Fine-grained Access Control Scheme Supporting Cloud-assisted Write Permission Control in Cloud-aided E-Health System

Kai He[1,3], Ziqi Wang[1,3], Jiaoli Shi[2], Anyuan Deng[2], and Shunlin Lv[2]
(Corresponding author: Jiaoli Shi)
School of Computer Science and Artificial Intelligence, Wuhan Textile University[1]
School of Computer and Big Data Science, Jiujiang University[2]
407081693@qq.com
3 Hubei Clothing Information Engineering Technology Research Center, China[3]
(Received July 30, 2021; Revised and Accepted Jan. 28, 2022; First Online Apr. 2, 2022)

Abstract

We consider a multi-reader-multi-writer scene in cloud-aided E-Health systems. Data is produced on all kinds of medical devices and encrypted to ciphertexts for security. These pieces of ciphertext would be aggregated as an electronic medical records file on controllers and uploaded onto the cloud. Doctors then download and decrypt the encrypted file, make diagnoses and treatment plans, and write them in the encrypted file. Nurses implement real-time treatment plans and record progress in the same file. This paper proposes a Fine-grained Access Control Scheme supporting Cloud-assisted Write Permission Control. In this scheme, multiple authorized users can read the same file but cannot write files until they are appropriate. We represent Users' statuses as a matrix and use a Viète formula to match them with a write access policy on the cloud to judge whether the user can modify the file or not.

Keywords: Attribute-based Encryption; Fine-grained Access Control; Write Permission Control

1 Introduction

With the fast development of cloud services, cloud-based PHR (personal health record) systems becomes popular more and more, such as Google Health and Microsoft HealthVault. In these PHR systems, lightweight medical devices and controllers are deployed gradually in hospital or home, and users can access PHR services anytime and anywhere. In the scene, most security issues stem from the plaintext transmission of data. Thus, the data need to be encrypted to be ciphertext before being sent to the controller.

These pieces of ciphertext then would be aggregated as an electronic medical records file on controllers and uploaded onto the cloud. Doctors then download and decrypt the encrypted file, make diagnoses and treatment plans, and write them in the encrypted file. Accordingly, nurses implement real-time treatment plans and record progress in the same file. This scenario is called Multi-Reader-Multi-Writer by us.

It is troublesome to carry out access control over a wide variety of data generated by all kinds of devices. Fortunately, ABE (Attribute-Based Encryption) may be the most suitable method. When the ABE method is adopted, the judgment of reading permission is in the user, while the control of write permission is generally migrated to the cloud server by the owners. On the one hand, owners want a cloud server to realize aggregating privilege control. However, on the other hand, the cloud server cannot get out anything about the ciphertext.

Our contributions can be summarized as follows.

1) We propose a fine-grained access control scheme supporting multiple authorized users to write the same encrypted medical record file. In this scheme, the encrypted file can be read by multiple users who have authorized read rights and can be modified by some doctors or nurses who has authorized write rights and be in an appropriate status (such as at work).

2) We present a representation and matching method of users' statuses. Users' statuses represent as a Matrix and match on the cloud by using the Viète formula to judge whether the user can modify the file or not.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 gives the system architecture. Section 4 demonstrates the construction. Section 5 presents all kinds of analyses. Section 6 demonstrates the efficiency of our proposed scheme. Finally, Section 7 concludes the paper. This paper concretizes the application scenario of our scheme, stores a medical records file using blockchain, and extends several parts over its earlier version [2]. These extended parts include

security model and formal proof, correctness analysis, performance analysis, *et al.* What is more, and we adjusted simulation.

2 Related Works

Fugkeaw *et al.* [4] presented a write access enforcement mechanism based on the proxy re-encryption method, in which users may have permission to update files stored on the cloud. Despite the data owner freeing from encrypting files to be updated and loading back to the cloud, the cloud servers asked the user to enter the passphrase to decrypt and re-encryption file.

Dong *et al.* [3] designed a SECO (Secure and scalable data collaboration services) scheme in cloud computing, in which the latest version of written data was decided by early in early write principle. Ruj *et al.* [15] compelled that data is written by a single user at a time using Claim Policy. Li *et al.* [13] only considered the Create-Writing situation. They divided time into slices and controlled write permission using Hash chain and signature.

Jahan *et al.* [7] extended the CP-ABE scheme to support write operation with coarse-grained write access. Fugkeaw *et al.* [5] represented read and write access privilege as an attribute of a user. Jahan *et al.* [9] provided authorized users with fine-grained read/write access without altering access policies specified by data owners.

Lee *et al.* [11] used attribute-based encryption as Self-updatable encryption (SUE) and presented revocable-storage attribute-based encryption (RS-ABE) by combining user revocation and ciphertext updating functionalities. Yang *et al.* [19] allow patients' vital signs to be measured and aggregated on medical devices and uploaded on a cloud for storage and healthcare workers access. They mainly focused on the lightweight break-glass access control system and did not investigate the aggregating privilege control. Wang *et al.* [17] proposed an RWAC (read and write access control) scheme, in which the write control was done on a user using the CP-ABE method. Jahan *et al.* [8] also agreed on a drawback of CP-ABE. Users can modify the access policy specified by the data owner if write operations are incorporated in the scheme. However, their scheme enabled the users to perform write operations without altering the access policy. Their write control was still done on a user. Alam [1] mentions five platforms to develop IoT systems using blockchain technology. They are IOTA, IOTIFY, Exec, Xage and SONM. IOTA solves various performance limitations of blockchains. IOTIFY provides an online web solution. Exec helps users' applications to the benefits of the cloud used. Xage is a secure IoT blockchain platform for adding automation. SONM is a medium-sized fog computing platform.

Many Cloud Service Providers (CollateBox, Microsoft Azure, Windows Azure, Google docs, Amazon S3, etc.) also allow multiple writers in one file, although most of them use role-based access control (RBAC), which is a centralized method to control the access of shared documents.

3 System Architecture

3.1 System Model

As shown in Figure 1, there are six entities in the system: AA (attribute authorities), Server (cloud servers), Medical devices, Controller (the data controller), Reader (data readers, such as patients or their families), and Writer (data modifier, such as doctors or nurses). We assume that: a) AA and Controller are trusted. b) Cloud is semi-trusted, which will execute all tasks correctly but is curious about ciphertexts' content. c) Unauthorized readers cannot read an out-sourced ciphertext, and unauthorized writers cannot write a ciphertext. Readers can read the ciphertext but cannot write it. Writers can read or write it.

AA (Trusted Attribute Authorities). They generate a public parameter set, and then generate, issue, revoke and update three keys (a global private key GSK_{u_t}, a global public key GPK_{u_t}, and a private attribute key SK_{u_t}) for each user called u_t in this paper. With GSK_{u_t}, users can make a digital signature to ensure data integrity. Using the private key SK_{u_t}, Readers can read out-sourced ciphertext if his/her attributes meet the read policy defined in the ciphertext. With SK_{u_t}, Writers can read and modify out-sourced ciphertext if his/her attributes meet the write policy defined in the ciphertext.

Medical Devices. They monitor the body's various parameters data (called M by us) and encrypt them to be Cm by running a certain symmetric encryption algorithm with a content key $Key_{content}$. The symmetric key $Key_{content}$ was negotiated in advance between the medical devices and their controllers.

Controller. It receives Cm from medical devices and encrypts the content key to Cp by running a CP-ABE encryption algorithm with a read policy defined by himself/herself. Next, Controllers defines a write policy (X, Y) and encrypts it with a part of Cp to Ct. As a result, the encrypted electronic health record file ($Cm||Cp||Ct$) is constructed and uploaded on Server.

Reader of an Out-Sourced Data. Anyone can download an encrypted electronic health record file from a cloud server and tries to read it by matching his/her private key SK_{u_t} to the read policy in Cp. If the match succeeds, he/she can get the content key and decrypt the encrypted electronic health record file to the plaintext data M.

Writer of an Out-Sourced Data. Any writer can download and read a ciphertext if his SK_{u_t} matches

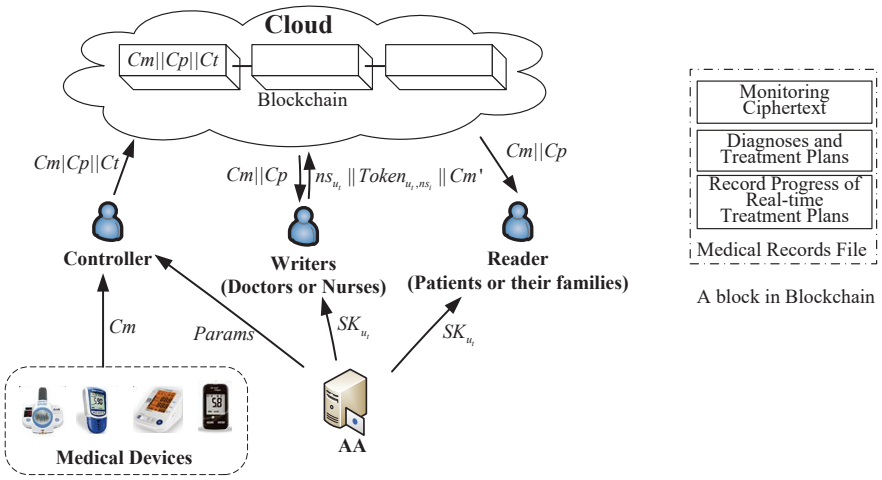


Figure 1: System Model

successfully to the read policy in Cp. Next, he/she may modify the plaintext data M and encrypts it to Cm' with the content key. Then he/she signs his/her write credential $ns_{u_t}||Token_{u_t,ns_i}$ with his/her global private key GSK_{u_t} and uploads the result and Cm' on Server.

Server. The semi-trusted cloud server in cloud-aided E-health system is always online, stores all encrypted electronic health record files submitted by controllers, and provides data access services anytime and anywhere. Let ns_{u_t} denotes the user's status of written data, ns_i the status vector of users' access,$Token_{u_t,ns_i}$ the write credential. When Writer uploads a writing request $ns_{u_t}||Token_{u_t,ns_i}||Cm'$, Server stores the user's status ns_{u_t}, and tries to match Ct with $ns_{u_t}||Token_{u_t,ns_i}$ and other users' statuses saved before. Suppose the match succeeds (means that the write policy is satisfied), Server updates Cm with Cm'. Otherwise, Server records the user's status ns_{u_t} and ignores the request. What is more, Server can be realized by a cluster of multiple servers not to make a bottleneck or a single point of failure in the system.

3.2 Security Requirements

Data Confidentiality. Unauthorized users or cloud servers (which are semi-trusted) should be prevented from accessing the plaintext of the data. This is because that they do not have enough attributes to satisfy the read access policy.

Collusion Resistance. Multiple users cannot improve the ability to decrypt a ciphertext by combining their attributes. It is assumed that doctors or nurses will not share their write credentials with others for their good.

Semi-Hiding and Unpredictability. Cloud servers will complete the match of the writing policy faithfully without knowing the details of write policies or write credentials. They also cannot predict whether or not a user's request writing data is accepted.

3.3 Security Model

We define the security for our scheme in terms of a game as follows:

Setup.

The challenger runs the algorithm $Setup$ to generate public parameters $params$ and a master secret key MSK. Then, he publishes the $params$ to an adversary Λ.

Phase 1.

The adversary Λ can submit a challenge (X^*, Y^*) to the challenger, and construct his/her $Token_{u_t,ns_i}$. When Λ queries User u_t on State ns_i, the challenger signs $Token_{u_t,ns_i}$ using GSK_{u_i}, and issues them to Λ.

Challenge.

The adversary Λ gives a challenge (X, Y), which must be satisfied with $X^* \cup X - X^* \cap X \geq 2$ *or* $Y^* \cup Y - Y^* \cap Y \geq 2$. Ct is constructed and sent to Λ, and then matched with $Token_{u_t,ns_i}$.

Phase 2.

Λ can query and construct more $Token_{u_t,ns_i}$, as long as they do not violate the constraints on the challenge (X^*, Y^*).

Guess.

Λ outputs a guess (X^*, Y^*) of (X, Y).

The advantage of an adversary Λ in this game is defined as:

$$Pr[X^* = X \wedge Y^* = Y] = 0.5$$

This completes the security game.

4 Proposed Scheme

4.1 Overview

When CP-ABE is used to control the access to medical record files in E-Health systems, the data generated by monitor devices should be uploaded onto the cloud immediately after being encrypted. The encrypted data is sent to a controller to be aggregated into a medical record file. The file is bound with a read policy and a write policy, uploaded on the cloud, and stored in blockchain. Then doctors or nurses can download the encrypted medical record file to make diagnoses and treatment plans or record the progress of real-time treatment plans remotely. These plans or progress are also uploaded onto the cloud and stored in the medical record file.

In the above scene, monitor devices, controllers, doctors, or nurses cannot always be online or restore massive data. Due to their limited storage ability or may only carry a lightweight mobile terminal,these data are usually uploaded on the cloud and downloaded when read or written by users. Then Server can process the written data by two methods:

1) *According to the content of data* (the first method). The data owner verifies and decides the latest version of written data submitted by multiple users according to the data content. Based on this method, we construct two collaborative access control schemes (see the previous research [10]).

2) *According to the write access policy of data* (the second method). Server decides the latest version of written data according to the write policy defined by Owner. In addition, Owner can specify an arbitrary on-demand policy to ensure data consistency. Based on the second method, this work constructs a new Access Control Scheme supporting Ciphertext Writing Privilege Management in Cloud-aided E-Health System.

In our scheme, the writer (doctors or nurses) can read the monitoring record file, make diagnoses and treatment plans, and write them in the medical records file. In contrast, nurses implement real-time treatment plans and record progress in the same medical records file. These writers can write these data to the medical records file and form an updated file M'. Then they encrypt M' to Cm' by symmetric encryption and upload the Cm' on the cloud along with their write access credentials $ns_{u_t}||Token_{u_t,ns_i}$. To prevent imitate attack, these writers sign $ns_{u_t}||Token_{u_t,ns_i}||Cm'$ with their global private key GSK_{u_t}. Subsequently, $Sign(ns_{u_t}||Token_{u_t,ns_i}||Cm')_{GSK_{u_t}}$ is sent to the cloud as a writing request.

When the writing request arrives in Server, Server firstly verifies the signature using the user's global public key GPK_{u_t} and then matches $ns_{u_t}||Token_{u_t,ns_i}$ to Ct. If the match is successful, Server covers Cm with Cm'. Otherwise, Server ignores the writing request. For convenience, we put aside the realization of concurrent mechanisms. Each write success triggers a blockchain transaction event so that the write operation can be recorded on blockchain and cannot be tampered with.

To facilitate the research, let us focus on the write privilege control and make a quick summary. To control the collaborative writing on a single file by multiple writers, Controller defines a write policy, constructs Ct by associating the policy with the ciphertext, and sends Ct and the ciphertext onto Server together. Server then aids Controller to match the collaborative write policy with the writer's write credential when a writer submits a writing request.

Our scheme addresses two issues:

1) Designing the structure of Ct and the writer's write credential.

2) Judging whether or not the writer's write credential is satisfied with Ct (that is, match of Ct).

4.2 Structure of Ct

Inspired by [14], we design a new access structure of Ct, wherein writers' statuses may be input manually or generated automatically.

Let $u_cs_d|u_as_b$ indicate the relationship between a device u_c and Status s_d when another device u_a accesses data in the status s_b. Let $u_cs_d|u_as_b = \{-1, +1, ^*\}$. $u_cs_d|u_as_b = -1$ indicates that the device u_a can access data in the status s_b only if u_c has ever been in the status s_d.$u_cs_d|u_as_b = +1$ indicates that u_a can access data in the status s_b only if u_c is being in the status s_d. s_d.$u_cs_d|u_as_b = *$ indicates that there is not any constraint in u_cs_d when the user u_a accesses data in the status s_b.

Let Nu denotes the number of writers, Ns the number of statuses, and $N = Nu \cdot Ns$. The access structure can be described by a matrix Ms (shown in Figure 2).

When the writer u_a writes data in the status s_b, the access structure, expressed as a line of Matrix Ms, must be satisfied by other writers' statuses. Therefore, we extract all of the elements of the line of Ms, and construct a vector $\overrightarrow{m}_{line_{a,b}}$, wherein $line_{a,b} = (a-1) * Ns + b$.

For instance, let $Nu = 3$, and $Ns = 4$. The vector $\overrightarrow{m}_1$ (the first line of Ms) is assumed as: $\overrightarrow{m}_1 = (1, *, *, *, *, *, *, *, *, *, *, *)$. It denotes that User u_1 can write the data in Status s_1 with independence of other writers or their statuses. The vector $\overrightarrow{m}_{12}$ (the latest line of Ms) is assumed as: $\overrightarrow{m}_{12} = (-1, *, -1, 1, *, *, *, *, *, *, -1, 1)$. It denotes that the writer u_1 has ever been in the status s_1 and the status s_3, the writer u_3 has ever been in the status s_3, while both u_1 and u_3 are being in the status s_4, when the writer u_3 writes the data in the status s_4.

According to different values of $\overrightarrow{m}_{line}$, we construct three sets: X, Y and P, wherein, the elements of

$$\begin{bmatrix} u_1s_1 \,|\, u_1s_1 & \cdots & u_1s_{Ns} \,|\, u_1s_1 & u_2s_1 \,|\, u_1s_1 & \cdots & u_{Nu}s_{Ns} \,|\, u_1s_1 \\ u_1s_1 \,|\, u_1s_2 & \cdots & u_1s_{Ns} \,|\, u_1s_2 & u_2s_1 \,|\, u_1s_2 & \cdots & u_{Nu}s_{Ns} \,|\, u_1s_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_1s_1 \,|\, u_{Nu}s_{Ns} & \cdots & u_1s_{Ns} \,|\, u_{Nu}s_{Ns} & u_2s_1 \,|\, u_{Nu}s_{Ns} & \cdots & u_{Nu}s_{Nm} \,|\, u_{Nu}s_{Ns} \end{bmatrix}$$

Figure 2: Access Structure of User Status Constraints

$X = \{x_i\}$ and $P = \{p_k\}$ are respectively the positions of +1, -1 and * in $\overrightarrow{m}_{line}$. For example, when $\overrightarrow{m}_{12} = (-1, *, -1, 1, *, *, *, *, *, *, -1, 1)$,$X = \{4, 12\}$,$Y = \{1, 3, 11\}$, and $P = \{2, 5, 6, 7, 8, 9, 10\}$.

4.3 Structure of the Write Credential

The write credential is designed as a monitor vector of a writer status consists with multiple elements of $u_a s_b$. $u_a s_b = +1$ denotes that the writer u_a is being in the status s_b, $u_a s_b = -1$ denotes that the writer u_a has ever been in the status s_b, and $u_a s_b = 0$ denotes that the writer u_a hasn't ever been in the status s_b.

Let $\vec{r}_a = (u_a s_1, u_a s_2, \cdots u_a s_{Ns})$, and $\vec{r} = (\vec{r}_1, \vec{r}_1, \cdots \vec{r}_N)$ can be described as follows.

$$\vec{r} = (u_1s_1, u_1s_2, \cdots u_1s_{Ns}, u_2s_1, u_2s_2, \cdots u_2s_{Ns}, \cdots u_{Ns}s_1, u_{Ns}s_2, \cdots u_{Ns}s_{Ns})$$

The number of elements of $\vec{r}$ is N:$|\vec{r}| = N$.

For example, $\vec{r} = (-1, -1, 0, 1, -1, 0, 0, 1, 0, -1, -1, 1)$. $\vec{r}_1 = (-1, -1, 0, 1)$ denotes that the writer u_1 has ever been in the status s_1 and s_2, and is being in the status s_4 now.

According to the values of different elements of $\vec{r}$, two sets can be constructed as: X' and Y'. The elements of $X' = \{x_i\}$ are positions of +1 in $\vec{r}$, and those of $Y' = \{y_i\}$ are positions of -1 in $\vec{r}$. For instance, $\vec{r} = (-1, -1, 0, 1, -1, 0, 0, 1, 0, -1, -1, 1)$, $X' = \{4, 8, 12\}$ and $Y' = \{1, 2, 5, 10, 11\}$.

4.4 Intuition of Matching Ct

Whether $\vec{r}$ is satisfied to $\vec{m}_{line}$ is the same as whether $X \subseteq X' \wedge Y \subseteq Y'$ is valid, wherein, $X \subseteq X'$ is equivalent to $X = X' - P$, and $Y \subseteq Y'$ to $Y = Y' - P$. In the above example, $X' - P = \{4, 12\}$, $Y' - P = \{1, 11\}$.

Inspired by [14], we use Viète formula on the wildcard set P to construct a proper coefficient a_j in Formula (1).

$$\prod_{p_\tau \in P} (x - p_\tau) = \sum_{l=0}^{n_p} (a_l \cdot x^l) \tag{1}$$

where, $p_\tau, a_l \in Z$, n_p denotes the number of elements in P, and $n_p \leq N$.

The coefficient a_l can be constructed by Viète Formula (2) as follows.

$$a_{n-t} = (-1)^t \sum_{1 \leq k_1 < k_2 < \ldots < k_t \leq n} p_{i_1} p_{i_2} \cdots p_{i_t} \tag{2}$$

We can get Formula (3) when replacing x of the above Formula (1) with $i = 1, 2, 3..., n_X$ and cast them up.

$$\sum_{i=1,2,3...,n_X} \prod_{p_\tau \in P} (i - p_\tau) = \sum_{i=0}^{n_p} (a_l \cdot \sum_{i=1,2,3...,n_X} i^l) \tag{3}$$

Based on Formula (3), we construct our scheme.

4.5 Sketch of Scheme

Our scheme includes three parts and four algorithms, which are described in Figure 3. To keep things simple, we focus on the write privilege control, especially matching a write credential to a write policy Ct.

Initialization. AA calls the algorithm *Setup* to generate a global public parameter set (*params* called by us) and a master private key MSK, and publishes it. Then AA generates three keys for each user: a global private key GSK_{u_t}, a global public key GPK_{u_t} and a private attribute key SK_{u_t}. AA sends GSK_{u_t} and SK_{u_t} to User by a key exchange protocol. SK_{u_t} is not marked in Figure 3, because that it was irrelevant with the write privilege access control.

Write Policy Definition. Owner defines a collaborative write policy, calls the algorithm *EncryptCt* to construct Ct, and uploads it onto Server. Server attaches Ct behind $Cm||Cp$, and then $Cm||Cp||Ct$ is stored.

Data Write. When a user writes data, he/she signs $ns_{u_t}||Token_{u_t,ns_i}||Cm'$ with his/her global private key GSK_{u_t}, and submits it onto Server. Then Server verifies $ns_{u_t}||Token_{u_t,ns_i}||Cm'$ with the user's global public key GPK_{u_t}, and runs the algorithm $MatchCt$ to match Ct. If the match succeeds, Server accepts the written data, and updates Cm to Cm'.

4.6 Construction of Our Scheme

In this section, we describe in detail our scheme. As introduced in Section above, our scheme has four algorithms: Setup, *EncryptCt*, *TokenGen* and *MatchCt*.

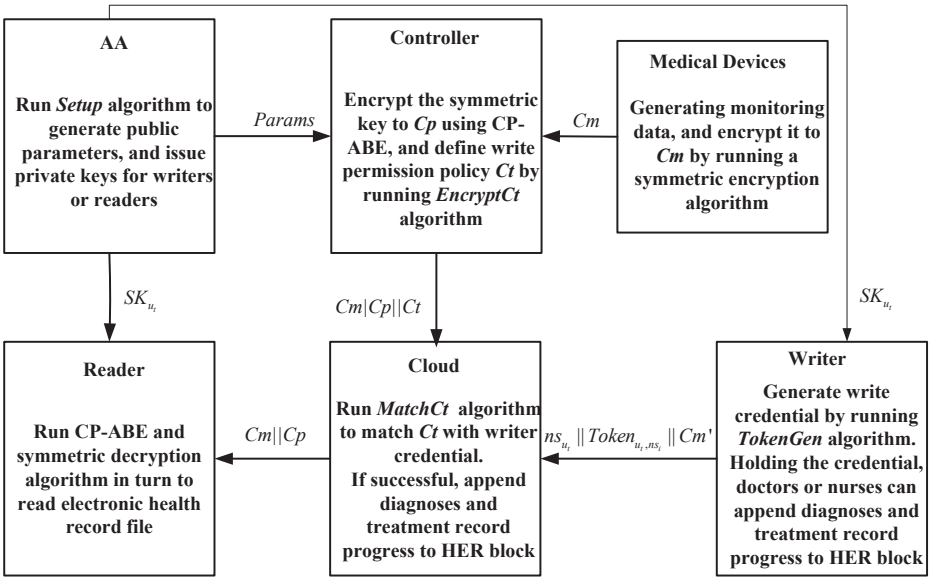


Figure 3: Sketch of our scheme

4.6.1 Setup

The algorithm *Setup* runs on AA. It chooses two group elements: h_0,$g \in G_q$ wherein G_q is a group with a prime order q. It also chooses $N+4$ random numbers: $\alpha_1, \beta_1, \beta_2, \delta, \delta_1, \delta_2, ..., \delta_N \in Z_q^*$, and computes:

$$\Omega_1 = e(g, h_0)^{\alpha_1 \beta_1},$$
$$\Omega_2 = e(g, h_0)^{\alpha_1 \beta_2},$$
$$\{h_w = h_0^{\delta_w}\}_{w=1}^{N}, \{\tilde{x}_w = g^{\delta^{x_w}}\}_{w=1}^{N}, \{\tilde{y}_w = g^{\delta^{y_w}}\}_{w=1}^{N}.$$

Then AA publishes public parameters and stores the master secret key MSK as follows:

$$params = (\{h_w, \tilde{x}_w, \tilde{y}_w\}_{w=0}^{N}, g^{\alpha_1}, \Omega_1, \Omega_2, q, G, e(.,.), g^{\delta}),$$
$$MSK = (\alpha_1, \beta_1, \beta_2, \delta, \delta_1, \delta_2, ..., \delta_N)$$

4.6.2 EncryptCt

The algorithm *EncryptCt* runs on a controller to construct *User Status Constraints*, the write policy Ct. The controller chooses two random numbers $\mu_1, \mu_2 \in Z_q^*$, and constructs Ct as follows.

$$\begin{aligned}
Ct &= (Ct_0, \{Ct_{1,w}\}_{w=0}^{n_P}, \{Ct_{2,w}\}_{w=0}^{n_P}, Ct', Ct''),\\
Ct_0 &= \Omega_1{}^{\mu_1} \Omega_2{}^{\mu_2},\\
Ct_{1,w} &= (h_w \cdot \prod_{i=1, x_w \in X}^{n_X} g^{i^w \cdot \delta^{x_w}})^{a_w \cdot (\mu_1 + \mu_2)},\\
Ct_{2,w} &= (h_w \cdot \prod_{i=1, y_w \in Y}^{n_Y} g^{i^w \cdot \delta^{y_w}})^{a_w \cdot (\mu_1 + \mu_2)},\\
Ct' &= g^{\alpha_1, \mu_1},\\
Ct'' &= g^{\mu_2}
\end{aligned}$$

Wherein, $\{a_w\}$ is computed by (2). Then the controller sends Ct onto the cloud, and stores it on the cloud.

4.6.3 TokenGen

The algorithm *TokenGen* runs on a writer (a doctor or nurse) to generate a write access credential.

We encrypt the set P concatenating with the content key $Key_{content}$:

$$\hat{C} = (Key_{content} || P) \cdot e(g, g)^{\alpha \cdot s}$$

We assume that these writers who possess reading permission can get the set P by decrypting $policy_{read}$. It is a reasonable assumption because that if writers can get the plaintext M, they will know which users or statuses the plaintext M it is related to.

When a writer makes a writing request, he/she computes $\theta = \sum_{w=0}^{n_P} \delta_w \cdot a_w$ based on P and Viète formula, chooses a random number $s \in Z_q^*$, and constructs $Token_{u_t, ns_t}$:

$$\begin{aligned}
Token_{u_t, ns_t} &= \left(\tilde{S}_0, \tilde{S}_1, \tilde{S}_2, \tilde{S}_3, \tilde{S}_4, P\right),\\
\tilde{S}_0 &= g^{\frac{\alpha_1 \cdot s}{\theta}},\\
\tilde{S}_1 &= h_0^{s_1} \cdot \prod_{\substack{i=1,2,\cdots,n_X \\ x_i \in X' - P}} g^{\frac{s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{x_i},\\
\tilde{S}_2 &= h_0^{\alpha_1 s_2} \cdot \prod_{\substack{i=1,2,\cdots,n_X \\ x_i \in X' - P}} g^{\frac{\alpha_1 \cdot s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{x_i},\\
\tilde{S}_2 &= h_0^{\alpha_1 s_2} \cdot \prod_{\substack{i=1,2,\cdots,n_X \\ x_i \in X' - P}} g^{\frac{\alpha_1 \cdot s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{x_i},\\
\tilde{S}_4 &= h_0^{\alpha_1 \cdot s_2} \prod_{\substack{i=1,2,\cdots,n_Y \\ y_i \in Y' - P}} g^{\frac{\alpha_1 \cdot s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{y_i}.
\end{aligned}$$

4.6.4 MatchCt

The algorithm $MatchCt$ runs on Server (any of cloud servers) to match Ct. Server calculates ρ based on P by the Viète formula: $\rho = \{a_w\}_{w=0}^{n_p}$, and computes M_1 and M_2 as below.

$$M_1 = Ct_0 \frac{e(\tilde{S}_0, \prod_{w=0}^{n_P} (Ct_{1,w})^{a_w})}{e(\tilde{S}_1, Ct') \cdot e(\tilde{S}_2, Ct'')},$$
$$M_2 = Ct_0 \frac{e(\tilde{S}_0, \prod_{w=0}^{n_P} (Ct_{2,w})^{a_w})}{e(\tilde{S}_3, Ct') \cdot e(\tilde{S}_4, Ct'')}$$

If M_1 and M_2 are both 1 ($M_1 = M_2 = 1$), the writing policy is satisfied. Otherwise, the cloud server ignores the writing request.

In our scheme, AA and Server must be online all the time, but others don't need to be.

5 Analysis

5.1 Correctness Analysis

The detail analysis is presented as below.

$$\begin{aligned}
M_1 =& Ct_0 \frac{e(\tilde{S}_0, \prod_{w=0}^{n_P} (Ct_{1,w})^{a_w})}{e(\tilde{S}_1, Ct') \cdot e(\tilde{S}_2, Ct'')}, \\
part1 \stackrel{def}{=}& e(\tilde{S}_0, \prod_{w=0}^{n_P} (Ct_{1,w})^{a_w}) \\
=& e\left(g^{\frac{a_1 \times s}{q}}, \left(h_0^q \times g^{\sum_{w=1}^{n_P} \left(\sum_{i=1}^{n_X} i^w \times d^{x_i} \times a_w\right)}\right)^{m_1+m_2}\right) \\
=& e(g, h_0)^{\alpha_1 \cdot s \cdot (\mu_1+\mu_2)} \\
& \cdot e(g, g)^{\frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1+\mu_2) \cdot \sum_{w=1}^{n_P} \left(\sum_{i=1}^{n_X} i^w \cdot \delta^{x_i} \cdot a_w\right)}, \\
part2 \stackrel{def}{=}& e(\tilde{S}_1, Ct') \\
=& e\left(h_0^{(s+\beta_1)} \cdot g^{\sum_{i=1}^{n_X} \left(\frac{s}{\theta} \cdot \prod_{\tau=1}^{n_P} (i-p_\tau)\right) \cdot \delta^{x_i}}, g^{\alpha_1 \mu_1}\right) \\
=& e(g, h_0)^{\alpha_1 \mu_1 \beta_1} \cdot e(g, h_0)^{s \cdot \alpha_1 \mu_1} \\
& \cdot e(g, g)^{\frac{\alpha_1 \mu_1 \cdot s}{\theta} \cdot \sum_{i=1}^{n_X} \left(\prod_{\tau=1}^{n_P} (i-p_\tau) \cdot \delta^{x_i}\right)}, \\
part3 \stackrel{def}{=}& e(\tilde{S}_2, Ct'') \\
=& e\left(h_0^{\alpha_1(s+\beta_2)} \cdot g^{\sum_{i=1}^{n_X} \left(\frac{\alpha_1 s}{\theta} \cdot \prod_{\tau=1}^{n_P} (i-p_\tau)\right) \cdot \delta^{x_i}}, g^{\mu_2}\right) \\
=& e(g, h_0)^{\alpha_1 \mu_2 \beta_2} \cdot e(g, h_0)^{s \cdot \alpha_1 \mu_2} \\
& \cdot e(g, g)^{\frac{\alpha_1 \mu_2 \cdot s}{\theta} \cdot \sum_{i=1}^{n_X} \left(\prod_{\tau=1}^{n_P} (i-p_\tau) \cdot \delta^{x_i}\right)}, \\
Ct_0 =& e(g, h_0)^{\alpha_1 \beta_1 \mu_1 + \alpha_1 \beta_2 \mu_2}, \\
M_1 =& e(g, h_0)^{\alpha_1 \beta_1 \mu_1 + \alpha_1 \beta_2 \mu_2} \cdot \\
& \frac{e(g, h_0)^{\alpha_1 \cdot s \cdot (\mu_1+\mu_2)}}{e(g, h_0)^{\alpha_1 \beta_1 \mu_1 + \alpha_1 \mu_2 \beta_2} \cdot e(g, h_0)^{s \cdot \alpha_1 \cdot (\mu_1+\mu_2)}} \cdot \\
& \frac{e(g, g)^{\frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1+\mu_2) \cdot \sum_{w=1}^{n_P} \left(\sum_{i=1}^{n_X} i^w \cdot \delta^{x_i} \cdot a_w\right)}}{e(g, g)^{\frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1+\mu_2) \cdot \sum_{i=1}^{n_X} \left(\prod_{\tau=1}^{n_P} (i-p_\tau) \cdot \delta^{x_i}\right)}} = 1.
\end{aligned}$$

The correctness analysis of our scheme has been finished.

5.2 Security Proof

Decisional q-parallel BDHE Assumption. Decisional q-parallel Bilinear Diffie-Hellman Exponent Assumption problem (q-parallel BDHE, for short) can be recalled as follows.

Choose a group G of prime order p, two random numbers $a, s \in Z_p$ and a random element $g \in G$. If an adversary is given y:

$$\begin{aligned}
y =& (g, g^s, g^a, ..., g^{a^q}, g^{a^{q+2}}, ..., g^{a^{2q}}, \\
\forall_{1 \le j \le q}& \left(g^{s \cdot b_j}, g^{\frac{a}{b_j}}, ..., g^{\frac{q}{b_j}}, g^{\frac{q+1}{b_j}}, ..., g^{\frac{2q}{b_j}}\right), \\
\forall_{1 \le j,k \le q, k \ne j}& \left(g^{\frac{a \cdot s \cdot b_k}{b_j}}, ..., g^{\frac{a^q \cdot s \cdot b_k}{b_j}}\right))
\end{aligned}$$

It is hard to distinguish a valid tuple $e(g, g^{a^{q+1} \cdot s}) \in G_T$ from a random element R in G_T.

An algorithm $\mathbb{B}$ that outputs $z \in \{0, 1\}$ has advantage ε in solving Decisional q-parallel BDHE problem if

$$\begin{aligned}
|Pr\left[\mathbb{B}\left(y, T = e(g, g)^{a^{q+1} \cdot s}\right) = 0\right] \\
- Pr\left[\mathbb{B}\left(y, T = R\right) = 0\right]| \varepsilon.
\end{aligned}$$

Theorem 1. *Suppose that an adversary Λ can find a polynomial time algorithm $\hat{A}$ that can success the game with the advantage ε.*

Proof. Permissions on out-sourced ciphertext should be read or written for the owner and writers, read for the readers, neither read nor written for all others. We need to prove that a reader cannot write the ciphertext. Thus, we assume that the adversary Λ can read a ciphertext, but cannot write the ciphertext.

We assume that the adversary Λ chooses a challenge (X^*,Y^*), and the (X^*,Y^*) is compared to (X,Y) with

two different elements at least. For simplicity, we assume that $X^* - X = \{\tilde{x}\}$ and $X - X^* = \{\hat{x}\}$.Our goal is to prove that the adversary Λ cannot distinguish X^* and X.

Setup. The challenger issues public parameters params to $\hat{A}$. $params = (\{h_w\}_{w=0}^{N}, \{g_w\}_{w=1}^{N}, g^{\alpha_1}, \Omega_1, \Omega_2, q, G, e(\cdot,\cdot), g^{\delta})$

Phase 1. $\hat{A}$ computes $\theta = \sum_{w=0}^{N} \delta_w \cdot a_w$, and chooses $s \in Z_q^*$, and constructs $Token_{u_t,ns_t}$:

$$Token_{u_t,ns_t} = \left(\tilde{S}_0, \tilde{S}_1, \tilde{S}_2, \tilde{S}_3, \tilde{S}_4, P\right),$$

$$\tilde{S}_0 = g^{\frac{\alpha_1 \cdot s}{\theta}},$$

$$\tilde{S}_1 = h_0^{s_1} \cdot \prod_{\substack{i=1,2,\cdots,n_{X^*} \\ x_i \in X^{*'} - P}} g^{\frac{s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{x_i},$$

$$\tilde{S}_2 = h_0^{\alpha_1 \cdot s_2} \prod_{\substack{i=1,2,\cdots,n_{X^*} \\ x_i \in X^{*'} - P}} g^{\frac{\alpha_1 \cdot s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{x_i},$$

$$\tilde{S}_3 = h_0^{s_1} \cdot \prod_{\substack{i=1,2,\cdots,n_{Y^*} \\ y_i \in Y^{*'} - P}} g^{\frac{s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{y_i},$$

$$\tilde{S}_4 = h_0^{\alpha_1 \cdot s_2} \prod_{\substack{i=1,2,\cdots,n_{Y^*} \\ y_i \in Y^{*'} - P}} g^{\frac{\alpha_1 \cdot s}{\theta}} \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{y_i}$$

Challenge. The challenger constructs and retains a table to record (X^*,Y^*). When the challenge (X^*,Y^*) exists in the table, the challenger reconstructs and returns Ct, otherwise, he/she returns the same Ct as before.

Ct is matched as follows.

$$\begin{aligned} M_1 &= e(g,g)^{\frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1 + \mu_2)\left(\sum_{w=1}^{n_P} i^w \cdot \widehat{\delta^x} \cdot a_w - \prod_{\tau=1}^{n_P} (i - p_\tau) \cdot \delta^{\tilde{x}}\right)} \\ &= e(g,g)^{\frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1 + \mu_2)\left(\sum_{w=1}^{n_P} i^w \cdot a_w\right) \cdot \left(\widehat{\delta^x} - \delta^{\tilde{x}}\right)} \end{aligned}$$

Phase 2. Same as **Phase 1**.

Guess. Let $\tilde{a}$ be defined as follows.

$$\tilde{a} \overset{def}{=} \frac{\alpha_1 \cdot s}{\theta} \cdot (\mu_1 + \mu_2)\left(\sum_{w=1}^{n_P} i^w \cdot a_w\right)$$

We then have: $M_1 = e(g,g)^{\tilde{a} \cdot \left(\widehat{\delta^x} - \delta^{\tilde{x}}\right)}$.

According to Decisional q-parallel DBHE problem, the adversary Λ cannot distinguish $e(g,g)^{\widehat{\delta^x}}$,$e(g,g)^{\delta^{\tilde{x}}}$, or a random element R. Correspondingly, he/she cannot distinguish X^* or X. □

5.3 Features of Matching Ct

The write policy is semi-hidden in our scheme. Firstly, it consists of five parts: three group elements (Ct_0, Ct', Ct'') and two sets $(\{Ct_{1,w}\}_{w=0}^{n_P}, \{Ct_{2,w}\}_{w=0}^{n_P})$. These five parts cannot derive out the corresponding write policy. Secondly, the algorithm $MatchCt$ can't leak out a write policy in the process of matching Ct either. The adversary can only get the set P. The write policy then holds a $semi-hidden$ feature.

What's more, the way of matching holds a noteworthy feature: $re-usability$. It is observed that the construction of Ct is independent of the random numbers μ_1 and μ_2. Therefore, we can reuse the writing access structure by refreshing Ct with different μ_1 and μ_2 after each successful match: Cloud server chooses a random number $\psi \in Z_q^*$, and computes: $Ct_{1,w} \leftarrow (Ct_{1,w})^{\psi}$,$Ct_{2,w} \leftarrow (Ct_{2,w})^{\psi}$, $Ct' \leftarrow (Ct')^{\psi}$,$Ct'' \leftarrow (Ct'')^{\psi}$.

As a result, the write policy holds two features: $semi-hidden$ and $reusability$. With these two features, our scheme holds $unpredictability$. That is to say, a server can help data owners with writing permission control, but cannot predict or determine the subsequent writing request.

5.4 Performance Analysis

In this section, we compare storage and computation costs to other two classic schemes (DAC-MACS [18] and Hur's [6]) on each entity and complete simulations of core algorithms in our scheme.

5.4.1 Storage Cost Comparison

Table 1 shows the comparison of storage costs. We ignore the storage cost of random integers like the other two schemes (DAC-MACS [18] and Hur's [6]).

Wherein, $|p|$: Storage cost of an elemental of groups; $n_{a,k}$: Number of attributes managed by AA_k; N_A: Number of AAs in the scheme; n_{a,k,u_t}: Number of attributes distributed by AA_k to u_i; $n_{u_t,k}$: Number of users managed by AA_k; t_r : Number attributes of the access tree $Tree_{Read}$ assigned by each owner; n_p: Number of wildcards in a vector of constraint of access structure; N: Number of elements in the constraint access structure.

In our scheme, AA_k stores $\{h_w\}_{w=0}^{N}$, $\{g_w\}_{w=1}^{N}$, g^{α_1}, Ω_1, Ω_2, which equals $N|p|$, $N|p|$, $|p|$, $|p|$ and $|p|$ respectively. The storage costs on both owner and user are the same as that of our previous work [10]. The storage cost of Ct on cloud is Ct_0, $\{Ct_{1,w}\}_{w=0}^{n_p}$, $\{Ct_{2,w}\}_{w=0}^{n_p}$, Ct' and Ct'', which equals $|p|$, $n_P|p|$,$n_P|p|$, $|p|$ and $|p|$ respectively. Compared to existing schemes, our scheme spends less storage cost on each user. But that on AA_k and cloud is the most because we have taken multi-writer access control into account, while they did not.

Table 1: Comparison of storage cost ($|p|$)

	DAC-MACS	Hur's	Our previous work	Our Scheme
AA_k	$n_{a,k}+3$	$n_{a,k}+3$	$(1+\sum_{k=1}^{N_A} n_{a,k})\,\|p\|$	4+2N
Owner	$3N_A+1+\sum_{k=1}^{N_A} n_{a,k}$	$2N_A+\sum_{k=1}^{N_A} n_{a,k}$	$(3+2\sum_{k=1}^{N_A} n_{a,k,u_t})\,\|p\|$	$1+\sum_{k=1}^{N_A}(n_{a,k})$
User	$1+2\sum_{k=1}^{N_A} n_{a,k,u_t}$	$3N_A+1+\sum_{k=1}^{N_A} n_{a,k,u_t}$	$(n_{a,k}+3+n_{u_t,k})\,\|p\|$	$1+\sum_{k=1}^{N_A} n_{a,k,u_t}$
Cloud	$3+3t_r$	$3+3t_r$	$(3+2t)\,\|p\|$	$5+2t_r+2n_P$

5.4.2 Computation Cost Comparison

Table 2 gives out the computation cost comparison of the core algorithms. We ignore the computation cost of multiplying and dividing like the other two schemes (DAC-MACS [18] and Hur's [6]).

$|E|$: Exponent arithmetic; $|Pe|$: An $e(g,g)$ bilinear mapping operation; k: Number of attributes of a user's private key; I_{A_k}: Attribute set in a ciphertext issued by AA_k; n_P: Number of elements in a wildcard set P; n_X: Number of elements in a status set X; n_Y: Number of elements in a status set Y.

As shown in Table 2, the running time of the algorithm Setup is proportional to M (the number of statuses). The algorithm $EncryptCt$ runs on each owner to construct Ct, which need $(6+n_X+n_Y)|E|$. Algorithm $MatchCt$ runs on Server to match Ct, which needs $(2n_P)|E|+6|Pe|$. It is related with N (the number of users). Compared to the other two schemes, our scheme works for multi-writer access control, but those of DAC-MACS [18] and Hur's [6] for a single user is an extra cost.

We can replace users' identities with an attribute set to reduce delay time if the number of attribute sets is lesser than the number of users' identities. This replacement can extend our scheme to a larger scale.

6 Simulation

We complete the simulation on Ubuntu system with an Intel Core i7 10^{th} Gen CPU. The Pairing-Based Cryptography library is installed onto Ubuntu to simulate all of the algorithms. The elliptic curve is chosen as, the order of all groups as 160 bit, and the field size as 512bit. Times are the mean of 10 trials to avoid the results of accidents.

Figure 4 (a) gives out the computational time comparison between Hur's [6], Li's [12], Teng's [16], and ours. It shows that the encrypting time spent on a controller is similar to these two schemes, Hur's [6] and Li's [12]. We have joined the data sharing and aggregating scheme by writing privilege permission control with a negligible performance impact. Figure 4(b), (c) and (d) show that the computational costs spent on each entity are remarkably correlated linearly with n_P without depend on n_X or n_Y. It is worth knowing that the write policy and token can be generated on a controller and a writer separately ahead of time. The computational delay spent on token matching are almost 50 ms, which falls within the user acceptable dealing tolerance range. This simulation gives the feasibility of this scheme.

7 Conclusion

This paper analyzes the control requirements when multiple doctors or nurses collaboratively write the same encrypted data in Cloud-aided E-Health scene. In response to these requirements, we propose an Access Control Scheme supporting Ciphertext Writing Privilege Management in Cloud-aided E-Health System by expressing a write access control policy as Matrix. Our scheme has two noteworthy features:

1) Fine-grained write privilege control. Authorized doctors or nurses can write data legally only when his/her write credential satisfies the write policy defined by the data's controller, while unauthorized users cannot.

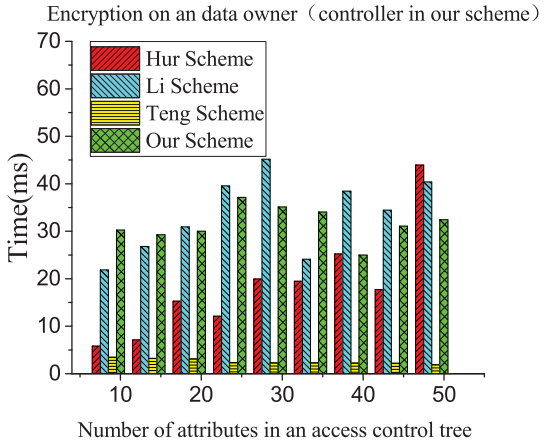
2) Data-binding-policy access control method. The outsourced data is bound to a collaborative write policy before being stored on the cloud. The policy can be on-demand now that it is defined by the data owner, bringing flexibility to our scheme.

Acknowledgments

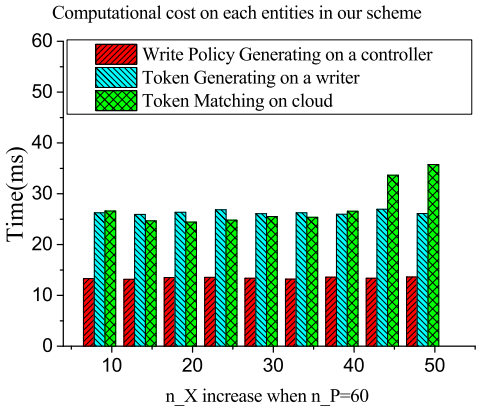
This work is supported by the National Science Foundation of China (No. 62062045), the Science and Technology Research Project of Jiangxi Provincial of Education

Table 2: Computational cost comparison

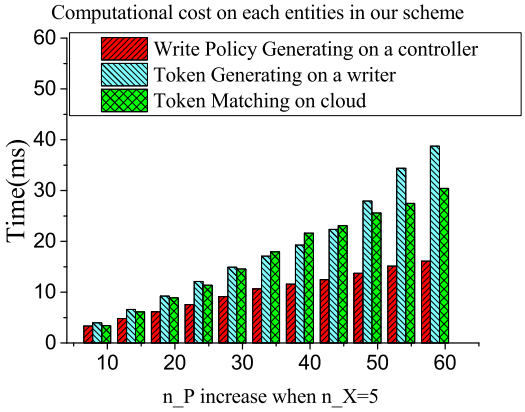
	EncryptData on controller	**DecryptData** on User	**DecryptData** on Cloud	**EncryptCt** on controller	**MatchCt** on Server
DAC-MACS	$(3+6t_r)\|E\|$	$\|E\|$	$N_A \times ((\sum_{k=1}^{I_{A_k}}(3\|Pe\|+\|E\|)) +2\|Pe\|$	-	-
Hur's	$(2+2t_r)\|E\|$	$(k+logt_r)\|E\| +(2k+1)\|Pe\|$	0	-	-
Our previous work	$(2+2t_r)\|E\|$	$\|E\|+\|P\|$	$(k+logt_r)\|E\|+(2k)\|Pe\|$	-	-
Our Scheme				$(6+n_X+n_Y)\|E\|$	$(2n_P)\|E\|+6\|Pe\|$



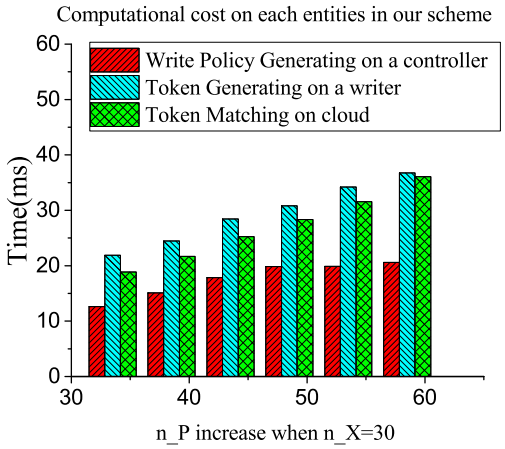
(a)Compare with other schemes



(b)n_X is changing when n_P=60



(c) n_P is changing when n_X=5



(d) n_P is changing when n_X=30

Figure 4: Simulation of Computational Cost

Department (No.GJJ180905), and the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (No.20YJAZH112).

References

[1] T. Alam, "A survey on the use of blockchain for the internet of things," *International Journal of Electronics and Information Engineering*, vol. 13, no. 3, pp. 119–130, 2021.

[2] A. Deng, J. Shi, and K. He, "Acs-wam: an access control scheme supporting write authority management in cloud-assisted cyber-physical systems," in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pp. 747–752, 2019.

[3] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Computers & security*, vol. 42, pp. 151–164, 2014.

[4] S. Fugkeaw and H. Sato, "Enforcing hidden access policy for supporting write access in cloud storage systems.," in *CLOSER*, pp. 530–536, 2017.

[5] S. Fugkeaw and H. Sato, "An extended cp-abe based access control model for data outsourced in the cloud," in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 3, pp. 73–78, 2015.

[6] J. Hur and K. Kang, "Secure data retrieval for decentralized disruption-tolerant military networks," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 16–26, 2012.

[7] M. Jahan, M. Rezvani, A. Seneviratne, and S. Jha, "Method for providing secure and private fine-grained access to outsourced data," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 406–409, 2015.

[8] M. Jahan, M. Rezvani, Q. Zhao, P.S. Roy, K. Sakurai, A. Seneviratne, and S. Jha, "Light weight write mechanism for cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 1131–1146, 2017.

[9] M. Jahan, P.S. Roy, K. Sakurai, A. Seneviratne, and S. Jha, "Secure and light weight fine-grained access mechanism for outsourced data," in *2017 IEEE Trustcom/BigDataSE/ICESS*, pp. 201–209, 2017.

[10] S. Jiaoli, H. Chuanhe, W. Jing, Q. Kuangyu, and H. Kai, "Multi-user collaborative access control scheme in cloud storage [j]," *Journal on Communications*, vol. 37, no. 1, pp. 88–99, 2016.

[11] K Lee, D.H. Lee, J.H. Park, M. Yung, and Y. Mu, "Cca security for self-updatable encryption: Protecting cloud data when clients read/write ciphertexts," *The Computer Journal*, vol. 62, no. 4, pp. 545–562, 2019.

[12] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2016.

[13] 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, 2012.

[14] T.V.X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE transactions on information forensics and security*, vol. 11, no. 1, pp. 35–45, 2015.

[15] S. Ruj, M. Stojmenovic, and A. Nayak, "Decentralized access control with anonymous authentication of data stored in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 384–394, 2013.

[16] W. Teng, G. Yang, Y. Xiang, T. Zhang, and D. Wang, "Attribute-based access control with constant-size ciphertext in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 617–627, 2015.

[17] J. Wang, B. Lang, and R. Zhu, "Rwac: A self-contained read and write access control scheme for group collaboration," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 97–103, 2018.

[18] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "Dacmacs: effective data access control for multiauthority cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, 2013.

[19] Y. Yang, X. Liu, and R.H. Deng, "Lightweight break-glass access control system for healthcare internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3610–3617, 2017.

Biography

Kai He received the B.S. degree in computer science from Wuhan Textile University, Wuhan, China, in 2010, and the Ph.D. degree from Wuhan University, Wuhan, China, in 2016. Since 2016, He has been a Lecturer at the School of Mathematics and Computer, Wuhan Textile University, Wuhan, Hubei, China. From Jan. to June 2015, he was a visiting Student at the University of Calgary. His research interest includes the Cloud security, auction, *et al.*

Ziqi Wang received the B.S. in computer science from Wuhan Textile University, Wuhan, Hubei, China in 2020. He is currently pursuing the master degree in Wuhan Textile University. His research interests include cloud security and data security, blockchain.

Jiaoli Shi received the Ph.D. degree from the School of Computing, Wuhan University, in 2017. Since 2012, she has been an Assistant Professor at the school of computer and big data science, Jiujiang University. Her research

interests include Cloud security, ICN security, SDN and CDN. Dr. Jiaoli twice won the third prize of Doctoral Forum of School of Computer Science, Wuhan University for Excellence in 2014 and 2015.

Anyuan Deng received the B.Sc. degree in computer science from Jiangxi Normal University, Nanchang, China, in 1995, M.Sc. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2006. Since 2003, He has been a professor at the school of computer and big data science, Jiujiang University, Jiujiang, China. His research interests include Cloud security, ICN security, Software Engineering and CDN.

Shunlin Lv is an undergraduate student in the school of computer and big data science, Jiujiang University. His research interests include Cloud security, Privacy protection.