PLOS ONE

RESEARCH ARTICLE

A data-sharing scheme that supports multi-keyword search for electronic medical records

Shufen Niu, Wenke Liu*, Song Han, Lizhi Fang

College of Computer Science and Engineering, Northwest Normal University, Lanzhou, Gansu, China

* liuwenke0315@foxmail.com

Abstract

As cloud storage technology develops, data sharing of cloud-based electronic medical records (EMRs) has become a hot topic in the academia and healthcare sectors. To solve the problem of secure search and sharing of EMR in cloud platforms, an EMR data-sharing scheme supporting multi-keyword search is proposed. The proposed scheme combines searchable encryption and proxy re-encryption technologies to perform keyword search and achieve secure sharing of encrypted EMR. At the same time, the scheme uses a traceable pseudo identity to protect the patient's private information. Our scheme is proven secure based on the modified Bilinear Diffie-Hellman assumption and Quotient Decisional Bilinear Diffie-Hellman assumption under the random oracle model. The performance of our scheme is evaluated through theoretical analysis and numerical simulation.

Citation: Niu S, Liu W, Han S, Fang L (2021) A data-sharing scheme that supports multi-keyword search for electronic medical records. PLoS ONE 16(1): e0244979. https://doi.org/10.1371/journal.pone.0244979

Editor: Hua Wang, Victoria University, AUSTRALIA

Received: September 3, 2020

Accepted: December 21, 2020

Published: January 7, 2021

Data Availability Statement: All relevant data are within the manuscript and its Supporting information files.

Funding: The National Natural Science Foundation of China(No.61562077, No.61662069, No.61662071, No.61772022) http://www.nsfc.gov.cn/ The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

1 Introduction

An electronic medical record (EMR) is a digital document that contains medical information about a patient; this document is stored, managed, transmitted, and reproduced with electronic devices (computers, health cards, and others) [1]. Compared to the traditional medical record in paper form, EMR has the advantages of large storage capacity, resource saving, convenient query, improved diagnosis and treatment efficiency. With the continuous development of cloud computing, EMR has been rapidly developed, widely used, and gradually improved. A growing number of institutions and individuals use EMR and upload these data to the cloud for storage. Cloud-based systems have more advantages than traditional systems. Users can store and maintain massive data quickly and enjoy high-quality data storage services formed by cloud computing [2].

As a pervasive storage platform, cloud server providers are willing to deploy their EMR storage and application services to cloud servers [3]. Since EMR involves a large amount of patient's private information, an important task is to prevent the EMR from being leaked by unauthorized users and cloud servers [4, 5]. To ensure data security and user privacy, the data are usually stored in the form of ciphertext in the cloud server, but users encounter the problem of how to search through the ciphertext. Searchable encryption is a cryptographic primitive that has been developed in recent years to assist users when performing keyword search on the ciphertext. This type of encryption fully utilizes abundant computing resources of cloud

servers to perform keyword search on the ciphertext [6, 7]. Using searchable encryption technology, users can efficiently search EMR on the cloud server [8]. As the ciphertext of EMR is encrypted with the patient's public key, the ciphertext can only be decrypted by the patient using the private key, which causes inconvenience in EMR sharing. The re-encryption technology realizes the conversion of the ciphertext [9], which can be converted into the ciphertext that can be decrypted by other users so that the patient's EMR can be shared.

1.1 Related works

To enable precise retrieval of encrypted data, Song et al. [10] first proposed symmetric searchable encryption (SSE) based on stream cipher. However, the key distribution of SSE is difficult, which means that they cannot be applied in many practical applications. To address this issue, Boneh et al. [11] proposed public key encryptions with keyword search scheme (PEKS) and proved its security under the random oracle model. However, a secure channel is necessary to transmit the key in this scenario. Baek et al. [12] first proposed a secure channel free proxy re-encryption with keyword search (SCF-PEKS) model. Xu et al. [13] proposed the concept of public key encryption based on fuzzy keyword search. After the server implements fuzzy keyword search on all ciphertexts, it returns the results to the receiver, and the receiver performs a more accurate keyword search on these results. As searchable encryption provides the capability to query encrypted data with a given keyword, it can be applied to the EMR to protect the patient's private information, such as information on identity, communication, and medical history. Liu et al. [14] proposed an efficient and secure fine-grained access control scheme, which realized authorized users' access to the EMR in cloud storage. Li et al. [15] proposed an attribute-based searchable encryption for the EMR system, which reduced the difficulty of key management in a multi-user environment and realized fine-grained access control of EMR by data owners. As certificateless public key cryptography solves the key escrow problem and avoids the use of certificates, Ma et al. [16] proposed a certificateless searchable public key encryption scheme for mobile healthcare systems. In most EMR data-sharing schemes based on searchable encryption, the EMR ciphertext is encrypted by the patient's public key, so only the patient uses its private key to decrypt. If the patient's condition is serious, more hospitals are needed for online consultation, and sharing of EMR becomes a problem.

The emergence of proxy re-encryption (PRE) is considered a superior solution to the aforementioned problems. In proxy re-encryption, a semi-trusted agent converts the ciphertext encrypted with the public key of the delegator Alice to the ciphertext encrypted with the public key of the delegatee Bob through the re-encryption key generated by the proxy re-encryption. Shao et al. [17] first proposed a new cryptography primitive called proxy re-encryption with keyword search (PRES) and constructed a bidirectional PRES scheme; the researchers proved the security of this scheme under the random oracle model. Guo et al. [18] proposed the definition and security model of proxy re-encryption with keyword search with a designated tester (dPRES), which can be proved secure under the standard model. Chen et al. [19] proposed the model of limited proxy re-encryption with keyword search (LPREKS) and proved its security under the mBDH assumption and q-DBDHI assumption in the random oracle model.

1.2 Our contributions

In this study, we propose an electronic medical record data-sharing scheme that supports multi-keyword search. We simplify the proposal of Chen et al. [19] and apply it to the data sharing of EMR to achieve secure storage, privacy preservation, and secure sharing of EMR. Roughly, the contributions of our scheme are described as follows:

1. We propose a framework for cloud-based EMR sharing with security and privacy preservation for diagnosis improvements in e-Health system. The doctor generates the EMR for the patient and encrypts it using the public key of the patient. The cloud server is responsible for storing the patient's EMR ciphertext and performs the search operation on EMR.
2. Our scheme can achieve conditional privacy preservation, in which each EMR encrypted by a patient is mapped to a distinct pseudo identity, while a legal hospital can retrieve the real identity of a patient from any pseudo identity. When the true identity of the patient needs to be obtained, the user can send an identity-tracking request to the hospital. After the verification request is legal, the hospital returns the true identity of the patient to the user.
3. We apply the searchable encryption to implement the secure search on the patient's EMR. The keyword index is stored in the cloud server. When the patient or data user needs to access the patient's EMR, the patient uses his/her private key and multi-keyword to generate a trapdoor and upload to the cloud server, then the cloud server performs the search operation.
4. In this scheme, EMR can be obtained not only by the patient, but also by the data user, such as medical institution and insurance company. We apply the proxy re-encryption to ensure secure sharing of the patient's EMR. When the patient wants to access his/her EMR, the patient sends the trapdoor to the cloud server. The cloud server returns EMR ciphertext to the patient. When the data user wants to obtain the patient's EMR, an authorization request is sent to the patient. After the authorization of the patient, the cloud server generates a re-encryption key to encrypt the EMR ciphertext. After obtaining the re-encryption ciphertext, the data user decrypts it with his/her private key to obtain the patient's EMR.

1.3 Paper organization

The rest of this paper is organized as follows. In section 2, we present some preliminaries. In section 3, we introduce the system architecture, threat model and design goals, and algorithm model of our scheme. In section 4, we provide an overview of our scheme and describe the scheme in detail. Section 5 provides the security analysis, including the achieving goals and security proof of our scheme. In section 6, we compare the proposed scheme with relevant schemes through theoretical analysis and numerical simulation. Finally, we conclude the paper in section 7.

2 Preliminaries

2.1 Bilinear map

Let G_1 and G_2 be two cyclic groups of a large prime q. Let g be a generator of G_1. A bilinear pairing e is a function defined by e: $G_1 \times G_1 \rightarrow G_2$ if the function e satisfies the following properties:

1. Bilinearity: For any $a, b \in Z_q^*$, $e(g^a, g^b) = e(g, g)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.
3. Computablility: $e(g, g)$ can be efficiently computed.

2.2 Hardness assumptions

Let G_1 be a cyclic group of a large prime q with a generator g. The following assumptions hold in our scheme.

Definition 1. (Modified Bilinear Diffie-Hellman (mBDH) Problem) [20]. *Given* $(g, g^a, g^b, g^c) \in G_1$ for $a, b, c \in Z_q^*$, the mBDH problem is to compute $e(g, g)^{ab/c}$.

mBDH assumption. We say the mBDH assumption holds if no probabilistic polynomial-time algorithm can solve the mBDH problem with a non-negligible advantage.

Definition 2. (Quotient Decisional Bilinear Diffie-Hellman (QDBDH) Problem) [21]. *Given* $(g, g^a, g^b) \in G_1$ and $Q \in G_2$ for $a, b \in Z_q^*, g \in G_1$, the QDBDH problem is to determine whether $Q = e(g, g)^{a/b}$ or not.

QDBDH assumption. We say the QDBDH assumption holds if no probabilistic polynomial-time algorithm can solve the QDBDH problem with a non-negligible advantage.

3 System model

In this section, we present an architecture for the EMR system. Moreover, we consider several threats and propose several design goals.

3.1 System architecture

As shown in Fig 1, five entities are involved in this system: patients, doctor, hospital, cloud server, and data users.

Patient. A patient is an entity who needs medical assistance. The patient first needs to register at the hospital to obtain his/her visiting token. When a patient visits a doctor for treatment,

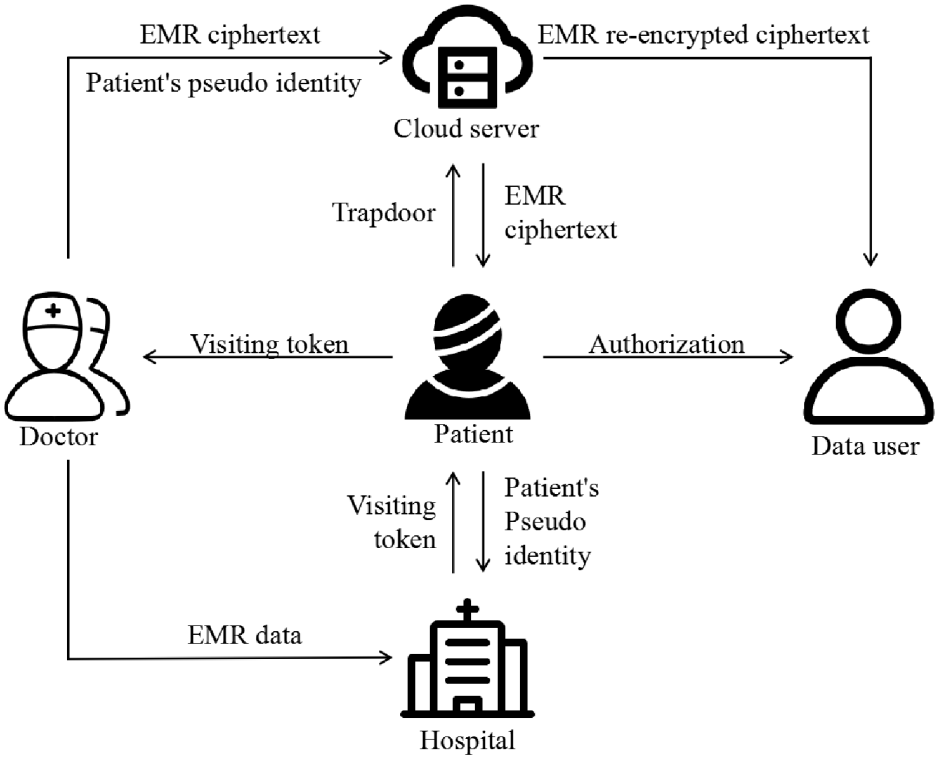

Fig 1. System model.

https://doi.org/10.1371/journal.pone.0244979.g001

his/her health information is generated by the doctor. When the patient's EMR is needed, he/she can access the EMR by sending the trapdoor to the cloud server. In addition, the patient calculates a pseudo-identity for himself/herself and sends it to the hospital.

Doctor. The doctor is an entity responsible for generating the EMR for the patient and uploading them to the hospital. The doctor is also responsible for encrypting the EMR with the patient's public key and sends the ciphertext to the cloud server. When the doctor wants to obtain the patient's historical EMR, the doctor sends a request to the patient. After receiving the EMR from the cloud server, the patient shows the EMR to the doctor.

Hospital. A hospital is an entity that is responsible for generating a visiting token with value τ for the patient and sending the token to the cloud server. The hospital is also responsible for calculating the true identity of the patient required by the data user.

Cloud server. The cloud server is an entity that takes responsibility for storing the patient's encrypted EMR ciphertext and providing the function of searching EMR. After receiving the trapdoor from the patient, the cloud server performs the search operation on EMR. The cloud server generates the re-encryption key by interacting with data users and patients. Then, the cloud server re-encrypts the EMR ciphertext using the re-encryption key and sends the re-encryption ciphertext to the data user.

Data user. In our scheme, the data user refers to the user authorized by the patient who wants to use the patient's EMR. For example, if a patient's condition is complicated, multiple experts are needed for consultation, and the experts come from different hospitals. After interacting with the patient and the cloud server, the data user receives the re-encryption ciphertext sent by the cloud server. The data user can decrypt it using his/her private key.

3.2 Threat model and design goals

In this study, we consider a semi-trust server that has been widely utilized in existing work. Specifically, the server honestly searches information for the benefit of patients, but curiously learns the underlying meaning of the sender's EMR. In addition, malicious outside attackers may intercept and analyze the information transferred in the public channel. Based on the preceding system architecture and threat model, the design goals of our scheme are as follows:

1. Data confidentiality and integrity. Whether the EMR is stored on the hospital server or transmitted through the public channel, no entity can retrieve or modify the EMR data.
2. Access control. The EMR data belongs to the patients who can control data access. In other words, only authorized users have the right to access the data. Simultaneously, data access activities should always be carried out with the participation and monitoring of patients and hospitals.
3. Secure search. When the doctor wants to access the patient's history EMR to improve diagnosis, the patient generates a trapdoor to search the EMR. During the process, only patients can generate the trapdoor. Moreover, the pseudo-identity of the patient is used in the search process, so the eavesdropper cannot deduce the real identity of the patient.
4. Privacy preservation. As the EMR data contains privacy-sensitive information of the patient, the patient's identity must be kept secret.

3.3 Algorithm description

The proposed scheme is composed of nine polynomial-time algorithms:

Setup$(1^{\lambda}) \rightarrow PP$: The algorithm takes a security parameter 1^{λ} as input, and outputs the public parameters PP.

KeyGen$(PP) \rightarrow (pk, sk)$: Given the public parameters PP, the algorithm outputs a public/private key pair (pk, sk).

Enc$(pk_P, W, M) \rightarrow C$: The algorithm inputs a public key of user P, an electronic medical record M, a keyword set $W = (w_1, \cdots, w_n)$, outputs an original ciphertext C.

Trapdoor$(sk_P, Q) \rightarrow T_{w'}$: Takes a private key of user P and a query keyword set $Q = (w_1^{'}, \cdots, w_n^{'})$ as input, the algorithm outputs a keyword trapdoor set $T_{w'} = (T_{w_1^{'}}, \cdots, T_{w_n^{'}})$.

Search$(T_{w'}, C) \rightarrow 1 \ or \ 0$: Given a trapdoor set $T_{w'} = (T_{w_1^{'}}, \cdots, T_{w_n^{'}})$ and a ciphertext C, the algorithm outputs 1 if $w_i^{'} = w_i$ for $1 \leq i \leq n$, or 0 otherwise.

Dec$_1$ $(sk_P, C) \rightarrow M$: The algorithm takes a private key of user P and an original ciphertext C, and output a record M if each input parameter is correct.

ReKeyGen$(sk_P, sk_R) \rightarrow rk_{P \rightarrow R}$: Given user P's private key sk_P and user R's private key sk_R, the algorithm outputs a re-encryption key $rk_{P \rightarrow R}$. This process is performed by user P, user R and the cloud server.

ReEnc$(rk_{P \rightarrow R}, C) \rightarrow C'$: Takes a re-encryption key $rk_{P \rightarrow R}$ from user P to user R and an original ciphertext C for user P, the algorithm converts the ciphertext C to C' for user R.

Dec$(sk_R, C') \rightarrow M$: The algorithm takes a private key of user R and a re-encryption ciphertext C', and output a record M if each input parameter is correct.

4 EMR sharing

4.1 Overview of scheme

Without loss of generality, we assume that a patient P registers to a hospital for medical assistance, and the hospital generates a visiting token τ for the patient and sends it to the patient. Here, τ works as the authorization for the doctor to generate EMR for the patient P. Meanwhile, the patient P computes a pseudo identity ID_P for himself/herself and returns it to the hospital. The hospital packs the tuple (ID_P, τ) and sends it to the cloud server. After the patient P physically visits the doctor, he/she provides τ to the doctor as accordance for generating his/her EMR. We assume that the doctor generates health record M for the patient P by the interaction. To safely store the data with interoperability, the doctor extracts a keyword set $W = (w_1, \cdots, w_n)$ for the EMR. Then, the doctor encrypts M and W with the patient's public key pk_P. The ciphertext $C = (C_M, C_W)$ is stored in the cloud server, where C_M is the ciphertext of EMR M and C_W is the ciphertext of keyword set W.

When the patient P visits another doctor in a different hospital, the doctor may think it is necessary to know the patient's history health record. The patient P can send an access request that includes keyword trapdoor to the cloud server. If the access request is valid, the cloud server sends the patient P the ciphertext C_M. The patient P can decrypt C_M with his/her private key to obtain the health record M. Then, the patient shows it to the doctor.

If the data user R wants to access the EMR of patient P, then he/she sends an interactive request to the patient and the cloud server. After the interaction, the cloud server generates a re-encryption key. The cloud server uses this key to re-encrypt the EMR ciphertext and obtains the re-encryption ciphertext. Then, the cloud server sent it to the data user R. The data user R uses his own private key to decrypt the re-encryption ciphertext. If the data user wants to obtain the true identity of the patient P, he/she can send a request to the hospital.

4.2 Our scheme

In this section, we introduce the details of our proposed scheme. The entities in our scheme involved at least one of the algorithms mentioned in "algorithm definition". Roughly, our

proposed scheme is composed of four main phases: initialization, data processing, search, and record retrieval.

Phase 1: Initialization. In this phase, the system generates the public parameter PP by operating the algorithm $Setup(1^{\lambda})$, where 1^{λ} is the security parameter. All the patients P, doctors D, and data users R generate their private and public keys by running the algorithm $KeyGen(PP)$.

Setup(1^{λ}): Select two bilinear groups (G_1, G_2) of prime order q and a bilinear map e. Pick g as a generator of G_1 and set $Z = e(g, g)$. Select four hash functions $H_1: G_1 \rightarrow \{0, 1\}^*$, $H_2: \{0, 1\}^* \rightarrow G_1$, $H_3: G_2 \rightarrow \{0, 1\}^{log2q}$, $H_4: G_2 \rightarrow \{0, 1\}^*$. Thus, the public parameter can be denoted as $PP = \{G_1, G_2, g, q, e, Z, H_1, H_2, H_3, H_4\}$.

KeyGen(PP): Each patient P randomly selects a secret value $p \in Z_q^*$ as its private key sk_P and computes the public key $pk_P = g^p$. Each doctor D randomly chooses a secret value $d \in Z_q^*$ as its private key sk_D and computes the public key $pk_D = g^d$. Each data user R randomly selects a secret value $r \in Z_q^*$ as its private key sk_R and computes the public key $pk_R = g^r$.

When the patient P registers at the hospital, the hospital randomly selects $\beta \in \{0, 1\}^*$ and computes $\tau = g^{1/\beta}$. Then, the hospital sends the token τ to the patient P securely. Meanwhile, the patient randomly selects $s \in Z_q^*$ and computes $S = g^s$. Thereafter, the patient calculates his/her pseudo identity $ID_P = RID_P \oplus H_1(\tau^s)$ where RID_P is the real identity of the patient P. The patient P returns the tuple (τ, S, ID_P) to the hospital. The hospital chooses a doctor D for the patient and sends the tuple to the cloud server with the doctor.

Phase 2: Data encryption and storage. As a patient P sees a doctor D for medical assistance, he/she shows the doctor token τ, which works as a proof of the patient's authorization to the doctor for generating his/her EMR. After interaction with the patient P, the doctor D generates health record $M \in G_2$ and extracts a keyword set $W = (w_1, \cdots, w_n)$ from the record. Then, the doctor stores M in the hospital and encrypts M and W with the patient's public key pk_P by operating the algorithm $Enc(pk_P, W, M)$.

Enc(pk_P, W, M): The doctor randomly selects a value $k \in Z_q^*$ and computes $C_1 = M \cdot Z^k$, $C_2 = pk_P^k$, $C_3 = H_4(C_1)$, $t = e(\sum_{i=1}^{n} H_2(w_i), g)^k$ for $1 \leq i \leq n$.

The output of encryption algorithm is $C = (C_M, C_W)$, where $C_M = (C_1, C_2)$ and $C_W = (t, H_3(t))$. Here, C_M is the record ciphertext and C_W is the keyword index. The doctor sends the ciphertext C and the patient's pseudo identity ID_P to the cloud server. To match the patient's token in the cloud server, the doctor performs the following operations:

- Randomly chooses value $a \in Z_q^*$ and computes $\alpha = g^{\frac{a+d}{H_1(\tau)}}$, $\tau' = H_1(g^a) \oplus H_1(\tau)$.

The doctor sends (α, τ') to the cloud server. Then, the cloud server checks whether the equation $H_1(\tau^*) = H_1(\tau)$ holds or not, where $H_1(\tau^*) = H_1(\alpha^{H_1(\tau)} \cdot pk_D^{-1}) \oplus \tau'$. If the equality holds, the EMR ciphertext C successfully matches the token τ of the patient P. The cloud server stores the ciphertext C and ID_P together.

Correctness:

$$\begin{aligned} H_1(\tau^*) &= H_1(\alpha^{H_1(\tau)} \cdot pk_D^{-1}) \oplus \tau' \\ &= H_1((g^{\frac{a+d}{H_1(\tau)}})^{H_1(\tau)} \cdot g^{-d}) \oplus H_1(g^a) \oplus H_1(\tau) \\ &= H_1(g^a) \oplus H_1(g^a) \oplus H_1(\tau) \\ &= H_1(\tau) \end{aligned}$$

Phase 3: Search. This phase is divided into two steps: trapdoor generation and test. On another day, the patient may visit another doctor in a different hospital. During the interaction process of the doctor and the patient, the doctor may find that it is necessary to access the patient's history record for a more accurate diagnosis. To search over the encrypted record C, the patient P needs to compute the trapdoor set for a query keyword set $Q = (w_1', \cdots, w_n')$ by invoking the algorithm $Trapdoor(sk_P, Q)$.

Trapdoor(sk_P, Q): The patient P computes $T_{w'} = \left\{ T_{w_i'} = H_2(w_i')^{\frac{H_1(\tau)}{p}}, 1 \leq i \leq n \right\}$.

Meanwhile, the patient P sets an effective access time tr for this request [22], and then sends a tuple $(tr, T_{w'})$ to the cloud server.

The cloud server checks the validity of tr after receiving the tuple. If tr is not effective, the message is ignored. Otherwise, the cloud server performs $Search\ (T_{w'}, C)$ to check whether the encrypted record C involves the keyword set Q. Precisely, for each w_i in Q, the cloud server checks whether the equation $H_3(\prod_{i=1}^{n} e(C_2, T_{w_i'})) = H_3(t^{H_1(\tau)})$ holds or not. If the equality holds, then the cloud server sends EMR ciphertext C_M to the patient P. Otherwise, it sends $\perp$.

Correctness:

$$\begin{aligned} H_3(\prod_{i=1}^{n} e(C_2, T_{w_i'})) &= H_3(\prod_{i=1}^{n} e(g^{p \cdot k}, H_2(w_i')^{\frac{H_1(\tau)}{p}})) \\ &= H_3(\prod_{i=1}^{n} e(g, H_2(w_i'))^{k \cdot H_1(\tau)}) \\ &= H_3(e(g, \sum_{i=1}^{n} H_2(w_i'))^{k \cdot H_1(\tau)}) \\ &= H_3(t^{H_1(\tau)}) \end{aligned}$$

Phase 4: Record retrieval. This phase involves two cases: the patient decrypts EMR and the data user decrypts EMR.

Case 1: The patient decrypts EMR.

Upon receiving EMR ciphertext C_M from the cloud server, the patient P decrypts the ciphertext C_M to retrieve the record M by invoking the algorithm $Dec_1(sk_P, C)$.

Dec(sk_P, C): The patient P calculates $M = \frac{C_1}{e(g, C_2)^{\frac{1}{p}}}$.

After obtaining the EMR M, the patient P shows it to the doctor.

Correctness:

$$\begin{aligned} \frac{C_1}{e(g, C_2)^{\frac{1}{p}}} &= \frac{M \cdot Z^k}{e(g, g^{p \cdot k})^{\frac{1}{p}}} \\ &= \frac{M \cdot e(g, g)^k}{e(g, g^k)} \\ &= M \end{aligned}$$

Case 2: The data user decrypts EMR.

To obtain the patient P's EMR, the data user R first requests the patient P and cloud servers to interact with him/her. The cloud server generates the re-encryption key by running the algorithm $ReKeyGen(sk_P, sk_R)$. More precisely, the re-encryption key is generated by the following steps:

- The patient P randomly chooses value $j \in Z_q^*$. Then, the patient P sends j to the cloud server and $sk_P \cdot j$ to the data user R.
- After receiving $sk_P \cdot j$ from the patient P, the data user R sends $sk_R/(sk_P \cdot j)$ to the cloud server.
- Finally, the cloud server computes the re-encryption key $rk_{P\to R} = r/p$.

Then, the cloud server re-encrypts the EMR ciphertext C_M with $rk_{P\to R}$ to generate the re-encryption ciphertext $C_M^{'}$ for the data user R by running the algorithm $ReEnc(rk_{P\to R}, C)$.

ReEnc$(rk_{P\to R}, C)$: The cloud server computes $C_1^{'} = C_1$, $C_2^{'} = C_2^{rk_{P\to R}\cdot H_4(C_1^{'})} = g^{r\cdot k\cdot H_4(C_1^{'})}$, $C_3^{'} = C_3$.

The cloud server sets the re-encryption ciphertext $C_M^{'} = (C_1^{'}, C_2^{'}, C_3^{'})$ and sends it to the data user R. After receiving EMR re-encryption ciphertext $C_M^{'}$ from the cloud server, the data user R decrypts it to retrieve the record M by invoking the algorithm $Dec_2(sk_R, C)$.

Dec(sk_R, C): The data user R calculates $M = C_1^{'}/e(g, C_2^{'})^{1/(r\cdot C_3^{'})}$.

When the real identity of the patient P needs to be obtained for treatment or medical insurance purposes, the data user R sends a request to the hospital. The hospital obtains the true identity of the patient P by calculating $RID_P = ID_P \oplus H_4(S^{1/\beta})$ and returns it to the data user R. In our scheme, only the hospital system knows the β value, so only the hospital can extract the real identity of the patient.

Correctness:

$$\begin{aligned}\frac{C_1^{'}}{e(g, C_2^{'})^{\frac{1}{r\cdot C_3^{'}}}} &= \frac{M\cdot Z^k}{e\left(g, g^{r\cdot k\cdot H_4(C_1)}\right)^{\frac{1}{r\cdot H_4(C_1)}}}\\ &= \frac{M\cdot e(g,g)^k}{e(g,g)^k}\\ &= M\end{aligned}$$

5 Security analysis

5.1 Achieving goals

In this section, we illustrate how the proposed scheme can effectively achieves the design goals presented in "System Model".

The proposed scheme achieves data confidentiality and integrity. The EMR data are encrypted before being outsourced to the hospital server. The doctor uses the patient's public key to encrypt the EMR. On the one hand, the patient uses his/her private key to decrypt the EMR ciphertext; on the other hand, the data user authorized by the patient uses his/her private key to decrypt the EMR re-encryption ciphertext.

The proposed scheme achieves access control. As mentioned in phase 4, if the data user wants to access the patient's EMR, he/she first sends an authorization request to the patient.

After the patient agrees, the cloud server generates a re-encryption key. The cloud server re-encrypts the EMR ciphertext with it to generate the re-encryption ciphertext that the data user can decrypt with his/her private key.

The proposed scheme achieves secure search. In phase 2 of our scheme, the EMR is encrypted with keyword search. In phase 3, the patient generates the trapdoor set to search his/her history health record to improve the doctor's diagnosis of the patient. In this scenario, the keyword trapdoor $T_{w'} = \{T_{w'_i} = H_2(w'_i)^{H_1(\tau)/p}, 1 \leq i \leq n\}$ contains the patient's private key, so only the patient can generate the trapdoor and perform search on EMR.

5.2 Security proof

As the data used by the patient is similar to that of the data user, we only demonstrate the safety of data used by data users.

Theorem 1. *Our scheme is IND-CKA secure in the random oracle model, if mBDH assumption holds in* G_1 and G_T.

Proof. We assume the existence of a polynomial-time adversary A_1 with non-negligible advantage $\epsilon(k)$ in attacking the privacy for keywords of our scheme, where $\epsilon(k)$ is a negligible function in the security parameter k. We construct a simulator B that can compute the solution of the mBDH problem.

Let $(g, g^{\alpha}, g^{\beta}, g^{\gamma} \in G_1)$ be an instance of the mBDH problem, where g is the generator of G_1 and $\alpha, \beta, \gamma \in Z_q^*$ are uniformly random choices. The goal of B is to output $e(g, g)^{\alpha\beta/\gamma} \in G_2$ by interacting with A_1 as follows:

***H_1* query**: B maintains an empty-initial table H_1^{list}. Input w in the hash function H_1, and B checks H_1^{list}. If $<w_i, h_i, a_i, c_i>$ exists in H_1^{list}, then B returns $H_1(w_i) = h_i$. Otherwise, B generates a random coin c_i such that $pr[ci = 0] = 1/(q_T + 1)$, where $c_i \in \{0, 1\}$, q_T is the maximum number of Trapdoor queries. B selects a random number $a_i \in Z_q^*$. If $c_i = 0$, B returns $h_i = (g^{\alpha})^{a_i}$ to A_1; if $c_i = 1$, B returns $h_i = (g^{\gamma})^{a_i}$ to A_1. Thereafter, B adds $<w_i, h_i, a_i, c_i>$ to H_1^{list}.

***H_2* query**: B maintains an empty-initial table H_2^{list}. Upon receiving H_2 query about $t' \in G_2$ from A_1, B checks H_2^{list}. If t' already exists in H_2^{list}, B returns V to A_1. Otherwise, B selects a value $V \in \{0, 1\}^{log}{}_2\, q$ randomly and adds $<t', V>$ to H_2^{list} by setting $H_2(t') = V$.

Phase 1. A_1 makes several queries.

Uncorrupted key query: On input an index i, B selects $x_i \in Z_q^*$ randomly and outputs the public key $pk_i = (g^{\gamma})^{x_i}$. Thus, the private key is defined as $sk_i = \gamma x_i$ implicitly. B adds $<i, pk_i, x_i>$ to L_U.

Corrupted key query: On input an index i, B selects $x_i \in Z_q^*$ randomly and outputs the public key $pk_i = g^{x_i}$. Thus, the private key is defined as $sk_i = x_i$ implicitly. B adds $<i, pk_i, x_i>$ to L_U.

Trapdoor query: When A_1 makes a trapdoor query on the keyword w_i, B responds as follows:

- B recovers $<w_i, h_i, a_i, c_i>$,$<i, pk_i, x_i>$,$<i, pk_i, x_i>$ from H_1^{list}, L_U, L_C, respectively.
- If $c_i = 0$, B aborts. Otherwise, B computes $h_i = (g^{\gamma})^{a_i}$ when $c_i = 1$.
- If $i \in L_C$, B computes $T_i = h_i^{N/x_i} = (g^{\gamma})^{Na_i/x_i}$; if $i \in L_C$, B computes $T_i = h_i^{N/(\gamma x_i)} = ((g^{\gamma})^{a_i})^{N/(\gamma x_i)} = g^{Na_i/x_i}$. Then, T_i is the trapdoor for keyword w_i and B returns T_i to A_1.

Re-encryption key query: When A_1 asks B about the re-encryption key $rk_{i\to j}$ for two public keys pk_i, pk_j, B responds as follows:

- If neither pk_i nor pk_j belongs to L_C, B aborts.
- Otherwise, B returns $rk_{i\to j} = x_j/x_i$ to A_1.

Challenge: Eventually, A_1 issues a challenge on two keywords w_0, w_1, a message m, and a public key pk_i. If pk_i belongs to L_C, then B aborts. Otherwise, B performs as follows:

- B conducts two H_1 queries to obtain $h_0, h_1 \in G_1$ such that $H_1(w_0) = h_0$, $H_1(w_1) = h_1$. If both $c_0 = 1$ and $c_1 = 1$ hold, then B aborts.
- Otherwise, at least one of c_0 and c_1 is equal to 0. Then B randomly picks $b \in \{0, 1\}$ so that $c_b = 0$.
- B returns $C_b^{'} = (g^{\beta x_i, V})$ to A_1, where $V \in \{0, 1\}^{log}{}_2\, q$.
- B implicitly defines $pk_i^{\beta/\gamma} = ((g^{\gamma})^{x_i})^{\beta/\gamma}$ and $V = H_2(e(H_1(w_b), g)^{\beta/\gamma}) = H_2(e(g^{\alpha a_b}, g)^{\beta/\gamma}) = H_2(e((g^{\alpha a_b})^{1/\gamma x_i}, g^{\beta x_i})) = H_2(e(g, g)^{\beta\alpha a_b/\gamma})$.

Phase 2. A_1 can continue to issue several queries as in phase 1 on keyword w_i, where $w_i \neq w_0$ and $w_i \neq w_1$.

Guess: Finally, A_1 outputs its guess $b' \in \{0, 1\}$ to check whether the challenge ciphertext $C_b^{'}$ is the result of keyword w_0 or w_1. Then B chooses the pair (t', V) from H_2^{list} and outputs $(t')^{1/a_b}$ as its guess to $e(g, g)^{\beta\alpha/\gamma}$.

Theorem 2. *Our scheme is IND-CPA secure in the random oracle model, if QDBDH assumption holds in* G_1 and G_T.

Proof. We assume the existence of a polynomial-time adversary A_2 with non-negligible advantage $\epsilon(k)$ in attacking our scheme, where $\epsilon(k)$ is a negligible function in the security parameter k. We construct a simulator B that can compute the solution of the QDBDH problem.

Let $(g, g^a, g^b \in G_1)$ be an instance of the mBDH problem, where g is the generator of G_1 and $a, b \in Z_q^*$ are uniformly random choices. The goal of B is to output $e(g, g)^{a/b} \in G_2$ by interacting with A_2 as follows:

***H_1* query**: B maintains an empty-initial table H_1^{list}. Once receiving H_1 query about $w \in Z_q^*$ from A_2, B checks H_1^{list}. If w already exists in H_1^{list}, B returns h to A_2. Otherwise, B selects a value $h \in \{0, 1\}^{log}{}_2\, q$ randomly and adds $<w, h>$ to H_1^{list} by setting $H_1(w) = h$.

***H_2* query**: B maintains an empty-initial table H_2^{list}. Once receiving H_2 query about $t' \in G_2$ from A_2, B checks H_2^{list}. If t' already exists in H_2^{list}, B returns V to A_2. Otherwise, B selects a value $V \in \{0, 1\}^{log}{}_2\, q$ randomly and adds $<t', V>$ to H_2^{list} by setting $H_2(t') = V$.

Phase 1. A_2 makes several queries.

Public key query: B generates a random coin $c \in \{0, 1\}$. If $c_i = 1$, B selects a random value $x_i \in Z_q^*$ and outputs the public key $pk_i = g^{ax_i}$. Otherwise, B outputs the public key $pk_i = g^{x_i}$ and adds $<c_i, pk_i, x_i>$ to table L_C, where the private key is implicitly defined as $sk_i = x_i$.

Private key query: B recovers $<c_i, pk_i, x_i>$ from L_C. If $c_i = 0$, the private key $sk_d = x_d$ is returned to A_2. Otherwise, it aborts.

Re-encryption key query: The adversary A_2 can adaptively ask B for the re-encryption key $rk_{i\to j}$ for any two public keys pk_i, pk_j and B generates the re-encryption key as follows:

- If $c_i = 1$ and $c_j = 1$, B aborts.
- Otherwise, B responds $rk_{i\to j} = x_j/x_i$ to A_2.

Re-encryption query: Based on the result of re-encryption query, B obtains the re-encryption ciphertext through the re-encryption algorithm and returns it to A_2.

Decryption query: After obtaining the re-encryption ciphertext $C_m^{'} = (C_1^{'}, C_2^{'}, C_3^{'})$, B recovers $<c_i, pk_i, x_i>$ associated with the data user from L_C. If $c_i = 0$, B set the message $m = \frac{C_1^{'}}{e(g, C_2^{'})^{1/(sk \cdot C_3^{'})}}$. Otherwise, B sets the message $m = \frac{C_1^{'}}{e(g^a, C_2^{'})^{1/(sk \cdot C_3^{'})}}$.

Challenge: Eventually, A_2 issues a challenge on two messages m_0, m_1 and a public key pk_i. B recovers the tuple $<c_i, pk_i, x_i>$ from L_C. If $c = 1$, then B reports failure and aborts. Otherwise, B randomly selects $\delta \in \{0, 1\}$ and sets the challenge ciphertext $C_\delta^{'}$ as follows:

$$C_1^* = m \cdot T, C_2^* = (pk^*)^{a/b}$$

Phase 2. A_2 can continue to issue several queries as in phase 1 on message m_i, where $m_i \neq m_0$ and $m_i \neq m_1$.

Guess: Finally, A_2 outputs its guess $b' \in \{0, 1\}$ to check whether the challenge ciphertext $C_\delta^{'}$ is the result of message m_0 or m_1. If $c_\delta = 1$, then the ciphertext $C_1^* = m \cdot T = m \cdot e(g, g)^{a/b}$ is a QDBDH instance.

6 Performance analysis

In this section, we expound a theoretical analysis on the performance of the proposed schemes. Then, we analyze the efficiency of the scheme by numerical simulation. To show the performance more intuitively, we have implemented our scheme, as well as the schemes used by Wu [23] and Wang [24] in the Linux operating system using Pairing-Based Cryptography (PBC) Library [25], programmed in C language, and ran in a virtual machine of a PC (HP PC, 3.1 GHz CPU, and 4 GB RAM). In the experiment, we used elliptical curves with a base field size of 512 bits and an embedding degree of 2. The security levels are selected as $|p| = 512$.

6.1 Theoretical analysis

In this section, we compare the computation overhead of the proposed scheme and other schemes from a theoretical perspective. We denote T_e, T_p, T_h, T_H, T_{mul} as the computation cost of exponentiation operation, bilinear pairing operation, general hash function, hash-to-point operation, and multiplication operation, respectively. The running time of those basic operations are presented in Table 1.

As shown in Table 1, T_H and T_{mul} are much smaller than the others, so the hash-to-point operation time and multiplication operation time are negligible. The descending order time of common cryptographic algorithms is T_e, T_p, T_h, T_H, T_{mul}, and the computational cost of the bilinear pairing operation is much higher than that in other cryptographic algorithms. The computation cost of the proposed schemes in the index generation and search phases is presented in Table 2. We specify n as the number of keywords.

As shown in Table 2, in the index generation phase, the descending order of the computation cost is Wang's scheme [24], our scheme, and Wu's scheme [23]. In the search phase, the descending order of the computation cost is Wang's scheme [24], our scheme, and Wu's scheme [23]. Since Wu's scheme [23] only implements single-keyword encryption and our

Table 1. The running time of basic operations (ms).

T_e	T_p	T_h	T_H	T_{mul}
3.485	3.724	9.714	0.002	0.017

https://doi.org/10.1371/journal.pone.0244979.t001

Table 2. Comparison of computation cost.

scheme	Index generation	Search
Wu's scheme [23]	$T_e + T_h$	$3T_e + 2T_p$
Wang's scheme [24]	$(2n + 1)T_e + nT_p + nT_h$	$(n + 1)T_e + nT_p + nT_h$
Our scheme	$T_e + T_p + nT_h$	$(n + 1)T_e + T_p + nT_h$

https://doi.org/10.1371/journal.pone.0244979.t002

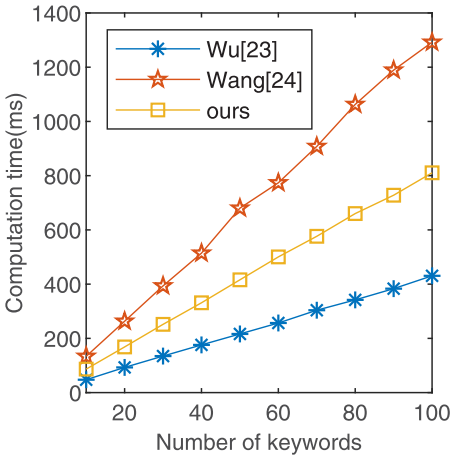

Fig 2. The performance comparison of index generation.

https://doi.org/10.1371/journal.pone.0244979.g002

scheme implements multi-keyword encryption, the computation cost of our scheme in the index generation and search phase is higher than that of Wu's scheme [23].

6.2 Numerical simulation

We compared our scheme with the schemes proposed by Wu [23] and Wang [24] through numerical simulation. Both our scheme and Wang's scheme [24] realize multi-keyword search function in ciphertext, whereas Wu's scheme [23] only realizes single-keyword search. In the numerical simulation, we use the same number of keywords in the index generation and search phases, and compare the computational overhead of different keyword quantities in each phase. We specify the number of keywords as n = 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000. The experimental result is the average time for the algorithm to run 10 times. For more information, see S1 Appendix and S1 File.

As illustrated in Fig 2, index generation time increases with the number of keywords. The index generation time of our scheme is less than that of Wang's scheme [24] but higher than that of Wu's scheme [23]. The reason is that our scheme uses bilinear pairing operations in the keyword encryption process, but Wu's scheme [23] is not used. In Fig 3, we present the time cost of the search phase in all schemes. The time spent linearly increases with the number of keywords. Wu's scheme [23] and our scheme have a subtle difference in the search phase, and both are higher than Wang's scheme [24].

7 Conclusion

We presented an EMR data sharing scheme with privacy protection, secure storage, and secure sharing based on searchable encryption and proxy re-encryption technology, which solves the security problems of data security and personal privacy in the process of EMR sharing based

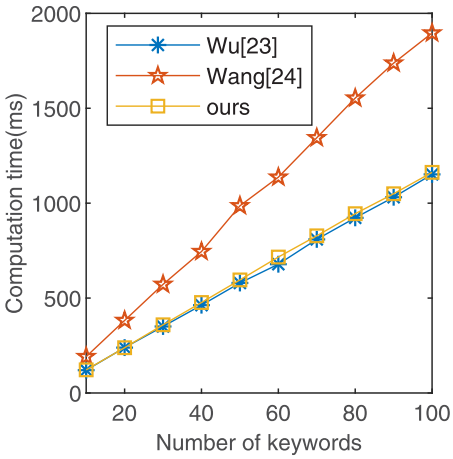

Fig 3. The performance comparison of search.

https://doi.org/10.1371/journal.pone.0244979.g003

on cloud storage. While protecting the privacy of the patient, this scheme enables patients to access their own EMR. After authorization is provided by the patient, the data users can also access the EMR, which is a practical approach. The EMR ciphertext and keyword index are stored in the cloud server to enable the patient to search EMR with keyword search. The cloud server generates a re-encryption key for the data user after the patient authorizes the data user to access his/her EMR. Then, the cloud server re-encrypts the EMR ciphertext with the re-encryption key and sends it to the data user, who can decrypt it using the private key.

Supporting information

S1 Appendix. Data used to build graphs. The experimental data used for plotting in Figs 2 and 3.
(DOCX)

S1 File. Procedure source code. The procedure source code for the numerical simulation of our scheme, Wu's scheme and Wang's scheme.
(ZIP)

Author Contributions

Conceptualization: Shufen Niu, Wenke Liu.

Investigation: Shufen Niu, Wenke Liu.

Methodology: Shufen Niu, Wenke Liu.

Project administration: Wenke Liu.

Resources: Shufen Niu, Wenke Liu.

Software: Shufen Niu, Wenke Liu, Song Han, Lizhi Fang.

Supervision: Shufen Niu.

Validation: Shufen Niu.

Visualization: Wenke Liu.

Writing – original draft: Shufen Niu, Wenke Liu, Song Han, Lizhi Fang.

Writing – review & editing: Shufen Niu, Wenke Liu.

References

1. Au MH, Yuen TH, Liu JK. A general framework for secure sharing of personal health records in cloud system. Journal of Computer and System Sciences, 2017, 90: 46–62. https://doi.org/10.1016/j.jcss.2017.03.002
2. Xia Z, Wang X, Zhang L. A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. IEEE Transactions on Information Forensics and Security, 2017, 11(11):2594–2608. https://doi.org/10.1109/TIFS.2016.2590944
3. Wu CH, Chiu RK, Yeh HM. Implementation of a cloud-based electronic medical record exchange system in compliance with the integrating healthcare enterprise's cross-enterprise document sharing integration profile. International Journal of Medical Informatics, 2017, 107: 30–39. https://doi.org/10.1016/j.ijmedinf.2017.09.001 PMID: 29029689
4. Chenthara S, Ahmed K, Wang H, Whittaker F. Security and privacy-preserving challenges of e-Health solutions in cloud computing. IEEE Access, 2019, 7(99):74361–74382. https://doi.org/10.1109/ACCESS.2019.2919982
5. Vimalachandran P, Zhang Y, Cao J. Preserving data privacy and security in Australian my health record system: A quality health care implication. In: International Conference on Web Information Systems Engineering. Springer, Cham, 2018: 111-120.
6. Fu Z, Huang F, Sun X, Vasilakos A, Yang C. Enabling semantic search based on conceptual graphs over encrypted outsourced data. IEEE Transactions on Services Computing, 2016, 99:1–1.
7. Xia Z, Wang X, Sun X, Wang Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(2):340–352. https://doi.org/10.1109/TPDS.2015.2401003
8. Sun J, Wang X, Wang S, Ren L. A searchable personal health records framework with fine-grained access control in cloud-fog computing. Plos One, 2018, 13(11):e0207543 https://doi.org/10.1371/journal.pone.0207543 PMID: 30496194
9. Shi Y, Liu J, Han Z, Zheng Q. Attribute-based proxy re-encryption with keyword search. Plos One, 2014, 9(12):e116325. https://doi.org/10.1371/journal.pone.0116325 PMID: 25549257
10. Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE Symposium on Security and Privacy. IEEE, 2000: 44-55.
11. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2004: 506-522.
12. Baek J, Safavi-Naini R, Susilo W. Public key encryption with keyword search revisited. In: International Conference on Computational Science and Its Applications. Springer, Berlin, Heidelberg, 2008: 1249-1259.
13. Xu P, Jin H, Wu Q. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. IEEE Transactions on Computers, 2012, 62(11): 2266–2277. https://doi.org/10.1109/TC.2012.215
14. Liu X, Xia Y, Yang W, Yang F. Secure and efficient querying over personal health records in cloud computing. Neurocomputing, 2018, 274: 99–105. https://doi.org/10.1016/j.neucom.2016.06.100
15. Li X, Song Z, Ren J, Xu L. Attribute-based searchable encryption of electronic medical records in cloud computing. Chinese Computer Science, 2017, 44(Z11): 342–347.
16. Ma M, He D, Khan MK, Chen J. Certificateless searchable public key encryption scheme for mobile healthcare system. Computers and Electrical Engineering, 2018, 65: 413–424. https://doi.org/10.1016/j.compeleceng.2017.05.014
17. Shao J, Cao Z, Liang X, Lin H. Proxy re-encryption with keyword search. Information Sciences, 2010, 180(13): 2576–2587. https://doi.org/10.1016/j.ins.2010.03.026
18. Guo L, Lu B. Efficient proxy re-encryption with keyword search scheme. Chinese Journal of Computer Research and Development, 2014, 51(6): 1221–1228.
19. Chen Z, Li S, Guo Y, Wang Y, Chu Y. A limited proxy re-encryption with keyword search for data access control in cloud computing. In: International Conference on Network and System Security. Springer, Cham, 2015: 82-95.
20. Sahai A, Waters B. Fuzzy identity-based encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2005: 457-473.

21. Ateniese G, Fu K, Green M, Hohenberger S. Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security, 2006, 9(1): 1–30. https://doi.org/10.1145/1127345.1127346

22. Wang X, Zhang A, Xie X, Ye X. Secure-aware and privacy-preserving electronic health record searching in cloud environment. International Journal of Communication Systems, 2019, 32(8):e3925.1–e3925.11. https://doi.org/10.1002/dac.3925

23. Wu Y, Lu X, Su J, Chen P. An efficient searchable encryption against keyword guessing attacks for sharable electronic medical records in cloud-based system. Journal of Medical Systems, 2016, 40(12): 258. https://doi.org/10.1007/s10916-016-0609-z PMID: 27722976

24. Wang T, Au M H, Wu W. An efficient secure channel free searchable encryption scheme with multiple keywords. In: International Conference on Network and System Security. Springer, Cham, 2016: 251-265.

25. The pairing-based cryptography library. Available from: http://crypto.stanford.edu/pbc/.