HCBE: Achieving Fine-Grained Access Control in Cloud-Based PHR Systems

Xuhui Liu[1], Qin Liu[1(✉)], Tao Peng[2], and Jie Wu[3]

[1]College of Computer Science and Electronic Engineering,
Hunan University, P. R. China, 410082
[2]School of Information Science and Engineering,
Central South University, P. R. China, 410083
[3]Department of Computer and Information Sciences,
Temple University, Philadelphia, PA 19122, USA
Correspondence to: gracelq628@hnu.edu.cn

Abstract. With the development of cloud computing, more and more users employ cloud-based personal health record (PHR) systems. The PHR is correlated with patient privacy, and thus research suggested to encrypt PHRs before outsourcing. Comparison-based encryption (CBE) was the first to realize time comparison in attribute-based access policy by means of the forward/backward derivation functions. However, the cost for encryption is linearly with the number of attributes in the access policy. To efficiently realize a fine-grained access control for PHRs in clouds, we propose a hierarchical comparison-based encryption (HCBE) scheme by incorporating an attribute hierarchy into CBE. Specifically, we construct an attribute tree, where the ancestor node is the generalization of the descendant nodes. The HCBE scheme encrypts a ciphertext with a small amount of generalized attributes at a higher level, other than lots of specific attributes at a lower level, largely improving the encryption performance. Furthermore, we encode each attribute node with the positive-negative depth-first (PNDF) coding. By virtue of the backward derivation function of the CBE scheme, the users associated with the specific attributes can decrypt the ciphertext encrypted with the generalized attributes, within the specified time. The experiment results show that the HCBE scheme has better performance in terms of the encryption cost, compared with the CBE scheme.

Keywords: Personal Health Record; Cloud Computing; Comparison-Based Encryption; Fine-Grained Access Control; Attribute Hierarchy.

1 Introduction

In recent years, personal health record (PHR) [1] as a patient-centric model of health information exchange has become popular with more and more users due to its convenience to access a patient's centralized profile by merging a wide range of health information sources. PHR allows medical practitioners to online

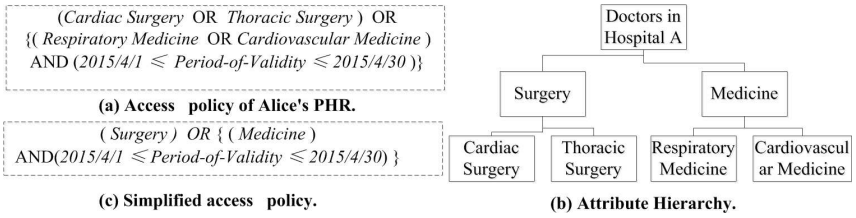


Fig. 1. Application scenario.

access a complete and accurate summary of a patient's medical history, thereby making the healthcare processes much more efficient and accurate [2].

Cloud computing is a model for enabling ubiquitous and convenient network access to data resources [3]. Due to its overwhelming advantages, e.g., rapid elasicity, high availability, and low cost, more and more patients decide to outsource their PHRs to the cloud for flexibility and convenience. The most popular cloud-basd PHR systems include Google Health [4] and Microsoft HealthVault [5], which promise the users to access the PHR services anytime and anywhere using any devices connected to the Internet.

However, a PHR which includes patient-centric health data, such as allergies and adverse drug reactions, family history, imaging reports (e.g. X-ray) and so on, is closely related to patient privacy. Allowing a cloud service provider (CSP), like Amazon, Google, and Microsoft, to take care of sensitive medical data, may raise potential issues. For instance, an untrustworthy CSP may intentionally leak PHRs to medical companies or medical instrument companies for making a profit. To preserve the patients's privacy while using the cloud-based PHR systems, research suggested to encrypt PHR before outsourcing [6].

Let us consider the following application scenario: Alice was hospitalized in Hospital A, requiring heart surgery. Unfortunately, the surgery may face high risk, since Alice also suffered from hypertension and asthma. The related attending doctors in Hospital A needed to hold a consultation to decide surgery program after carefully studying Alice's PHR. For convenience and flexibility, Alice uploaded her encrypted PHR to Google Health, specifying an access control policy, as shown in Fig. 1-(a).

The access policy can be viewed as a description of attributes and time condition, specifying that only the users whose attributes satisfying the access policy can decrypt the ciphertext within the specified time. For instance, Fig. 1-(a) stipulates that the cardiologists and the respiratory physicians, can view Alice's PHR during the consultation (April 1, 2015 $\sim$ April 30, 2015), as well as the cardiac surgeons and the cardiothoracic surgeons can view it anytime.

To ensure fine-grained access control in above application scenario, the adopted encryption scheme needs meeting the following requirements: (1) Supporting attribute-based access policy. For example, for a given ciphertext associated with access policy $[(A_1 \wedge A_2) \vee A_3]$, only the users who possess both attributes A_1 and A_2 or those who possess attribute A_3 can recover it using their own decryption keys. (2) Supporting time-based comparison. For example, the time condition of

the ciphertext is $t_x \leq A_k \leq t_y$, which means that the users possess attribute A_k can access the data during time $[t_x, t_y]$.

Comparison-based encryption (CBE) [7] as a promising tool facilitating a fine-grained access control in cloud computing was proposed by Zhu et. al. in 2012. CBE utilizes the forward and backward derivation functions to achieve time comparison in attribute-based encryption. For example, suppose that the access policy of the ciphertext is $A_t \wedge [t_x, t_y]$, and the authorization time of the user with attribute A_t is $[t_a, t_b]$. Then, the user can decrypt the ciphertext only when the current time $(t_c \in [t_x, t_y]) \wedge (t_c \in [t_a, t_b])$. Meanwhile, the key delegation mechanism was applied to assign a majority of decryption cost to the cloud, so as to take full advantage of cloud resources.

However, the main drawback of CBE is that the encryption cost grows linearly with the number of attributes in the access policy. For a system of a large number of attributes, the cost for encryption may be extensive. To solve this problem, we propose a hierarchical comparison-based encryption (HCBE) scheme for efficiently achieving a fine-grained access control in cloud-based PHR systems. The main idea of the HCBE scheme is building a hierarchical structure for attributes, where the attribute at a higher level is a generalization of the attributes at lower levels. Specifically, we encrypt the ciphertext with a small amount of generalized attributes at the higher level, other than lots of specific attributes at the lower level. For example, if we construct an attribute tree as shown in Fig. 1-(b), the access policy can be simplified as shown in Fig. 1-(c), with which the computation cost for encryption may be largely reduced compared to that in Fig. 1-(a).

To realize the attribute hierarchy, we encode each node in an attribute tree with the positive-negative depth-first (PNDF) coding. Then, we apply the backward derivation function of CBE to allow the descendant attribute node to deduce the secrets associating with its ancestor attribute nodes. Therefore, the users with the specific attributes can decrypt the ciphertext encrypted with the generalized attributes. For example, when the ciphertext is encrypted with access policy $(\textit{medicine}) \wedge [2015-4-1, 2015-4-30]$, only the cardiologists and the respiratory physicians can decrypt it during April 1, 2015 $\sim$ April 30, 2015. Our main contributions in this work are summarized as follows:

1. We proposed a hierarchical comparison-based encryption (HCBE) scheme, by incorporating attribute hierarchy into CBE, so as to efficiently achieve a fine-grained access control in cloud-based PHR systems.
2. We constructed an attribute hierarchy tree, and encode each attribute node with the PNDF coding. By applying the backward derivation function, the users with the specific attributes can decrypt the ciphertext encrypted with the generalized attributes.
3. We analyze the security of the proposed scheme, and conduct experiments to validate its effectiveness and efficiency.

The rest of this paper is organized as follows. In Section 2, we introduce our models, design goals, and technical preliminaries. Then, we overview our HCBE

scheme in Section 3 and provide its construction in Section 4. We analyze the security of our scheme in Section 5 and conduct experiments in Section 6. Finally, we introduce the related work in Section 7, and conclude this paper in Section 8.

2 Preliminaries

2.1 System Model

The system consists of the following parties: the cloud service provider (CSP), the data owner, and the data users. The CSP operates the cloud-based PHR system, which locates on a large number of interconnected cloud servers with abundant hardware resources. The data owner is the individual patient who employs the cloud-based PHR system to manage her PHR. The data users are the entities who is authorized by the data owner to access the cloud-based PHR system. Take the scenario in Fig. 1 as an example, Alice is the data owner, Google is the CSP, and Alice's lead doctors in Hospital A are the data users. Specially, when all the data users located in the same trusted domain, a proxy server responsible for part of the decryption operation can be deployed inside.

Suppose that the universal attribute set $\mathcal{A} = \{A_1, \ldots, A_m\}$, from which an attribute hierarchy $\widehat{\mathcal{A}}$ of L levels is built. In the tree structure, each attribute A_k contains two hierarchy codes, $\{Pcode_k, Ncode_k\}$, s.t. the descendant node's codes are larger than those of its ancestors. To efficiently achieve a fine-grained access control while using the cloud-based PHR services, our HCBE scheme will be employed as follows. We describe each user with an attribute-based access privilege $\widehat{\mathcal{L}}$, where each attribute $A_k \in \widehat{\mathcal{L}}$, denoted as $A_k(t_a, t_b, Pcode_k, Ncode_k)$, is associated with the authorization time $[t_a, t_b]$ and hierarchy codes $\{Pcode_k, Ncode_k\}$.

The PHR is encrypted with an attribute-based access policy, $\widehat{AP}$, where each attribute $A_l \in \widehat{AP}$, denoted as $A_l(t_i, t_j, Pcode_l, Ncode_l)$, is also associated with the time condition $[t_i, t_j]$ and hierarchy codes $\{Pcode_l, Ncode_l\}$. The data user can decrypt the PHR only when the following conditions are simultaneously satisfied: (1) user attributes satisfy the access policy, denoted $\widehat{\mathcal{L}} \subseteq \widehat{AP}$; (2) the current time $(t_c \in [t_x, t_y]) \wedge (t_c \in [t_a, t_b])$; (3) the attributes in $\widehat{\mathcal{L}}$ are either the same as or more specific than those in $\widehat{AP}$, denoted as $Pcode_k \geq Pcode_l$ and $Ncode_k \geq Ncode_l$.

2.2 Adversary Model

Our design goal is to preserve privacy for the data owner while using the cloud-based PHR services. There are two main attacks under such a circumstance, i.e., *external attacks* initiated by unauthorized outsiders, and *internal attacks* initiated by an *honest but curious* CSP and the untrusted data users. The communication channels are assumed to be secured under existing security protocols such as SSL and SSH, thus we only consider the internal attacks. We assume that the honest but curious CSP will always correctly execute a given protocol, but may try to learn some additional information about the stored data. The untrusted data users may collude to access the PHRs outside their permissions.

The HCBE scheme is considered fail if the following cases happens:

– **CASE 1.** The data user uk of access privilege $\widehat{\mathcal{L}}$ can access the PHR of access policy $\widehat{AP}$ while (1) $\widehat{\mathcal{L}} \not\subseteq \widehat{AP}$; or for attribute $A_k[t_a, t_b, Pcode_k, Ncode_k] \in \widehat{\mathcal{L}}$ and $A_l[t_i, t_j, Pcode_l, Ncode_l] \in \widehat{AP}$ (2) the intersection of the authorization time $[t_a, t_b]$ in $\widehat{\mathcal{L}}$ and the time condition $[t_i, t_j]$ in $\widehat{AP}$ is empty; or (3) $Pcode_k < Pcode_l \vee Ncode_k < Ncode_l$.
– **CASE 2.** The CSP can access the PHR without permission.

2.3 Composite Order Bilinear Map

Let p, q be two large primes, and $N = pq$ be the RSA-modulus. Following the work in [8], we define a bilinear map group system $S_N = (N, \mathbb{G}, \mathbb{G}_\mathrm{T}, e)$, where $\mathbb{G}$ and $\mathbb{G}_\mathrm{T}$ are cyclic groups of prime order $n = sp'q'$ [1], and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathrm{T}$ is a bilinear map with the following properties:

– **Bilinearity**: for $a, b \in \mathbb{Z}_n$ and $g_1, g_2 \in \mathbb{G}$, it holds that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
– **Non-degeneracy**: $e(g_1, g_2) \neq 1$, where g_1, g_2 are the generators of group $\mathbb{G}$;
– **Computability**: $e(g_1, g_2)$ is efficiently computable.

As the work in [7], we make N public and keep n, s, p', q' secret in this system. Let $\mathbb{G}_s$ and $\mathbb{G}_{n'}$ denote the subgroups of order s and $n' = p'q'$ in $\mathbb{G}$, respectively. We have $e(g, h) = 1$, when $g \in \mathbb{G}_s$ and $h \in \mathbb{G}_{n'}$.

2.4 Comparison-Based Encryption (CBE)

In CBE, time is denoted as a set of discrete values $U = \{t_1, t_2, \ldots, t_T\}$, with total ordering $0 \leq t_1 \leq t_2 \leq \ldots, \leq t_T \leq Z$, where Z is the maximal integer. Let $\varphi, \overline{\varphi}$ be two random generators in $\mathbb{G}_{n'}$, where $n' = p'q'$ and p' and q' are two large primes. The functions $(\psi(\cdot), \overline{\psi}(\cdot))$ mapping from integer set $U = \{t_1, t_2, \ldots, t_T\}$ to $V = \{v_{t_1}, \ldots, v_{t_T}\} \in \mathbb{G}_{n'}$ and $\overline{V} = \{\overline{v}_{t_1}, \ldots, \overline{v}_{t_T}\} \in \mathbb{G}_{n'}$ are defined as follows:

$$v_{t_i} \leftarrow \psi(t_i) = \varphi^{\lambda^{t_i}}, \qquad \overline{v}_{t_i} \leftarrow \overline{\psi}(t_i) = \overline{\varphi}^{\mu^{Z-t_i}} \tag{1}$$

where λ, μ are randomly chosen from $\mathbb{Z}^*_{n'}$.

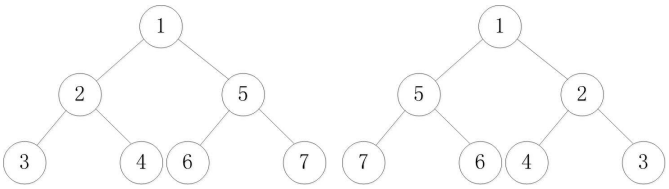
Based on Eq. 1, the forward derivation function (FDF), $f(\cdot)$, and the backward derivation function (BDF), $\overline{f}$, are defined as follows:

$$\begin{aligned} v_{t_j} \leftarrow f(v_{t_i}) &= (v_{t_i})^{\lambda^{t_j - t_i}}, \quad t_i \leq t_j \\ \overline{v}_{t_j} \leftarrow \overline{f}(\overline{v}_{t_i}) &= (\overline{v}_{t_i})^{\mu^{t_i - t_j}}, \quad t_i \geq t_j \end{aligned} \tag{2}$$

FDF and BDF have the *one-way* property, under the RSA assumption that λ^{-1} and φ^{-1} cannot be efficiently computed based on the secrecy of n'. That is, Eq. 2 is efficiently computable; However, it is intractable to obtain v_{t_j} from v_{t_i} while $t_i > t_j$, and obtain $\overline{v}_{t_j}$ from $\overline{v}_{t_i}$ while $t_i < t_j$.

For a given set of attributes $\mathcal{A} = \{A_1, \ldots, A_m\}$, CBE consists of the following algorithms: *Setup*, *GenKey*, *Encrypt*, *Delegate*, *Decrypt1*, and *Decrypt2*. For improving the efficiency, the output of the *Encrypt* algorithm is a random session key ek, which can be used to encrypt files using symmetrical-key cryptosystem.

[1] Let s_1, s_2 be two secret large primes. We have $n = sn' = s_1 s_2 p'q' | lcm(p+1, q+1)$, where $n' = p'q' | n$, $s = s_1 s_2$, $p = 2p's_1 - 1$, and $q = 2q's_2 - 1$.



(a) Positive depth-first coding (b) Negative depth-first coding

Fig. 2. Sample PNDF coding.

3 Overview of the HCBE scheme

3.1 Positive-Negative Depth-First Coding

From the attribute set $\mathcal{A} = \{A_1, \ldots, A_m\}$, we build an attribute hierarchy $\widehat{\mathcal{A}}$ of L levels. In attribute tree, the attribute at a higher level is a generalization of the attributes at lower levels. We associate each node with two hierarchical codes, i.e., the positive depth-first code (Pcode) and the negative depth-first code (Ncode). Suppose that each node has four fields: $Pcode$, $Ncode$, $rchild$ describing the right subtree, and $lchild$ describing the left subtree. First, we push the root node R to two stacks, $PStack$ and $NStack$. For the $PStack$, the right subtree of each node will be first pushed in, thus the left subtree's Pcodes will be larger than those of right subtree. In contrast, the left subtree of each node will be first pushed in the $NStack$, thus the right subtree's Ncodes will be larger than those of left subtree. Take the attribute tree shown in Fig. 1-(b) as an example. The PNDF coding is shown in Fig. 2.

Let $Pcode_i$ and $Ncode_i$ denote the Pcode and Ncode of node i, respectively. The PNDF coding has the property that $Pcode_i > Pcode_j$ and $Ncode_i > Ncode_j$, if i is the descendant node of j. For example, the $Pcode$ and $Ncode$ of attribute *Surgery* are 2 and 5, respectively; the $Pcode$ and $Ncode$ of attribute *Cardiac Surgery* are 3 and 7, respectively; the $Pcode$ and $Ncode$ of attribute *Respiratory Medicine* are 6 and 4, respectively. *Cardiac Surgery* as the descendant of *Surgery*, with both $Pcode$ and $Ncode$ larger than those of *Surgery*. *Respiratory Medicine* is not the descendant of *Surgery*, and its $Ncode$ is smaller than that of *Surgery*.

3.2 The Definition of the HCBE Scheme

Suppose that the number of nodes in the attribute hierarchy is m. In HCBE, the hierarchical codes are denoted as a set of discrete values $U_m = \{(Pcode_1, Ncode_1), (Pcode_2, Ncode_2), \ldots, (Pcode_k, Ncode_k), \ldots, (Pcode_m, Ncode_m)\}$, with total ordering $0 \leq Pcode_1 \leq Pcode_2 \leq \ldots, \leq Pcode_m \leq Z_m$ and $0 \leq Ncode_1 \leq Ncode_2 \leq \ldots, \leq Ncode_m \leq Z_m$, where Z_m is the maximal integer.

We apply the BDF to accomplish the attribute hierarchy. Let $\mathbb{G}_{n'}$ be a multiplicative group of RSA-type composite order $n' = p'q'$, where p', q' are two large primes. First, we choose random generators φ_1, φ_2 in $\mathbb{G}_{n'}$ and random numbers θ_1, θ_2 in $\mathbb{Z}^*_{n'}$, where the order of θ_1, θ_2 are sufficiently large in $\mathbb{Z}^*_{n'}$. Next, we define mapping functions $\psi_1(.), \psi_2(.)$ from an integer set $U_m =$

$\{(Pcode_1, Ncode_1), \ldots, (Pcode_k, Ncode_k), \ldots, (Pcode_m, Ncode_m)\}$ into $V_m = \{(v_{Pcode_1}, v_{Ncode_1}), \ldots, (v_{Pcode_k}, v_{Ncode_k}), \ldots, (v_{Pcode_m}, v_{Ncode_m})\}$ as follows:

$$v_{Pcode_k} = \varphi_1^{\theta_1^{Z_m - Pcode_k}}, \qquad v_{Ncode_k} = \varphi_2^{\theta_2^{Z_m - Ncode_k}} \tag{3}$$

According to the definitions of $\psi_1(.), \psi_2(.)$, it is easy to define BDFs $f_1(.), f_2(.)$ as follows:

$$\begin{aligned} v_{Pcode_j} \leftarrow f_1(v_{Pcode_k}) &= (v_{Pcode_k})^{\theta_1^{Pcode_k - Pcode_j}}, \quad Pcode_k \geq Pcode_j \\ v_{Ncode_j} \leftarrow f_2(v_{Ncode_k}) &= (v_{Ncode_k})^{\theta_2^{Ncode_k - Ncode_j}}, \quad Ncode_k \geq Ncode_j \end{aligned} \tag{4}$$

The definition of the HCBE scheme consists of the following algorithms:

- $Setup(1^\kappa, \widehat{\mathcal{A}}) \rightarrow (MK, PK_{\widehat{\mathcal{A}}})$: The data owner takes a security parameter κ and the attribute hierarchy $\widehat{\mathcal{A}}$ as inputs, and outputs the master key MK and the system public key $PK_{\widehat{\mathcal{A}}}$;
- $GenKey(MK, uk, \widehat{\mathcal{L}}) \rightarrow SK_{\widehat{\mathcal{L}}}$: The data owner utilizes her master key MK to generate a private key $SK_{\widehat{\mathcal{L}}}$ on an access privilege $\widehat{\mathcal{L}}$ for user uk, where each attribute $A_k \in \widehat{\mathcal{L}}$, denoted as $A_k(t_a, t_b, Pcode_k, Ncode_k)$, is associated with the authorization time $[t_a, t_b]$ and hierarchy codes $\{Pcode_k, Ncode_k\}$.
- $Encrypt(PK_{\widehat{\mathcal{A}}}, \widehat{AP}) \rightarrow (\widehat{\mathcal{H}_\mathcal{P}}, ek)$: The data owner takes the public key $PK_{\widehat{\mathcal{A}}}$ and an access policy $\widehat{AP}$ as inputs to generate a session key ek and a ciphertext header $\widehat{\mathcal{H}_\mathcal{P}}$, where each attribute $A_l \in \widehat{AP}$, denoted as $A_l(t_i, t_j, Pcode_l, Ncode_l)$, is associated with the time condition $[t_i, t_j]$ and hierarchy codes $\{Pcode_l, Ncode_l\}$.
- $Delegate(SK_{\widehat{\mathcal{L}}}, \widehat{\mathcal{L}}') \rightarrow SK_{\widehat{\mathcal{L}}'}$: The data user takes the private key $SK_{\widehat{\mathcal{L}}}$ and an access privilege $\widehat{\mathcal{L}}'$ as inputs to generate a derived private key $SK_{\widehat{\mathcal{L}}'}$ for the proxy server if $\widehat{\mathcal{L}}' \preceq \widehat{\mathcal{L}}$ [2].
- $Decrypt1(SK_{\widehat{\mathcal{L}}'}, \widehat{\mathcal{H}_\mathcal{P}}) \rightarrow \widehat{\mathcal{H}'_\mathcal{P}}$: The proxy server takes the derived private key $SK_{\widehat{\mathcal{L}}'}$ and a ciphertext header $\widehat{\mathcal{H}_\mathcal{P}}$ as inputs, and outputs a new ciphertext header $\widehat{\mathcal{H}'_\mathcal{P}}$ if $\widehat{\mathcal{L}}'$ satisfies $\widehat{AP}$.
- $Decrypt2(SK_{\widehat{\mathcal{L}}}, \widehat{\mathcal{H}'_\mathcal{P}}) \rightarrow ek$: The data user takes the private key $SK_{\widehat{\mathcal{L}}}$ and the new ciphertext header $\widehat{\mathcal{H}'_\mathcal{P}}$ as inputs, and outputs a session key ek, which can be used to decrypt the stored data.

4 Our Construction

$Setup(1^\kappa, \widehat{\mathcal{A}}) \rightarrow (MK, PK_{\widehat{\mathcal{A}}})$: Given a bilinear map system $S_N = (N = pq, \mathbb{G}, \mathbb{G}_\mathrm{T}, e)$, where $\mathbb{G}, \mathbb{G}_\mathrm{T}$ are cyclic groups of composite order $n = sn'$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_\mathrm{T}$, this algorithm first chooses the random generators $\omega \in \mathbb{G}, g \in \mathbb{G}_s$, and

[2] Let S and S' denote the set of attributes in $\widehat{\mathcal{L}}$ and $\widehat{\mathcal{L}}'$, respectively. $\widehat{\mathcal{L}}' \preceq \widehat{\mathcal{L}}$ iff $S' \subseteq S$, and for each attribute $A_k[t_a, t_b, Pcode_k, Ncode_k] \in \widehat{\mathcal{L}}$ and $A_l[t_i, t_j, Pcode_l, Ncode_l] \in \widehat{\mathcal{L}}'$, $t_a \leq t_i$, $t_b \geq t_j$, $Pcode_k \geq Pcode_l$, and $Ncode_k \geq Ncode_l$.

$\varphi, \overline{\varphi}, \varphi_1, \varphi_2 \in \mathbb{G}_{n'}$, where $\mathbb{G}_s$ and $\mathbb{G}_{n'}$ are two subgroups of $\mathbb{G}$. Thus, we have $e(g,\varphi) = e(g,\overline{\varphi}) = e(g,\varphi_1,) = e(g,\varphi_2) = 1$, but $e(g,\omega) \neq 1$.

Then, it chooses four random numbers $\lambda, \mu, \theta_1, \theta_2 \in \mathbb{Z}_n^*$, and employs a hash function $H : \{0,1\}^* \to \mathbb{G}$, mapping the root attribute, R, described as a binary string to a random group element. Next, it chooses two random exponents $\alpha, \beta \in \mathbb{Z}_n^*$ and sets $h = \omega^\beta, \eta = g^{1/\beta}, \zeta = e(g,\omega)^\alpha$. The master key is set as $MK = (g^\alpha, \beta, p, q, n')$, and the public key is set as:

$$PK_{\widehat{\mathcal{A}}} = (S_N, \omega, g, \varphi, \overline{\varphi}, \varphi_1, \varphi_2, h, \eta, \zeta, \lambda, \mu, \theta_1, \theta_2, H). \tag{5}$$

$GenKey(MK, uk, \widehat{\mathcal{L}}) \to SK_{\widehat{\mathcal{L}}}$: Given a user uk with license $\widehat{\mathcal{L}}$, this algorithm chooses two random numbers $\tau_{uk}, r \in \mathbb{Z}$, and then for each attribute $A_k[t_a, t_b, Pcode_k, Ncode_k] \in \widehat{\mathcal{L}}$, it calculates:

$$\begin{aligned} D_{A_k} &= (D_t, D'_{t_a}, \overline{D'}_{t_b}, D''_t, D_{K_1}, D_{K_2}) \\ &= (g^{\tau_{uk}} {H_{A_k}}^r, (v_{t_a})^r, (\overline{v}_{t_b})^r, \omega^r, (v_{Pcode_k})^r, (v_{Ncode_k})^r). \end{aligned} \tag{6}$$

where $H_{A_k} = H(R) \cdot v_{Pcode_k} \cdot v_{Ncode_k}$, $v_{t_a} = \varphi^{\lambda^{t_a}}$, $\overline{v}_{t_b} = \overline{\varphi}^{\mu^{Z-t_b}}$, $v_{Pcode_k} = \varphi_1^{\theta_1^{Z_m - Pcode_k}}$ and $v_{Ncode_k} = \varphi_2^{\theta_2^{Z_m - Ncode_k}}$. Then, uk's private key is set as:

$$SK_{\widehat{\mathcal{L}}} = (D = g^{(\alpha+\tau_{uk})/\beta}, \{D_{A_k}\}_{A_k \in \widehat{\mathcal{L}}}). \tag{7}$$

$Encrypt(PK_{\widehat{\mathcal{A}}}, \widehat{AP}) \to (\widehat{\mathcal{H}_{\mathcal{P}}}, ek)$: Given an access policy tree $\mathcal{T}$ over access policy $\widehat{AP}$, the ciphertext header $\widehat{\mathcal{H}_{\mathcal{P}}}$ can be calculated with:

$$\begin{aligned} \widehat{\mathcal{H}_{\mathcal{P}}} = (\mathcal{T}, C = h^\sigma, \{(\bar{E}_{t_i}, {E'}_{t_i}), (E_{t_j}, {E'}_{t_j}), \\ (E_{Pcode_l}, {E'}_{Pcode_l}), (E_{Ncode_l}, {E'}_{Ncode_l})\}_{A_l[t_i,t_j,Pcode_l,Ncode_l] \in \mathcal{T}}). \end{aligned} \tag{8}$$

Here, each component is set as follows:

$$\begin{aligned} &(\bar{E}_{t_i}, {E'}_{t_i}) = (\bar{v}_{t_i}\omega)^x, H_{A_l}^x), (E_{t_j}, {E'}_{t_j}) = ((v_{t_j}\omega)^y, H_{A_l}^y), \\ &(E_{Pcode_l}, {E'}_{Pcode_l}) = ((v_{Pcode_l} \cdot \omega)^{z_1}, H_{A_l}^{z_1}), \\ &(E_{Ncode_l}, {E'}_{Ncode_l})) = ((v_{Ncode_l} \cdot \omega)^{z_2}, H_{A_l}^{z_2}). \end{aligned} \tag{9}$$

where $H_{A_l} = H(R) \cdot v_{Pcode_l} \cdot v_{Ncode_l}$. The session key ek is set as $\zeta^\sigma = e(g^\alpha, \omega)^\sigma$ where σ is a main secret in $\mathbb{Z}_n$ for tree $\mathcal{T}$, and $\triangle_\sigma(A_l) = x + y + z_1 + z_2$ is the secret share of σ in the tree $\mathcal{T}$ for an attribute A_l (see Ref. [9]).

$Delegate(SK_{\widehat{\mathcal{L}}}, \widehat{\mathcal{L}'}) \to SK_{\widehat{\mathcal{L}'}}$: Given a specified access privilege $\widehat{\mathcal{L}'}$, and the private key $SK_{\widehat{\mathcal{L}}} = (D, \{(D_t, D'_{t_a}, \overline{D'}_{t_b}, D''_t, D_{K_1}, D_{K_1})\}_{A_k[t_a,t_b,Pcode_k,Ncode_k] \in \widehat{\mathcal{L}}})$, this algorithm checks for each attribute $A_l[t_i, t_j, Pcode_l, Ncode_l] \in \widehat{\mathcal{L}'}$ whether A_l is a generalized attribute of A_k, $t_a \leq t_j$ and $t_b \geq t_i$. If so, this algorithm uses Eq. 2 and Eq. 4 to compute:

$$\begin{aligned} D'_t &\leftarrow g^{\tau_{uk}} {H_{A_k}}^r \cdot \frac{f_1(D_{K_1}) \cdot f_2(D_{K_2})}{(v_{Pcode_k})^r \cdot (v_{Pcode_k})^r} \\ &= g^{\tau_{uk}} (H(R) \cdot v_{Pcode_k} \cdot v_{Ncode_k})^r \cdot \frac{f_1((v_{Pcode_k})^r) \cdot f_2(f_1((v_{Ncode_k})^r))}{(v_{Pcode_k})^r \cdot (v_{Pcode_k})^r} \\ &= g^{\tau_{uk}} H(R)^r \cdot {v_{Pcode_l}}^r \cdot {v_{Ncode_l}}^r = g^{\tau_{uk}} H_{A_l}^r \\ D'_{t_j} &\leftarrow f(D'_{t_a}) \cdot D''_t = f((v_{t_a})^r) \cdot \omega^r = (v_{t_j})^r \cdot \omega^r, \\ \overline{D}'_{t_i} &\leftarrow \overline{f}(\overline{D}'_{t_b}) \cdot D''_t = \overline{f}((\overline{v}_{t_b})^r) \cdot \omega^r = (\overline{v}_{t_i})^r \cdot \omega^r, \\ D'_{Pcode_l} &\leftarrow f_1(D_{K_1}) \cdot D''_t = f_1((v_{Pcode_k})^r) \cdot \omega^r = (v_{Pcode_l})^r \cdot \omega^r, \\ D'_{Ncode_l} &\leftarrow f_2(D_{K_2}) \cdot D''_t = f_2((v_{Ncode_k})^r) \cdot \omega^r = (v_{Ncode_l})^r \cdot \omega^r, \end{aligned} \tag{10}$$

where

$$
\begin{aligned}
&f((v_{t_a})^r) = (\varphi^{r\lambda^{t_a}})^{\lambda^{t_j - t_a}} = \varphi^{r\lambda^{t_j}} = (v_{t_j})^r,\\
&\overline{f}((\overline{v}_{t_b})^r) = (\overline{\varphi}^{r\mu^{Z-t_b}})^{\mu^{t_b - t_i}} = \overline{\varphi}^{r\mu^{Z-t_i}} = (\overline{v}_{t_i})^r,\\
&f_1((v_{Pcode_k})^r) = (\varphi_1^{r\theta_1^{Z_m - Pcode_k}})^{\theta_1^{Pcode_k - Pcode_l}} = \varphi_1^{r\theta_1^{Z_m - Pcode_l}} = (v_{Pcode_l})^r,\\
&f_2((v_{Ncode_k})^r) = (\varphi_2^{r\theta_2^{Z_m - Ncode_k}})^{\theta_2^{Ncode_k - Ncode_l}} = \varphi_2^{r\theta_2^{Z_m - Ncode_l}} = (v_{Ncode_l})^r.
\end{aligned} \tag{11}
$$

Next, it chooses a random $\delta \in \mathbb{Z}$ and computes:

$$
\begin{aligned}
&\widetilde{D}_t = D'_t \cdot (gH_{A_l})^\delta = g^{\tau_{uk}} {H_{A_l}}^r \cdot (gH_{A_l})^\delta = g^{\tau_{uk}+\delta} H_{A_l}^{r+\delta} = g^{\tau'_k} H_{A_l}^{r'},\\
&\widetilde{D}'_{t_j} = D'_{t_j} \cdot (v_{t_j}\omega)^\delta = (v_{t_j}\omega)^{r+\delta} = (v_{t_j}\omega)^{r'},\\
&\widetilde{\overline{D}'}_{t_i} = \overline{D}'_{t_i} \cdot (\overline{v}_{t_i}\omega)^\delta = (\overline{v}_{t_i}\omega)^{r+\delta} = (\overline{v}_{t_i}\omega)^{r'},\\
&\widetilde{D}'_{Pcode_l} = D'_{Pcode_l} \cdot (v_{Pcode_l}\omega)^\delta = (v_{Pcode_l}\omega)^{r+\delta} = (v_{Pcode_l}\omega)^{r'},\\
&\widetilde{D}'_{Ncode_l} = D'_{Ncode_l} \cdot (v_{Ncode_l}\omega)^\delta = (v_{Ncode_l}\omega)^{r+\delta} = (v_{Ncode_l}\omega)^{r'},
\end{aligned} \tag{12}
$$

where $H_{A_l} = H(R) \cdot v_{Pcode_l} \cdot v_{Ncode_l}$, and $\tau'_k = \tau_{uk} + \delta, r' = r + \delta$. Finally, the derivation privacy key is set as $SK_{\widehat{\mathcal{L}'}} = \{\widetilde{D}_t, \widetilde{D}'_{t_j}, \widetilde{\overline{D}'}_{t_i}, \widetilde{D}'_{Pcode_l}, \widetilde{D}'_{Ncode_l}\}_{A_l \in \mathcal{L}'}$.

$Decrypt1(SK_{\widehat{\mathcal{L}'}}, \widehat{\mathcal{H}_{\mathcal{P}}}) \rightarrow \widehat{\mathcal{H}'_{\mathcal{P}}}$: Given the private key $SK_{\widehat{\mathcal{L}'}}$ and a ciphertext header $\widehat{\mathcal{H}_{\mathcal{P}}}$, we check whether each attribute $A_l[t_i, t_j, Pcode_l, Ncode_l] \in \widehat{\mathcal{L}'}$ is consistent with $A_l[t_i, t_j, Pcode_l, Ncode_l] \in \widehat{AP}$. If true, the secret share $\triangle_\sigma(A_l)$ of σ over $\mathbb{G}_\mathrm{T}$ is reconstructed by using

$$
\begin{aligned}
F_1 &\leftarrow \frac{e(\widetilde{D}_t, E_{t_j})}{e(\widetilde{D}'_{t_j}, E'_{t_j})} = \frac{e(g^{\tau'_k} H_{A_l}^{r'}, (v_{t_j}\omega)^x)}{e((v_{t_j}\omega)^{r'}, H_{A_l}^x)}\\
&= e(g^{\tau'_k}, v_{t_j}^x) \cdot e(g^{\tau'_k}, \omega^x) = e(g^{\tau'_k}, \omega)^x
\end{aligned} \tag{13}
$$

$$
\begin{aligned}
F_2 &\leftarrow \frac{e(\widetilde{D}_t, \overline{E}_{t_i})}{e(\widetilde{\overline{D}'}_{t_i}, E'_{t_j})} = \frac{e(g^{\tau'_k} H_{A_l}^{r'}, (\overline{v}_{t_i}\omega)^y)}{e((\overline{v}_{t_i}\omega)^{r'}, H_{A_l}^y)}\\
&= e(g^{\tau'_k}, \overline{v}_{t_i}^y) \cdot e(g^{\tau'_k}, \omega^y) = e(g^{\tau'_k}, \omega)^y
\end{aligned} \tag{14}
$$

$$
\begin{aligned}
F_3 &\leftarrow \frac{e(\widetilde{D}_t, E_{Pcode_l})}{e(\widetilde{D}'_{Pcode_l}, E'_{t_j})} = \frac{e(g^{\tau'_k} H_{A_l}^{r'}, (v_{Pcode_l}\omega)^{z_1})}{e((v_{Pcode_l}\omega)^{r'}, H_{A_l}^{z_1})}\\
&= e(g^{\tau'_k}, v_{Pcode_l}^{z_1}) \cdot e(g^{\tau'_k}, \omega^{z_1}) = e(g^{\tau'_k}, \omega)^{z_1}
\end{aligned} \tag{15}
$$

$$
\begin{aligned}
F_4 &\leftarrow \frac{e(\widetilde{D}_t, E_{Ncode_l})}{e(\widetilde{D}'_{Ncode_l}, E'_{t_j})} = \frac{e(g^{\tau'_k} H_{A_l}^{r'}, (v_{Ncode_l}\omega)^{z_2})}{e((v_{Ncode_l}\omega)^{r'}, H_{A_l}^{z_2})}\\
&= e(g^{\tau'_k}, v_{Ncode_l}^{z_2}) \cdot e(g^{\tau'_k}, \omega^{z_2}) = e(g^{\tau'_k}, \omega)^{z_2}
\end{aligned} \tag{16}
$$

$$
F_t = F_1 \cdot F_2 \cdot F_3 \cdot F_4 = e(g^{\tau'_k}, \omega)^{\triangle_\sigma(A_l)} \tag{17}
$$

where $H_{A_l} = H(R) \cdot v_{Pcode_l} \cdot v_{Ncode_l}$. We have $e(g^{\tau'_k}, v_{t_j}^x) = e(g^{\tau'_k}, \overline{v}_{t_i}^y) = e(g^{\tau'_k}, v_{Pcode_l}^{z_1}) = e(g^{\tau'_k}, v_{Ncode_l}^{z_2}) = 1$ due to $g^{\tau'_k} \in \mathbb{G}_s$ and $v_{t_j}^x, \overline{v}_{t_i}^y, v_{Pcode_l}^{z_1}, v_{Ncode_l}^{z_2} \in \mathbb{G}_{n'}$. Next, the value $C_2 = e(g^{\tau'_k}, \omega)^\sigma$ is computed from $\{e(g^{\tau'_k}, \omega)^{\triangle_\sigma(A_l)}\}_{A_l \in \mathcal{T}}$

by using the aggregation algorithm (see Ref. [9]). Finally, the new ciphertext header $\widehat{\mathcal{H}'_{\mathcal{P}}} = (C = h^{\sigma}, C_2)$ is returned.

$Decrypt2(SK_{\widehat{\mathcal{L}}}, \widehat{\mathcal{H}'_{\mathcal{P}}}) \rightarrow ek$: After receiving $\widehat{\mathcal{H}'_{\mathcal{P}}} = (C, C_2) = (\omega^{\beta\sigma}, e(g^{\tau'_k}, \omega)^{\sigma})$, the data user uses the secret δ to compute

$$D' = D \cdot \eta^{\delta} = g^{(\alpha+\tau_{uk})/\beta} g^{\delta/\beta} = g^{(\alpha+\tau_{uk}+\delta)/\beta} = g^{(\alpha+\tau'_k)/\beta}. \tag{18}$$

Next, the session key is computed by

$$ek = \frac{e(C,D')}{C_2} = \frac{e(g^{(\alpha+\tau'_k)/\beta}, (\omega^{\beta})^{\sigma})}{e(g^{\tau'_k}, \omega)^{\sigma}} = e(g^{\alpha}, \omega)^{\sigma}. \tag{19}$$

5 Security Analysis

As described in Section 2, the HCBE scheme is considered failed if either CASE 1 or CASE 2 happens. In this section, we will sketch the security of our scheme as follows:

The data file stored in the cloud is in the encrypted with a session key $ek = e(g^{\alpha}, \omega)^{\sigma}$. For ease of illustration, we assume that ek is encrypted with the access policy $\widehat{AP} = A_l[t_i, t_j, Pcode_l, Ncode_l] \wedge A_x[t_i, t_j, Pcode_x, Ncode_x]$. We consider the first condition in CASE 1 is true, if user uk_1, whose access privilege $\widehat{\mathcal{L}_1} = A_l[t_i, t_j, Pcode_l, Ncode_l]$, can recover ek, by colluding with uk_2, whose access privilege $\widehat{\mathcal{L}_2} = A_x[t_i, t_j, Pcode_x, Ncode_x]$. The construction of the HCBE scheme allows them to recover $F_{t1} = e(g^{\tau'_{k1}}, \omega)^{\triangle_{\sigma}(A_l)}$ and $F_{t2} = e(g^{\tau'_{k2}}, \omega)^{\triangle_{\sigma}(A_x)}$, with private keys $SK_{\widehat{\mathcal{L}_1}}$ and $SK_{\widehat{\mathcal{L}_2}}$, respectively. However, τ'_{k1}, τ'_{k2} are uniquely chosen to distinguish different users. Therefore, with F_{t1}, F_{t2}, they cannot obtain either $T1 = e(g^{\tau'_{k1}}, \omega)^{\sigma}$ or $T2 = e(g^{\tau'_{k2}}, \omega)^{\sigma}$, to recover ek, and the first condition in CASE 1 is false.

Next, we assume that ek is simply encrypted with the access policy $\widehat{AP} = A_l[t_i, t_j, Pcode_l, Ncode_l]$. We consider the second condition in CASE 1 is true, if user uk_1, whose access privilege $\widehat{\mathcal{L}_1} = A_l[t_a, t_b, Pcode_l, Ncode_l]$ can recover ek while $t_j < t_a$ (or $t_i > t_b$). Note that, due to the one-way property of the FDF and BDF in CBE, uk_1 cannot derive D'_{t_j} and $\overline{D}'_{t_i}$ from D'_{t_a} and $\overline{D}'_{t_b}$ while $t_j < t_a$ (or $t_i > t_b$). Therefore, uk_1 cannot obtain F_1 and F_2 to recover ek, and the second condition in CASE 1 is false.

Finally, we consider the third condition in CASE 1 is true, if user uk_1, whose access privilege $\widehat{\mathcal{L}_1} = A_l[t_a, t_b, Pcode_l, Ncode_l]$ can recover ek while access policy $\widehat{AP} = A_x[t_a, t_b, Pcode_x, Ncode_x]$, while $Pcode_x > Pcode_l$ or $Ncode_x > Ncode_l$. Note that, due to the one-way property of the BDF in CBE, uk_1 cannot derive D'_{Pcode_x} from D'_{Pcode_l} while $Pcode_x > Pcode_l$. The same situation holds for $Ncode_x > Ncode_l$. Therefore, the thrid condition in CASE 1 is false, and CASE 1 will not happen. ■

The proof of CASE 2 is similar with that of CASE 1. To obtain ek, the CSP needs to calculate $F_1 \cdot F_2 \cdot F_3 \cdot F_4$ to obtain $e(g^{\tau'_k}, \omega)^{\triangle_{\sigma}(A_l)}$ for sufficient attributes $A_l \in \mathcal{T}$. Since the CSP is not allowed to access the PHR system, it cannot obtain sufficient private keys. As proved in CASE 1, the entities that do not meet the access policy cannot recover ek. Therefore, CASE 2 will not happen. ■

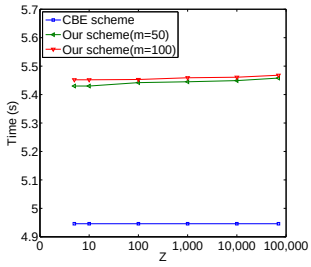


Fig. 3. Computation cost of *Setup*.

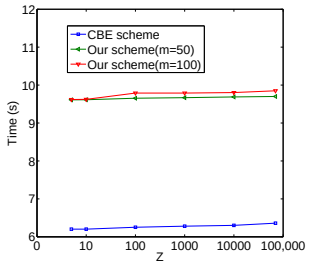


Fig. 4. Computation cost of *Genkey*.

6 Experimental Results

In this section, we will compare our proposed scheme with the CBE scheme in terms of computation cost. Our experiments are conducted with Java programming language. We implement our scheme on an stand-alone mode, on a PC with Intel Core i3 CPU running at 2.3GHz and 2G memory. The practical computational costs of algorithms *Setup*, *Genkey*, *Encrypt*, *Delegate*, *Decrypt*1, and *Decrypt*2 in both schemes are shown in Fig 3∼Fig. 8.

The parameter settings in the experiments are as follows : N_A is the number of specific attributes in the access policy, m indicates the number of attribute nodes in an attribute hierarchy tree. Here, we take $m = 50, N_A = 10$ and $m = 100$, respectively. In our experiments, we generate a private key with privilege $[t_1, t_2]$, where $t_1 \in_R [1, Z/4]$ and $t_2 \in_R [3Z/4, Z]$, for a certain comparison range $[1, Z]$. Furthermore, the message is encrypted by the time condition $t \in_R [Z/4, 3Z/4]$ to ensure that $\max(t - t_1, t_2 - t) \geq Z/4$.

Our experimental results are shown in Fig 3∼Fig. 8. We observe that the growth of time overhead is not significant as the value of Z increases in terms of the HCBE scheme and the CBE scheme. Meanwhile, in our scheme, the growth of time overhead is not significant while m grows from 50 to 100. Due to the introduction of attribute hierarchy in our scheme, the computational overhead of *Setup* and *GenKey* algorithms in our scheme is larger than that of the CBE scheme. However, the difference is minor. For example, as shown in Fig. 3, the computation time of our *Setup* algorithm grows from $5.43s$ to $5.46s$ under the setting of $m = 50$, and the computation time of the *Setup* algorithm of the CBE scheme grows from $4.94s$ to $4.95s$, while Z ranges from 7 to $70,000$; as shown in Fig. 4, the computation time of our *Genkey* algorithm grows from $9.62s$ to $9.85s$ under the setting of $m = 100$, and the computation time of the *Genkey* algorithm of the CBE scheme grows from $6.20s$ to $6.36s$, while Z ranges from 7 to $70,000$.

In the experiment, in order to get better comparison results between our scheme and CBE scheme, we use $N_A = 10$ specific attributes for CBE encryption. As shown in Fig. 5, the encryption time of our scheme is much smaller than that of the CBE scheme. For example, the computation time of our *Encrypt* algorithm

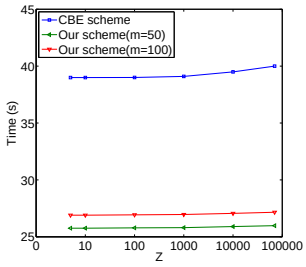


Fig. 5. Computation cost of *Encrypt*.

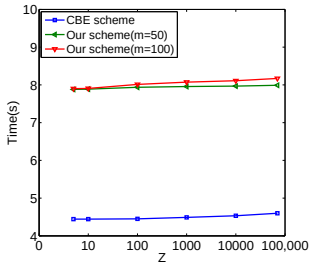


Fig. 6. Computation cost of *Delegate*.

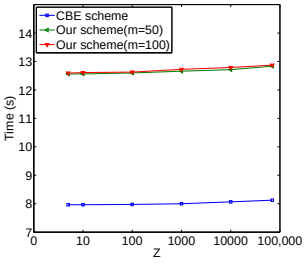


Fig. 7. Computation cost of *Decrypt*1.

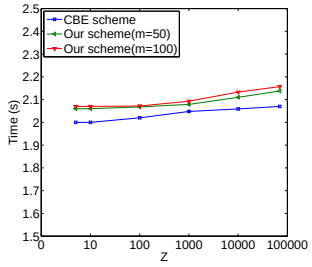


Fig. 8. Computation cost of *Decrypt*2.

grows from $25.75s$ to $25.98s$ under the setting of $m = 50$, and the computation time of the *Encrypt* algorithm of the CBE scheme grows from $38.95s$ to $40.00s$, while Z ranges from 7 to $70,000$; Furthermore, with the decrease of the number of attribute, m, in access policy, our scheme has better performance. Therefore, in our scheme, the data owner's time overhead will be reduced, thereby getting better service experience.

The algorithms run by the data user include *Delegate* and *Decrypt*2. The comparisons of time overhead are shown in Fig. 6 and Fig. 8. The comparison of time overhead of *Decrypt*1 algorithm is shown in Fig. 7, which is executed by the proxy server. Therefore, for the data users in our scheme, they have the same experience as those in the CBE scheme. The above experimental results verify our theoretical analysis in Section 5.

7 Related Work

Today, many CSP, like Amazon, Google, and Microsoft, provide PHR services. PHR contains a significant amount of sensitive information, thus how to preserve individual privacy while using cloud-based PHR system becomes a key problem [2]. To prevent the exposure of health information to unauthorized individuals, cryptographic tools and access control mechanism are proposed as promising solutions. For example, Jin et. al [10] proposed a multi-level access control scheme to support patient-centric health information sharing. Benaloh

et. al [11] designed a Patient Controlled Encryption (PCE) system to secure the storage of patients' medical records. With the PCE system, patients can share partial access rights with others, and to perform searches over their records in a secure way. Li et. al [12] proposed a novel framework for scalable and efficient access control to PHRs in cloud computing environment. Yao et. al [13] utilized order preserving symmetric encryption (OPSE) [14] for preserving data privacy in multi-source personal health record clouds. Li et.al [15] utilized predicate encryption [16] to achieve authorized search on PHRs in cloud computing.

Most existing work adopted Attribute Based Encryption (ABE) [17, 9] as the cryptographic tool to achieve fine-grained access control in cloud-based PHR systems. The original ABE systems only support monotone access policy and assume the existence of a single private key generator (PKG). A lot of research has been done to achieve more expressive access policy [18], and distributed key management [19]. To achieve dynamic access control in cloud computing, Yu et. al [20] applied the proxy re-encryption (PRE) technique [21] to ABE. Wang et. al [6] proposed a hierarchical ABE scheme to achieve key delegation in cloud environment. On the basis of ABE scheme, Zhu et. al [7] proposed the CBE scheme by making use of the forward/backward derivation functions, and applied CBE to the cloud environment. However, the encryption cost of the CBE scheme will grow linearly with the number of attributes in the access policy. To solve this problem, we proposed the HCBE scheme by incorporating the attribute hierarchy to the CBE scheme.

8 Conclusion and Future Work

In this paper, we proposed a HCBE scheme for achieving a fine-grained access control in cloud-based PHR systems. Our scheme supports time comparison in attribute-based encryption in an efficient way, by incorporating attribute hierarchy into CBE. However, due to the the limited space, we only sketch the security of the proposed scheme. In our future work, we will try to prove that the HCBE scheme has key security under chosen derivation-key attacks (KS-CDA) and semantical security under chosen derivation-key attacks (SS-CDA).

Acknowledgments

This work was supported in part by NSFC grants 61402161, 6147213161272546; NSF grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672,CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963.

References

1. P. Tang, J. Ash, D. Bates, et al, "Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption", *Journal of the American Medical Informatics Association*, 2006, vol. 13, No. 2, pp. 121-126.
2. L. Guo, C. Zhang, J. Sun, et al, "PAAS: A privacy-preserving attribute-based authentication system for ehealth networks", in *Proceedings of IEEE ICDCS*, 2012, pp. 224-233.

3. M. Armbrust, A. Fox, R. Griffith, et al, "A view of cloud computing", *Communications of the ACM*, 2010, 53(4): 50-58.
4. Googlehealth. https://www.google.com/health/.
5. Healthvault. http://www.healthvault.com/.
6. G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services", in *Proceedings of ACM CCS*, 2010, pp. 735-737.
7. Y. Zhu, H. Hu, G. Ahn, et al, "Comparison-based encryption for fine-grained access control in clouds", in *Proceedings of ACM CODASPY*, 2012, pp. 105-116.
8. D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing", *Advances in Cryptology–CRYPTO*, 2001, vol. 2139, pp.213-229.
9. J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption", in *Proceedings of IEEE S&P*, 2007, pp.321-349.
10. J. Jin, G.-J. Ahn, H. Hu, "Patient-centric authorization framework for sharing electronic health records", in *Proceedings of ACM SACMAT*, 2009, pp.125-134.
11. J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records", in *Proceedings of ACM CCSW*, 2009, pp. 103-114.
12. M. Li, S. Yu, et al, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner setting", in *Security and Privacy in Communication Networks*, 2010, vol. 50, pp. 89-106.
13. X. Yao, Y. Lin, Q. Liu, et al, "Efficient and privacy-preserving search in multi-source personal health record clouds", accepted to appear in *Proceedings of IEEE ISCC*, 2015.
14. A. Boldyreva, N. Chenette, and A. O. Neill, *Order-preserving encryption revisited: Improved security analysis and alternative solutions*, *Advances in CryptologyCCRYPTO*, 2011, vol. 6841, pp. 578-595.
15. M. Li, S. Yu, N. Cao, et al, "Authorized private keyword search over encrypted data in cloud computing", in *Proceedings of IEEE ICDCS*, 2011, pp. 383-392.
16. T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products", *Advances in Cryptology–ASIACRYPT*, 2009, vol. 5912, pp. 214-231.
17. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data", in *Proceedings of ACM CCS*, 2006, pp.89-98.
18. B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization", in *Public Key Cryptography–PKC*, 2011, vol. 6571, pp. 53-70.
19. A. Lewko and B. Waters, "Decentralizing attribute-based encryption", in *Advances in Cryptology–EUROCRYPT*, 2011, vol. 6632, pp. 568-588.
20. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure,scalable, and fine-grained data access control in cloud computing", in *Proceedings of IEEE INFOCOM*, 2010, pp. 534-542.
21. B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption", in *Public Key Cryptography–PKC*, 2008, vol. 4939, pp. 360–379.