10-2018

A lightweight cloud sharing PHR system with access policy updating

Zuobin YING

Wenjie JANG

Shuanlong CAO

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Jie CUI

Citation

YING, Zuobin; JANG, Wenjie; CAO, Shuanlong; LIU, Ximeng; and CUI, Jie. A lightweight cloud sharing PHR system with access policy updating. (2018). *IEEE Access*. 6, 64611-64621.
Available at: https://ink.library.smu.edu.sg/sis_research/5142

Received September 30, 2018, accepted October 20, 2018, date of publication October 25, 2018,
date of current version November 30, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2877981

A Lightweight Cloud Sharing PHR System With Access Policy Updating

ZUOBIN YING[1], (Member, IEEE), WENJIE JANG[1], SHUANGLONG CAO[1], XIMENG LIU[2,3], (Member, IEEE), AND JIE CUI[1]

[1]School of Computer Science and Technology, Anhui University, Hefei 230601 China
[2]College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China
[3]School of Information Systems, Singapore Management University, Singapore 178902

Corresponding author: Cui Jie (cuijie@mail.ustc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1405255, Grant 61572001, Grant 61502008, Grant 61702005, Grant 61502248, Grant 61427801, and Grant 61702105.

ABSTRACT The rapid development of smart wearable devices makes personal health management feasible, which also stimulates the evolution of personal health records (PHRs). However, PHRs face many security challenges ever since it has been created. Besides, the complicated policy adjusting operation makes the PHRs stored in the cloud not so easy to use. In this paper, we propose a lightweight PHRs system on the basis of attribute-based encryption with policy updating. To update an outsourced ciphertext PHRs in the cloud, PHRs owners only need to generate an updating key, then upload it to the cloud server instead of retrieving the entire ciphertexts. We proved the security of our scheme, and the experiment result indicates that our system is much more efficient than the brute force way of ciphertext updating.

INDEX TERMS Personal health records, device-to-device, ciphertext-policy attribute-based encryption, policy update, smart wearable device, cloud computing.

I. INTRODUCTION

Internet of Things (IoTs) and cloud computing are the main building block of the new generation information technology. They become the next ‘‘major productivity’’ to promote the world’s high-speed development. Meanwhile, the rapid development of device-to-device (D2D) technology makes the communication much more reliable and convenient. D2D communication is a promising technology that permits devices to communicate directly without access points or base station interactions. In the approaching 5G era, D2D communication will also be popularized to alleviate the shortage of spectrum resources in wireless communication systems. D2D has different applications in different networks, such as Peer to Peer (P2P) in Ad hoc network and Machine to Machine (M2M) in the IoTs. There is no essential difference between these different jargons, but they are adapted to meet their specific needs in their respective application scenarios. One of the most typical applications is Smart Wearable Devices (SWDs), which are integrated with physiological sensors, low-power computing, communication and storage modules as a bridge connecting the human body and the information world. These devices can sense different information from humans, such as physiological parameters, health status, movement and location. Therefore, through D2D technology, we can transmit data to the smart phones and other mobile devices around us at any time, then the smart phones will process and upload the data to the cloud. In addition, we can share real-time data on our SWDs with friends through social network, such as travels or health-related information etc. However, some security issues may happen when we transmit data to smart phones or share data to peripheral devices. For example, malicious nodes may illegally intercept and discard packets to be forwarded. In addition, malicious nodes can obtain some private data by distributing forged public keys. Thus, privacy became one of the main restrictions that encumbers the development of D2D, especially in some privacy high demand scenarios, health care, for example.

With the increasing concern about the health, health related research is becoming more and more crucial. PHRs are directly formed by people in health-related activities and have the value of preservation and reference characterized with security and confidentiality. Personal health information in PHRs includes basic information, summaries of major diseases and health problems, records of major health services. The information recorded in PHRs mainly comes from medical service records, health examination, disease investigation and the SWDs. The standardization and digitization of various records have enabled information sharing among medical institutions, patients and health departments. After the PHRs system is completely established, the management of people's health information will be more simple, faster and safer.

The following five points need to be considered in order to implement the PHRs system:

(1) In theory, the PHRs system should be able to cover all residents, which makes the PHRs system have a great demand for physical storage resources.

(2) The requirements of PHRs system: one side of the input while multi-use, which means that the PHRs system needs to have an appropriate way of sharing.

(3) Because the PHRs are the residents' personal health statistics, the PHRs system has a higher requirement to protect users' privacy.

(4) According to the privacy required by the third point, users should have the right to manage their own health records (which means that users can determine the access rights of health records), so the PHRs system should have the function of dynamic updating access rights.

(5) PHRs system should have the function of statistics users' real-time health data through SWDs.

Cloud computing can meet the requirements of physical storage system and data sharing of PHRs mentioned in the first and second points above. However, cloud is generally considered to be untrustworthy because it will sniffer as much personal information as possible, which is particularly sensitive in PHRs. At the same time, these data are also of great commercial value, thus become a target of the attackers. This makes it necessary to solve this user privacy preserving problem in order to construct a fully-functional PHRs system.

According to the characteristics of PHRs system and cloud environment, the encryption scheme should have good security, fine-grained access control function, and can be applied to cloud computing environment. Therefore, attribute based encryption scheme (ABE) [5] is our preferred option.

It should be noted that the CP-ABE scheme itself does not provide efficient policy updating methods. In the past, when the access policy was updated in CP-ABE, the following steps were normally completed:

(1) Generate new access policy.

(2) Generate new ciphertext by using new access policy.

(3) Upload the new ciphertext to the cloud server to replace the old ciphertext.

From the above three steps we can see that to update access policy needs to regenerate the entire ciphertext, which consumes a lot of computing resources and makes the update very inefficient. In fact, policy updating in CP-ABE do not require regenerating the entire ciphertext, but only some parts of the ciphertext. We can briefly describe this process as follows:

(1) Generate new access policy.

(2) Compare the difference between the new access strategy and the old access strategy.

(3) Generate part of new ciphertext based on the results of the above step comparison.

(4) Upload the results of step (3) to the cloud server and replace part of the values in the old ciphertext.

Comparing the above two processes, we can find that the complete ciphertext is regenerated in the first process, and some ciphertext is regenerated in the second process. When the user makes subtle adjustments to the access policy, the first process consumes a large amount of computing resources, while the second process consumes only a small amount of computing resources compared to the first process. When the user makes considerable changes to the access policy, the first process still regenerates the entire ciphertext, while the second process generates only a portion of the ciphertext. This shows that the second process always generates only a portion of the ciphertext, and the first process always generates the entire ciphertext without the user completely changing the access policy. When the access policy is completely changed, the two processes consume a lot of computing resources.

However, CP-ABE itself cannot encrypt complex data, so the application of CP-ABE in PHRs system should adopt mixed encryption, that is, symmetric encryption and asymmetric encryption. Generally speaking, in the case of encrypting complex data, the efficiency of symmetric encryption is better than asymmetric encryption. Therefore, in our scheme, CP-ABE is used to encrypt the symmetric encryption key, and the symmetric key is used to encrypt the PHRs data.

Besides, SWDs only have some simple, data-gathering capabilities, and statistics of these data will be placed on the user's mobile device or on the cloud, but as we have discussed above, cloud is considered not entirely credible, so we think statistics of collected data should be computed on personal mobile devices. In order to make the data collected by SWDs can be timely transmitted to mobile devices, we believe that D2D communication technology will be a good choice. As shown in Fig. 1, it is a structural diagram of a PHRs system.

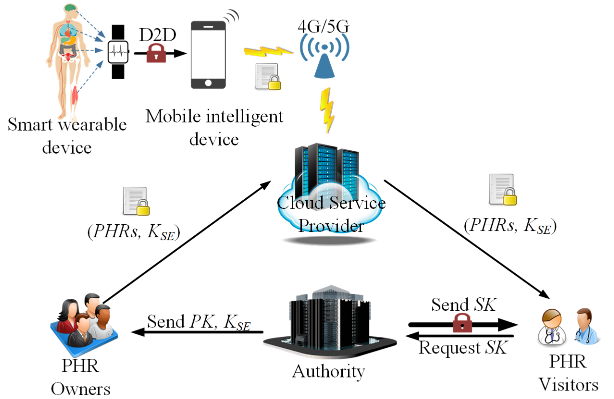


FIGURE 1. A Personal Health Records (PHRs) system.

The main contribution of our work can be summarized as follows:

(1) We construct a concrete PHRs privacy preserving system which can either protect the individuals health information at the primary phase of data collection from SWDs, or preserve the privacy at the cloud sharing phase.

(2) A practical and lightweight policy updating CP-ABE algorithm is designed to reduce the computation as well as the communication cost of the PHRs owners, especially in the energy constraint scenarios. The simulation indicates that our proposed scheme only introduces a small overhead compared with the scheme without health information protection.

(3) The scheme is proposed to be adaptively secure under random oracle model. We also proved that the policy updating procedure will not reveal further helpful information to the adversary.

The rest of this paper is organized as follows. In Section II, we introduce the background of some technologies in PHRs systems, these include PHRs, CP-ABE, SWDs, and D2D communication related background knowledge. In Section III, we made some necessary preliminaries for our policy updating plan of CP-ABE. In Section IV, we give the definition, system model and security model of CP-ABE scheme with policy updating function. In Section V, the CP-ABE scheme with policy updating function is described in detail. In Section VI, we carried out security analysis and performance analysis. Finally, Section VII gives the conclusion.

II. RELATED WORK

- **Device-To-Device (D2D):** Initially, device-to-device (D2D) communication was defined as direct data communication between two devices without access point (AP) or intermediate base station (BS). The most typical D2D applications are Bluetooth [7], WiFi-Direct [8] and Near Field Communications (NFC) [9]. D2D communication is considered to be the key technology to realize the next generation mobile communication network and wireless system (5G). To solve the problem of limited computing power in 5G mobile networks, [12] proposes a Knowledge-Centric Edge (KCE) architecture, which can achieve reliable network access and maximize the utilization of network resources. Besides communication efficiency, security issues in D2D have attract more and more attention. Wu *et al.* [13] propose a new Resisting On-Off Attack Data Forwarding Mechanism (OADM) to resist switching attacks for mobile agents. Mobile Social Networks (MSNs) detects switching attacks, which not only prevents malicious nodes from intercepting packets, but also makes use of nodes to cooperatively forward packets. In order to solve the problem of using forged public keys to obtain privacy data. We *et al.* [11] proposes a dynamic trust relationships aware data privacy protection (DTRPP) mechanism.

- **Personal Health Records (PHRs):** Information and communication technology (ICT) has changed the health care world around the world. One of the main drivers of this change is the electronic health records (EHRs). However, there are still some open problems and challenges because EHRs usually reflects a one-sided view of a healthcare provider without the ability of patients to control or interact with their data. In addition, with the development of mobile computing and pervasive computing, the number of PHRs has increased exponentially. This movement is described as the IoTs, which includes the extensive development of wearable computing technology and a variety of health-related sensors. This leads to the need for a comprehensive approach to store health-related data, defined as PHRs, which can be used by health care providers and patients. This method can combine EHRs with data collected from sensors or other SWDs. This unified view of patient health can be shared with providers, who can use not only previous health-related records, but also data generated through interactions to expand them. Another advantage of PHRs is that patients can interact with their health data to make decisions that may have a positive impact on their health [15]. Among them, [14] reviewed the changing clinical challenges posed by the implementation of PHRs, which are fully integrated with electronic medical records (EMRs).

- **Attribute Based Encryption (ABE):** Attribute-Based Encryption is a one-to-many cryptography prototype. With the purpose of achieving fine-grained access control, access policy is introduced. Data owners set the appropriate access policies, then attached them to either the ciphertext side to form Ciphertext Policy-ABE (CP-ABE) [16] or the attribute key side to form Key Policy-ABE (KP-ABE) [17]. However, only a few of the state-of-art ABE schemes take *arbitrary form* of policy updating into account. Some of the policy adjusting works only support policy retrench, which basis on the theory of credential authorization [18]. Yang *et al.* first propose a CP-ABE scheme which can realize any kinds of policy updating besides policy retrench [19]. Data owner only have to generate update keys then upload them to the cloud without retrieving the entire ciphertext, after receiving the updating key element, the cloud server will update the corresponding ciphertext components. The scheme is constructed on the generic order group and random oracle model. In order to realize higher security level. Ying *et al.* [20] proposed another policy updating scheme, which is proved to be adaptively secure under standard model. But their scheme is constructed on composite order group, so it is not practical to be used. Yuan and Wei [21] put forward an policy updating algorithm based on matrix update. Moving one step forward, we proposed a more efficient policy updating mechanism which could face the energy constraint as well as personal health information preserving needs of PHRs with SWDs scenario.

III. PRELIMINARIES

First, the formal definition and related background of access structure in Linear Secret Sharing Scheme (*LSSS*) are given, then the structure of CP-ABE and its security definition are given. Finally, we give the background information of bilinear mapping.

A. ACCESS STRUCTURES

Access Structure [10]: Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\cdots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure(respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\cdots,P_n\}}\backslash\{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

In our context, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. We restrict our attention to monotone access structures. However, it is also possible to (inefficiently) realize general access structures using our techniques by having the not of an attribute as a separate attribute altogether. Thus, the number of attributes in the system will be doubled. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

B. LINEAR SECRET SHARING SCHEMES

We will make essential use of linear secret-sharing schemes. We adapt our definitions from those given in [10]:

A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$.

2. There exists a matrix an M with ℓ rows and n columns called the share-generating matrix for Π. For all $i = 1, \cdots, \ell$, the i'th row of M we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $\upsilon = (s, r_2, \cdots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \cdots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $M\upsilon$ is the vector of ℓ shares of the secret s according to Π. The share $(M\upsilon)_i$ belongs to party $\rho(i)$.

It is shown in [10] that every linear secret sharing-scheme according to the above definition also enjoys the linear reconstruction property, defined as follows: Suppose that Π is an *LSSS* for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \in \{1, 2, \cdots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\prod_{i \in I} \omega_i \lambda_i = s$. Furthermore, it is shown in [10] that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

We note that we use the convention that vector $(1, 0, 0, \cdots, 0)$ is the target vector for any linear secret sharing scheme. For any satisfying set of rows I in M, we will have that the target vector is in the span of I. For any unauthorized set of rows I the target vector is not in the span of the rows of the set I. Moreover, there will exist a vector ω such that $\omega \cdot (1, 0, 0, \cdots, 0) = -1$ and $\omega \cdot M_i = 0$ for all $i \in I$.

Prior works on ABE (e.g., [11]) typically described access formulas in terms of binary trees. Using standard techniques [10] one can convert any monotonic boolean formula into an *LSSS* representation. An access tree of ℓ nodes will result in an *LSSS* matrix of ℓ rows.

C. CP-ABE SCHEME

A CP-ABE scheme consists of four algorithms: Setup, Encrypt, KeyGen, Decrypt.

Setup (λ, U) The setup algorithm takes security parameter and attribute universe description as input. It outputs the public parameters PK and a master key MSK.

Key Generation (MSK, S) The key generation algorithm takes as input the master key MSK and a set of attributes S that describe the key. It outputs a private key SK.

Encrypt $(PK, msg, \mathbb{A})$ The encryption algorithm takes as input the public parameters PK, msg, and an access structure $\mathbb{A}$ over the universe of attributes. The algorithm will encrypt msg and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains $\mathbb{A}$.

Decrypt (PK, CT, SK) The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy $\mathbb{A}$, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathbb{A}$ then the algorithm will decrypt the ciphertext and return the msg.

D. BILINEAR MAPS

We present a few facts related to groups with efficiently computable bilinear maps and then give our number theoretic assumptions.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bilinear map e has the following properties:

(1) Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.

(2) Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

IV. DEFINITIONS

In this section, we give a brief introduction of the policy update system model. Then we define the proposed scheme as well as the security model. In order to make our proposed scheme easier to read. We notate the symbols and the corresponding explainations in Table 1.

A. SYSTEM MODEL

The policy update scheme we provide consists of the following parts: **Setup**, **KeyGen**, **Encrypt**, **Decrypt**, **Update**. Besides in our scheme, **Encrypt** consists of **SE-Enc**, **KEY-Enc**. Correspondingly, the **Decrypt** also consists of two sub-algorithms, **KEY-Dec** and **SE-Dec**. Specially, in our scheme **SE** represents a symmetric encryption scheme which consists of two functions, **SE.Enc** and **SE.Dec**.

Setup$(U) \rightarrow (PK, MSK, M_{key})$: The setup algorithm takes as input the number of attributes in the system. It outputs the public key PK, the master secret key MSK and M_{key} which is selected randomly from $\mathbb{G}_T$.

KeyGen $(MSK, S) \rightarrow SK$: The key generation algorithm takes as input the master secret key and a set S of attributes. It outputs the corresponding secret key SK.

Encrypt$(PK, M_{key}, (\mathbb{M}, \rho), msg) \rightarrow (CT_{msg}, CT_{key})$: The encryption algorithm consists of two sub-algorithm, namely, **KEY-Enc** and **SE-Enc**.

- **SE-Enc**$(PK, M_{key}, msg) \rightarrow CT_{msg}$: The SE-Enc sub-algorithm takes as input the M_{key}, PK and compute $K_{se} \leftarrow \mathcal{H}(M_{key})$ which $\mathcal{H}(x)$ is from PK, then it takes K_{se} and msg to encrypt. It output the ciphertext CT_{msg}.
- **KEY-Enc**$(PK, (\mathbb{M}, \rho), M_{key}) \rightarrow CT_{key}$: The KEY-Enc sub-algorithm takes as input the public key PK and M_{key}. In addition, it takes as input an $LSSS$ access structure $(\mathbb{M}, \rho)$. The function ρ associates rows of M to attributes. It takes the ciphertext CT_{key} as an output.

Decrypt$((CT_{msg}, CT_{key}), SK) \rightarrow msg$: Corresponding to encryption algorithm, the decryption algorithm also consists of two sub-algorithm, namely, **KEY-Dec** and **SE-Dec**.

- **KEY-Dec**$(CT_{key}, SK) \rightarrow M_{key}$: The KEY-Dec sub-algorithm takes as input CT_{key} and a private key SK for a set S. It outputs M_{key} if the attributes satisfy the policy, or$\perp$if not.
- **SE-Dec**$(CT_{msg}, M_{key}) \rightarrow msg$: The SE-Dec sub-algorithm takes as input CT_{msg} and the output of KEY-Dec sub-algorithm, and then it computes $K_{se} \leftarrow \mathcal{H}(M_{key})$ for decrypting. The output of SE-Dec sub-algorithm is msg.

Update$(PK, \mathbb{A}, \mathbb{A}') \rightarrow CT'_{key}$: The Update algorithm consists of two parts, namely, **LOCAL-Update** and **CLOUD-Update**.

- **LOCAL-Update**$(PK, \mathbb{A}, \mathbb{A}') \rightarrow CU$: The LOCAL-Update algorithm takes as input system public key PK, the old access structure $\mathbb{A}$ and the new access structure $\mathbb{A}'$ as input. Output ciphertext update component CU.
- **CLOUD-Update**$(CU, CT_{key}) \rightarrow CT'_{key}$: The CLOUD-Update algorithm takes as input the update component CU and the ciphertext CT_{key}, Output the new ciphertext CT'_{key} which has been updated and this part will be executed in cloud.

B. SECURITY MODEL

We now describe a security model for our lightweight and efficient policy update CP-ABE schemes. Like CP-ABE and IBE schemes [6], [12] the security model allows the adversary to query for any private keys that cannot be used to decrypt the challenge ciphertext. In our scheme the ciphertexts are identified with access structures and the private keys with attributes. It follows that in our security definition the adversary will choose to be challenged on an encryption to an access structure $\mathbb{A}^*$ and can ask for any private key S such that S does not satisfy $\mathbb{A}^*$. We now give the formal security game Security Model for our policy update CP-ABE.

Setup. The challenger runs the Setup algorithm and gives the public parameters, PK to the adversary.

Phase 1. The adversary makes repeated private keys corresponding to sets of attributes $S_1, \cdots, S_{q1}$.

TABLE 1. Notations.

Symbols	Descriptions
$PK,\ SK$	public/private key pairs
MSK	system master secret key
$(\mathbb{M},\rho),\ \mathbb{A},\ \mathbb{A}'$	access policy in policy-update CP-ABE
M_{key}	M_{key} is selected randomly from $\mathbb{G}_T$ for generating the K_{se}
$M_{phr},\ msg$	the plaintext
K_{se}	symmetric key which is generated by M_{key} through $H(x)$ function
$\mathcal{H}(x)$	hash function which is used for generating the symmetric key K_{se}
$h_1,\cdots,h_n$	the elements which are from the group $\mathbb{G}$ for mapping the attributes
CT_{key},CT_{phr}	the ciphertext
CU	components for updating the ciphertext
$U,\ S$	a set of attribues

Challenge. The adversary submits two equal length messages M_0 and M_1. In addition the adversary gives a challenge access structure $\mathbb{A}^*$ such that none of the sets $S_1, \cdots, S_{q1}$ from Phase 1 satisfy the access structure. The challenger flips a random coin b, and encrypts M_b under $\mathbb{A}^*$. The ciphertext $\mathbb{CT}^*$ is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that none of sets of attributes $S_{q1+1}, \cdots, S_q$ satisfy the access structure corresponding to the challenge.

Guess. The adversary outputs a guess b' of b.

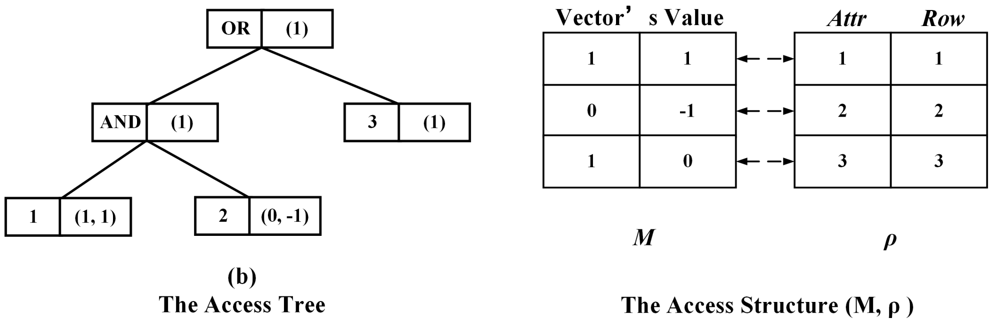
The advantage of an adversary in this game is defined as $Pr[b' = b] - 1/2$. We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

V. CONSTRUCTIONS OF OUR SCHEME

A. MAIN IDEA

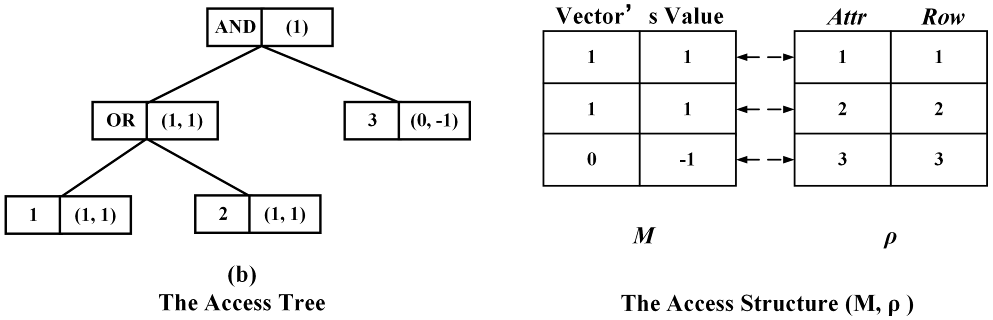
Changes in access policies in CP-ABE will directly lead to changes in access structure. The access structure $(\mathbb{M}, \rho)$ consists of two parts: access matrix $\mathbb{M}$ and mapping function ρ. The mapping function ρ is used to preserve the correspondence between the attribute associated with a row in the access matrix $\mathbb{M}$ and the row number of the attribute in the access matrix $\mathbb{M}$, so the mapping function ρ changes with the order of the rows in the access matrix $\mathbb{M}$ (note here that the order of the rows changes, not the value of the rows). So for the change of access structure, we mainly focus on the change of access matrix $\mathbb{M}$. On the other hand, when the values of some rows in the access matrix $\mathbb{M}$ change, according to the principle of the linear secret sharing scheme ($LSSS$), we find that some ciphertext components in the ciphertext will change directly. In summary, we can find that when the access policy changes, it will directly lead to the change of access matrix $\mathbb{M}$, and the change of access matrix will indirectly lead to the change of mapping function ρ and ciphertext components. However, the changes in ciphertext components are more noticeable than changes in access matrices $\mathbb{M}$ and mapping functions ρ, because the computational overhead of generating ciphertext components is extraordinarily large compared to other operations.

In CP-ABE, the access structure $(\mathbb{M}, \rho)$ is generated with the input of the access policy according to the principle of



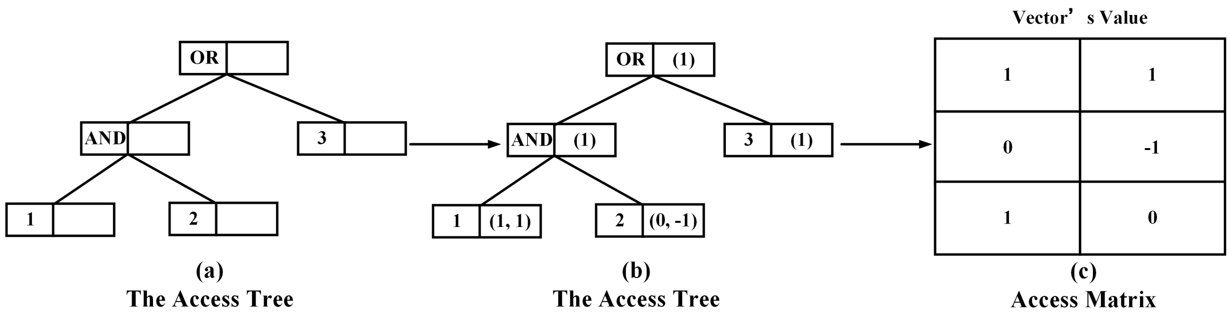
Vector's Value			Attr	Row
1	1	←→	1	1
0	-1	←→	2	2
1	0	←→	3	3

FIGURE 2. Access Policy: (1 AND 2) OR 3.



Vector's Value			Attr	Row
1	1	←→	1	1
1	1	←→	2	2
0	-1	←→	3	3

FIGURE 3. Access Policy: (1 OR 2) AND 3.



Vector's Value	
1	1
0	-1
1	0

FIGURE 4. Access Policy: (1 AND 2) OR 3.

the *LSSS*. The access structure is shown in Fig. 2 and Fig. 3.

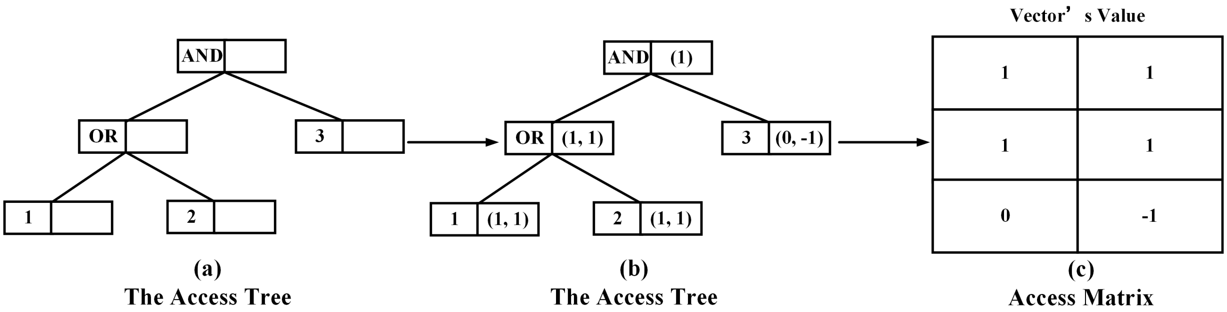
As can be seen from the graph, the generation of access matrix $\mathbb{M}$ is transformed from the access policy binary tree, and the access policy binary tree is transformed from the access policy, the specific process is as follows:

(1) The structure of the node in the binary tree is composed of two parts, namely *type* and *value*, in which *type* is a string and *value* is a multi-dimensional vector; in the binary tree, there are two kinds of nodes, namely, intermediate node and attribute node, we can see thin in Fig. 4(a) and Fig. 5(a).

(2) First, the access strategy is transformed into a binary tree.

(3) Assign value to the root node of the binary tree. The root node is assigned to vector $\vec{1}$ by default. The assignment of a non-root node depends on its parent node: assuming that the parent node's vector value is $\vec{V}$, when the parent node's *type* is 'OR', the vector *value* of its two children's nodes is $\vec{V}$; when the parent node's *type* is 'AND', the vector *value* of its left child's node is equal to $(1, \vec{V})$, and the vector *value* of its right child's node is equal to $(0, \cdots, \underset{\to}{0}, -1)$ the number of '0' is equal to the length of the vector $\vec{V}$, and we can see this in Fig. 4(b), Fig. 5(b).

(4) To store the values of all the attribute nodes in the access policy tree into the matrix $\mathbb{M}$, it is necessary to note that when the dimension of the vector *values* of the attribute nodes is different, the dimension of all the vectors is enlarged based on the vector with the largest dimension, and the rule is to supplement '0' at the end, for example, the vector $\vec{V}$



(a)
The Access Tree

(b)
The Access Tree

(c)
Access Matrix

FIGURE 5. Access Policy: (1 OR 2) AND 3.

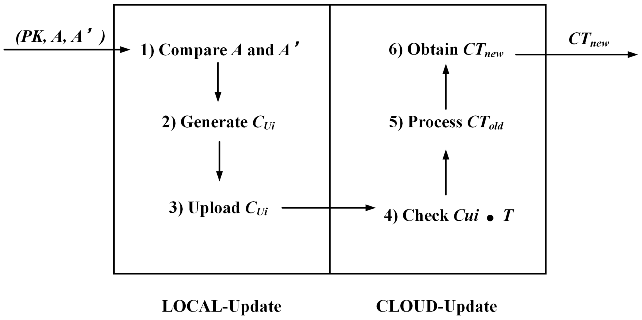


FIGURE 6. A policy update process ($A = \mathbb{A}$, $A' = \mathbb{A}'$).

is supplemented by '0' to obtain ($\vec{V}$, 0),we can see this in Fig. 4(c) and Fig. 5(c).

(5) Save the row number of the generated access matrix and the *type* of its corresponding attribute node into the mapping function.

When the access structure is generated, the ciphertext component needs to be generated according to the access matrix and *LSSS*,the concrete process is as follows:

(1) Vector $\vec{v} = (s, y_2, y_3, \cdots, y_n)$ is a vector related to the linear secret sharing scheme, whose dimension is equal to the number of columns of the access matrix $\mathbb{M}$, and s which is in the $\vec{v}$ is the secret who is going to be shared.

(2) First, we compute

$$\lambda_i = \overrightarrow{\mathbb{M}}_i \cdot \vec{v}$$

$\overrightarrow{\mathbb{M}}_i$ is the vector value of the ith row of the access matrix $\mathbb{M}$. Then, we compute

$$C_i = g^{a\lambda_i} h_{\rho(i)}^{-r_i}, D_i = g^{r_i}, r_i \in \mathbb{Z}_p$$

Remark: we can find that the influence factors of ciphertext components are

$$\textit{policy-string} \rightarrow (\mathbb{M}, \rho) \rightarrow \lambda_i \rightarrow (C_i, D_i)$$

and comparing Fig. 4(c) and Fig. 5(c), we can see that the vector values in the access matrix $\mathbb{M}$ do not change entirely when the policy changes, for example, the vector corresponding to the attribute '1' is still (1, 1) after the change of access policy. This means that λ_i does not change entirely. Similarly, this means that ciphertext components do not change entirely.

Therefore, we can conclude that when the access policy changes, there is no need to change the entire ciphertext component, but only a part of it.

B. SCHEME CONSTRATION

(1) Setup(U) The setup algorithm takes as input the number of attributes in the system. It then chooses a group $\mathbb{G}$ of prime order p, a generator g and U random group elements $h_1, \cdots, h_U \in \mathbb{G}$ that are associated with the U attributes in the system. In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$ and an anti-collision hash function $\mathcal{H}(x)$.

The public key is published as

$$PK = g, e(g,g)^{\alpha}, g^a, h_1, ..., h_U, \mathcal{H}(x)$$

The authority sets $MSK = g^{\alpha}$ as the master secret key. A randomly selected element $M_{key} \in \mathbb{G}_T$, which $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

(2) KeyGen (MSK, S) The key generation algorithm takes as input the master secret key and a set S of attributes. The algorithm first chooses a random $t \in \mathbb{Z}_p$. It creates the private key as

$$K = g^{\alpha} g^{\alpha t}, L = g^t, \forall x \in S, K_x = h_t^x$$

(3) Encrypt($PK, M_{key}, (\mathbb{M}, \rho), M_{phr}$) The encryption algorithm consists of two sub-algorithm, namely, **KEY-Enc** and **SE-Enc**.

- **SE-Enc**(M_{key}, M_{phr}) This sub-algorithm takes as input the M_{key} and M_{phr}, then computes:

$$K_{se} \leftarrow \mathcal{H}(M_{key})$$

the ciphertext CT_{phr} is published as

$$CT_{phr} = \text{SE.Enc}(K_{se}, M_{phr})$$

- **KEY-Enc**($PK, (\mathbb{M}, \rho), M_{key}$) This sub-algorithm takes as input the public parameters PK and M_{key} to encrypt. In addition, it takes as input an *LSSS* access structure $(\mathbb{M}, \rho)$. The function ρ associates rows of $\mathbb{M}$ to attributes.

Let $\mathbb{M}$ be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_n^p$. These values will be used to share the secret s. For $i = 1$ to ℓ, it calculates $\lambda_i = \vec{v} \cdot \mathbb{M}_i$, where $\mathbb{M}_i$ is the vector corresponding to the ith row of $\mathbb{M}$. In addition, the algorithm chooses random $r_1, \cdots, r_\ell \in \mathbb{Z}_p$.

The ciphertext is published as $CT_{key} =$

$$C_{key} = M_{key} \cdot e(g, g)^{\alpha s}, C' = g^s,$$
$$(C_1 = g^{a\lambda_1} h_{\rho(1)}^{-r_1}, D_1 = g^{r_1}), \cdots, (C_\ell = g^{a\lambda_\ell} h_{\rho(\ell)}^{-r_\ell}, D_\ell = g^{r_\ell})$$

along with a description of $(\mathbb{M}, \rho)$.

4) Decrypt(CT_{key}, CT_{phr}, SK) The decryption algorithm also consists of two sub-algorithm, namely, **KEY-Dec** and **SE-Dec**.

KEY-Dec(CT_{key}, SK) This sub-algorithm takes as input a ciphertext CT_{key} for access structure $(\mathbb{M}, \rho)$ and a private key for a set S. Suppose that S satisfies the access structure and let $I \subset 1, 2, \cdots, \ell$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to $\mathbb{M}$, then $\prod_{i \in I} \omega_i \lambda_i = s$. (Note there could potentially be different ways of choosing the ω_i values to satisfy this).

The decryption algorithm first computes

$$e(C', K)/(\prod_{i \in I}(e(C_i, L)e(D_i, K_{\rho)(i)})^{\omega_i}) =$$
$$e(g, g)^{\alpha s} e(g, g)^{ast}/(\prod_{i \in I} e(g, g)^{ta\lambda_i \omega_i}) = e(g, g)^{\alpha s}$$

The decryption algorithm can then divide out this value from C_{key} and obtain the M_{key}.

SE-Dec(CT_{phr}, M_{key}) This sub-algorithm takes as input the ciphertext CT_{phr} and the output of **KEY-Dec** sub-algorithm, it first computes:

$$K_{se} \leftarrow \mathcal{H}(M_{key})$$

the **SE-Dec** sub-algorithm then computes:

$$M_{phr} = SE.DEC(CT_{phr}, K_{se})$$

(5) Policy-Update$(PK, \mathbb{A}, \mathbb{A}')$ The Update algorithm consists of two parts, namely, Local-Update and Cloud-Update. The Local-Update algorithm takes as input system public key PK, the old access structure $\mathbb{A}$ and the new access structure $\mathbb{A}'$ as input,output ciphertext update component CU, The Cloud-Update algorithm takes as input the update component CU, output the new ciphertext which has been updated.

- **Local-Update**$(PK, \mathbb{A}, \mathbb{A}')$ The access structure consists of $(\mathbb{M}, \rho)$, where $\mathbb{M}$ is a matrix composed of vectors $\mathbb{M}_i$ to share secret s, and ρ is used to represent the correspondence between rows and attributes in the matrix $\mathbb{M}$.

When we want to update the policy, we first compare the matrix $\mathbb{M}'$ in the newly generated access structure $\mathbb{A}'$ with the access matrix $\mathbb{M}$ in the old access structure $\mathbb{A}$, and get an update vector set $\mathbb{M}_U$. $\mathbb{M}_U$ contains all the changed row vectors $\mathbb{M}_{U_i}$ relative to the original access matrix $\mathbb{M}$. Then we calculate $\lambda_{U_i} = \mathbb{M}_{U_i} \cdot (s, y_2, \cdots, y_n)$ according to the method of linear secret sharing.

The I' is an updated attribute set which $I' = \{U_1, U_2, \cdots, U_n\}$, then we calculate $CU_i, i \in I'$ by encryption algorithm.

Update component is published as $CU_i =$

$$(T_{U_i}, C_{U_i} = g^{a\lambda_{U_i}} h_{\rho(U_i)}^{-r_{U_i}}, D_{U_i} = g^{r_{U_i}})$$

T_{U_i} indicates what operations should be performed when the cloud server receives the updated component CU_i. T_{U_i} refers to three operations, adding ciphertext components, deleting the original ciphertext component and modifying the original ciphertext component.

- **Cloud-Update**(CU, CT_{key})This part is executed in the cloud. The input of the algorithm is the original ciphertext CT_{key} and the update component CU, the output is the new ciphertext CT'_{key} corresponding to the new access policy $\mathbb{A}'$.

When the algorithm is executed, the operations (modification, insertion, deletion) for the ciphertext component corresponding to attribute $i(i = U_i)$ are determined according to the T_{U_i} in the update component. When the T_{U_i} representation is modification, the ciphertext components C_i and D_i corresponding to the attribute $i(i = U_i)$ in the original ciphertext CT are found and replaced with C_{U_i} and D_{U_i}; if the T_{U_i} representation is insertion, C_{U_i} and D_{U_i} are inserted into the original ciphertext CT; if the T_{U_i} representation is deletion, the ciphertext components C_i and D_i corresponding to the original ciphertext CT are found and deleted according to the attribute $i(i = U_i)$. In general, the updating process can be described as Fig. 6.

VI. SECURITY AND PERFORMANCE ANALYSIS

A. SECURITY ANALYSIS

Our lightweight policy update scheme is constructed on prime order groups, which is much faster than those ones on composite order groups. In this part, we will prove that our lightweight policy update scheme is secure in the generic bilinear group model [3], [23], [24] and random oracle model [25].

Theorem 3. *Our scheme is secure in the generic bilinear group model and random oracle model, if no polynomial time adversary can get non-negligible advantage in the security game defined in Section IV-B.*

Proof: In this paper, an access control scheme is constructed based on the CP-ABE method in reference [22]. It is proved that the scheme is secure under the generalized bilinear group model and the random oracle model. This means that if there is any vulnerability in our scheme, these vulnerabilities must take advantage of the special mathematical properties of the elliptic curve group. Or the encryption hash function used in the actual scenario. To make a adversary $\mathcal{A}$ an advantage that we can't ignore. We will construct a $\mathcal{A}'$, which will break the [22] scheme and have an advantage that can not be ignored.

In our security game, the adversary returns access structure $\mathbb{A}^*$ which is $(\mathbb{M}^*, \rho^*)$ with two message (m_0, m_1), and the adversary receives the ciphertext CT_b which is obtained by plaintext M_b through the access structure $(\mathbb{M}^*, \rho^*)$

TABLE 2. Comparison of encryption and decryption time of AES 128.

File Size(MB)	10	20	30	40	50	60	70	80	90	100
Encryption and Decryption Procedure (*Second*)										
Encryption	0.030	0.114	0.175	0.359	0.523	0.661	0.840	0.933	1.116	1.260
Decryption	0.033	0.116	0.256	0.413	0.570	0.702	0.856	0.989	1.197	1.267

encryption. The adversary $\mathcal{A}'$ initialize CP-ABE security game, and forward the public key PK to $\mathcal{A}$.

To simulate the update ciphertext component generation of $\mathcal{A}$, $\mathcal{A}'$ queries its update ciphertext component generation. The simulation of challenge ciphertext of $\mathcal{A}$ is the same with the one of $\mathcal{A}'$.

Now, we prove that the update ciphertext component query in our security game will not increase the advantage of $\mathcal{A}'$. Considering two update ciphertext component queries $UC(m_0, (\mathbb{M}^*, \rho^*))$ and $UC(m_1, (\mathbb{M}^*, \rho^*))$, the update ciphertext component generation returns the same update ciphertext component which do not involve with the challenge data. If we take into account the random numbers used in encrypting m_0 and m_1, we can assume that the random numbers that we have used are the same, because the simulator tosses a coin in a secure game and selects only one challenge message. Therefore, the update ciphertext component will not reveal any information on the chosen challenging message. This completes the proof.

B. PERFORMANCE ANALYSIS

We proposed two update sub-algorithms on the basis of CP-ABE, which are LOCAL-Update and CLOUD-Update. We deploy the experiment environment on the Ubuntu Linux Desktop 64 bit-system with an Intel Core i7 CPU at 3.4GHz and 8.00GB RAM. The code utilizes the charm library version 0.50 and an asymmetric elliptic curve α-curve, where the base field size is 512-bit and the embedding degree is 2, so that the security parameter is 1024-bit. In order to generate symmetric keys from $\mathbb{G}_T$ group, we employ the MurmurHash3. All the experimental result are the mean of 100 trails.

It can be seen from the Fig. 7(a) that with the increase of the number of attributes, the time of Kegen, encrypy, decrypt, and update algorithms all show an increasing trend. The update time is obviously much less than other time overhead, and with the increase of the number of attributes, the time of update algorithm does not increase significantly, and in this experiment, AES 128 encryption and decryption algorithm is used to test the encryption and decryption time of different size files. From the Fig. 7(b), as the size of the file increases, encryption and decryption also increased significantly, and the increase in encryption is indeed close to the same.

In Fig. 8(a) we tested the relationship between the number of 'AND' and 'OR' and the time. The initial number of 'AND' and 'OR' is 0, and then increases by 4, 8, 12, 16 in turn. As the number of 'AND' and 'OR' increases, their time overhead increases, and the increase in time is roughly the same. The relationship between the number of 'AND' and 'OR' reduction and time was tested in Fig. 8(b). The initial number of 'AND' and 'OR' is 20, and then decreases by 4, 8, 12, 16 in turn. From the images, it can be seen that when the number of 'AND' decreases, its time overhead shows a downward trend, while the initial time overhead of 'OR' is very small, with the number of 'OR' decreases, the time changes little.

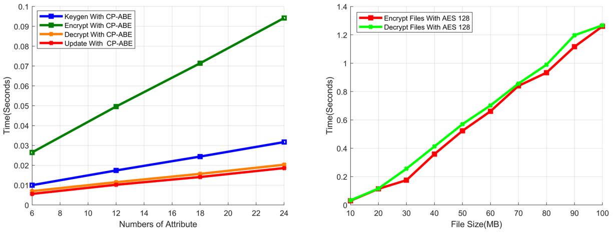


FIGURE 7. Computation cost of our scheme.

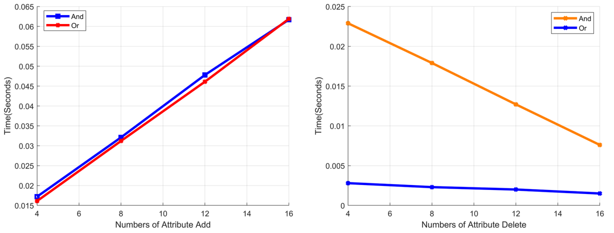


FIGURE 8. Overhead of our policy update scheme.

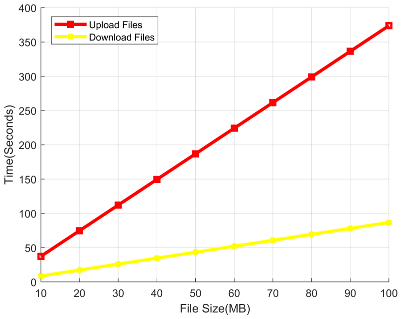


FIGURE 9. Communication cost of our scheme.

In Fig. 9 we tested the time cost of uploading different small files to the server and downloading from the server. Because the network speed of the local network is stable, the time of uploading and downloading files increases with the increase of file size, and the time cost of uploading increases obviously.

We also evaluate the encryption and decryption overhead of AES-128, the extra overhead comparison is listed

TABLE 3. Compare the time of adding 'AND' and the time of adding 'OR'.

Number of Attributes	$n = 4$	$n = 8$	$n = 12$	$n = 16$
Add 'AND' and 'OR' Procedure (*Second*)				
AND	0.0172	0.0321	0.0478	0.0616
OR	0.0161	0.0312	0.0460	0.0619

in Table 2. Besides, in Table 3 we give specific time cost for adding 'AND' and adding 'OR'.

VII. CONCLUSION

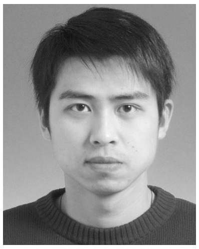
In this paper, we applied the CP-ABE scheme to the electronic health records system. Based on the difference between the PHRs and other data, we propose a policy update method based on the CP-ABE scheme. Unlike other policy update methods, we implement partial ciphertext update, which greatly reduces the computational overhead of ciphertext update in policy update. But at the same time, we need the cloud server to identify and process the updated ciphertext components, which to some extent increases the computing overhead of the cloud server. In addition, our scheme does not take into account the special privacy requirements of PHRs, which may be achieved by policy hiding. But the new problem will be how to combine policy hiding with policy updating.

ACKNOWLEDGMENT

The authors appreciate the help from the unsigned researchers and students for their constructive advises and suggestions.

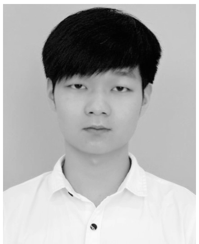
REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Eurocrypt*, May 2005, pp. 457–473.
[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Nov. 2006, pp. 89–98.
[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Comput. Soc. (SP)*, May 2007, pp. 321–334.
[4] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2010.
[5] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, Mar. 2011, pp. 53–70.
[6] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, May 2004, pp. 223–238.
[7] J. Xiong *et al.*, "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2842773.
[8] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, "RSE-PoW: A role symmetric encryption PoW scheme with authorized deduplication for multimedia data," *Mobile Netw. Appl.*, vol. 23, no. 3, pp. 650–663, 2018, doi: 10.1007/s11036-017-0975-x.
[9] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1908–1920, Aug. 2017.
[10] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Proc. IMA Int. Conf.*, 2001, pp. 360–363.
[11] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2958–2970, Aug. 2018, doi: 10.1109/JIOT.2017.2768073.
[12] R. Wang, J. Yan, D. Wu, H. Wang, and Q. Yang, "Knowledge-centric edge computing based on virtualized D2D communication systems," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 32–38, May 2018.
[13] D. Wu, F. Zhang, H. Wang, and R. Wang, "Security-oriented opportunistic data forwarding in mobile social networks," *Future Gener. Comput. Syst.*, vol. 87, pp. 803–815, Oct. 2018.
[14] J. E. Cahill, M. R. Gilbert, and T. S. Armstrong, "Personal health records as portal to the electronic medical record," *J. Neurooncol.*, vol. 117, no. 1, pp. 1–6, Mar. 2014.
[15] A. Roehrs, C. A. da Costa, R. da Rosa Righi, and K. S. F. de Oliveira, "Personal health records: A systematic literature review," *J. Med. Internet Res.*, vol. 19, no. 1, p. e13, Jan. 2017, doi: 10.2196/jmir.5876.
[16] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant-size keys for lightweight devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 5, pp. 763–771, May 2014.
[17] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Proc. Int. Workshop Public Key Cryptogr.*, Mar. 2014, pp. 293–310.
[18] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2012, pp. 199–217.
[19] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 2013–2021.
[20] Z. Ying, H. Li, J. Ma, J. Zhang, and J. Cui, "Adaptively secure ciphertext-policy attribute-based encryption with dynamic policy updating," *Sci. China Inf. Sci.*, vol. 59, p. 042701, Apr. 2016.
[21] W. Yuan, "Dynamic policy update for ciphertext-policy attribute-based encryption," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 457, 2016.
[22] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*. Berlin, Germany: Springer, 2011, pp. 568–588.
[23] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. EUROCRYPT*, 2005, pp. 440–456.
[24] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. EUROCRYPT*, 1997, pp. 256–266.
[25] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. CCS*, 1993, pp. 62–73.



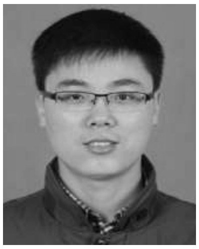
ZUOBIN YING (M'17) received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2016. He is currently a Lecturer with the School of Computer Science and Technology, Anhui University, China. His research interests include cloud security and applied cryptography.

WENJIE JANG is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei, China. His research focuses on applied cryptology.



SHUANGLONG CAO is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei, China. His research focuses on applied cryptology.

XIMENG LIU (S'13–M'16) received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the Ph.D. degrees in cryptography from Xidian University, China, in 2015. He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University, China. He is also a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published over 100 research articles include the IEEE TIFS, the IEEE TDSC, the IEEE TC, the IEEE TII, the IEEE TSC, and the IEEE TCC. His research interests include cloud security, applied cryptography, and big data security.



JIE CUI was born in Zhoukou, Henan, China, in 1980. He received the Ph.D. degree with the University of Science and Technology of China in 2012. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University. He has published over 50 papers. His current research interests include applied cryptography, IoT security, vehicular ad hoc networks, and software-defined networking.