Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 6808190, 10 pages
https://doi.org/10.1155/2017/6808190

Research Article

Modified Ciphertext-Policy Attribute-Based Encryption Scheme with Efficient Revocation for PHR System

Hongying Zheng,[1] Jieming Wu,[2] Bo Wang,[2] and Jianyong Chen[2]

[1]*School of Software Engineering, Shenzhen Institute of Information Technology, Shenzhen, China*
[2]*School of Computer and Software Engineering, Shenzhen University, Shenzhen, China*

Correspondence should be addressed to Jianyong Chen; jychen@szu.edu.cn

Received 26 January 2017; Accepted 3 August 2017; Published 30 August 2017

Academic Editor: Haipeng Peng

Attribute-based encryption (ABE) is considered a promising technique for cloud storage where multiple accessors may read the same file. For storage system with specific personal health record (PHR), we propose a modified ciphertext-policy attribute-based encryption scheme with expressive and flexible access policy for public domains. Our scheme supports multiauthority scenario, in which the authorities work independently without an authentication center. For attribute revocation, it can generate different update parameters for different accessors to effectively resist both accessor collusion and authority collusion. Moreover, a blacklist mechanism is designed to resist role-based collusion. Simulations show that the proposed scheme can achieve better performance with less storage occupation, computation assumption, and revocation cost compared with other schemes.

1. Introduction

Personal health record (PHR) system is a novel application that can bring great convenience in healthcare. The privacy and security of PHR are the major concerns of the users, which could hinder further development and wide adoption of the system [1, 2]. PHR is a typical usage of cloud storage, taking advantages of elastic computing resources to provide flexible, pervasive, and on-demand health cloud service. Patients store their PHRs in cloud storage servers and therefore can share these data with friends or doctors conveniently. However, such promising cloud-based application meets new security challenges: (1) Since PHRs need to be shared among doctors, researchers, patients, and so on, the sharing scenario is complicated. Patients should be able to control the access in a fine-grained manner. (2) PHRs may be migrated among different cloud storage servers which cannot be fully trusted. Therefore, patients cannot rely on servers to protect their PHRs. Traditionally, outsourced data is usually encrypted with cipher-key and the storage servers are responsible for distributing cipher-keys to legal accessors. However, such mechanism is just secure in specific domain, but not suitable for PHR system which works across several domains.

It is significant to find out a fine-grained access control technique for PHR system. In recent years, attribute-based encryption (ABE) [3–8] seemed to be a promising technique for such one-file-multiaccess cloud storage scenario. In ABE algorithm, patient can control the security by directly specifying access policies for their outsourced PHRs, while the third-party entities, named authorities, are responsible for attribute management and key distribution. Cloud storage only needs to store the encrypted PHRs. In this way, PHR service is oriented to patients across several domains.

Typically, ABE schemes work in two models, key-policy ABE (KP-ABE) [9] and ciphertext-policy ABE (CP-ABE) [10]. KP-ABE applies policy in attribute keys of accessors. Therefore, once a key is predefined and is used to encrypt PHRs, accessors who can decrypt them are limited. Accessor can only decrypt the PHRs associated with a set of attributes that satisfies the key. That is to say, PHR owner should know all attributes that accessors own before he encrypts one PHR, so that he can associate a correct set of attributes. It is not natural and practical, unless the attributes of accessors are generated and distributed by PHR owner himself. CP-ABE scheme works in the opposite manner, which is conceptually

closer to the traditional access control methods, such as Role-Based Access Control (RBAC) [10]. The access policy is set by PHR owner during PHR encryption, where the policy is a Boolean formula consisting of public attributes and logical operations, like "AND" and "OR." PHR owner does not need to know who can access his PHRs because it is responsibility of authority. Only the accessors with attributes that satisfy access policy can decrypt ciphertext of PHR. Evidently, it is more reasonable to implement CP-ABE scheme in public attributes scenario, and it is also convenient for PHR owner without keeping online all the time.

Based on the application scenarios of KP-ABE and CP-ABE, Li et al. [11] proposed a PHR system framework that combines KP-ABE and CP-ABE together. In the framework, users are divided into personal domains (PSDs) and public domains (PUDs) according to their roles. Usually, PHR owners (patients) normally knows users who access the system via PSDs. It would be better to apply revocable KP-ABE scheme for PSDs [12], so that patients are responsible for defining attributes and authorizing accessors. Professional users access the system via PUDs. They should have public roles, such as doctor and researcher. Therefore, it is better for the attributes in PUD to be defined and authorized by third-party attribute authorities (abbreviated as AA_s in this paper). Li et al. uses Chase-Chow multiauthority ABE scheme (CC MA-ABE) [13] with an attribute revocation method to control the attributes in PUDs.

Although there are some advantages for the division of user domains, several shortcomings still exist for Li's ABE scheme [11] (abbreviated as Li's MA-ABE), which are listed as follows: (1) Since it works based on CC MA-ABE which is exactly a variant KP-ABE scheme, it is limited on a strict "AND" policy over a predetermined set of authorities. As commented by Lewko and Waters [14], such policy is not flexible and expressive. In order to get the same function of CP-ABE, it uses an additional conjunctive normal form (CNF) rule for generation of both policy and encryption. (2) PUDs and PSDs have to apply different ABE schemes and work in parallel. However, our paper reveals an implicit collusion, named role-based collusion, between users from PUDs and PSDs. Specifically, users in PSDs may also have professional roles, such as doctors with public attributes in PUDs. In this situation, one PHR owner can prevent specific accessor from PSD by associating his PHR with a set of PSD attributes but may fail to prevent this accessor from accessing via PUD. For example, patient A has a friend B who works as a physician in hospital C. Patient A goes to hospital C for diagnosis. He specifies an access policy for his encrypted PHR to allow all the physicians in hospital C access. However, he suddenly remembers that his friend B also works there and he does not want him to know the diagnosis. Although patient A does not authorize friend B to decrypt via PSD, he cannot stop friend B from accessing via PUD.

There exist several MA-CP-ABE schemes [11, 13, 15–18], but they are not designed for PUD's scenario. Commented by paper [14], CC MA-ABE [13, 17] is limited by the strict "AND" policy. Muller et al. proposed an ABE scheme that can realize any access structure but needs an authentication center [16]. The usage of authentication center may face security and performance bottleneck because all the authorities should be controlled by center. Lin et al. [15] gave a scheme without authentication center but needs to fix the set of authorities ahead of time. It can resist collusion of users less than m, where m is a chosen parameter at setup phase. Lewko's ABE solution [14] is flexible but lacks attribute revocation mechanism. Ruj et al. proposed a solution based on Lewko's ABE to make attribute revocable [19]. However, it requires PHR owner to stay online for revocation and its efficiency is quite low.

More importantly, the role-based collusion which is significant for PHR system is not solved in these previous MA CP-ABE schemes. In order to resist the collusion, our proposed MA CP-ABE scheme designs a blacklist for owner. Each user (PHR owner) can specify a blacklist of accessor identities that cannot decrypt his data from PUD. This blacklist is delegated to a third-party authority that the owner trusts. The authority tags each blacklist with a unique public attribute in PUD, so that the owner can use this unique public attribute to specify his access policy. However, the amount of public attributes will increase linearly with PUD users, which results in a heavy burden for authorities.

Consequently, our paper aims to construct the CP-ABE scheme for PUD scenario which has efficient revocation and supports multiple authorities without an authentication center. Compared with Li's ABE scheme in PUD, our proposed scheme realizes access control with flexible access policy. Moreover, the proposed role-based collusion is also solved efficiently. Our contributions are concluded as follows.

(1) We propose a modified multiauthority CP-ABE scheme based on Lewko's scheme [14]. With it, PHR owner can specify flexible and expressive access policy to protect their outsourced PHRs. Meanwhile, authorities need not communicate with each other or be controlled by an authentication center. The number of attributes is almost unrestricted since the increase of attributes does not occupy more resources.

(2) We proposed an efficient attribute revocation mechanism for our scheme. Attribute can be revoked efficiently through the proxy reencryption and lazy revocation, while the scheme does not need an authentication center and any additional communications among authorities.

(3) To resist the role-based collusion, we suggest a blacklist solution to prevent it. By replacing the specific attribute master key and public key with hash value of attribute's descriptive name, the storages in authorities keep small even when number of attributes increases.

2. Related Work

Sahai and Waters [8] proposed the first ABE scheme, in which ciphertext is encrypted and associated with a set α of attributes. An accessor can successfully decrypt ciphertext if and only if he gets a set β of attributes components where the set overlap between the two attributes sets, that is, $|\alpha \cap \beta|$, is beyond a predefined threshold. Afterwards, Goyal et al.

[9] proposed KP-ABE scheme, in which a set of attributes from an accessor is constructed through a tree-like policy which is taken as key of the accessor. The leaf nodes of the tree associated with attributes and the nonleaf nodes are logical operations, such as "or" and "and." Data owner associates his ciphertext with a set of attributes. Once the associated attributes satisfy a specific key-policy of accessor, the accessor can decrypt the ciphertext. However, the data owner should know all the keys of accessors before he encrypts the data and then he can suitably associate the ciphertext with corresponding attributes. Such requirements of KP-ABE are not suitable for public access scenario, where the data owner cannot predict which person can access his data.

Consequently, Bethencourt et al. [10] proposed CP-ABE which is conceptually closer to the traditional access control methods, such as RBAC. CP-ABE scheme attaches access policy in ciphertext instead of attributes of accessors. It is more intuitive for the data owner to specify such policy at the time he encrypts the data. For accessors, they should own enough attributes issued by the third party, named authorities, to decrypt the ciphertext correctly. Furthermore, ordered binary decision diagram (OBDD) is used to describe access policies in CP-ABE. The system makes full use of both the powerful description ability and the high calculating efficiency of OBDD and improve both performance and efficiency [20]. However, only one single authority may cause bottleneck of performance [21]. Moreover, it is more natural and practical with multiple professional organizations (authorities) to manage distinct sets of attributes. Security can be improved with the multiauthority because an attacker should compromise several authorities at the same time to get the keys associated with enough sets of attributes for decryption.

There are already some attempts to solve multiauthority ABE problem with new cryptographic solutions. Chase and Chow [13] firstly proposed a multiauthority ABE scheme (CC MA-ABE) in which each user is authorized based on a global identifier (GID), such as a social security number. The GID plays a linchpin to associate users' keys from different authorities together. But the solution still relies on an authentication center and the access policy is not flexible and expressive which is limited on "AND" gate policy over the predetermined set of authorities. Later, Li et al. [11] proposed an ABE scheme with attribute revocation mechanism based on CC MA-ABE, which is limited on a rule of CNF in the access policy. A threshold multiauthority CP-ABE access control scheme was proposed for public cloud storage with which both security and performance are improved [22].

Actually, it is important for MA CP-ABE to support an expressive and flexible access policy. For example, American Medical Association (AMA) authorizes attributes of medical professional licenses, such as junior nurse license and experienced nurse license, while American Hospital Association (AHA) authorizes attributes of affiliations, such as hospital A and hospital B. If one patient thinks that the diagnosis and treatment in hospital A are better than those in hospital B, he may specify an access policy that permits the nurses with any level of license in hospital A to access his PHR files, and only allow the nurses with junior level of license from hospital B access. Such expressive policy is presented as policy = ((/junior nurse level/ $\vee$ /experienced nurse level/) $\wedge$ /hospital A/) $\vee$ (/junior nurse level/ $\wedge$ /hospital B/). The policy can be transformed to the "AND" policy; for example, policy = $\{(A_1 = a_{1,1}) \vee \cdots \vee (A_1 = a_{1,d_1})\} \wedge \cdots \wedge \{(A_m = a_{m,1}) \vee \cdots \vee (A_m = a_{m,d_m})\}$, where A_m refers to the mth authority and a_{m,d_i} refers to the policy managed by A_m and one authority has only one clause [11].

There are some other schemes which can set the access policy in any Boolean formula over attributes from any number of authorities. Among them, Muller proposed another MA-ABE scheme which is realized on any access structure with an authentication center. Yang and Jia [18] proposed a variant CP-ABE scheme to support multiauthority, but it still requires an additional authentication center to generate user secret key and authority secret key. Moreover, it is weak in revocation security. Based on Yang's scheme, an extensive scheme was proposed to withstand the vulnerability [23]. For MA-ABE scheme with an authentication center to control multiple authorities, once the authentication center is broken, the entire ABE system will be compromised. Therefore, it should be fully trusted which is hard to guarantee. Moreover, the whole ABE system is hard to be expanded. Some researches try to remove the authentication center from MA CP-ABE schemes. Chase and Chow [13] used pseudorandom functions (PRFs) between different authorities without the center. However, it is still limited on "AND" access policy over a determined set of authorities. Lin et al. [15] proposed a threshold based ABE scheme that is decentralized and enforces an efficient attribute revocation scheme. The system is collusion-resistant for fewer m users, where m is chosen statically during the setup phase. However, the authorities set should be configured before the setup phase and is fixed in running. The authorities should interact with each other at the setup phase and the access policy is inflexible. Later, Lewko and Waters [14] proposed a scheme for decentralized ABE scenario, in which the authorities work independently without coordination among them. A main drawback is that the scheme has no revocation function. Although a further paper (DACC) [19] addressed it, the computations of key update and communication overhead for attribute revocation are quite heavy. Besides, DACC requires the data owner to take part in revocation and transmit an updated ciphertext component to every unrevoked user. It means that the data owner should keep being online all the time, as is unreasonable in practical application scenario.

Attribute revocation is an important issue for an ABE system and benefits security of the system. Once a malicious user is identified by an authority, all his attributes or one of his specific attributes should be revoked by the authority, which means the malicious user can no longer decrypt the ABE-generated ciphertext associated with those attributes. In single authority ABE scheme, Yu et al. [7] introduced the concept of proxy reencryption into CP-ABE to realize attribute revocation, in which the affected attribute components of ciphertext and the attributes components stored in terminals of unrevoked users are updated via reencryption. Inspired by paper [7], Yang and Jia [18] proposed the CP-ABE scheme

Table 1: Comparison among previous MA CP-ABE schemes and ours.

	Lin [15]	Muller [16]	Chase [17]	Lewko [14]	DACC [19]	Li [11]	Yang [18]	Ours
Flexible access policy	√	√		√	√		√	√
Resistance of accessor collusion		√	√	√	√	√	√	√
Without an authentication center	√		√	√	√	√		√
Authority independence	√	√		√	√	√	√	√
Efficient revocation			√	—		√	√	√

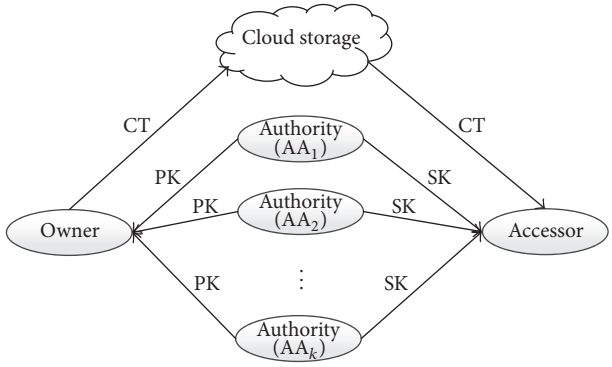

Figure 1: MA CP-ABE system model.

with a more efficient revocation than that in [19]. However, it requires an authentication center to control the multiple authorities. Based on the above depiction, the comparisons among previous MA CP-ABE schemes and our proposed scheme are listed in Table 1.

3. System Model and Security Definition for MA CP-ABE

3.1. System Model. The MA CP-ABE scheme for PUD involves three kinds of participants, that is, cloud storage, authorities, and users (including data owner and accessors), as shown in Figure 1. The scheme consists of five basic algorithms: *System Setup*, *Authority Setup*, *Encrypt*, *KeyGen*, and *Decrypt*. They are described as follows.

System Setup $(\lambda) \rightarrow (para)$. The setup algorithm takes security parameter λ as input and outputs global parameters *para*.

Authority Setup $(para) \rightarrow (msk, pk)$. Each attribute authority (AA) runs its own authority setup process. The setup algorithm takes system global parameters *para* and AA's descriptive attributes as input. Then, for each attribute that AA manages, AA generates a master key *msk* and the corresponding public key *pk*. The master keys msk_s are kept secret, while the public keys pk_s are published.

Encrypt $(D, para, policy\ \mathcal{T}, pk_s) \rightarrow (CT = \{D^{/}, policy\ \mathcal{T}, pAC_s\})$. Once the data owner gets public keys pk_s from authorities, he can execute encryption process in his own terminal. The algorithm takes pk_s from several authorities, data D for encryption, and an access policy $\mathcal{T}$ specified by the data owner as inputs. Then, the algorithm encrypts D to a ciphertext $D^{/}$ and generates a public attribute component (abbreviated as pAC) for each leaf node of $\mathcal{T}$. The whole data tuple of $CT = \{D^{/}, policy\ \mathcal{T}, \mathrm{pAC}_s\}$ is the final ciphertext tuple and is uploaded to cloud storage.

KeyGen $(para, msk) \rightarrow (SK : \{uAC_s\})$. Each authority manages its own attributes set and is responsible for key distribution to legal users (accessors). Once an authority authenticates identity of an accessor, it will process key generation which takes the master keys mk_s for a requested set of attributes $\omega^{/}$ as input and outputs user attribute components (abbreviated as uAC_s) for each attribute. All the attributes uAC_s generated for the specific accessor are collected as secret key of the accessor SK and sent back to the accessor secretly.

Decrypt $(para, CT, SK_s, pk_s) \rightarrow (M)$. An accessor executes the decryption algorithm which takes the ciphertext tuple CT from cloud storage and the public keys pk_s and secret keys SK_s from authorities as inputs. If the attributes set associated with SK_s satisfies access policy $\mathcal{T}$, the accessor can decrypt the plaintext data M. Otherwise, it returns an error symbol $\perp$.

3.2. PHR Upload and Access. Based on CP-ABE scheme (Figure 1), we can easily figure out the PHR upload and PHR access procedures. Specifically, once a data owner needs to upload his specific PHR file *"pFile"* to cloud storage, he does the following steps: (1) Cut the data into contents segments *s*. (2) Pick random content key *ck* for each content segment. (3) Encrypt the segment via symmetric cryptography and get result $s^{/} = E_{ck}(s)$. (4) Define an access policy over a set of attributes, encrypt content key *ck* as owner data *M* via our proposed MA CP-ABE scheme, and get the ciphertext tuple CT. (6) Finally upload $s^{/}$ and CT together as an integrated tuple to the cloud storage. The data owner can go offline and authorities perform other key distribution workflows.

When an accessor needs to read the plaintext of one specific PHR on the cloud storage, he should process the following steps: (1) Get the whole ciphertext tuple $s^{/}$ and CT from the cloud storage. (2) Read the access policy from the CT and know a minimal set of attributes required for decryption. (3) Get identity authenticated by several authorities, with which these authorities can return the keys associated with attributes (uAC_s) to the accessor, respectively. (4) Collect enough keys to recover content key *ck* from CT. (5) Decrypt $s^{/}$ to *s* via symmetric cryptography by content key *ck* and then construct the original PHR file *"pFile."*

4. Modified MA CP-ABE Scheme for PUD

4.1. Scheme Construction. Our proposed MA CP-ABE scheme has five algorithms, that is, *System Setup, Authority Setup, KeyGen, Encrypt, and Decrypt*. They are depicted as follows.

System Setup $\rightarrow$ (*para*). System first selects a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$ and bilinear map function $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and then picks a generator g_1 of $\mathbb{G}_{p_1}$ [14, 24]. A hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is used to map global identities GID_s of an accessor and descriptive names of his attributes, such as *doctor*, to elements in $\mathbb{G}$. Once the hash function is fixed, the value $H(\mathrm{GID})$ is modelled as a random oracle. Finally, all these system parameters are published as $para = (\hat{e}, g_1, H(\cdot), N)$.

Authority Setup $(para) \rightarrow (msk, pk)$. For each authority AA_k which manages attributes set $\mathbb{A}_k$, AA_k takes *para* as input and generates two public keys as $g_1^{\alpha_k}$, $g_1^{\beta_k}$ where the two values α_k, β_k are picked randomly from $\mathbb{Z}_N$. The values $msk_k = (\alpha_k, \beta_k)$ are stored secretly by AA_k as master keys, while the public keys $pk_k = (g_1^{\alpha_k}, g_1^{\beta_k})$ are published.

KeyGen $(para, msk) \rightarrow (SK = \{uAC_s\})$. Suppose that a legal accessor with *GID* requests authority AA_k for attributes set A_u and he owns attributes set $A_{u,k}$ in AA_k. Then AA_k will generate secret key (SK) of the accessor which is associated with attributes set $A_u \cap A_{u,k}$. Specifically, for each attribute $i \in A_u \cap A_{u,k}$, AA_k generates a user attribute component $(\mathrm{uAC}_i = H(i)^{\alpha_k} \cdot H(\mathrm{GID})^{\beta_k})$ for the accessor. Finally, all the components $\{\mathrm{uAC}_i\}|_{i \in A_u \cap A_{u,k}}$ are combined as secret keys of the accessor and $\mathrm{SK} = \{\mathrm{uACs}\}$ is sent back to the accessor secretly for further decryption.

Encrypt $(D, para, policy\ \mathcal{T}, pk_s) \rightarrow (CT = \{D^{/}, policy\ \mathcal{T}, pAC_s\})$. In encryption phase, the data owner specifies an access policy tree $\mathcal{T}$ to restrict the accessors. The encryption algorithm encrypts data D into $D^{/} = D \cdot \hat{e}(g_1, g_1)^s$, where the value $s \in \mathbb{Z}_n$ is selected randomly. Meanwhile, a set of public attribute components (pAC_s) will be generated according to the value s and the access policy $\mathcal{T}$.

Specifically, as shown in previous paper [11], any monotone access tree $\mathcal{T}$ can be translated to an access structure $(\mathscr{M}, \rho)$ over the involved attributes, where $\mathscr{M}$ is a $\ell \times n$ matrix and ℓ denotes the number of leaf nodes in the access tree $\mathcal{T}$. The function ρ maps the xth row of matrix $\mathscr{M}_x$ to an attribute $i = \rho(\mathscr{M}_x)$. The encryption algorithm chooses two random vectors $\overrightarrow{v} = (s, r_2, \ldots, r_n) \in \mathbb{Z}_N^n$ and $\overrightarrow{v'} = (0, r_2', \ldots, r_n') \in \mathbb{Z}_N^n$ and then computes $\lambda_x = \overrightarrow{v} \cdot \mathscr{M}_x$ and $\nu_x = \overrightarrow{v'} \cdot \mathscr{M}_x$. Notice that the former vector $\overrightarrow{v}$ is used to distribute the value s, while the latter vector formula distributes the zero value 0. For each leaf node x of $\mathcal{T}$ associated with attribute $i = \rho(\mathscr{M}_x)$, the algorithm computes the three pAC_s as follows, where the value μ_x is picked arbitrarily in $\mathbb{Z}_n$

$$\begin{aligned} \mathrm{pAC}_{0\,(x,i)} &= \hat{e}(g_1, g_1)^{\lambda_x} \cdot \hat{e}(H(i), g_1)^{\alpha_k \cdot \mu_x}, \\ \mathrm{pAC}_{1\,(x,i)} &= g_1^{\mu_x}, \\ \mathrm{pAC}_{2\,(x,i)} &= g_1^{\beta_k \cdot \mu_x + \omega_x}. \end{aligned} \tag{1}$$

Finally, the owner sends the ciphertext $D^{/}$ together with pAC_s and access structure $(\mathscr{M}, \rho)$ to the semitrust cloud storage. The uploaded data CT is presented as

$$\begin{aligned} CT = \Big(D^{/}, (\mathscr{M}, \rho), \{\mathrm{pAC}_{0(x,i)}, \mathrm{pAC}_{1(x,i)}, \mathrm{pAC}_{2(x,i)}\} \mid \\ (i = \rho(\mathscr{M}_x)) \,\&\, (1 \le x \le R)\Big). \end{aligned} \tag{2}$$

Decrypt $(para, CT, SK_s, pk_s) \rightarrow (D)$. An accessor receives CT from the cloud storage, finds out the minimal set of attributes $\mathbb{A}_u$ for decryption according to the policy $\mathcal{T}$, and then requests corresponding AA_s for attributes (uAC_s). Notice that the minimal attributes set $\mathbb{A}_u$ is mapped to ℓ' rows of matrix $\mathscr{M}$. The rows set is labeled as $\{I_x\}$, where $|\{I_x\}| = \ell'$ and $\ell' \le \ell$. According to submatrix $\{I_x\}$, the algorithm can compute ℓ' values $\{\zeta_x \in \mathbb{Z}_n\}|_{x \in \{I_x\}}$, which has the relationship with $s = \sum_{x \in \{I_x\}} \zeta_x \lambda_x$ and $0 = \sum_{x \in \{I_x\}} \zeta_x \omega \omega_x$ *(interpolation)*.

Consequently, for each leaf node which is associated with the xth row of $\{I_x\}$, the algorithm can decrypt it via the following formula:

$$\begin{aligned} &\frac{\mathrm{pAC}_{0\_(x,i)} \cdot \hat{e}\left(H(\mathrm{GID}), \mathrm{pAC}_{2_{x,i}}\right)}{\hat{e}\left(\mathrm{uAC}_i, \mathrm{pAC}_{1_{x,i}}\right)} \\ &= \frac{\hat{e}(g_1, g_2)^{\lambda_x} \cdot \hat{e}(H(i), g_1)^{\alpha_k \cdot \mu_x} \cdot \hat{e}\left(H(\mathrm{GID}), g_1^{\beta_k \cdot \mu_x + \omega_x}\right)}{\hat{e}\left(H(i)^{\alpha_k} \cdot H(\mathrm{GID})^{\beta_k}, g_1^{\mu_x}\right)} \\ &= \hat{e}(g_1, g_1)^{\lambda_x} \cdot \hat{e}(H(\mathrm{GID}), g_1)^{\omega_x}. \end{aligned} \tag{3}$$

By collecting ℓ' decryption values of leaf nodes, the algorithm can easily recover value $\hat{e}(g_1, g_1)^s$ via interpolation depicted as follows:

$$\begin{aligned} &\prod_{x \in \{I_x\}} \left(\hat{e}(g_1, g_1)^{\lambda_x} \cdot \hat{e}(H(\mathrm{GID}), g_1)^{\omega_x}\right)^{\zeta_x} \\ &= \hat{e}(g_1, g_1)^{\sum_{x \in \{I_x\}} (\zeta_x \cdot \lambda_x)} \cdot \hat{e}(H(\mathrm{GID}), g_1)^{\sum_{x \in \{I_x\}} (\zeta_x \cdot \omega_x)} \\ &= \hat{e}(g_1, g_1)^s \cdot \hat{e}(H(\mathrm{GID}), g_1)^0 = \hat{e}(g_1, g_1)^s. \end{aligned} \tag{4}$$

Finally, the plaintext D is computed by $D = D'/\hat{e}(g_1, g_1)^s$.

4.2. Efficient Lazy Revocation. There are two levels of revocation, that is, attribute revocation and accessor revocation. The attribute revocation is done by updating the attribute associated pACs stored in cloud storage, so that the previous authenticated pACs is no longer useful for decryption. The accessor revocation can be done by revocation of all the attributes that an accessor owns.

Normally, the command of attribute revocation is started from authority when there are changes in management of accessors. Firstly, authority AA_k sends update parameter to

the cloud storage and then the cloud storage updates pAC_s via proxy reencryption technique [12]. In our revocation scheme, the corresponding pAC_s will not be updated until someone requests them. Specifically, the cloud storage stores the update parameters in an attribute history list (AHL) for each attribute revocation command. Once a ciphertext (associated with a set of pAC_s) is requested, it can be updated only once according to AHL, although the update parameters have been updated many times and recorded in AHL. Such mechanism is called lazy revocation, which can accumulate update of parameters over time. Our revocation model is more efficient than DACC's solution [19] when AA_k delegates most computation workloads to the cloud storage and the lazy revocation is used.

For accessors, once pAC_s stored in the cloud storage is updated, their corresponding uAC_s can no longer decrypt the ciphertext. Consequently, these accessors need to request authorities to update parameters. Instead of regenerating the accessors' uAC_s, the authorities can simply generate parameters, that is, update keys (UK_s), and let these accessors update their uAC_s at their terminal.

In previous papers [11, 12, 25], the revocation methods will generate the same update keys for all accessors. This is efficient but weak in security. Therefore, our proposed revocation scheme can support two methods. One method is to generate the same update parameters for all accessors, and the other one is to generate different update parameters for different accessors. It is obvious that the former method is efficient but has potential risk in some circumstance. The latter method is the opposite. PHR system can choose either method according to its strategy and environment.

Attribute Revocation $(para, msk) \rightarrow (UK_{aAC}, UK_{pAC})$. To execute the revocation command for attribute i, its corresponding authority AA_k takes public system parameters *para* and its own master key (α_k, β_k) as input. Then AA_k generates regeneration key UK_{pAC} for the cloud storage and generates UK_{aAC} for the accessors. All these regeneration keys are transmitted secretly.

Method 1 (Same Update Parameter). Specifically, AA_k selects a random value $\alpha' \in Z_N$ and then generates $UK_{aAC_i} = UK_{pAC_i} = H(i)^{\alpha'_k - \alpha_k}$. The cloud storage updates the attribute i associated $pAC_{0\_(x,i)}$ through (5). uAC_i of the accessor is updated through (6) at the terminals of accessors or at the authority

$$\begin{aligned} pAc'_{0\_(x,i)} &= pAC_{0\_(x,i)} \cdot \hat{e}\left(UK_{pAC_i}, pAC_{1_{x,i}}\right) \\ &= \hat{e}\left(g_1, g_1\right)^{\lambda_x} \cdot \hat{e}\left(H\left(i\right), g_1\right)^{\alpha'_k \cdot \mu_x}, \end{aligned} \tag{5}$$

$$uAC'_i = uAC_i \cdot UK_{aAC_i} = H\left(i\right)^{\alpha'_k} \cdot H\left(\mathrm{GID}\right)^{\beta_k}. \tag{6}$$

Method 2 (Different Update Parameters). Specifically, AA_k selects random values $\alpha'_k, \beta'_k \in \mathbb{Z}_n$ and generates $UKpACi = H(i)^{\alpha'_k - \alpha_k}$ and $UKpACi = \beta'_k - \beta k$ for the cloud storage. For each accessor with GID, AA_k generates specific $UKaACi, GID = H(i)^{\alpha'_k - \alpha_k} \cdot H(\mathrm{GID})^{\beta'_k - \beta_k}$. The cloud storage updates the attribute i associated $pAC0\_(x,i)$ and $pAC2\_(x,i)$ through (7) and (8). The accessor's $uACi$ is updated through (9)

$$\begin{aligned} pAC'_{0\_(x,i)} &= pAC_{0\_(x,i)} \cdot \hat{e}\left(UK_{pAC_i}, pAC_{1_{x,i}}\right) \\ &= \hat{e}\left(g_1, g_1\right)^{\lambda_x} \cdot \hat{e}\left(H\left(i\right), g_1\right)^{\alpha'_k \cdot \mu_x} \end{aligned} \tag{7}$$

$$pAC'_{2\_(x,i)} = pAC_{2\_(x,i)} \cdot PAC_{1_{x,i}}^{UK_{pAC_i}} = g_1^{\beta'_k \cdot \mu_x + \omega_x} \tag{8}$$

$$uAC'_i = uAC_i \cdot UK_{\alpha AC_i, \mathrm{GID}} = H\left(i\right)^{\alpha'_k} \cdot H\left(\mathrm{GID}\right)^{\beta'_k}. \tag{9}$$

Accessor Revocation. Supposing that the attributes set $\mathbb{A}_\alpha$ is owned by the accessor, the corresponding authority AA_k can execute attribute revocations for these $|\mathbb{A}_\alpha|$ attributes in total. Moreover, to avoid fake revocation commands, both the authority and the cloud storage use digital signature technique to confirm validity as implemented in paper [12].

4.3. Collusion Resistant. The same as most of previous papers [11, 18], our proposed MA CP-ABE scheme can resist both accessor collusion and authority collusion. Besides, the malicious but implicit role-based collusion can also be resisted.

As discussed in Introduction, role-based collusion is caused by the fact that PHR owner cannot predict the exact user identity who is an accessor from PUD because the attribute authentication is controlled by the third authority party. To resist the collusion, it is essential for PHR owner to specify a blacklist, which contains the access identities that are not allowed access from PUD and delegates the blacklist to a third authority party. The authority maps each blacklist to an attribute, such as attribute "*Alice*$'$*s Blacklist*1," so that an owner can combine such attributes in his access policy in PUD to restrict specific identity from access. Normally, the amount of blacklist attributes will grow linearly with users in PHR system. Fortunately, our proposed ABE construction is efficient in managing attributes because the algorithms replace attribute master keys with the hash values of attributes' descriptive names. The storage for attribute management can keep small at the authority even when the number of attributes increases. It means that the blacklist solution is highly efficient.

Accessor collusion denotes that different accessors will combine their attribute components (pACs) together for decryption of a file despite the fact that they do not have enough attributes to decrypt it alone. Our proposed MA CP-ABE scheme can resist the accessor collusion by embedding the accessor's hash value into their pACs. Consequently, the temporary result in decryption phase, that is, $\hat{e}(g1, g1)\lambda x \cdot \hat{e}(H(\mathrm{GID}), g1)\mu x$, differs among accessors. Therefore, the decryption process is resisted.

Authority collusion is an important security metric in multiauthority scenario. In our proposed scheme, since the authorities do not communicate with each other or have no predefined parameters among them, the authority collusion is impossible in our proposed scheme.

Table 2: Storage overhead on each entity.

	DACC	Yang	Ours
Authority	$2 * n_{\text{att}}$	$n_{\text{att}} + 2 * n_{\text{user}} + 3$	2
Owner	$n_c + 2 * n_{\text{att}} + 2$	$3 * n_{\text{AA}} + 2 * n_{\text{att}} + 3$	$2 * n_{\text{AA}} + 1$
Accessor	$n_{\text{pAC}_s} + n_{\text{att}}$	$2 * n_{\text{AA}} + n_{\text{att}} + 2$	n_{att}
Cloud storage	$(3 * \text{avg} + 1) * n_{\text{cipher}}$	$(4 * \text{avg} + 3) * n_{\text{cipher}}$	$(3 * \text{avg} + 1) * n_{\text{cipher}}$

Table 3: Time consumption of different types of operation.

Type	Description	Time for 1000 operations
$T0$	Time for two-vector multiplication	Depending on the vector length
$T1$	Time for one PBC pairing operation	875443 (us)
$T2$	Time for one PBC exponent operation	1419140 (us)
$T3$	Time for one PBC multiply operation	13264 (us)
$T4$	Time for one PBC addition operation	1196 (us)

Table 4: Computation efficiency.

	Time for encryption	Time for decryption
DACC	$n_{\text{pAC}_s} \cdot (2 \cdot T0 + 5 \cdot T2 + 2 \cdot T3) + (T2 + T3)$	$n^{/} \cdot (2 \cdot T1 + T2 + 3 \cdot T3)\text{pAC}_s$
Yang	$n_{\text{pAC}_s} \cdot (T0 + 5 \cdot T2 + 2 \cdot T3) + (3 \cdot T2 + n_{\text{AA}} \cdot T3)$	$n \cdot (4 \cdot T1 + 2 \cdot T2 + 4 \cdot T3) + n_{\text{AA}} \cdot (2 \cdot T1 + T3) + (T2 + T3)\text{pAC}_s$
Ours	$n_{\text{pAC}_s} \cdot (2 \cdot T0 + T1 + 4 \cdot T2 + 2 \cdot T3) + (T2 + T3)$	$n^{/} \cdot (2 \cdot T1 + T2 + 3 \cdot T3) \cdot n_{\text{pAC}_s}$

5. Performance

In this section, we will compare performances between our proposed scheme and previous MA CP-ABE schemes in aspects of storage cost, computation efficiency, and revocation cost. Since Li's ABE scheme for PUD is actually a variant KP-ABE scheme, we will compare our scheme with both DACC's [19] and Yang's scheme [18].

5.1. Storage. The storage overheads on each entity are listed in Table 2. Notice that n_{user} is the amount of users (accessors) in PHR system, n_{att} denotes the number of all attributes, n_{AA} denotes the number of authorities, n_{cipher} is the number of all ciphertext tuples n_c stored in cloud storage, and n_{pAC_s} denotes the number of generated pAC_s at terminal of accessor. For comparison, the storage overheads of these parameters are n_c, n_{cipher}, n_{user}, and $n_{\text{pAC}_s} > n_{\text{att}} > n_{\text{AA}}$. Specifically, storage overhead at authority (AA) is mainly the space occupation of master keys and public keys for attributes. Since our proposed scheme uses hash values to replace keys for attributes, the storage space at authorities can be saved evidently. We suppose that each ciphertext is associated with avg attributes on average. From Table 2, it is evident that our scheme has the smallest storage overhead at authority, terminal of owner, terminal of accessor, and cloud storage compared with both DACC's and Yang's schemes.

5.2. Computation Efficiency. In this section, we compare the computation costs for these three schemes by implementing them on a Linux system with an Intel Core i7 CPU at 2.20 GHz and 1.00 GB RAM. The codes are constructed based on the Pairing-Based Cryptography (PBC) library version 0.5.14. A symmetric elliptic curve α-curve whose base field size is 512 bits is set up to execute the pairing operation. The group order of α-curve is of 160 bits; that is, p_1 is a 160-bit length prime. All the simulation results come from the average of 20 trials.

Before the simulations, time consumption values of four PBC functional operations are compared which are listed in Table 3. It is obvious that pairing operation and exponent operation consume more time than multiplication and addition. Furthermore, time consumption for encryption and decryption is shown in Table 4 where n' denotes the number of pACs required in each decryption.

We compare the computation efficiencies of both encryption and decryption in two criteria: (1) The number of authorities is changeable while the number of attributes in each authority is fixed. (2) The number of authorities is fixed while the number of attributes in each authority is changeable. The result is shown in Figure 2. In the first simulation, the number of related authorities (x-axis) changes from 2 to 20, and the involved attributes of each authority are set to be 10. Time for encryption is shown in Figure 2(a), while time for decryption is presented in Figure 2(b). The second simulation is the opposite. The number of involved attributes in each authority changes from 2 to 20, and related authorities are set to be 10. Time for encryption and time for decryption are shown in Figures 2(c) and 2(d), respectively. Evidently, our proposed scheme has better performance in computation efficiency because of less number of PBC exponent operations.

5.3. Revocation Cost. As shown in Table 5, we use expressions to denote the communication overheads between terminals and the cloud storage. In DACC, it is the responsibility

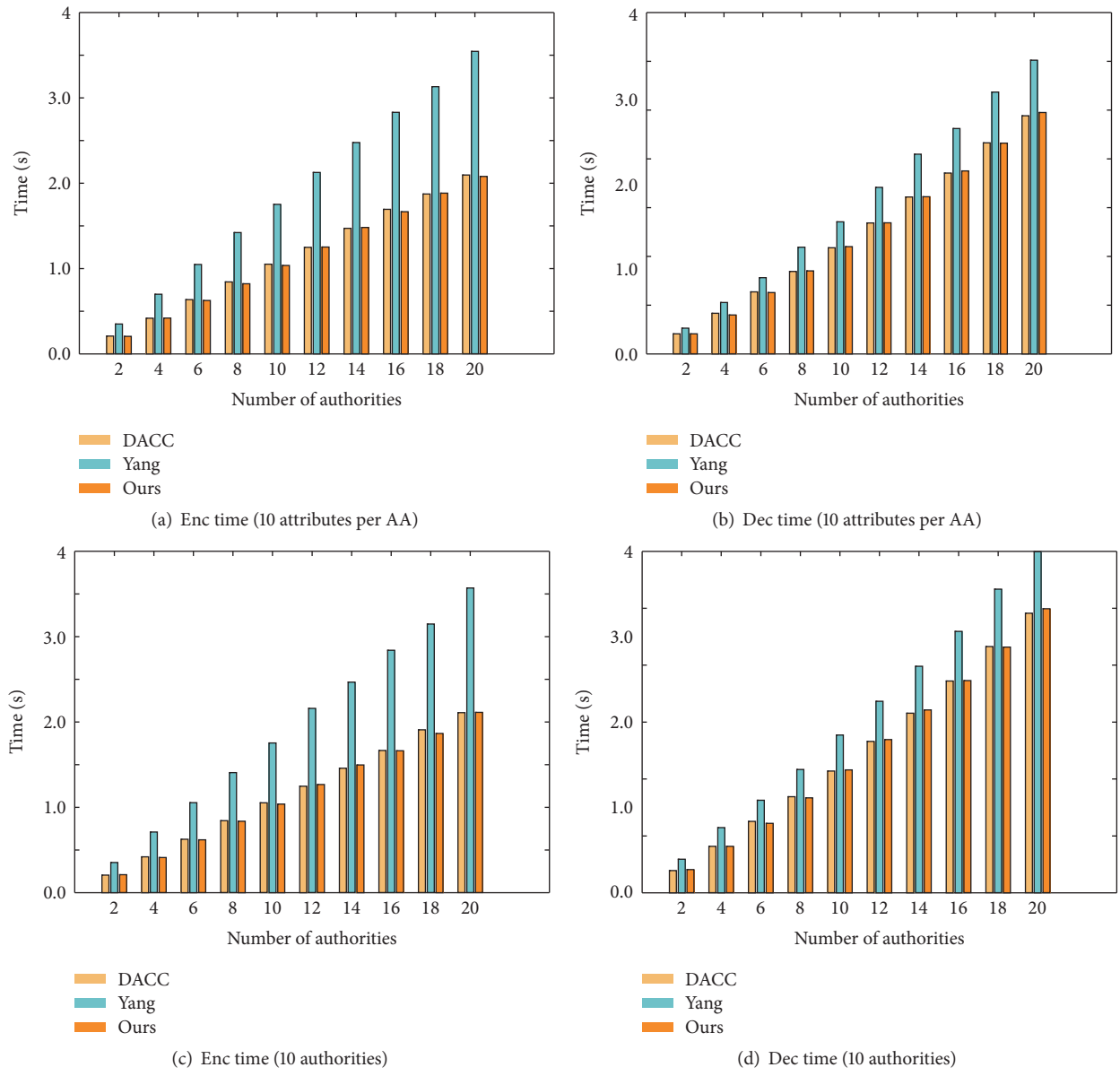

Figure 2: Time for encryption (Enc time) and decryption (Dec time).

Table 5: Communication overhead of attribute revocation.

	DACC	Yang's scheme	Ours (method 1)	Ours (method 2)
Update parameters for accessors	$(n'_{\mathrm{pAC}_s} * n'_{\mathrm{user}} + 1) * \lvert p_1 \rvert$	$n'_{\mathrm{user}} * \lvert p_1 \rvert$	$n'_{\mathrm{user}} * \lvert p_1 \rvert$	$n'_{\mathrm{user}} * \lvert p_1 \rvert$
Update parameters for cloud storage server	$n'_{\mathrm{pAC}_s} * \lvert p_1 \rvert$	$2 * \lvert p_1 \rvert$	$\lvert p_1 \rvert$	$2 * \lvert p_1 \rvert$

Notes. n'_{pAC_s} is the number of ciphertexts which is associated with the revoked attribute i. n'_{user} is the number of unrevoked accessors. $\lvert p_1 \rvert$ is the length of each update parameter.

of data owner to generate update parameters for attribute revocation. In some other schemes, authority generates the update parameters and the data owner can stay offline. It is clear that DACC is inefficient because the data owner should regenerate all the related pACs manually. Both Yang's scheme and our two revocation methods (the same update parameters and different update parameters) use the proxy reencryption technique to reduce communication cost and computation cost.

Time revocation for different number of attributes is shown in Figure 3 where the x-axis denotes number of the revoked attributes and the y-axis is time consumption. For

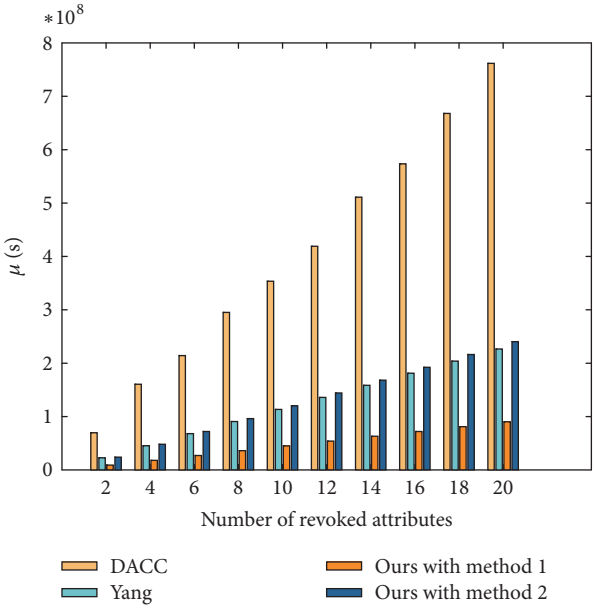

Figure 3: Revocation time with different number of attributes.

simplify, we set the related ciphertext as n tuples and each ciphertext is associated with 10 attributes (so that $n'_{\mathrm{pAC}_s} = 1000 * 10$).

It is inefficient for the data owner to generate update parameters for each attribute associated pAC in DACC, which means the data owner should always keep being online. Our second revocation method (different update parameters) is as efficient as Yang's scheme [18], while our first revocation method (same update parameter) is more efficient because it generates the same update parameters for all accessors. It is noticed that the difference of computation time will be more obvious if n'_{pAC_s} or n'_{user} are getting bigger. From both Table 5 and Figure 3, we can conclude that our scheme has higher efficiency in in communication and computation.

6. Conclusion

In this paper, we proposed a modified MA CP-ABE scheme to implement fine-grained access control. Our proposed scheme supports expressive access policy and can resist user collusion without an authentication center. Moreover, two types of attribute revocation methods, which can revoke attribute efficiently, are proposed. The system can choose one of them according to different application scenarios. Simulations and analysis show that the proposed scheme can achieve less in storage occupation, computation assumption, and revocation cost compared with other schemes.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant 61402291 and the Technology Planning Project from Guangdong Province, China, under Grant no. 2014B010118005.

References

[1] J. Li, "Ensuring privacy in a personal health record system," *Computer*, vol. 48, no. 2, Article ID 7042698, pp. 24–31, 2015.

[2] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 746–759, 2016.

[3] A. Ge, J. Zhang, R. Zhang, C. Ma, and Z. Zhang, "Security analysis of a privacy-preserving decentralized key-policy attribute-based encryption scheme," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2319–2321, 2013.

[4] M. Li, "Fractal time series—a tutorial review," *Mathematical Problems in Engineering*, Article ID 157264, Art. ID 157264, 26 pages, 2010.

[5] M. Li, "Record length requirement of long-range dependent teletraffic," *Physica A. Statistical Mechanics and its Applications*, vol. 472, pp. 164–187, 2017.

[6] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1265–1277, 2016.

[7] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security, (ASIACCS '10)*, pp. 261–270, April 2010.

[8] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in cryptology*, vol. 3494 of *Lecture Notes in Comput. Sci.*, pp. 457–473, Springer, Berlin, 2005.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, November 2006.

[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, May 2007.

[11] M. Li, S. Yu, Y. Zheng, K. Ren, and 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.

[12] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, March 2010.

[13] M. Chase and 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 (CCS '09)*, pp. 121–130, Chicago, Ill, USA, November 2009.

[14] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in cryptology*, vol. 6632 of *Lecture Notes in Comput. Sci.*, pp. 568–588, Springer, Heidelberg, 2011.

[15] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Information Sciences. An International Journal*, vol. 180, no. 13, pp. 2618–2632, 2010.

[16] S. Muller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Information security and cryptology*, vol. 5461 of *Lecture Notes in Comput. Sci.*, pp. 20–36, Springer, Berlin, 2009.

[17] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography*, vol. 4392 of *Lecture Notes in Computer Science*, pp. 515–534, Springer, Berlin, Germany, 2007.

[18] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1735–1744, 2014.

[19] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: distributed access control in clouds," in *Proceedings of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '11)*, pp. 91–98, Changsha, China, November 2011.

[20] L. Li, T. L. Gu, L. Chang, Z. B. Xu, Y. N. Liu, and J. Y. Qian, "A ciphertext-policy attribute-based encryption based on an ordered binary decision diagram," *IEEE Access*, vol. 5, pp. 1137–1145, 2017.

[21] L. Ibraimi, M. Asim, and M. Petković, "Secure management of personal health records by applying attribute-based encryption," in *Proceedings of the 6th International Workshop on Wearable, Micro, and Nano Technologies for Personalized Health*, pp. 71–74, Oslo, Norway, June 2009.

[22] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A Robust and Verifiable Threshold Multi-Authority Access Control System in Public Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, 2016.

[23] X. Wu, R. Jiang, and B. Bhargava, "On the security of data access control for multiauthority cloud storage systems," *IEEE Transactions on Services Computing*, vol. PP, no. 99, 2015.

[24] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of cryptography*, vol. 3378 of *Lecture Notes in Comput. Sci.*, pp. 325–341, Springer, Berlin, 2005.

[25] S. Wang, K. Liang, J. K. Liu, J. Chen, J. Yu, and W. Xie, "Attribute-Based Data Sharing Scheme Revisited in Cloud Computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1661–1673, 2016.