and data storage services, Fig. 3 shows its basic structure below.

In Fig. 3, the object storage interface which provided by LP for data access, is up to standard with cloud management interface CDMI [12], support PUT, GET, Delete, FTP, NFS and other operations. In addition, we also designed a permission configure interface to support specified data access strategy before uploading.

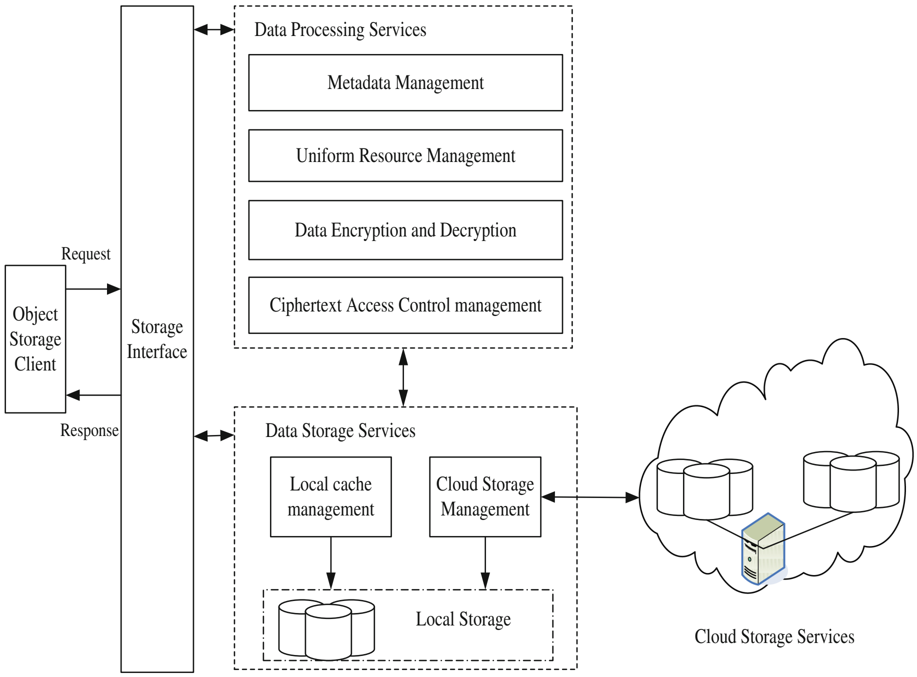

Fig. 3. Local resided proxy structure

The main function of Data Processing Services is to interact with user and receive user data, put data to the lower data storage services, and publish data to cloud. Furthermore, it receives user's instructions and gets response data from Data Storage Services, then put them to user. Data Processing Services include Metadata Management, Uniform Resource Management, Data Encryption and Decryption, Ciphertext Access Control Management and other functions.

LP generate meta data for each user data object, which include the information of data object identifier, data access policy, access time, the state of data objects, data location, key location and ciphertext location etc., also contains LP identifier which is used to distinguish released data form different proxy. URM includes resource scheduling module and duplicate management module, which is mainly responsible for the application and dispatch of resources. Data Encryption and Decryption is mainly used for data confidentiality protection, we use AES-128 algorithm in the scheme. Ciphertext Access Control Management can complete main work of access control based on hierarchical CP-ABE, including system initialization, user management, key generation, key distribution and other operations, it can take on most of overhead when users use cloud storage services, and can improve access efficiency.

Data Storage Services is responsible for manage local data cache and data storage in cloud. Local cache use LP itself storage, that to store higher frequency access data objects by using principle of the most recently used priority. Cloud Storage Management is mainly responsible for the storage of data stored in the cloud storage services, including upload/download data, optimizing data layout and other means to reduce user costs.

Let local proxy be $LP_1, LP_2, \ldots, LP_m$, m represents the total number of local proxy, let *LPj.urls* be the data published by LP_j, let data publishing as an example, the algorithm of data processing by local proxy are shown as Algorithm 1 below.

```
Algorithm 1. Data Publishing algorithm by Local resided Proxy
INPUT:  Data Owner $DO$,Local resided Proxy $LP_j$, Data Object $O_i$, Public parameters $Params$, Access strategy $T$
OUTPUT: Data location in Cloud Storage $LP_j.urls$

1. $DO$ send $O_i$ to $LP_j$, Specifies the access strategy $T$ of $O_i$
2. $LP_j$ creat Metadata
   { meta.state = creating; meta.LPID = LP.ID; meta.ID = $O_i$.ID; meta.T = $T$;
     meta.urla = urla        //urla: $O_i$ storage Localtion
     meta.ekurl = urlb       //urlb: cipher of key storage Localtion   }
3. $LP_j$ creat radom document key $K_i$
4. URM creat Replication strategy
5. $LP_j$ run AES-128 encryption algorithm $E$ encrypt $O_i$, get $E_{ki}(O_i)$
6. $LP_j$ run Setup(), GenDM(), GenUser()
7. $LP_j$ run Encrypt algorithm $E'$, use $T$ encrypt $K_i$, get cipher of key $E'_T(\mathbf{K}_i)$
8. $LP_j$ upload $E_{ki}(O_i)$, $E'_T(K_i)$ to cloud storage
9. $LP_j$ return $LP_j.urls$
```

4 Experimental Results and Analysis

4.1 Experimental Data Selection

We use similar data sets to create simulation experimental data for lack of real data set using in access control based on hierarchy CP-ABE scheme. We select posting data of Netease Forum and TianYa BBS to create testing data set, among these data we select 2,854,693 posting record of Netease forum and 3,463,786 posting records in TianYa BBS from six different forums of each, all the data have filtered out pictures, video, audio etc. for testing. NetEase and TianYa BBS are used to simulate different Affiliation in the hierarchy structure, and the different forums section are used to simulate different departments, the post landlord, the post follower, the landlord's fans are used to emulate different positions, we set the user's login name as the identity ID, and area, integration, post time etc. as the attribute information.

We created each post as a data object, because the actual data objects are too large for testing, we selected data objects within 10 M. There are totally 173,598 data objects that are created, the total size of the data objects are 1329.3 GB. We set 2,618,952 data access operations in total, which include 481,248 writing operation data, about 252 GB, and 2,137,704 reading operation data objects, about 587.6 GB. We assume that each data object has 12 leaf nodes access control tree, and each read operation have been authorized.

4.2 Experimental Results and Analysis

The prototype system for testing is written in C++ running on Linux platform. The prototype system includes a local agent and a client program, local agent performs the operation of data publishing, data fetching, and ciphertext access control based on hierarchical CP-ABE etc., the client program access data through local agent. We use part of others work for reference in prototype system, which include: AES-128 and SHA256 algorithms; CP-ABE tools library [13]; The information dispersal algorithms routines in Jerasure library [14] which are used to implement the (k, n) threshold scheme. We use Hadoop distributed file system HDFS as the local cloud storage system, that are running on UBuntu platform server. Figures 4 and 5 show the time cost of publishing data and reading and writing data in different ways.

From the experimental data, we conclude that the time cost of ciphertext access control based on hierarchical CP-ABE are less than direct access control of CP-ABE, whether the operation is data publish operation or data retrieval operation, the experiment show that the access control scheme based on hierarchical CP-ABE can effectively provide access efficiency when using cloud storage service, but its time cost are greater than direct access storage cloud, the reason is to have increased the operation of ciphertext access control, which performs the protection of data privacy.

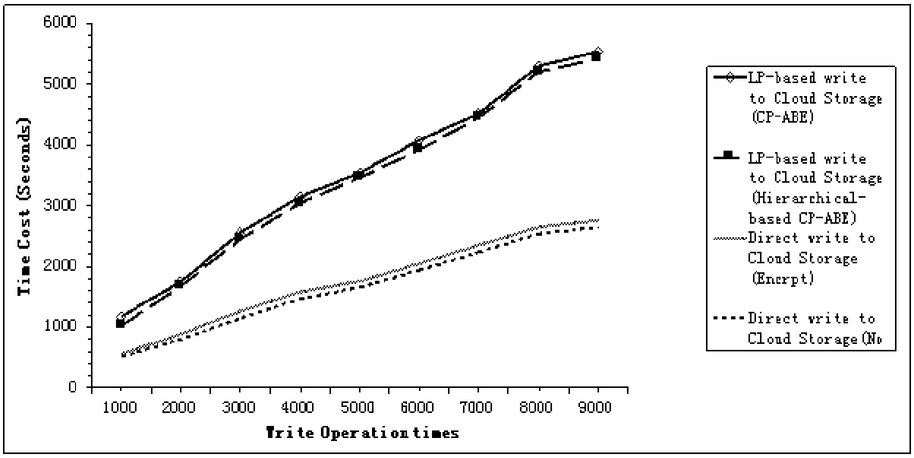

Fig. 4. Time cost comparison when publishing data in different ways

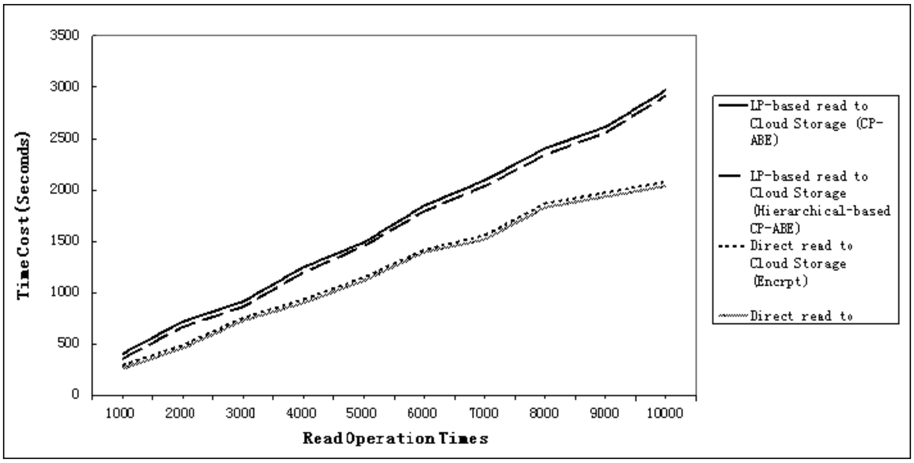

Fig. 5. Time cost comparison when retrieving data in different ways

5 Conclusion

In this paper we proposed solution about protection of data privacy when using cloud storage service from the user point of view. On the one hand, we proposed a scheme of ciphertext access control policy based on hierarchical CP-ABE which increase hierarchical structure in access right distribution and precise identity information in attribute distribution, the use of both precise identity and attributes at the same time can efficiently implement fine-grained access control, and support for key generation in hierarchy structure, this scheme can be more suitable for actual scenarios in cloud storage services. On the other hand, we proposed an efficiency access control optimizing technique based on local agency, which can implement data privacy protection. We showed the basic structure of local agents and data processing etc. experiments show that the scheme can effectively reduce the impact of using ciphertext access control mechanism and improve the access efficiency of cloud storage. Further research is the integrity protection of ciphertext, replication policy, access right revocation and optimization techniques that affect the efficiency of cloud storage access.

Acknowledgments. This research was financially supported by the Science Foundation of Education Department of Liaoning Province (No. L2014533). We thank for their support.

References

1. iResearch: China Cloud Storage Industry and User Behavior Research Report. http://report.iresearch.cn/1763.html
2. Borgmann, M., Hahn, T., Herfert, M.: On the security of cloud storage services. http://www.sit.fraunhofer.de/content/dam/sit/en/studies/Cloud-Storage-ecurity_a4.pdf
3. Shi, X.: The Core of Cloud Storage is Information Security, pp. 107–108. China Information Security, Beijing (2013)
4. Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privcy. O'Reilly, Media, Inc., Houston (2009)
5. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) RLCPS, WECSR, and WLC 2010. LNCS, vol. 6054, pp. 136–149. Springer, Heidelberg (2010)
6. Zhang, R., Chen, P.: A dynamic cryptographic access control scheme in cloud storage services. In: Proceedings of the 8th International Conference on Computing and Networking Technology, pp. 50–55. IEEE Press, New York (2012)
7. Lv, Z., Zhang, M., Feng, D.: Cryptographic access control scheme for cloud storage. Jisuanji Kexue yu Tansuo, pp. 835–844. Computer Research and Development, Beijing (2011)
8. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
9. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient and provably secure realization. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography, pp. 53–70. Taormina, Italy (2011)
10. Goyal, V., Pandey, O., Sahai, A.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM Press, New York (2006)
11. Yu, S., Wang, C., Ren, K.: Achieving secure, scalable and fine-grained data access control in cloud computing. In: Proceedings of the IEEE INFOCOM 2010, pp. 19. IEEE Press, New York (2010)
12. SNIA: Cloud Data Management Interface (CDMI). http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf
13. Bethencourt, J., Sahai, A., Waters, B.: Advanced crypto software collection ciphertext–policy attribute–based encryption. http://acsc.cs.utexas.edu/cpabe/
14. Plank, J.S., Simmerman, S., Schuman, C.D.: Jerasure: A library in C/C++ facilitating erasure coding for storage applications – version 1.2. http://web.eecs.utk.edu/~plank/plank/papers/CS-08-627.html