FSSR: Fine-Grained EHRs Sharing via Similarity-Based Recommendation in Cloud-Assisted eHealthcare System

Cheng Huang
School of Electrical and Electronic Engineering
Nanyang Technological Univeristy
50 Nanyang Avenue, 639798, Singapore
huangcheng@ntu.edu.sg

Rongxing Lu
School of Electrical and Electronic Engineering
Nanyang Technological Univeristy
50 Nanyang Avenue, 639798, Singapore
rxlu@ntu.edu.sg

Hui Zhu
School of Telecommunications Engineering
Xidian University
Xi'an, 710126, China
zhuhui@xidian.edu.cn

Jun Shao
School of Computer and Information Engineering
Zhejiang Gongshang University
Hangzhou, 310018, China
chn.junshao@gmail.com

Xiaodong Lin
Faculty of Business and Information Technology
University of Ontario Institute of Technology
Oshawa, ON, Canada
xiaodong.lin@uoit.ca 6th. author

Hui Li
School of Telecommunications Engineering
Xidian University
Xi'an, 710126, China
lihui@mail.xidian.edu.cn

ABSTRACT

With the evolving of ehealthcare industry, electronic health records (EHRs), as one of the digital health records stored and managed by patients, have been regarded to provide more benefits, especially considering that patients can easily and conveniently share health records with doctors and build up a complete picture of their health. However, due to the confidentiality and sensitivity of EHRs, how to guarantee the security and privacy of EHRs when sharing becomes one of most important issues concerned by patients. To tackle these privacy challenges such as how to make a fine-grained access control on the shared EHRs, how to keep the confidentiality of EHRs stored in cloud, how to audit EHRs for patients and how to find the suitable doctors, in this paper, we propose a new fine-grained EHRs sharing scheme via similarity-based recommendation accelerated by Locality Sensitive Hashing (LSH) in cloud-assisted ehealthcare system, called FSSR, which allows a patient to securely share his/her EHRs with some suitable doctors under fine-grained privacy access control. In addition, we also conduct extensive simulations by developing a prototype of FSSR, and the performance evaluations demonstrate the FSSR's effectiveness in terms of computational cost, storage and communication cost while minimizing the privacy disclosure.

AsiaCCS '16 Xi'an, Shaanxi China

DOI: 10.475/123_4

CCS Concepts

•**Computer systems organization** → **Embedded systems;**

Keywords

Privacy-preserving, fine-grained access control, EHRs sharing, locality sensitive hashing

1. INTRODUCTION

With the development of technology, paper-based records are gradually being phased out in ehealthcare system due to the inconvenient storage and management, while digitization and virtualization are creating a whole market with the evolving of global ehealthcare industry [1]. Thus, electronic health records (EHRs), as one of the digital health records stored and managed by patients, have been regarded to provide more benefits [2], some of which include improving care coordination, reducing cost, reducing workflows and enhancing health care quality. Especially, patients can easily and conveniently share their health records with doctors through internet and build up a complete picture of their health.

However, considering EHRs sharing between patients and doctors in the real-world EHR systems, how to guarantee the security and privacy of EHRs becomes one of most important challenges concerned by patients. As EHRs are confidential and sensitive, if these records end up in the hands of a person who is not privy to the information, the consequences can be overwhelming. Breach of EHRs may leak patients' sensitive health information, and could further lead to some serious consequences such like identity theft, which can destroy a person's finances, credit and reputation [3]. From a privacy law perspective, to handle the privacy and security issues of EHRs sharing, many regulations have been enacted by various governments, and the most widely used regulations include the Health Insurance Portability and Ac-

countability Act (HIPAA) and the European Data Protection Directive 95/46/EC [4].

In addition to the aforementioned laws, plenty of secure EHRs sharing schemes [5–9] have also been proposed by researchers in recent years from the view of techniques, and some of them are based on attribute-based encryption (ABE) techniques [10], to realize a fine-grained access control on EHRs. The access policy for each EHR file may look like $physician \wedge internalmedicine \wedge hospitalA$ [5], which means only hospital A's physician, whose specialty is internal medicine, can obtain the EHR file. Nevertheless, we believe these schemes are not flexible and fine-grained enough. On the one hand, if patients want to modify/update the policy (e.g sharing the EHR with hospital B's physicians), they have to encrypt the same EHR with new policy again, which is absolutely not flexible. On the other hand, the access policy allows all physicians of internal medicine in hospital A to obtain the EHR, which means all physicians, no matter authorized or unauthorized by the patient (the owner of EHR), can obtain the EHR. It seems like sharing EHRs with all physicians is acceptable, but according to TrustWave's 2015 security health check report [11], enticed by potential high illegal earnings [12], more and more insider attackers (e.g., malicious doctors and nurses) may steal patients' EHRs for selling, so sharing EHRs with the above access policy is dangerous and not fine-grained enough. Accordingly, some improved techniques like attribute-based proxy re-encryption (ABPRE) [9], are used to provide a secure and flexible EHRs sharing approach, but they still have some drawbacks such as high computational cost at patient side when updating policies.

Hence, inspired by the technique called attribute-based conditional proxy re-encryption (ABCPRE) [13], we consider a new privacy-preserving EHRs sharing scheme to achieve a more flexible and fine-grained access control on patients' EHRs. Moreover, we also consider that, before sharing their EHRs, patients should firstly offer their demands and locate some suitable and highly qualified shared targets (doctors). In fact, the requirements of finding a better doctor are necessary and emergent by many patients, and some applications such as BetterDoctor [14], have already been developed. Obviously, when patients share EHRs with some suitable and trustable doctors according to their information, the exposing risks of shared EHRs will decrease a lot. Under this circumstance, doctors need to upload their information, and patients can match some suitable ones through all doctors' information. Considering the privacy of patients and doctors, the information of doctors is sensitive and should be privacy-preserving, and patients' demands should also be privacy-preserving when matching doctors' information.

Contribution. In this paper, to address the privacy issues for EHRs sharing, we propose a new fine-grained EHRs sharing scheme via similarity-based recommendation, called FSSR, which allows a patient to securely share his/her EHRs with some suitable doctors under fine-grained privacy access control. To make our scheme more scalable, we consider a cloud server in ehealthcare system to store data and accomplish most of the heavy computations. Specifically, the contribution of this paper are four-fold.

- Firstly, to help patients locate some suitable doctors, we propose a privacy-preserving doctor recommendation protocol based on similarity comparison of Euclidean distance. With this protocol, patients can easily and efficiently locate some doctors which meet their demands while minimizing the privacy disclosure.
- Secondly, to achieve a more fine-grained access control on patients' EHRs, we propose a fine-grained privacy-preserving EHRs sharing protocol based on attribute-based conditional proxy re-encryption [13]. With this protocol, patients can achieve a flexible and fine-grained privacy access control when sharing EHRs.
- Thirdly, through combining the above two protocols, we present a fine-grained EHRs sharing scheme via similarity-based recommendation in cloud-assisted ehealthcare system, called FSSR. In addition, we also analyze the security of our proposed FSSR scheme.
- Finally, we develop a prototype for FSSR and implement all protocols to demonstrate its substantial improvement in terms of computational cost, storage and communication cost. The results show that our scheme is efficient at patient/doctor side and most of heavy computations are delegated to the cloud server.

The remainder of this paper is organized as follows. In Section 2, we introduce the system model, security model, and design goal. In Section 3, we recall some preliminaries. Then, we present a fine-grained EHRs sharing (FSSR) scheme via similarity-based recommendation in cloud-assisted eHealthcare system in Section 4 and its security analysis in Section 5. In Section 6, we evaluate the performance of FSSR in terms of computational cost, storage and communication cost. We discuss the related work in Section 7. Finally, we draw our conclusions in Section 8.

2. MODELS AND DESIGN GOAL

In this section, we formalize the system model and security model, and identify our design goal as well.

2.1 System Model

In our system model, we mainly consider four types of entities, namely a healthcare center (HC), a cloud server (CS), a group of patients $P = \{P_1, P_2, ..., P_N\}$ and doctors $D = \{D_1, D_2, ..., D_M\}$, as shown in Figure 1.

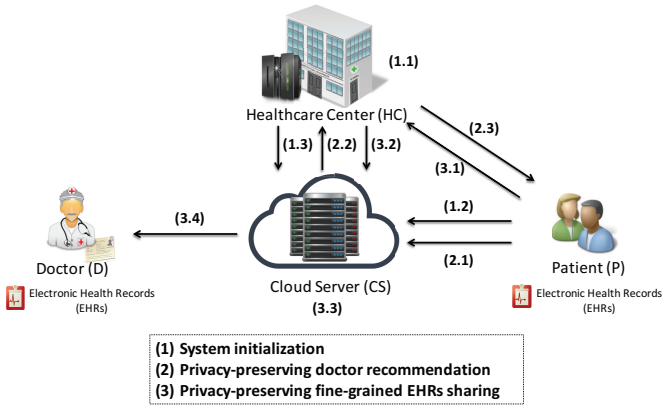


Figure 1: System model under consideration

Healthcare Center (HC): HC is a core and fully trustable entity, which is a legal institution authorized by government.

HC is responsible for initializing the whole system, which means all patients and doctors need to register themselves in HC. Meanwhile, in order to help patients find the suitable doctors, HC will maintain and publish a criterion which defines n entries used to characterize a doctor. According to the criterion, one doctor's information can be represented as a binary vector $\overrightarrow{C} = \{\phi_1, \phi_2, ..., \phi_n\}$ in a n-dimensional space, where each $\phi_i \in \overrightarrow{C}$ indicates whether this doctor has this character, i.e., $\phi_i = 1$ if the doctor has this character, and $\phi_i = 0$ otherwise. By combining all registered doctors' information, HC can create a *doctor information database* and outsource this database to CS. Moreover, HC can also help patients to audit the logs of EHRs sharing.

Cloud Server (CS): CS is a powerful entity. For one thing, CS can help patients and HC to store data; for another, CS can help HC, patients and doctors to achieve most of heavy computations in our FSSR scheme.

Patient ($P = \{P_1, P_2, ..., P_N\}$): each patient $P_i \in P$ possesses his/her own EHRs, and stores the encrypted EHRs in CS. With the help of HC and CS, P_i can share his/her EHRs with the doctors who meets his/her specific demands. That is, according to the criterion published by HC, the demand of a patient can be represented as a n-dimensional vector $\overrightarrow{C} = \{\phi_1, \phi_2, ..., \phi_n\}$, which can be used to match the suitable doctors by similarity comparison of Euclidean distance.

Doctor ($D = \{D_1, D_2, ..., D_M\}$): each doctor $P_i \in P$ knows his/her information and uploads the information based on the criterion published by HC, $\overrightarrow{C} = \{\phi_1, \phi_2, ..., \phi_n\}$ to HC when registering. After authorized by a patient, the doctor can get some shared EHRs of the patient under fine-grained access control.

Concretely, there are three phases in our FSSR scheme: (1) system initialization, (2) privacy-preserving doctor recommendation, and (3) fine-grained privacy-preserving EHRs sharing.

- *System initialization:* (1.1) **SystemSetup:** HC firstly activates the system by generating key materials, and all doctors and patients registered themselves in HC; (1.2) **PatientStore:** patients encrypt their EHRs and store the encrypted files in CS; (1.3) **HCOutsource:** HC encrypts and outsources the *doctor information database* to CS. Finally, HC publishes some parameters as the system-wide information.

- *Privacy-preserving doctor recommendation:* (2.1) **PatientQuery:** a patient generates his/her demand and sends the query to CS; (2.2) **DoctorMatch:** CS helps the patient to match the suitable doctors and sends the results to HC; (2.3) **DoctorRecommend:** HC recommends some suitable doctors to the patient.

- *Fine-grained privacy-preserving EHRs sharing:* (3.1) **PatientAuth:** according to the recommended doctors, the patient authorizes one doctor to share EHRs under fine-grained access control; (3.2) **RekeyGen:** HC generates the re-encryption key according to the patient's authorization, and sends the re-encryption key to CS; (3.3) **EHRReEnc:** CS re-encrypts this patient's EHRs using his/her re-encryption key; (3.4) **DoctorDec:** the authorized doctor decrypts the patient's encrypted EHRs obtained from CS.

Different from the previous models [5, 15], we consider a new entity HC, which needs to participate in the process of sharing. The advantages of introducing HC involve two aspects. Firstly, HC can help patients to find the suitable doctors due to its doctor information database and can help patients to achieve authorization when generating rekey. Secondly, based on the sharing logs stored in HC, patients can easily achieve auditing if needed. Note that, while performing privacy-preserving EHRs sharing, HC only needs to generate rekey once, and all EHRs of patients can be shared through CS without the attendance of HC later.

2.2 Security Model

Consider the existing security models when dealing with cloud server [5, 15], we assume that CS is *honest-but-curious*, which means CS will faithfully follow the proposed scheme, but could launch passive attacks to obtain sensitive information about patients and doctors as much as possible. Specifically, CS may try to decrypt the encrypted EHRs or encrypted doctors' information stored in its database, but it won't maliciously modify the communication data in the scheme or collude with others to disclose the privacy. Finally, we assume that the communication channels in the scheme are protected by existing methods, such as SSL. Therefore, in this paper, we only focus on the privacy protection of the doctors' information, patients' demands and the shared EHRs.

2.3 Design Goals

Considering the aforementioned system model and security model, our design goal is to propose a flexible and fine-grained privacy-preserving EHRs sharing scheme. Specifically, the following design goals should be satisfied:

Fine-grained access control: Any doctors who are shared with EHRs should be consent and authorized by the patients (the owner of EHRs), and patients can achieve a fine-grained access control on the shared EHRs in the meantime, i.e., the doctor, even though authorized by one patient, cannot access this patients' all EHRs and can just decrypt part of encrypted EHRs with specific access policies.

Flexible sharing: Patients only need to encrypt and upload the shared EHRs once, i.e., the shared EHRs should be flexible and do not need to be updated for different shared targets (doctors) by patients.

Data confidentiality and auditability: All data, including doctors' information, patients' demands and the shared EHRs, should be encrypted and stored in CS. When performing doctor recommendation, none of patients' demands and doctors' information will be disclosed. When performing EHRs sharing, only the doctor who is authorized by a patient can decrypt this patient's encrypted EHRs. Additionally, some audit logs should be left for future use.

Computational cost should be acceptable: Considering efficiency and utility of the whole system, after system intialization, patients and doctors in the proposed scheme only need to do minimum operations, and most of the heavy computations should be delegated to CS.

3. PRELIMINARIES

In this section, we outline the bilinear pairing technique [16], locality sensitive hashing [17], proxy re-encryption [18], access structure and linear secret-sharing schemes [10], which will serve as the basis of the proposed FSSR scheme.

3.1 Bilinear Pairing

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of prime order q with the multiplication. Let g and h be two random generators of $\mathbb{G}$ and e be a bilinear map. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map has the following properties: i)Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q$, we have $e(u^a, v^b) = e(u, v)^{ab}$; ii) Non-degeneracy: $e(g, g) \neq 1$; and iii) Computability: there is an efficient algorithm to compute bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

Notice that, in group $\mathbb{G}$, the Computational Diffie-Hellman (CDH) problem is hard, i.e., given g, g^a, g^b for $g \in \mathbb{G}$ and unknown $a, b \in \mathbb{Z}_q$, it is intractable to compute g^{ab} in a polynomial time. However, the Decisional Diffie-Hellman (DDH) problem is easy, i.e., given g, g^a, g^b, g^c for $g \in \mathbb{G}$ and unknown $a, b, c \in \mathbb{Z}_q$, it is easy to judge whether $c = ab \bmod q$ by checking $e(g^a, g^b) \stackrel{?}{=} e(g^c, g)$. Reference [16] can provide more detailed description of pairing technique, and complexity assumptions.

Definition 1. *A bilinear pairing generator algorithm* Gen() *can take a security parameter τ as input, and outputs a 5-tuple parameters $(q, g, h, \mathbb{G}, \mathbb{G}_T, e)$.*

3.2 Locality Sensitive Hashing

Locality sensitive hashing (LSH) is a commonly used technique for performing efficient approximate nearest-neighbor search. With some specially-designed LSH functions, all input similar high dimensional vectors can be hashed into the same bucket with high probability than distant ones.

Definition 2. (Locality Sensitive Hashing [17]) Let V be a set of vectors and D be the distance measure between vectors. Given two distance r_1, r_2 $(r_1 < r_2)$, and two probability p_1, p_2 $(p_1 > p_2)$, a family of hash function $H = \{h : O \rightarrow U\}$ is (r_1,r_2,p_1,p_2)-locality-sensitive iff for any v_i,$v_j \in V$: if $D(v_1, v_2) < r_1$ then $P[h(o_i) = h(o_j)] \geq p_1$; if $D(v_1, v_2) > r_2$ then $P[h(o_i) = h(o_j)] \leq p_2$.

To improve the accuracy, multiple hash functions and hash tables are needed to construct another function family $G = \{g : O \rightarrow U^\alpha\}$, where $g(v) = (h_1(v), h_2(v), ...h_{num_h}(v))$ is the concatenation of num_h LSH functions $h_i \in H$. In this paper, we choose the LSH for Euclidean distance (E2LSH) as it is easy to implement and is comparatively more efficient. For more detailed description of E2LSH, we refer to [17].

3.3 Proxy Re-Encryption

Proxy re-encryption allows a secure ciphertext transformation in a way that a semi-trusted proxy can use a re-encryption key (rekey) delegated from Alice (and Bob) to re-encrypt a ciphertext under Alice's public key into a new ciphertext that Bob can decrypt by using his own private key. However, the proxy cannot do any decryption on the ciphertexts of either Alice or Bob. We refer to [19] for more comprehensive description of proxy re-encryption.

3.4 Access Structure and Linear Secret-Sharing Schemes

Definition 3. (Access Structure [10]) Let $\{P_1, P_2, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}}$ 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 $\{P_1, P_2, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}} \backslash \emptyset$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Definition 4. (Linear Secret-Sharing Schemes (LSSS) [10]) A secret-sharing scheme Π over a set of parties P is called linear (over $\mathbb{Z}_q$) if *i)The shares for each party form a vector over $\mathbb{Z}_q$. ii)There exists a matrix an M with l rows and n columns called the share-generating matrix for Π. For all $i = 1, 2, ..., l$, the i-th row of M we let the function ρ defined the party labeling row i as $\rho(i)$. When we consider the column vector $\overrightarrow{v} = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_q$ is the secret to be shared, and $r_2, r_3, ..., r_n \in \mathbb{Z}_q$ are randomly chosen, then $M \cdot \overrightarrow{v}$ is the vector of l shares of the secret s according to Π. The share $\lambda_i = (M \cdot \overrightarrow{v})_i$ belongs to party $\rho(i)$.*

In the rest of this paper, we denote the pair (M, ρ) as the policy of the access structure. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, ..., l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_q\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \lambda_i = s$. For more details about access structures and linear secret sharing schemes, we refer the readers to [10].

4. DESCRIPTION OF FSSR SCHEME

In this section, we present a fine-grained EHRs sharing scheme via similarity-based recommendation in cloud-assisted ehealthcare system, called FSSR, which mainly consists of the following three phases: system initialization, privacy-preserving doctor recommendation, and fine-grained privacy-preserving EHRs sharing.

4.1 System Initialization

The high level description of system initialization can be found in Figure 2.

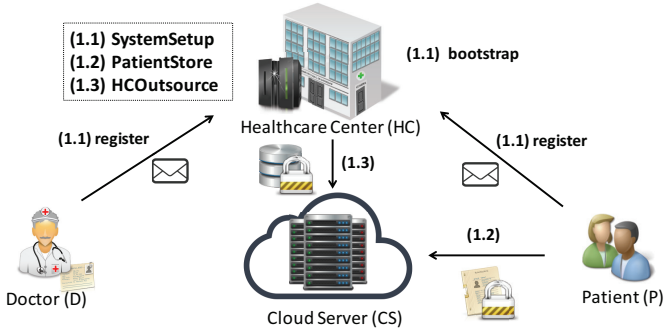


Figure 2: The high level description of system initialization.

• **SystemSetup.** The healthcare center (HC) firstly is responsible for activating the whole system and executes the bootstrap as follows.

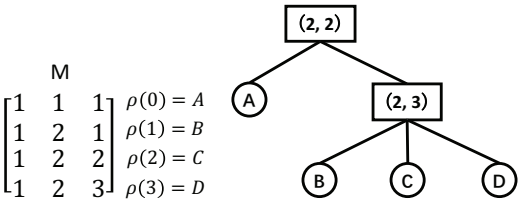
- Given a security parameter τ in system initialization, HC firstly generates $(q, g, \mathbb{G}, \mathbb{G}_T, e)$ by running *Gen()*.
- HC chooses a secure symmetric encryption/decryption algorithm $Enc_k()/Dec_k()$, i.e., AES, where k is the symmetric key, and selects a secure cryptographic hash function $H() : \{0, 1\}^* \rightarrow \mathbb{G}$.
- Given the security parameters τ_1 and τ_2 $(\tau_1 > 2\tau_2)$, HC chooses two large primes p and α, such that $|p| =$

τ_1 and $|\alpha| = \tau_2$, and chooses a large random number $s \in Z_p^*$ as the shared secret key, which will be shared with registered patients.

- HC generates a private key $x_h \in \mathbb{Z}_q^*$, a public key $y_h = g^{x_h}$ and an identity ID_h for itself.

Then, assume that there are total N registered doctors (D) and M patients (P), any patient or doctor can register themselves in HC as follows.

- Patient $P_i \in P$ generates a private key $x_{pi} \in \mathbb{Z}_q^*$, a public key $y_{pi} = g^{x_{pi}}$ and an identity ID_{pi}. when registering, he/she will provide y_{pi} and ID_{pi} for HC, and receive the shared secret key s from HC via an authenticated and secure channel.
- Doctor $D_j \in D$ generates a private key $x_{dj} \in \mathbb{Z}_q^*$, a public key $y_{dj} = g^{x_{dj}}$ and an identity ID_{dj}. When registering, he/she will provide y_{dj} and ID_{dj} to HC, and submit his/her information to HC via an authenticated and secure channel.
- HC will verify each registered doctor's information according to the government's records and hospital's records.



(a) (M, ρ) (b) Access Tree

Figure 3: Examples of access policy

• **PatientStore.** Suppose that patient $P_i \in P$ wants to store m in CS, he/she needs to perform the following steps to encrypt m as CT firstly, and then uploads CT to CS via an authenticated and secure channel.

- *Step-1.* P_i chooses a random number $r \in \mathbb{Z}_q^*$ and generates a random 256-bits symmetric key k_{pi}.
- *Step-2.* P_i chooses a set of attributes with respect to the content of EHR m, such as personal information (name, address, contact), diseases (heart disease, obesity, mental disease), and medication history. Then, P_i generates an access policy (M, ρ) for m, where M is an $l \times n$ matrix, as shown in Figure 3 (a). (M, ρ) can be explained as an access tree as shown in Figure 3 (b). The leaf node represents the attributes (A, B, C and D), non-leaf node is a (t, n)-threshold node, where n is the number of children and t is a threshold value. If and only if at least t child nodes are satisfied, the threshold node is said to be satisfied. If and only if the root node of the access tree is satisfied, the access tree is said to be satisfied.
- *Step-3.* P_i randomly chooses a random vector $\overrightarrow{v} = \{r, y_2, ..., y_n\} \in \mathbb{Z}_q^n$, and for $x = 1$ to l, P_i calculates $\lambda_x = \overrightarrow{v} \cdot M_x$, where M_x is the vector corresponding to the x-th row of M. P_i computes $CT = (c_1, c_m, \{c_{x,1}, c_{x,2}, c_{x,3}\}_{x=1}^{l}, (M, \rho))$ as follows.

$$c_1 = k_{pi} \cdot e(g,g)^{r x_{pi}}, c_m = Enc_{k_{pi}}(m),$$
$$c_{x,1} = y_h^{\lambda_x}, c_{x,2} = h^{\lambda_x}, c_{x,3} = H(\rho(x))^{\lambda_x} \quad (1)$$

If necessary, HC would fetch the CT stored in CS and perform the following steps to decrypt CT.

- *Step-1.* HC randomly selects an authorized set S' which satisfies (M, ρ) and obtains $I_h = \{i' : \rho(i') \in S'\}$.
- *Step-2.* HC computes $w_{i'} \in \mathbb{Z}_q$ such that $\sum_{i' \in I_h} w_{i'} M_{i'} = (1, 0, ..., 0)$, where $M_{i'}$ is the i'-th row of the matrix M.
- *Step-3.* HC computes $\prod_{i' \in I_h} c_{i',1}^{w_{i'}} = y_h^r$. By using y_h^r, k_{pi} can be decrypted as $\frac{c_1}{(e(y_h^r, y_{pi}))^{x_h{}^{-1}}} = \frac{k_{pi} \cdot e(g,g)^{r x_{pi}}}{e(g,g)^{r x_{pi}}} = k_{pi}$.
- *Step-4.* $m = Dec_{k_{pi}}(c_m) = Dec_{k_{pi}}(Enc_{k_{pi}}(m))$ can be decrypted by HC.

Algorithm 1 Build index for Ξ with Locality Sensitive Hashing (LSH)

Input:
All doctors' information, Ξ; the number of non-cryptographic hash function number, num_h; the number of hash tables, num_t; a cryptographic hash function, $H()$.
Output:
Index $\mathbb{INDEX}$ consists of num_t secure hash tables, and keys K for index.

```
for i = 0; i < num_t; i++ do
  for j = 0; j < num_h; j++ do
    HC chooses β⃗ = {beta_1, beta_2, ..., beta_n} as a n-dimensional random vector following a normal distribution.
    HC chooses a constant ω as a bucket width.
    b_{i,j} is chosen uniformly from the range [0, ω).
    HC generates a non-cryptographic hash function as hash_{i,j}(Φ) = (β⃗ · C⃗ + b_{i,j}) / ω.
  end for
end for
for i = 0; i < num_t; i++ do
  HC creates a hash table T_i (an example of secure hash table is shown in Tab. 1).
  HC chooses a random number k_i ∈ 𝔾 for T_i.
  for k = 1; k ≤ N; k++ do
    HC computes the hash value as key_{i,k} = H(h_{i,1}(Φ_k)||h_{i,2}(Φ_k)||...||h_{i,num_h}(Φ_k)||k_i) as the key of doctor D_k.
    if key_{i,k} exists in T_i then
      HC stores (key_{i,k}, PID_{dk}) pair in T_i.
    else
      HC creates a new key key_{i,k} in T_i.
      HC stores (key_{i,k}, PID_{dk}) pair in T_i.
    end if
  end for
end for
return 𝕀𝕟𝔻𝔼𝕏 = {T_1, T_2, ..., T_{num_t}} and K = {k_1, k_2, ..., k_{num_t}}.
```

• **HCOutsource.** HC performs the following steps to encrypt $\Xi = \{C_1, C_2, ..., C_N\}$, and outsources the secure hash index $\mathbb{INDEX}$ and encrypted information $\{Z_1, Z_2, ..., Z_N\}$ to CS.

- *Step-1.* For each registered doctor $D_j \in D$, HC generates a random number $r_{id} \in \mathbb{Z}_q^*$ and computes a pseudo identity PID_{dj} as $PID_{dj} = H(ID_{dj}||r_{id})$.
- *Step-2.* HC runs **Algorithm 1** to map the doctors' information with LSH. According to the property of LSH, doctors with the similar character will be mapped into the same bucket, i.e. obtaining the same hash value.
- *Step-3.* For each doctor's information $\overrightarrow{C} = \{\phi_1, \phi_2, ..., \phi_n\}$, HC computes $Z = \{z_1, z_2, ..., z_n\}$. For each $z_i \in Z$, $z_i = s(\alpha\phi_i + r_i) \bmod p$, where r_i is a random number with length of η_1 ($\eta_1 < \tau_2$). Accordingly, Ξ is encrypted as $\{Z_1, Z_2, ..., Z_N\}$.

Table 1: An example of secure hash table T_i

Keys (buckets)	Values (identifications of doctors)
$key_{i,1}$	$PID_{d1}, PID_{d2}, PID_{d6}, ...$
$key_{i,4}$	$PID_{d4}, PID_{d8}, PID_{d9}, ...$
$key_{i,5}$	$PID_{d5}, PID_{d11}, PID_{d12}, ...$
...	...

Finally, HC publishes the public system parameters $params = (q, g, \mathbb{G}, \mathbb{G}_T, e, E_k(), H(), \tau, \tau_1, \tau_2, \alpha, p, y_h, ID_h)$. For each registered patient $P_i \in P$ and doctor $D_j \in D$, HC publishes (ID_{pi}, y_{pi}) and (ID_{dj}, y_{dj}). In addition, HC also publishes all generated hash functions $\{\{h_{i,j}()\}_{j=1}^{j=num_h}\}_{i=1}^{i=num_t}$ and $K = \{k_1, k_2, ..., k_{num_t}\}$.

4.2 Privacy-Preserving Doctor Recommendation

The high level description of privacy-preserving doctor recommendation can be found in Figure 4.

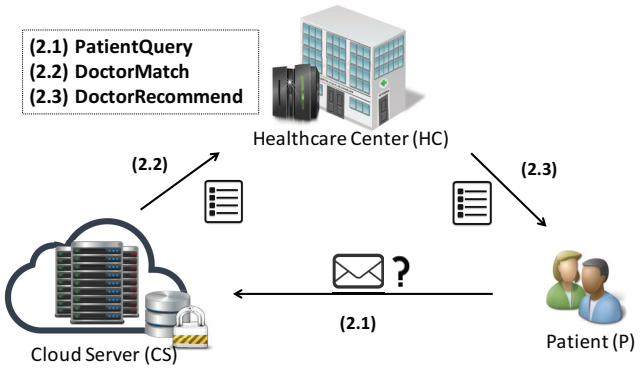


Figure 4: The high level description of recommendation.

• **PatientQuery.** Suppose that patient $P_i \in P$ wants to find some suitable doctors, he/she needs to perform the following steps to encrypt the demands $\overrightarrow{C} = \{\phi_1, \phi_2, ..., \phi_n\}$, and then sends the query $(\{hash_x\}_{x=1}^{num_t}$ and $CC)$ to CS via an authenticated and secure channel.

- For each $\phi_i \in C$, P_i computes $cc_i = s(\alpha\phi_i + r_i) \bmod p$, where r_i is a random number whose length is η_2 ($\eta_2 < \tau_2$). Finally, P_i obtains $CC = \{cc_1, cc_2, ..., cc_n\}$.
- For each hash table $T_x \in T$, P_i calculates a hash value $hash_x = H(h_{x,1}(C)||h_{x,2}(C)||...||h_{x,num_h}(C)||k_x)$, where $\{h_{x,j}\}_{j=1}^{num_h}$ are the hash functions generated by HC for hash table T_x and $k_x \in K$. Then, P_i gets a set of hash values $\{hash_x\}_{x=1}^{num_t}$.

• **DoctorMatch.** After receiving P_i's query, CS and HC perform the following steps to find the doctors whose information may match the demands of P_i using the measurement of Euclidean distance.

- *Step-1.* According to property of LSH, CS runs **Algorithm 2** to locate the doctors $D' \subseteq D$ who may satisfy the demands of P_i.
- *Step-2.* For each doctor $D'_i \in D' = \{D'_1, D'_2, ..., D'_{N'}\}$, CS retrieves the $Z = \{z_1, z_2, ..., z_n\}$ and computes $ED_i = \sum_{j=1}^{n}(z_j - co_j)^2 \bmod p$. Next, CS sends $\mathbb{ED} = \{ED_1, ED_2, ..., ED_{N'}\}$ and the corresponding pseudo identities to HC via an authenticated and secure channel.
- *Step-3.* HC maps the pseudo identities to real identities firstly. Then, for each $ED_i \in \mathbb{ED}$, HC computes $ED'_i = s^{-2}ED_i$ and decrypts the similarity sim_i as $sim_i = \frac{ED'_i - ED'_i \bmod \alpha^2}{\alpha^2}$. Next, HC sorts $D'_i \in D'$ from small to large according to similarity sim_i and regards the top-N'' doctors D'' from D' as the recommended doctors.

Algorithm 2 Matching suitable doctors according to P_i's demands

Input:
P_i's hash values, $\{hash_x\}_{x=1}^{num_t}$; index of all doctors' information, $\mathbb{INDEX} = \{T_1, T_2, ..., T_{num_t}\}$; a cryptographic hash function, $H()$.

Output:
Set Set includes doctors who may match the requirements of P_i.

```
CS creates and initializes a empty set Set = {}.
for i = 0; i < num_t; i++ do
  for each (key, values) in T_i do
    if hash_x == key then
      Set' ← values.
      Set ← Set' ⋃ Set.
    end if
  end for
end for
return Set.
```

• **DoctorRecommend.** HC will recommend these specific doctors D'' to P_k. HC chooses a random number $r' \in \mathbb{Z}_q^*$, and calculates the consent request rq_{ci} for each doctor $D''_i \in D'' = \{D''_1, D''_2, ..., D''_{N''}\}$ as follows.

$$rq_{ci} = g^{r'} \cdot {y_{di}}^{{x_h}^{-1}} = g^{r'} \cdot g^{x_{di}{x_h}^{-1}}. \quad (2)$$

Finally, the consent request $rq_c = \{rq_{c1}, rq_{c2}, ..., rq_{cN''}\}$ is sent to P_i via an authenticated and secure channel.

4.3 Fine-Grained Privacy-Preserving EHRs Sharing

The high level description of fine-grained privacy-preserving EHRs sharing can be found in Figure 5.

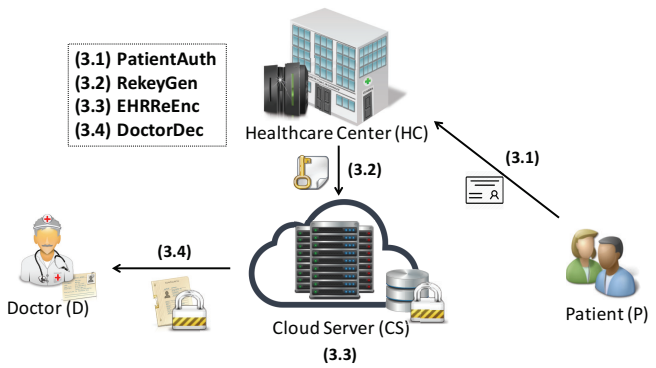


Figure 5: The high level description of sharing.

• **PatientAuth.** When P_i receives the consent request rq_c from HC, he/she authorizes $D_j \in D''$ to share his/her EHRs. To achieve fine-grained access control on the shared EHRs, P_i firstly sets the attribute set $S_i = \{attr_1, attr_2, ..., attr_n\}$ (e.g $\{A, C, D\}$). Then, P_i calculates rp_c as follows.

$$rp_c = rq_{cj}^{x_{pi}} = g^{r' x_{pi}} \cdot g^{x_{dj} x_{pi} {x_h}^{-1}} \tag{3}$$

Finally, P_i sends the consent response rp_c and S_i back to HC via an authenticated and secure channel.

• **RekeyGen.** To generate rekey, HC firstly chooses a random number $r_{pi} \in \mathbb{Z}_q$. For each attribute $attr_i \in S_i$, HC chooses a random number $r_i \in \mathbb{Z}_q^*$, and calculates the re-encryption key $rk_{P_i \to D_j}$ as follows.

$$\begin{aligned} &k_z = rp_c \cdot (y_{pi}^{r'})^{-1} h^{r_{pi}} = g^{x_{pi} x_{dj} {x_h}^{-1}} h^{r_{pi}}, \\ &\forall attr_i \in S_i : \{k_{z1} = h^{r_i}, k_{z2} = y_h^{r_{pi}} H(attr_i)^{r_i}\}, \\ &rk_{P_i \to D_j} = (k_z, \forall attr_i \in S_i : \{k_{z1}, k_{z2}\}). \end{aligned} \tag{4}$$

Finally, HC sends the rekey $rk_{P_i \to D_j}$ and S_i to CS. Meanwhile, HC records the sharing logs as $(P_i, D_j, S_i, Timestamp)$ for P_i, where $Timestamp$ is the current time stamp in HC. Notice that, unless P_i wants to update the rekey for D_j, after generating rekey for D_j, HC does not need to participate in the following sharing process.

• **EHRReEnc.** After receiving rekey $rk_{P_i \to D_j}$, for each P_i's encrypted EHR CT, CS filters out the encrypted EHRs whose access policy can be satisfied by S_i, and sets $I_{pi} = \{i'' : \rho(i'') \in S_i\}$. Then, CS computes $w_{i''} \in \mathbb{Z}_q$ such that $\sum_{i'' \in I_{pi}} w_{i''} M_{i''} = (1, 0, ..., 0)$ where $M_{i''}$ is the i''-th row of the matrix M. Subsequently, CS re-encrypts CT using rekey $rk_{P_i \to D_j}$ as follows.

$$\begin{aligned} &\prod_{i'' \in I_{pi}} \left(\frac{e(k_z, c_{i'',1}) \cdot e(k_{z1}, c_{i'',3})}{e(k_{z2}, c_{i'',2})} \right)^{w_{i''}} \\ =&\prod_{i'' \in I_{pi}} \left(\frac{e(g^{x_{pi} x_{dj} {x_h}^{-1}} h^{r_{pi}}, {y_h}^{\lambda_{i''}}) \cdot e(h^{r_{i''}}, H(\rho(i''))^{\lambda_{i''}})}{e({y_h}^{r_{pi}} H(\rho(i''))^{r_{i''}}, h^{\lambda_{i''}})} \right)^{w_{i''}} \\ =&\prod_{i'' \in I_{pi}} \left(e(g,g)^{x_{pi} x_{dj} \lambda_{i''}} \right)^{w_{i''}} \\ =&\prod_{i'' \in I_{pi}} e(g,g)^{x_{pi} x_{dj} \lambda_{i''} w_{i''}} = e(g,g)^{x_{pi} x_{dj} r} \end{aligned} \tag{5}$$

Finally, CS stores the converted encrypted EHR files, which can be represented as $CT' = (c_1, c_m, e(g,g)^{x_{pi} x_{dj} r})$.

• **DoctorDec.** D_j can decrypt the converted encrypted EHR file CT' straightforward as follows.

$$\frac{c_1}{(e(g,g)^{x_{pi} x_{dk'} r})^{x_{di}^{-1}}} = \frac{k_{pi} \cdot e(g,g)^{r x_{pi}}}{e(g,g)^{r x_{pi}}} = k_{pi}. \tag{6}$$

Finally, the EHR $m = Dec_{k_{pi}}(c_m) = Dec_{k_{pi}}(Enc_{k_{pi}}(m))$ is obtained by D_j.

5. SECURITY ANALYSIS

In this section, we analyze the security properties of the proposed FSSR scheme. In specific, our analyses focus on how the proposed FSSR scheme achieves privacy-preserving doctor recommendation and fine-grained privacy-preserving EHRs sharing as follows.

•*Patients' demands and doctors' information are privacy-preserving:* Firstly, HC uses all doctors' information to create secure hash index. The keywords in each hash table of index are encrypted by one-way hash function $H()$. Since HC creates different keys $K = \{k_1, k_2, ..., k_{numt}\}$ for different hash tables, the keywords $key_{i,x}$ are indistinguishable from random. Similarly as doctors' information, patients' demands are also secure and well protected with hash functions. Secondly, patients' demands and doctors' information are encrypted as $z_i = s(\alpha\phi_i + r_i) \bmod p$ and $cc_j = s(\alpha\phi_j + r_j) \bmod p$ for accurate comparison. s is a secret key sharing between registered patients and HC. r_i and r_j are two random numbers to ensure that every z_i and cc_j are indistinguishable even though c_i and c_j are same as $c_{i'}$ and $c_{j'}$. Without knowing s, CS cannot decrypt z_i and cc_j to obtain c_i and c_j. Therefore, patients' demands and doctors' information are privacy-preserving.

•*EHRs are privacy-preserving:* When patient P_i encrypts EHRs m with specific access policies, the ciphertext are $CT = (c_1, c_m, \{c_{x,1}, c_{x,2}, c_{x,3}\}_{x=1}^{l}, (M, \rho))$, where $c_1 = k_{pi} \cdot e(g,g)^{r x_{pi}}$, $c_2 = Enc_{k_{pi}}(m)$, $\{c_{x,1} = y_h^{\lambda_x}, c_{x,2} = h^{\lambda_x}, c_{x,3} = H(\rho(x))^{\lambda_x}\}_{x=1}^{l}$. According to attribute-based techniques [10], only the authorized doctor can recover the patient's EHR m. HC generates the rekey as $rk_{P_i \to D_j} = (k_z, \forall attr_i \in S_i : \{k_{z1}, k_{z2}\})$, and CS uses the rekey to re-encrypt the ciphertext as $CT' = (c_1, c_m, e(g,g)^{x_{pi} x_{dj} r}$ based on attributes S_i. Only doctor D_j can use his/her private key x_{dj} to decrypt c_1 and get $k_{pi} = \frac{c_1}{(e(g,g)^{x_{pi} x_{dk'} r})^{x_{di}^{-1}}}$, and then perform AES decryption algorithm to decrypt c_m and obtain $m = D_{k_{pi}}(c_m)$. Therefore, the EHRs are privacy-preserving.

•*Fine-grained access control can be achieved:* With patients' specific demands, HC will recommend some suitable doctors. The consent request is calculated as $g^{r'} \cdot {y_{di}}^{{x_h}^{-1}}$ and consent response is calculated as $g^{r' x_{pi}} \cdot g^{x_{dj} x_{pi} {x_h}^{-1}}$. r' is a random number to ensure that every consent request and response are indistinguishable. x_{pi} is the private key which belongs to P_i, and HC cannot generate rekey for CS without consent response. Meanwhile, when encrypting EHRs, patients can choose the access policy for each EHR file with attribute-based techniques, such as $CT = (c_1, c_m, \{c_{x,1}, c_{x,2}, c_{x,3}\}_{x=1}^{l}, (M, \rho))$. Namely, when generating rekey for CS by the authorized attributes from patients, HC can control which EHRs can be re-encrypted and decrypted by the authorized doctors. Therefore, fine-grained access control has been achieved.

6. PERFORMANCE EVALUATION

In this section, we present an experimental evaluation of the proposed FSSR scheme. The performance of FSSR scheme is evaluated in terms of computational cost, storage and communication cost. The performance of FSSR are dependent on several parameters, including the security parameters $(\tau, \tau_1, \tau_2, \eta_1, \eta_2)$, the number of non-cryptographic hash functions num_h, the number of secure hash tables, num_t, and the bucket width ω. Experimentally, we choose the parameters as $\tau = 160$, $\tau 1 = 1024$, $\tau 2 = 400$, $\eta 1 = \eta 2 = 128$, $num_h = 3$, $num_t = 6$ and $\omega = 100$ (note that, the parameters used here are not claimed to be the optimum ones). Our experiment environment is a desktop with 3.1 GHz processor, 8GB RAM, and Window 7 platform using Java Pairing-Based Library [20]. In addition, different from most of the existing attribute-based techniques implementation (access tree and polynomial), we implement the attribute-based techniques based on LSSS [21] and develop a prototype of FSSR based on Java.

6.1 Computational Cost

According to our proposed scheme, we will discuss the computational cost of system initialization, privacy-preserving doctor recommendation and fine-grained privacy-preserving EHRs sharing respectively.

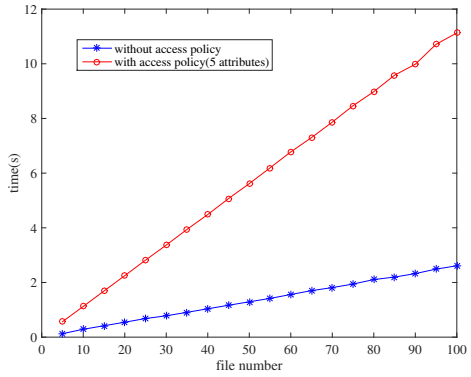
6.1.1 Computational Cost of System Initialization

• *Computational Cost of SystemSetup.* HC needs to initialize the system by executing the bootstrap to generate some key materials. We run this part for 100 times, and the average running time is about 100 ms.

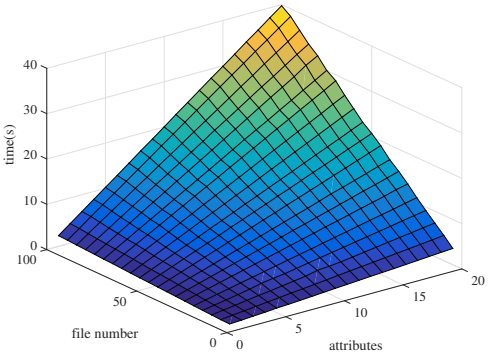
• *Computational Cost of PatientStore.* In order to measure the computational cost of PatientStore, we obtain a public XML-formatted EHR (635KB) file from VA Personal Health Record Sample Data [22]. In our scheme, patients need to encrypt the EHRs with specific access policy, while it can definitely increase the computational cost. Thus, based on the public EHR file, we compare the computational overhead of two encryption types (with/without access policy) with different file numbers. Moreover, we also measure the computational overhead while EHRs are encrypted with different access policies (different attributes) and different file numbers.

As shown in Figure 6 (a), it is clear that encryption without access policy is more efficient than encryption with access policy. When encrypting 100 EHR files (6.35MB) without access policy, the computation time is almost 2 seconds. By contrast, the computation time is almost 12 seconds when encrypting the same 100 EHR files with access policy (5 attributes). However, EHRs encryption with access policy can provide a fine-grained access control on the shared EHR files, which efficient and simple proxy re-encryption cannot achieve. Therefore, based on different requirements, patients can find a trade-off between privacy and efficiency when encrypting and uploading EHRs. Additionally, one EHR file of patients may have 2-3 attributes at most in real world, and the computation time is almost 5-7 seconds when encrypting 100 files, as shown in Figure 6 (b). Except for some emergent situations, EHRs just need to be encrypted and uploaded once. Thus, the computational cost is acceptable.

• *Computational Cost of HCOutsource.* When encrypting all doctors' information, firstly, HC builds a secure hash index $\mathbb{INDEX}$ according to Algorithm 1. Secondly, HC needs to encrypt each doctor's information. Obviously, the doctor number N and the number of criterion's entries n are two main parameters to affect the computational overhead of HCOutsource. Hence, N is chosen from 10,000 to 100,000 and n is chosen from 10 to 50 for different situations. Moreover, we generate all doctor's information randomly.



(a) With/without access policy



(b) Different attributes and file number

Figure 6: Computational cost of PatientStore.

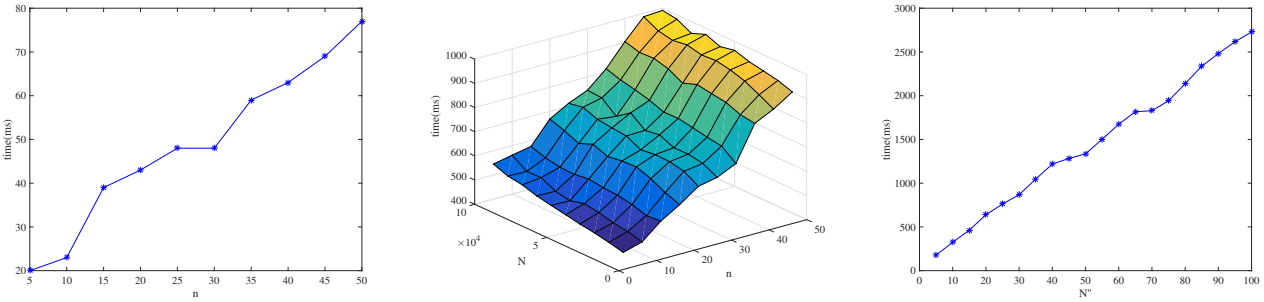
As shown in Table 2, more doctors and entries of criterion lead to the more computational overhead. The running time is almost 9 minutes when there are 100,000 doctors and 50 entries for each doctor' information. However, all doctors' information only needs to be encrypted and outsourced to CS once at the stage of system initialization. Thus, the computational cost is acceptable.

6.1.2 Computational Cost of Privacy-Preserving Doctor Recommendation

• *Computational Cost of PatientQuery.* Patients firstly perform PatientQuery to submit their encrypted demands to CS. The encrypted demands are related to the number of criterion's entries n according to our scheme. Apparently, with different n, the computational cost of PatientQuery will be different. Thus, we set n as 5,10,...,50 and generate different demands randomly. As shown in Figure 7 (a), although the increase of n leads to the increase of the running time, the maximum running time is less than 100 ms. Therefore, PatientQuery is very efficient when computing at

Table 2: Computational cost of HCOutsource

Doctor number (N)	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000	100,000
Time/s ($n = 10$)	10.4	20.6	30.9	41.2	51.4	60.6	70.5	80.4	90.3	101.0
Time/s ($n = 20$)	18.7	38.7	61.5	78.9	95.3	134.8	161.5	184.8	206.9	230.2
Time/s ($n = 30$)	33.6	67.3	101.2	135.2	168.4	202.1	236.1	269.7	302.8	337.6
Time/s ($n = 40$)	44.5	88.4	133.1	177.8	221.7	266.1	310.2	356.0	399.4	443.4
Time/s ($n = 50$)	55.0	110.0	165.0	219.9	275.0	329.8	384.5	435.4	481.2	546.7



(a) PatientQuery (b) DoctorMatch (c) DoctorRecommend

Figure 7: Computational cost of privacy-preserving doctor recommendation

patient side.

• *Computational Cost of DoctorMatch.* When CS runs DoctorMatch to find the doctor using LSH, the number of doctors N and the number of criterion's entries n are still two core parameters. Accordingly, N is chosen from 10,000 to 100,000, and n is chosen from 5 to 50 to measure the computational overhead of different situations. We run the Doctor Match 100 times and the results are as shown in Figure 7 (b). Despite that N and n increases greatly, the running time of DoctorMatch is around 1 3 seconds, which is very efficient and supports real-time searching. However, the approach of LSH exists some errors, which may make the matching results inaccurate. The accuracy of results is relevant with num_h, num_t and ω according to the definition of LSH [17]. In our experiments, DoctorMatch can locate some suitable doctors according to the patient's demands, but cannot ensure the most similar one can be found according to the criterion. Therefore, there is also a trade-off between the efficiency and accuracy for DoctorMatch. Namely, if a patient wants to find the most suitable doctor based on his/her demands, HC and CS may need to perform the linear search approach in ciphertext which can also be achieved in our scheme, but the efficiency should be concerned if the scheme wants to support real-time searching.

• *Computational Cost of DoctorRecommend.* According to different situations, different top-N'' doctors will be chosen from D', the number of recommended doctors D'' determines the computational overhead of DoctorRecommend. Thus, we choose N'' from 5 to 100 to evaluate the computational efficiency of DoctorRecommend. As shown in Figure 7 (c), with the increase of N'', the running time also increases, but the running time is less than 3 seconds while HC recommends 100 doctors to patient. However, in the real world, the recommended doctors are no more than 10. Therefore, DoctorRecommend is also efficient.

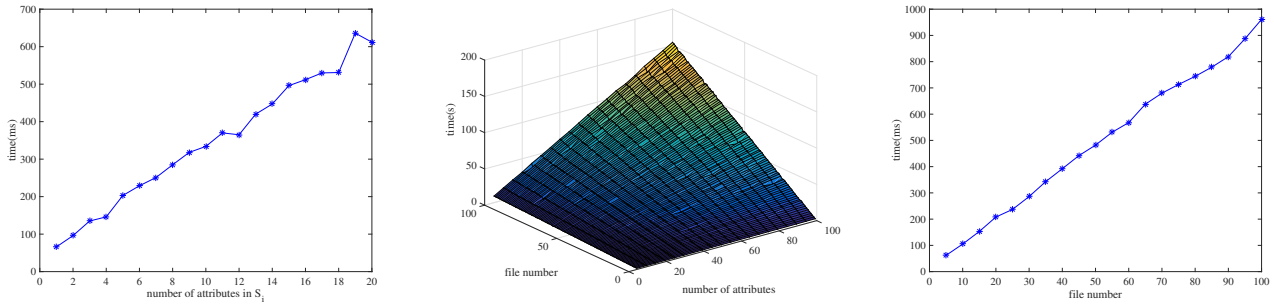
6.1.3 *Computational Cost of Fine-Grained Privacy-Preserving EHRs Sharing*

• *Computational Cost of PatientAuth.* One patient receives all recommended doctors from HC and only chooses to authorize one doctor. To measure the computational overhead of PatientAuth, we run PatientAuth 100 times, and get the average running time as 50 ms. Therefore, the PatientAuth is very efficient.

• *Computational Cost of RekeyGen.* When HC generates the rekey for a patient P_i, the computational cost is mostly related to the authorized set S_i. With more attributes in authorized set S_i, HC should generates more elements in the final re-encryption keys. Considering different situations, we simulate the RekeyGen by choosing different number of attributes in S_i, which is from 1 to 20. As shown in Figure 8 (a), with more attributes, the running time is slower, but the running time of RekeyGen is about 500 ms when there are maximum 20 attributes in S_i, which is also very efficient.

• *Computational Cost of EHRReEnc.* After CS receives the rekey from HC, CS needs to re-encrypt all encrypted EHR files which satisfy the access policy based on the authorized set S_i. However, the computational cost of EHRReEnc is not only related to the file number of EHRs, but also related to the number of attributes which is needed for re-encrypting the original ciphertext. Thus, we choose the number of attributes from 5 to 100 and file number of EHRs from 5 to 100 to simulate the different processes. As shown in Figure 8 (b), more file numbers and the numbers of attributes lead to more computational overhead of EHRReEnc. However, since the process of re-encryption is done by CS with great computational power and only needs to be done once, the computational cost is acceptable.

• *Computational Cost of DoctorDec* Finally, the authorized doctor D_j can decrypt and obtain the shared EHRs from P_i. The computational cost of DoctorDec is related to the shared file number. With more shared files, the doctor needs to decrypt more encrypted files. As shown in Figure 8



(a) RekeyGen (b) EHRReEnc (c) DoctorDec

Figure 8: Computational cost of fine-grained privacy preserving EHRs sharing

(c), due to simple and straightforward decryption operations (no attributes-based decryptions), the computational cost at doctor side is also efficient. Apparently, while the doctor decrypts 100 shared EHR files, the running time is less than 1 seconds.

Table 3: Computational cost of the whole sharing process

Situations	N''	Number of attributes	File number	Time/s
(1)	5	2	100	4.37
(2)	10	2	100	4.53
(3)	10	2	1000	40.37
(4)	5	5	100	8.71
(5)	10	5	100	8.93
(6)	10	5	1000	85.29

Finally, we evaluate the computational cost of the whole sharing process. We consider several situations with different access policies (different attributes) and different numbers of shared EHRs as shown in Table 3. Actually, it is easy to find that the file number and the number of attributes are two key factors to affect the computational efficiency. In the real world, most of EHRs' attributes are fewer than 5, and the number of shared EHR files is fewer than 100. Therefore, considering that the total sharing time is less than 5 seconds, our sharing protocol is efficient enough.

Table 4: Notations for storage and communication cost analysis

S_1	Bit size of an element in $\mathbb{G}$
S_t	Bit size of an element in $\mathbb{G}_T$
S_p	Bit size of an element in $\mathbb{Z}_p$
S_f	Bit size of an EHR file
S_i	Bit size of a identification
S_v	Bit size of a value in hash table
N_k	Number of keys in one hash table
N_d	Number of doctors
N_p	Number of patients
N_s	Number of one patient's EHR files
N_{a1}	Number of attributes for when encrypting
N_{a2}	Number of authorized attributes when sharing
N_{sf}	Number of shared EHR files for a doctor

6.2 Storage and Communication Cost

In addition, we also analyze the storage and communication efficiency by looking at the storage and communication overhead. For ease of presentation, we firstly give some notations in Table 4 that we will use in the analysis. Considering different phases in our scheme, it is clear that the storage overhead is related to the phase of system initialization, while the communication overhead can be determined by the phases of privacy-preserving doctor recommendation and fine-grained privacy-preserving EHRs sharing.

• *Storage Cost.* The total storage overhead in CS can be summarized as $num_t(N_kS_1+S_v)+nNS_p+N_sN_p(S_1(3N_{a1}+1)+S_t+S_f)$. Concretely, the storage overhead of secure hash index is $num_t(N_kS_1+S_v)$, the storage overhead of encrypted doctors' information is nNS_p, and the storage overhead of patients' encrypted EHR files is $N_sN_p(S_1(3N_{a1}+1)+S_t+S_f)$ as shown in Table 5.

Table 5: Storage cost of FSSR

Data	Storage cost
Secure hash index	$num_t(N_kS_1 + S_v)$
Doctors' information	nNS_p
EHR files	$N_sN_p(S_1(3N_{a1} + 1) + S_t + S_f)$

• *Communication Cost* For clarity, we just consider the communication cost of EHRs sharing between one patient and one doctor. Specifically, the communication overhead of privacy-preserving doctor recommendation is $nS_p+num_hS_1+N'(S_i+S_p)+N''S_1$, and the communication overhead of fine-grained privacy-preserving EHRs sharing is $S_1 + S_1(2N_{a2} + 1) + N_{sf}(S_f + 2S_t)$ as shown in Table 6.

Table 6: Communication cost of FSSR

Phases	Communication cost
Doctor recommendation	$nS_p + num_hS_1 + N'(S_i + S_p) + N''S_1$
EHRs sharing	$S_1 + S_1(2N_{a2} + 1) + N_{sf}(S_f + 2S_t)$

7. RELATED WORK

Our work is related to privacy-preserving EHRs sharing solutions. In this section, we introduce some related work that can be used to realize privacy-preserving EHRs sharing,

though some of solutions are designed for privacy-preserving data sharing but not particularly for EHRs sharing in ehealthcare system.

Solutions based on proxy re-encryption. In the solutions based on proxy re-encryption [18, 23, 24], all patients' EHRs are encrypted by their public key and stored in cloud. When a patient wants to share EHRs, by using his/her private and the shared target's public key, he/she can compute a re-encryption key and send the re-encryption key to cloud. Then, cloud helps patients to re-encrypt EHRs and relay the results to target. Proxy re-encryption is very efficient especially without pairing [24]. Therefore, although proxy re-encryption has no access control on the shared EHRs, proxy re-encryption is the basis of other secure sharing approaches.

Solutions based on identity-based encryption. In order to achieve accurate access control, some solutions [15, 25–27] combine identity-based encryption with other techniques, such like proxy re-encryption and searchable encryption together. By controlling the identity of doctors and other factors like time, patients can encrypt EHRs with different identities, and only the doctors who have the corresponding identities can decrypt and obtain the EHRs. However, the schemes based on identity-based encryption cannot realize a flexible and fine-grained access control. Meanwhile, identity-based encryption is very communication and computationally costly and not flexible when patients need to update the EHRs.

Solutions based on attribute-based encryption. The solutions [5–8, 28] based on attribute-based encryption is an improvement of schemes based on identity-based encryption. Patients can encrypt the shared EHRs with more attributes and select the doctors with specific attributes to obtain the EHRs. That is, patients encrypt the EHRs with some access policies, and the doctors whose attributes satisfy the access policy can decrypt the corresponding EHR file. However, as we discussed above, these schemes are not flexible and fine-grained enough to support all kinds of situations. Moreover, they have the some problem of high computational cost.

Solutions based on attribute-based proxy re-encryption. By combining proxy re-encryption and attribute-based encryption, some solutions [29, 30] can realize a secure and flexible EHRs sharing. Patients just need to encrypt the EHRs once and store the encrypted EHRs in cloud. When the access policy of a EHR file changes, the patient can offer a re-encryption key and the cloud can help to re-encrypt the EHR with the updated access policy. However, the decryption of EHR file is also controlled by the attributes instead of patients (the owner of EHRs) in these solutions. Therefore, the access control is also not very fine-grained according to our definition.

Different from aforementioned solutions, we propose a more fine-grained EHRs sharing scheme via similarity-based recommendation in cloud-assisted ehealthcare system, called FSSR. With FSSR, every EHR file is under the control of patients and only the authorized doctors can read the EHRs under fine-grained privacy access control. Meanwhile, the patients just need to encrypt the EHRs once, and can share them with different doctors under different access policies.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a fine-grained EHRs sharing scheme via similarity-based recommendation in cloud-assisted ehealthcare system, called FSSR, which mainly exploits how to achieve secure and flexible EHRs sharing with suitable doctors under fine-grained access control. Detailed security analysis shows that the proposed FSSR scheme can achieve privacy-preserving doctor recommendation and privacy-preserving EHRs sharing. In addition, through extensive performance evaluation, we have also demonstrated the proposed FSSR scheme is efficient at patients/doctor side, and can balance the computational efficiency and minimizing the privacy disclosure in cloud-assisted ehealthcare system. In our future work, we intend to carry on smartphone-based experiments to further verify the effectiveness of the proposed FSSR scheme for mobile device. In addition, we will also exploit the security issues of FSSR under collusion attacks.

Availability

The Java implementation of the proposed FSSR can be downloaded at https://www.dropbox.com/s/t74cw0vaomdc1yg/FSSR.zip?dl=0

References

[1] B. Monegain, "Why the ehr market continues to grow," http://www.healthcareitnews.com/news/why-ehr-market-grow-2020, 2015, [Online; accessed 28-October-2015].

[2] P. Fusion, "What are the benefits of electronic health records (ehrs)?" http://www.practicefusion.com/blog/benefits-electronic-health-records-ehrs/, 2014, [Online; accessed 30-October-2015].

[3] S. Radcliffe, "Patients beware: Hackers are targeting your medical information," http://www.healthline.com/health-news/hackers-are-targeting-your-medical-information-010715#1, 2015, [Online; accessed 28-October-2015].

[4] J. L. F. Alemán, I. C. Señor, P. Á. O. Lozoya, and A. Toval, "Security and privacy in electronic health records: A systematic literature review," *Journal of Biomedical Informatics*, vol. 46, no. 3, pp. 541–562, 2013.

[5] 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 Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, 2013.

[6] S. Narayan, M. Gagné, and R. Safavi-Naini, "Privacy preserving EHR system using attribute-based infrastructure," in *Proceedings of the 2nd ACM Cloud Computing Security Workshop, CCSW 2010, Chicago, IL, USA, October 8, 2010*, 2010, pp. 47–52.

[7] C. Wang, X. Liu, and W. Li, "Implementing a personal health record cloud platform using ciphertext-policy attribute-based encryption," in *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*, 2012, pp. 8–14.

[8] J. A. Akinyele, C. U. Lehmann, M. Green, M. W. Pagano, Z. N. J. Peterson, and A. D. Rubin, "Self-protecting electronic medical records using attribute-based encryption," *IACR Cryptology ePrint Archive*,

vol. 2010, p. 565, 2010. [Online]. Available: http://eprint.iacr.org/2010/565

[9] K. Liang, L. Fang, D. S. Wong, and W. Susilo, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," *IACR Cryptology ePrint Archive*, vol. 2013, p. 236, 2013.

[10] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, 2011, pp. 53–70.

[11] T. Company, "2015 security health check report," https://www.trustwave.com/Resources/Library/, 2015, [Online; accessed 28-October-2015].

[12] R. Lowes, "Stolen ehr charts sell for $50 each on black market," http://www.medscape.com/viewarticle/824192, 2014, [Online; accessed 28-October-2015].

[13] Y. Yang, H. Lu, J. Weng, Y. Zhang, and K. Sakurai, "Fine-grained conditional proxy re-encryption and application," in *Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings*, 2014, pp. 206–222.

[14] BetterDoctor, "Betterdoctor," https://betterdoctor.com, 2015, [Online; accessed 02-November-2015].

[15] Y. Tong, J. Sun, S. S. M. Chow, and P. Li, "Cloud-assisted mobile-access of health data with privacy and auditability," *IEEE J. Biomedical and Health Informatics*, vol. 18, no. 2, pp. 419–429, 2014.

[16] D. Boneh and M. K. Franklin, "Identity based encryption from the weil pairing," *IACR Cryptology ePrint Archive*, vol. 2001, p. 90, 2001.

[17] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, 2004, pp. 253–262.

[18] L. Xu, X. Wu, and X. Zhang, "CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, 2012, pp. 87–88.

[19] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.

[20] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

[21] Z. Liu, Z. Cao, and D. S.Wong, "Efficient generation of linear secret sharing scheme matrices from threshold access trees," *IACR Cryptology ePrint Archive*, vol. 2010, 2010. [Online]. Available: https://eprint.iacr.org/2010/374

[22] U. G. S. Administration, "Va personal health record sample data," https://catalog.data.gov/dataset/va-personal-health-record-non-identifiable-data, 2015, [Online; accessed 28-October-2015].

[23] H. Guo, Z. Zhang, J. Zhang, and C. Chen, "Towards a secure certificateless proxy re-encryption scheme," in *Provable Security - 7th International Conference, ProvSec 2013, Melaka, Malaysia, October 23-25, 2013. Proceedings*, 2013, pp. 330–346.

[24] A. Srinivasan and C. P. Rangan, "Certificateless proxy re-encryption without pairing: Revisited," in *Proceedings of the 3rd International Workshop on Security in Cloud Computing, SCC@ASIACCS '15, Singapore, Republic of Singapore, April 14, 2015*, 2015, pp. 41–52.

[25] J. Benaloh, M. Chase, E. Horvitz, and K. E. Lauter, "Patient controlled encryption: ensuring privacy of electronic medical records," in *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, 2009, pp. 103–114.

[26] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Inf. Sci.*, vol. 258, pp. 355–370, 2014.

[27] J. Sun, X. Zhu, C. Zhang, and Y. Fang, "HCPP: cryptography based secure EHR system for patient privacy and emergency healthcare," in *2011 International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, June 20-24, 2011*, 2011, pp. 373–382.

[28] L. Ibraimi, M. Asim, and M. Petkovic, "Secure management of personal health records by applying attribute-based encryption," in *Wearable Micro and Nano Technologies for Personalized Health (pHealth), 2009 6th International Workshop on*, June 2009, pp. 71–74.

[29] K. Liang, L. Fang, D. S. Wong, and W. Susilo, "A ciphertext-policy attribute-based proxy re-encryption scheme for data sharing in public clouds," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 2004–2027, 2015.

[30] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, 2015, pp. 2677–2685.