An Efficient PHR Service System Supporting Fuzzy Keyword Search and Fine-grained Access Control

Fatos Xhafa · Jianfeng Wang · Xiaofeng Chen · Joseph K. Liu · Jin Li · Paul Krause

Received: date / Accepted: date

Abstract Outsourcing of personal health record (PHR) has attracted considerable interest recently. It can not only bring much convenience to patients, it also allows efficient sharing of medical information among researchers. As the medical data in PHR is sensitive, it has to be encrypted before outsourcing. To achieve fine-grained access control over the encrypted PHR data becomes a challenging problem. In this paper, we provide an affirmative solution to this problem. We propose a novel PHR service system which supports efficient searching and fine-grained access control for PHR data in a hybrid cloud environment, where a private cloud is used to assist the user to interact with the public cloud for processing PHR data. In our proposed solution, we make use of attribute-based encryption (ABE) technique to obtain fine-grained access control for PHR data. In order to protect the privacy of PHR owners, our ABE is anonymous. That is, it can hide the access policy information in ciphertexts. Meanwhile, our solution can also allow efficient *fuzzy search* over PHR data, which can greatly improve the system usability. We also provide security analysis to show

Fatos Xhafa (*Corresponding author*)
Dept. of Languages and Informatics Systems, Technical University of Catalonia, Spain.

Jianfeng Wang
State Key Laboratory of Integrated Service Networks, Xidian University, China

Xiaofeng Chen
State Key Laboratory of Integrated Service Networks, Xidian University, China

Joseph K. Liu
Institute for Infocomm Research (I^2R), Singapore

Jin Li
School of Computer Science and Educational Software, Guangzhou University, Guangzhou, P.R. China

Paul Krause
Department of Computing, University of Surrey, UK

that the proposed solution is secure and privacy-preserving. The experimental results demonstrate the efficiency of the proposed scheme.

1 Introduction

A personal health record (PHR) is a collection of individual's health related information, which is created and managed by himself. Recently there has been a remarkable upsurge in activity surrounding the adoption of personal health record (PHR) systems for patients [14]. There are many potential benefits that would promote the use of PHRs. PHR offers a large number of trusted health information, data, and knowledge to patients, who can rely on that to easily manage and monitor their health and diseases. PHR can also provide a continuous connection between patients and physicians, which substantially shortens the time to solve problems that may arise. On the other hand, accumulated information from different patients' PHR can help physicians to make better decisions. Despite of various advantages, the PHR data contains very sensitive information. Secure storage and access to PHR is a must in the design of such system.

Cloud computing, an emerging technology in information system industry, makes on-demand computing resources a reality. Nowadays cloud service providers can offer both high-available storage services and massive parallel computing resources at relatively low costs [10]. With the rapid adoption of cloud storage services, more and more PHR data are being centralized into the cloud. By outsourcing their data in the cloud, data owners can obtain high quality data storage services, while reducing the burden of data storage and maintenance. Due to the fact that data owners and cloud server are not in the same trusted domain, in order to securely store PHR data on an untrusted cloud server, it should be encrypted before outsourcing [11]. However, it is intractable to search PHR data in the server efficiently. One of the most popular methods is to selectively retrieve files through keyword-based search instead of retrieving all the encrypted files back, which is known as searchable encryption technology.

Although searchable encryption allows secure and efficient searching among encrypted data, the existing schemes do not suit for cloud computing scenario since they support only exact keyword search. To overcome this drawback, Li et al. [9] proposed a new way to enable fuzzy keyword search over encrypted data by introducing the concept of *edit distance* in the encrypted keywords. However, they have not considered the construction in a multi-user system with different searching privileges.

1.1 Contribution

In this paper, we propose a novel PHR service system that allows efficient search and fine-grained access control to PHR data in a hybrid cloud environment. In our proposed construction, users are able to utilize the private cloud as a proxy to securely deploy their PHR data to the public cloud. Our system mainly consists of two aspects: 1) To achieve secure and fine-grained access control for PHR data, we deploy anonymous attribute-based encryption technique to encrypt a symmetric key which is then used to encrypt the PHR files. Thus it can hide the access policy information in ciphertexts. 2) To improve system usability, PHR data can be efficiently retrieved by exploiting the fuzzy keyword search technique. Based on the constructed fuzzy keyword sets, we

further propose an advanced symbol-based *trie*-traverse search technique for efficiency enhancement.

1.2 Related Work

Searchable encryption is a broad concept that deals with searches in encrypted data. The goal is to outsource encrypted data and be able to conditionally retrieve or query data without having to decrypt all data records [2]. The first practical searchable encryption scheme was proposed by Song et al. [13] in 2000. Later, various searchable encryption schemes [4], [5], [6], [7], [8] have been proposed in recent years. To achieve more efficient search, Goh [8] proposed to use Bloom filters to construct the index for each file. The index makes the search scheme independent of the file encryption. Moreover, the complexity of each search request is roughly proportional to the number of files in the collection. Chang et al. [6] developed a similar per-file index scheme. Curtmola et al. [7] presented the formal security notion of searchable encryption. Furthermore, they proposed similar "index" approaches, where a single encrypted hash table index is built for the entire file collection. In the index table, each entry consists of the trapdoor of a keyword and an encrypted set of related file identifiers. As a complementary approach, Boneh et al. [4] proposed a searchable encryption scheme in asymmetric setting in 2004, where anyone with the public key can encrypt data but only the authorized users with the private key are able to search. Subsequently, Abdalla et al. [1] proposed a novel public-key encryption with temporary keyword search. Compared to symmetric searchable encryption, public key solutions are usually more computational expensive. Note that the above schemes are efficient in general, but they can only support single-keyword queries. Thus they may not be suitable for real-world PHR search applications.

To achieve scalable and fine-grained access control, attribute-based encryption (ABE) [12] has been proposed. Using this technique, differential yet flexible access rights can be assigned to individual users. Specifically, ciphertext-policy attribute-based encryption (CP-ABE) [3] enables data owners to specify an access policy over an universe of attributes and encrypt the data under the access policy with the corresponding public key components. Though ABE can be directly applied to design secure access control, there is an increasing need to protect the privacy of user's access policy (that is, to hide the access policy information) in access control systems. In order to address this problem, anonymous-ABE was introduced in [15]. In an anonymous CP-ABE, a user obtains his attribute secret key and if the attribute set associated with the secret key does not satisfy the access policy in the ciphertexts, the user cannot decrypt and guess what access policy was specified by the data owner. We notice that anonymous CP-ABE is especially suitable for the PHR scenario, which is used to address the privacy-preserving PHR search problem in our work.

1.3 Organization

The rest of paper is organized as follows: Section 2 gives the system overview and security model of the proposed scheme. Section 3 provides the detailed description of our proposed scheme. Section 4 presents the security analysis. In Section 5 we report on our performance analysis. Finally, Section 6 concludes the paper.

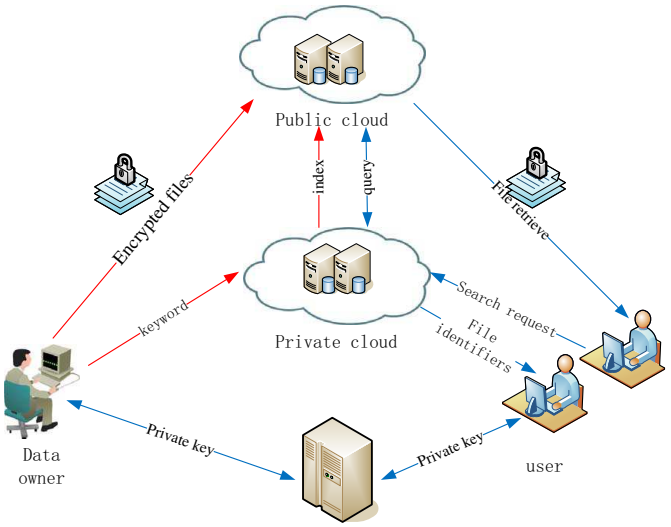


Fig. 1 Architecture for PHR system

2 Problem Formulation

2.1 Overview

We consider a PHR service system in a cloud environment. There are four entities defined in our system: data owners/users, trusted authority (TA), private cloud and public cloud. The owners refer to the patients who create their encrypted PHR data and upload them to the public cloud. Moreover, the owners can specify the ciphertext-policy such that only users whose attributes satisfy the policy are able to decrypt the ciphertext. Users refer to those who want to access the encrypted PHR data, such as family, friends, clinicians and researchers. The TA generates private key for all users. The public cloud stores all encrypted PHRs in their database and performs the search for users. The private cloud is additionally introduced to facilitate users for secure usage of cloud service. In addition, the ciphertext policy is kept hidden. Users are able to check whether their attributes satisfy the ciphertext-policy or not.

More specifically, the data owner stores his encrypted PHR data on the public cloud and shares the files with those users whose attributes satisfy the specific access policy. Since the computing resources at user side are restricted and the public cloud is not fully trusted in practice, the private cloud is able to provide users with an execution environment and an interface between users and the public cloud. The interface offered by the private cloud allows users to securely submit files. Queries can be also securely stored and computed.

The fuzzy keyword search algorithm returns the search results according to the following rules: 1) if the user's searching input exactly matches the pre-set keyword, the server is expected to return the files containing the keyword; 2) if there exist typos and/or format inconsistencies in the searching input, the server will return the closest possible results based on pre-specified similarity semantics. The architecture of PHR system is shown in Fig. 1.

2.2 System Definition

Our system includes seven algorithms, which are defined as follows:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm is run by the TA. It takes a security parameter λ as input and outputs a public key PK and a master key MSK.
- $\mathsf{KeyGen}(PK, MSK, \Omega)$: The key extraction algorithm is run by the TA. It takes a public key PK, an attribute set Ω and a master key MSK as inputs and outputs the a private key SK_Ω.
- $\mathsf{Encrypt}(PK, K, \mathcal{P})$: The encryption algorithm is run by the data owner. It takes a public key PK, a random symmetric encryption key K and an access policy $\mathcal{P}$ as inputs and outputs a ciphertext $CT_\mathcal{P}$.
- $\mathsf{KeywordIndexGen}(\mathcal{P}, FID, W)$: The algorithm is run by the private cloud to generate a keyword index set corresponding to a file identifier FID. It takes a file identifier FID, an access policy $\mathcal{P}$ and a distinct keyword set $W = \{w_i\}$ (which may contain different keywords w_i) as inputs and outputs a set of keyword indexes $\{I_{w_i}\}$ where $w_i \in W$.
- $\mathsf{SearchQueryGen}(\Omega, w)$: The algorithm is run by the private cloud to generate a search query for all fuzzy keywords of a user-input keyword w. It takes attributes Ω and a query keyword w as inputs and outputs a search query SRH.
- $\mathsf{Search}(\mathcal{T}, \mathsf{SRH})$: This algorithm is run by the public cloud to search for files in its database that contain keyword w. It takes a table of keywords $\mathcal{T}$ (which contains all keyword indexes) and a search query SRH as inputs and outputs the corresponding file identifiers $\{FID\}$.
- $\mathsf{Decrypt}(PK, CT_\mathcal{P}, SK_\Omega)$: The decryption algorithm is run by the user. It takes a public key PK, a ciphertext $CT_\mathcal{P}$ which was encrypted under the access policy $\mathcal{P}$ and a private key SK_Ω. It outputs a symmetric encryption key K if Ω satisfies the policy $\mathcal{P}$ (denoted as $\mathcal{P}(\Omega) = 1$), otherwise outputs the error symbol $\perp$.

2.3 Security Model

In our system, we assume both the public cloud and the private cloud are "honest-but-curious". That means the servers will honestly follow our proposed protocol, but try to find out as much secret information in the encrypted PHR data as possible. Users would try to access data files beyond their privileges. To do this, they may collude with other users, or even with the cloud server. Therefore, two kinds of adversaries are considered in our system: 1) external attackers, including the cloud servers and other unauthorized users, who try to learn security information as much as possible; and 2) internal attackers who share their privileges with other users who do not have these privileges. Without loss of generality, we assume the TA is trusted.

Index Privacy: Due to the computation of index and query of the same keyword, we only consider the index privacy. That means the cloud cannot learn additional knowledge of the keyword. It should be noted that the index is only sensitive for the public cloud, where the private cloud is allowed to know the keyword.

IND-sCP-CPA: The goals of an adversary in an anonymous CP-ABE system include extracting information of a plaintext from the ciphertext and distinguishing underlying access policies in ciphertexts, which can be integrated into the following $\mathsf{IND\text{-}sCP\text{-}CPA}$ game involving an adversary $\mathcal{A}$ and a challenger $\mathcal{S}$.

1. Initial: The adversary $\mathcal{A}$ commits to the challenge ciphertext policies W_0^*, W_1^*.
2. Setup: The challenger $\mathcal{S}$ chooses a sufficiently large security parameter λ, and runs the Setup algorithm to get a master key SK and the corresponding public key PK. It retains MSK and gives PK to $\mathcal{A}$.
3. Phase 1: In addition to hash queries, the adversary $\mathcal{A}$ issues a polynomially bounded number of queries to the following key generation oracle:
   - KeyGen oracle $\mathcal{O}_{KeyGen}$: The adversary $\mathcal{A}$ submits an attribute list Ω, if $(\Omega \models W_0^* \wedge \Omega \models W_1^*)$ or $(\Omega \not\models W_0^* \wedge \Omega \not\models W_1^*)$ (we use the notation $L \models W$ to represent the fact that L satisfies W, and the case of L does not satisfy W is denoted by $L \not\models W$), the challenger $\mathcal{S}$ gives $\mathcal{A}$ the secret key SK_Ω. Otherwise, it outputs $\perp$.
4. Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length messages M_0, M_1 from the message space, on which it wishes to be challenged with respect to W_0^* and W_1^*. It is required that $M_0 = M_1$ if any secret key on Ω satisfying $\Omega \models W_0^* \wedge \Omega \models W_1^*$ has been queried. The challenger $\mathcal{S}$ randomly chooses a bit $\nu \in \{0, 1\}$, computes $CT_{W_\nu^*} = \mathsf{Encrypt}(PK, M_\nu, W_\nu^*)$ and sends $CT_{W_\nu^*}$ to $\mathcal{A}$.
5. Phase 2: The same as Phase 1.
6. Guess: The adversary $\mathcal{A}$ outputs a guess bit $\nu' \in \{0, 1\}$ and wins the game if $\nu' = \nu$.

The advantage of an adversary $\mathcal{A}$ in the IND-sCP-CPA game is defined as

$$\mathsf{Adv}_{\text{CP-ABE}}^{\text{IND-sCP-CPA}}(\mathcal{A}) = \left|\Pr[\,\nu' = \nu\,] - \tfrac{1}{2}\right|.$$

3 Constructions of Fuzzy Keyword Search with Anonymous-ABE

3.1 Preliminaries

We first describe some preliminaries required in our system.

Definition 1 (Edit Distance) Edit distance is a measure of similarity between two strings. The edit distance $ed(w_1, w_2)$ between two words w_1 and w_2 is the minimum number of operations required to transform one to the other. There are three primitive operations. 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single character into a word. Given a keyword w, we let $\mathcal{W}_{w,d}$ denote the set of keywords w' satisfying $ed(w, w') < d$ for a certain integer d.

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

In our construction, we consider access structures consisting of AND-gates supporting multi-value attributes.

Definition 3 (Decision BDH) Given $g, g^x, g^y, g^z \in G_1$ for random exponents $x, y, z \in \mathbb{Z}_p^*$ and $Z \in \mathbb{G}_T$, the DBDH assumption says that no polynomial-time algorithm can decide whether $Z = e(g, g)^{xyz}$ or Z is a random element chosen from $\mathbb{G}_T$ with non-negligible probability.

Definition 4 (Decision Linear) Given $g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, Z \in \mathbb{G}_1$ for random exponents$z_1, z_2, z_3, z_4 \in \mathbb{Z}_P$, The D-Linear assumption is that no polynomial-time algorithm can decide whether $Z = g^{z_3+z_4}$ or Z is a random element chosen from $\mathbb{G}_1$ with non-negligible probability.

Definition 5 (Bilinear Pairings) Let $\mathbb{G}_1, \mathbb{G}_2$ be the cyclic groups of prime order p, let g be a generator of $\mathbb{G}_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a map with the following properties.

1. Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$, $a, b \in \mathbb{Z}_p$.
2. Non-degeneracy: There exist $x, y \in \mathbb{G}_1$ such that $e(x, y) \neq 1$.
3. Computable: For all $x, y \in \mathbb{G}_1$, $e(x, y)$ has to be computable in an efficient manner.

3.2 The Proposed Construction

Initialization. In our proposed system, we construct fuzzy keyword set with the wildcard-based method [9]. Different from the straightforward method of listing all the variants of the keywords, we use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of keyword w with edit distance d can be denoted as $\mathcal{W}_{w,d} = \{w'_{w,d}\}$, where $w'_{w,d}$ denotes a fuzzy keyword of w with d wildcards. For example, for the keyword *cat* with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as $\mathcal{W}_{cat,1} = \{cat, \star cat, \star at, c\star at, c\star t, ca\star t, ca\star, cat\star\}$. Furthermore, we define a map as follow: $f : \Omega \to P_\Omega$, where Ω and P_Ω represent user's attributes and the corresponding access policy respectively.

- $\mathsf{Setup}(1^\lambda)$: Let $\mathbb{G}, \mathbb{G}_T$ be cyclic multiplicative groups of prime order p, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Define two hash functions $H : \{0,1\}^* \to \mathbb{G}$ and $h : \{0,1\}^* \to \{0,1\}^l$ for some security parameter l. Assume there are n attributes in universe. Let $\mathcal{U} = \{u_1, \cdots, u_n\}$ denotes the universe of attributes. Each attribute has multiple values. Let $\mathcal{S}_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,n_i}\}$ be the multi-value set for u_i. The TA chooses $y \in_R \mathbb{Z}_p$, $g_1, g_2 \in_R \mathbb{G}$, then computes $Y = \hat{e}(g_1, g_2)^y$. The system public key is published as $PK = (g, g_1, g_2, Y)$ and the master key MSK is y.
- $\mathsf{KeyGen}(PK, MSK, \Omega)$: To generate the attribute secret key for the user who has the attribute set $\Omega = \{\Omega_1, \Omega_2, \cdots, \Omega_n\}$. TA randomly picks $r_1, r_2, \cdots, r_{n-1} \in_R \mathbb{Z}_p$ and computes $r_n = y - \sum_{i=1}^{n-1} r_i \mod p$. Furthermore, it randomly chooses $r \in_R \mathbb{Z}_p$ and $\{\hat{r}_i, \lambda_i, \widehat{\lambda}_i \in_R \mathbb{Z}_p\}_{1 \le i \le n}$, sets $\hat{r} = \sum_{i=1}^{n} \hat{r}_i$, and computes $\left(D_0, D_{\Delta,0}\right) = [g_2^{y-\hat{r}}, g_1^r]$. For $1 \le i \le n$, the TA computes

$$\left(D_{\Delta,i}, D_{i,0}, D_{i,1}, \widehat{D}_{i,0}, \widehat{D}_{i,1}\right) =$$
$$\left(g_2^{\hat{r}_i} H(i||v_{i,k_i})^r, g_2^{\lambda_i}, g_1^{r_i} H(0||i||v_{i,k_i})^{\lambda_i}, g_1^{\widehat{\lambda}_i}, g_2^{r_i} H(1||i||v_{i,k_i})^{\widehat{\lambda}_i}\right),$$

  where $\Omega_i = v_{i,k_i}$. The secret key is

$$SK_\Omega = \left(D_0, D_{\Delta,0}, \{D_{\Delta,i}, D_{i,0}, D_{i,1}, \widehat{D}_{i,0}, \widehat{D}_{i,1}\}_{1 \le i \le n}\right).$$

- $\mathsf{Encrypt}(PK, K, \mathcal{P})$: To encrypt a symmetric key $K \in \mathbb{G}_T$ under a ciphertext policy $\mathcal{P} = \{P_1, \cdots, P_n\}$, data owner chooses $s, s', s'' \in_R \mathbb{Z}_p$, and computes $\widetilde{C} = KY^s$, $C_\Delta = \hat{e}(g, g)^s Y^{s'}$, $C_0 = g^s$, $\widehat{C}_0 = g_1^{s'}$, $C_1 = g_2^{s''}$, $\widehat{C}_1 = g_1^{s-s''}$. Then for $1 \le i \le n$ and $1 \le t \le n_i$, he computes $\left(C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\right)$ as follows:

1. If $v_{i,t} \in P_i$, then $\left[C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\right] =$

$$\left(H(i||v_{i,t})^{s'}, H(0||i||v_{i,t})^{s''}, H(1||i||v_{i,t})^{s-s''}\right).$$

2. If $v_{i,t} \notin P_i$, then $\left[C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\right]$ are random elements in $\mathbb{G}$.

Then the ciphertext of K with respect to P is

$$CT_{\mathcal{P}} = \langle C_\Delta, C_0, \widehat{C}_0, \widetilde{C}, C_1, \widehat{C}_1, \{\{C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\}_{1\le t\le n_i}\}_{1\le i\le n}\rangle.$$

– KeywordIndexGen($\mathcal{P}$, FID, W): After running $Encrypt(PK, K, \mathcal{P})$, the data owner uploads the ciphertext $CT_{\mathcal{P}}$ to the public cloud and obtains the file identifier FID. Then, the data owner sends $(\mathcal{P}, FID, W)$ to the private cloud to generate indexes for the set of keywords W. For every $w_i \in W$, the private cloud computes the indexes $\{T_{w_i',\mathcal{P}} = h(\mathcal{P}, w_i')\}_{w_i' \in \mathcal{W}_{w_i,d}}$.[1] The set of keyword indexes denoted as $\left\{I_{w_i} = \left\{\{T_{w_i',\mathcal{P}}\}_{w_i' \in \mathcal{W}_{w_i,d}}, FID\right\}\right\}_{w_i \in W}$ is outsourced to the public cloud. The public cloud stores the keyword index set into the table $\mathcal{T}$.
– SearchQueryGen(Ω, w): To search with keyword w, the user sends (Ω, w) to the private cloud, who gets the corresponding access policy $P_\Omega \leftarrow f(\Omega)$ and generates the search query as $\{\mathsf{SRH} = h(P_\Omega, w'_{w,d})\}_{w'_{w,d} \in \mathcal{W}_{w,d}}$.[2] Next, the private cloud sends the search queries $\{\mathsf{SRH}\}$ to the public cloud.
– Search($\mathcal{T}, \{\mathsf{SRH}\}$): Upon receiving the search queries $\{\mathsf{SRH}\}$, the public cloud compares them with the indexes in the table $\mathcal{T}$ with the search queries and returns the corresponding matched file identifiers $\{FID\}$ to the private cloud.
– Decrypt($PK, CT_{\mathcal{P}}, SK_\Omega$): The ciphertext $CT_{\mathcal{P}}$ is tested and decrypted by a user with secret key SK_Ω as follows:

1. Matching Phase: The user checks whether $\Omega \models \mathcal{P}$. To be specific, $\Omega \models \mathcal{P}$ if and only if Equality (1) holds:

$$\frac{C_\Delta}{\hat{e}(g, C_0)} = \frac{\hat{e}\left(\widehat{C}_0, D_0 \prod_{i=1}^n D_{\Delta,i}\right)}{\hat{e}\left(\prod_{i=1}^n C_{i,t,\Delta}, D_{\Delta,0}\right)}, \tag{1}$$

where $\Omega_i = v_{i,t}$. If $\Omega \not\models \mathcal{P}$, it returns $\perp$. Otherwise, it initiates the Decryption Phase.

2. Decryption Phase: The user decrypts and computes the symmetric K as follows:

$$K = \frac{\widetilde{C} \prod_{i=1}^n \hat{e}\left(C_{i,t,0}, D_{i,0}\right) \hat{e}\left(\widehat{C}_{i,t,0}, \widehat{D}_{i,0}\right)}{\prod_{i=1}^n \hat{e}\left(C_1, D_{i,1}\right) \hat{e}\left(\widehat{C}_1, \widehat{D}_{i,1}\right)},$$

where $\Omega_i = v_{i,t}$.

[1] We assume that the edit distance d is implicitly defined by the private cloud.
[2] Here we assume that the edit distance d is implicited defineded by the private cloud. If it is dynamically set by the user, SearchQueryGen further takes d as an additional input.

Indeed, for a valid ciphertext that satisfies $\Omega \models \mathcal{P}$, we have

$$\begin{aligned}
&\frac{\hat{e}\left(\widehat{C}_0, D_0 \prod_{i=1}^{n} D_{\Delta,i}\right)}{\hat{e}\left(\prod_{i=1}^{n} C_{i,t,\Delta}, D_{\Delta,0}\right)} \\
&= \frac{\hat{e}\left(g_1^{s'}, g_2^{y-\hat{r}} \prod_{i=1}^{n}\left(g_2^{\hat{r}_i} H(i||v_{i,t})^{r}\right)\right)}{\hat{e}\left(\prod_{i=1}^{n} H(i||v_{i,t})^{s'}, g_1^{r}\right)} \\
&= \hat{e}\left(g_1^{s'}, g_2^{y-\hat{r}}\left(\prod_{i=1}^{n} g_2^{\hat{r}_i}\right)\right) = \hat{e}\left(g_1^{s'}, g_2^{y-\hat{r}} g_2^{\hat{r}}\right) \\
&= \hat{e}\left(g_1^{s'}, g_2^{y}\right) = \hat{e}\left(g_1, g_2\right)^{y s'} \\
&= \frac{C_{\Delta}}{\hat{e}(g, C_0)}.
\end{aligned}$$

On the other hand, the plaintext K can be successfully recovered as follows:

$$\begin{aligned}
&\frac{\widetilde{C} \prod_{i=1}^{n} \hat{e}\left(C_{i,t,0}, D_{i,0}\right) \hat{e}\left(\widehat{C}_{i,t,0}, \widehat{D}_{i,0}\right)}{\prod_{i=1}^{n} \hat{e}\left(C_1, D_{i,1}\right) \hat{e}\left(\widehat{C}_1, \widehat{D}_{i,1}\right)} \\
&= \frac{\widetilde{C} \prod_{i=1}^{n} \hat{e}\left(H(0||i||v_{i,t})^{s''}, g_2^{\lambda_i}\right) \hat{e}\left(H(1||i||v_{i,t})^{s-s''}, g_1^{\widehat{\lambda}_i}\right)}{\prod_{i=1}^{n} \hat{e}\left(g_2^{s''}, g_1^{r_i} H(0||i||v_{i,t})^{\lambda_i}\right) \hat{e}\left(g_1^{s-s''}, g_2^{r_i} H(1||i||v_{i,t})^{\widehat{\lambda}_i}\right)} \\
&= \frac{K Y^{s}}{\prod_{i=1}^{n} \hat{e}\left(g_2^{s''}, g_1^{r_i}\right) \hat{e}\left(g_1^{s-s''}, g_2^{r_i}\right)} \\
&= \frac{K \hat{e}(g_1, g_2)^{y s}}{\prod_{i=1}^{n} \hat{e}\left(g_1, g_2\right)^{s r_i}} = \frac{K \hat{e}(g_1, g_2)^{y s}}{\hat{e}\left(g_1, g_2\right)^{y s}} = K.
\end{aligned}$$

3.3 Efficient Fuzzy Search Scheme based on Symbol-based *Trie*

To enhance the search efficiency, we utilize a symbol-based *trie* to build the index and store it in the public cloud. We construct, a multi-way tree for storing the fuzzy keyword set $\{w'_{w,d}\} \in \mathcal{W}_{w,d}$ over a finite symbol set. Each index (hash value) is divided into l/ℓ parts and each part can be denoted by ℓ bits where ℓ is some security parameter. Assume $\Delta = \{\alpha_i\}$ is a predefined symbol set, where the number of different symbols is $|\Delta| = 2^{\ell}$. Thus each part of the search query can be denoted as a specific symbol.

The improved keyword search scheme works as follows:

- KeywordIndexGen($\mathcal{P}$, FID, W): Upon receiving $(\mathcal{P}, FID, W)$ from the data owner, the private cloud generates the index with the following steps.
  1) The private cloud computes each $T_{w'_i,\mathcal{P}} = h(\mathcal{P}, w_i^{'})$ for each $w'_i \in \mathcal{W}_{w_i,d}$ and each $w_i \in W$. It then divides each index into $\alpha_{i_1}, \cdots, \alpha_{i_{l/\ell}}$ where $|\alpha_{i_j}| = 2^{\ell}$, for $j \in [1, l/\ell]$.
  2) The private cloud builds a tree T_W covering all fuzzy keywords $w_i \in W$ based on the predefined symbol set Δ. In the initialization phase, a root of the T_W denoted as Φ is created. Then the first symbol α_1 of the sequence is added into T_W as the child of the root, if there is no existing node equal to the symbol. Subsequently, the current node is moved to the node of the child of the root. The algorithm is carried out recursively. When the last symbol is performed, the corresponding identifier FID is attached the last node as a leaf node.

- $\mathsf{SearchQueryGen}(\Omega, w)$: To search with keyword w, the user sends (Ω, w) to the private cloud, who transforms Ω into the corresponding access policy P_Ω and generates the trapdoor set $\{\mathsf{SRH}\}_{w'_{w,d} \in \mathcal{W}_{w,d}}$. Next, the policy translates the search query set $\{\mathsf{SRH}\}$ into symbol sequence and sends them to the public cloud.
- $\mathsf{Search}(T_W, \{\mathsf{SRH}\})$: Upon receiving the search request, the public cloud performs search in the index tree T_W (instead of the table $\mathcal{T}$ in previous setting) and returns all the matched file identifiers $\{FID\}$ to the private cloud.

4 Security Analysis

Data Privacy: The file contents are separately encrypted with symmetric cryptography. By using a cryptographic strong cipher, it is sufficient to assume that encrypted files leak zero information. This implies even if the PHR database is compromised, the adversary learns nothing about the PHR data contained in the server without the relevant symmetric keys. Since the system requires a trusted authority to issue private keys, the security of the system relies on the trust and reliability of the authority. Besides, since privacy-preserving query can be understood as a collection of $l-$length strings, the confidentiality of which are guaranteed by the underlying hash function.

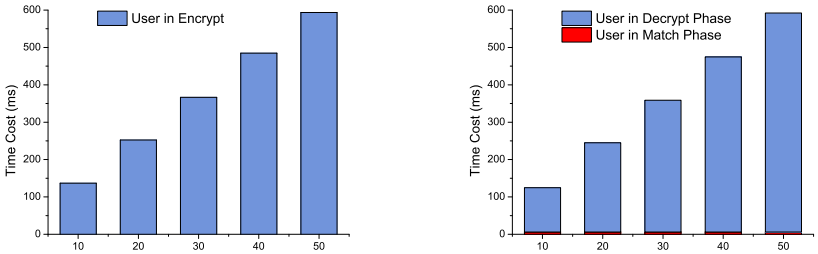
Search Privacy: The proposed scheme is based on the original fuzzy keyword search scheme, which is proved to have searchable privacy under random oracle model [9]. That is, any computationally bound adversary does not obtain the underlying information of keyword from the index. In other words, the adversary cannot obtain a search query that distinguishes two ciphertexts of keywords unless it obtains the key K. Thus, the searchable privacy is obtained. Note that the private cloud is allowed to know the plaintext of keyword. As the private cloud in practice may be maintained by some organizations themself, the leakage of keyword information is innocuous.

Secure Fine-grained Access for PHR Data: In our system it is sufficient to assume that the encrypted PHR data is accessed securely as we deploy ABE mechanism. In other words, any attackers cannot learn additional information about the underlying trapdoors of any keyword. Specifically, for the external attacker, such as the cloud server, they do not have the attributes that satisfy the access policy, thus they cannot get any information about the user attribute key. As a result, they are not able to decrypt any encrypted PHR data. On the other hand, for the internal attacker, such as a user who shares the privileges with others without such ones, they may combine their attributes for obtaining additional privilege that they do not have. Due to a secure ABE scheme can resist the collusion attack, no user can combine their attributes to obtain additional privilege.

Since the access privilege need to be protected in some scenarios, we adopt anonymous ABE scheme in this work. The proposed scheme is proven to be selective ciphertext policy and chosen plaintext secure under the Decisional Bilinear Diffie-Hellman assumption and the Decisional Linear assumption, which is described in [16]. For more efficient decryption, a *matching phase* is added before the *decryption phase*. It can perform the matching test if the attribute private key matches the hidden attributes policy in ciphertexts without decryption.

5 Performance Analysis

To facilitate the description, we introduce some marks. We denote $|\Omega|$ as the number of user attributes, Exp and P denote the exponentiation multiply and pairing operation respectively. In the proposed construction, TA runs the setup and keygen algorithm, In setup, the need just one Exp and P operation, which is very efficient. To generate the privacy key for some user, the computations of keygen is $8*|\Omega|+2$ Exp. Consider that the process is just once for each user, it is can be acceptable. The keywordindexgen, searchquerygen and search algorithms are run by the private public cloud. Consider that the high computation capacity of cloud, it will not influence the efficiency of system, so we omit it in our discuss.



(a) User in Encrypt

(b) User in Decrypt

Fig. 2 Efficiency Analysis of Encrypt and Decrypt

The encrypt and decrypt algorithms run by the data owner user. It should be pointed out that the proposed construction enjoys the desirable property of match-then-decrypt, and hence can be used to alleviate the computation overload of users. In order to precisely measure the overhead of user side, all experiments were performed on the same machine with Intel Core 3 i5-3470 3.20 GHz CPU and 4G memory running Linux. In experiments, we assume multi-value of each attribute $n_i = 3$ and vary the number of attributes $|\Omega|$ in the universe from 10 to 50.

Fig 2(a) shows the overload of encrypt at user side. From this figure, we find that the overload grows linearly as the number of attributes. This is because the data owner must generate increased number of variables of ciphertext.

The efficiency evaluation of decrypt algorithm is shown in Fig. 2(b). It can clearly be seen that the time cost in matching phase keeps constant with the variation of attribute number Ω, which is far less than the overload in decrypt phase. That is, by introducing the matching phase, user can efficiently decide before full decryption whether the hidden policy in a ciphertext matches his attributes, many unnecessary decryption operations are saved. The result is that our construction is more suitable to be applied for the resource constrained environment.

6 Conclusion

In this paper, we investigated the fuzzy search and fine-grained access control of personal health record in cloud environment. In our construction, the user can use different access privileges to control data access through anonymous attribute-based encryption scheme. In addition, the access privilege is hidden in the ciphertext so that the user privacy is also protected. We further exploit a fuzzy keyword search mechanism. Users can search the ciphertext for some keywords, without needing to decrypt all ciphertexts in the database. We also suggested a symbol-tree to realize the fuzzy keyword search more efficiently. Through security and efficiency analysis, we show that the proposed solution is secure and privacy-preserving, while it can achieve efficient fuzzy keyword search and fine-grained access control over the encrypted PHR data.

Acknowledgement

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 61272455 and 61100224), and Natural Science Foundation of Guangdong Province (Grant No. S2013010013671), and the Fundamental Research Funds for the Central Universities (K50511010001 and JY10000901034) and China 111 Project (No. B08038).

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Proceedings of Advances in Cryptology - CRYPTO 2005. pp. 205–222 (2005)
2. Agudo, I., Nuñez, D., Giammatteo, G., Rizomiliotis, P., Lambrinoudakis, C.: Cryptography goes to the cloud. In: Lee, C., Seigneur, J.M., Park, J., Wagner, R. (eds.) Secure and Trust Computing, Data Management, and Applications, Communications in Computer and Information Science, vol. 187, pp. 190–197. Springer (2011)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007. pp. 321–334 (may 2007)
4. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Proceedings of Advances in Cryptology - EUROCRYPT 2004. pp. 506–522. Springer, Interlaken, Switzerland (2004)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th Theory of Cryptography Conference. vol. 4392, pp. 535–554. Springer, Amsterdam, The Netherlands (2007)
6. Chang, Y., Mitzenmacher, M.: Privacy preserving keyword searches on remote encypted data. In: Proceedings of the 3rd Applied Cryptography and Network Security. pp. 391–421. New York, NY, USA (2005)
7. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definition and efficient constrcutions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. pp. 79–88. ACM, Alexandria, Virginia, USA (2006)
8. Goh, E.: Secure indexes. Report 2003/216, Cryptology ePrint Archive (2003), http://eprint.iacr.org/2003/216
9. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K.and Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of the 29th IEEE International Conference on Computer Communications(INFOCOM'10). pp. 441–445. IEEE, San Diego, CA, USA (2010)

10. Li, J., Li, J., Chen, X., Jia, C., Liu, Z.: Efficient keyword search over encrypted data with fine-grained access control in hybrid cloud. In: Proceedings of the 6th International Conference on Network and System Security(NSS'12). Lecture Notes in Computer Science, vol. 7645, pp. 490–502. Springer (2012)
11. Lu, Y., Tsudik, G.: Privacy-preserving cloud database querying. Journal of Internet Services and Information Security 1(4), 5–25 (2011)
12. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer Berlin / Heidelberg (2005)
13. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy. pp. 44–55. IEEE, Berkeley, California, USA (2000)
14. TANG, P.C., ASH, J.S., BATES, D.W., OVERHAGE, J.M., SANDS, D.Z.: Personal health records: Definitions, benefits, and strategies for overcoming barriers to adoption. J Am Med Inform Assoc 13(2), 121–126 (2006)
15. Yu, S., Ren, K., Lou, W.: Attribute-based content distribution with hidden policy. In: Secure Network Protocols, 2008. NPSec 2008. 4th Workshop on. pp. 39–44 (2008)
16. Zhang, Y., Chen, X., Li, J., Wong, D.S., Li, H.: Anonymous attribute-based encryption supporting efficient decryption test. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security(ASIACCS'13). pp. 511–516. ACM, Hangzhou, China (2013)