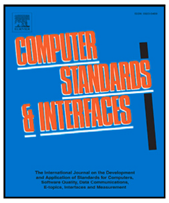
Computer Standards & Interfaces 87 (2024) 103746

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi



Efficient and privacy-preserving similar electronic medical records query for large-scale ehealthcare systems

Chang Xu [a,*], Zijian Chan [b], Liehuang Zhu [a], Rongxing Lu [c], Yunguo Guan [c], Kashif Sharif [b]

[a] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
[b] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[c] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada

ARTICLE INFO

Keywords:
Privacy-preserving
Similar EMRs querying
Large-scale ehealthcare systems

ABSTRACT

The advancements and adoption of cloud-assisted ehealthcare systems have enabled the storage of massive electronic medical records (EMRs) in the cloud for efficient and easy access. A direct benefit of EMRs is the ability of patients to search for EMRs that are similar to their own in the cloud for use as references. These similar EMRs can help a patient find appropriate medical services quickly. However, for large-scale ehealthcare systems, challenges remain with respect to ensuring the efficiency and privacy of these queries. In this study, we construct an efficient and privacy-preserving similar EMR query scheme to help patients find similar EMRs to reference in a large-scale ehealthcare system. Specifically, we propose a coarse-grained query method based on a binary decision tree to find a set of EMRs corresponding to the patient's set of medical-symptom keywords. We also design a fine-grained query method to find similar EMRs that meet the threshold set by the patient. A detailed security analysis shows that the proposed scheme is secure. The efficiency of the proposed method in a large-scale ehealthcare system is verified experimentally.

1. Introduction

With the rapid progress of cloud computing and the Internet of Things (IoT), the traditional healthcare industry is moving toward a more efficient and flexible ehealthcare paradigm. This evolution provides a platform for effectively sharing health care data among different stakeholders [1]. In this platform, traditional paper-based records can be converted into digitalized electronic records, such as electronic medical records (EMRs), which typically consist of a wide variety of data (e.g., symptom information, health data, and medical service records). In the current era of big data, a growing number of EMRs are generated, and some large-scale ehealthcare systems must be combined with cloud computing to reduce storage and computation costs [2].

The cloud-assisted ehealthcare system enables efficient EMR querying. After patients register in a cloud-assisted ehealthcare system, they can find appropriate medical services quickly by searching for similar EMRs. Specifically, a patient can deploy portable sensors to form a wireless body sensor network, and the sensors collect health data (e.g., blood pressure and blood sugar). Then, the collected health data and the patient's health symptoms (e.g., cough and fever) will be delivered to the cloud server, which could help the patient find similar EMRs by calculating the squared Euclidean distance between data vectors. Finally, with EMRs that are similar to their own, patients can more easily find appropriate medical services. However, the privacy of EMRs and trapdoors should be considered due to the sensitive nature of health care data.

Encrypting data before transmission to a cloud server is the standard method to preserve the privacy of information but is inconvenient during querying [3]. Thus, a privacy-preserving model that can query encrypted data is required. A review of existing encrypted data query methods shows that most searchable encryption schemes are based on the keyword-match query or keyword-range query, which are coarse-grained or primarily targeted at spatial data. Therefore, these searchable encryption schemes cannot be directly applied to help patients find similar EMRs. A more appropriate method is to query similar EMRs based on health symptom keywords and health data similarity. However, most existing methods cannot achieve a better balance between security, accuracy, and efficiency, which is unacceptable in large-scale ehealthcare systems. Thus, we design an efficient and privacy-preserving similar EMR query scheme for a large-scale ehealthcare system to help patients find similar EMRs for use as references. In our proposed scheme, we introduce an efficient and privacy-preserving

* Corresponding author.
E-mail addresses: xuchang@bit.edu.cn (C. Xu), 3220190776@bit.edu.cn (Z. Chan), liehuangz@bit.edu.cn (L. Zhu), RLU1@unb.ca (R. Lu), yguan4@unb.ca (Y. Guan), kashif@bit.edu.cn (K. Sharif).

https://doi.org/10.1016/j.csi.2023.103746
Received 20 June 2022; Received in revised form 9 March 2023; Accepted 7 April 2023
Available online 12 April 2023

coarse-grained query method based on a binary decision tree. Our method supports multi-keyword queries, is non-interactive, and provides trapdoor unlinkability. Furthermore, our performance analysis shows that our matching method, which is designed with a tree, is more efficient than the SSE scheme that does not utilize a tree.

1.1. Related works

Searchable encryption was first introduced in [4]. To enable secure searching, outsourced data is encrypted before being stored in the cloud server [5]. In this section, we review the related literature about searchable encryption.

Keyword query over encrypted data: Most existing studies on searchable encryption are based on keyword queries, which can be divided into several categories, including match query and range query, based on the query predicate. Many existing studies focused on multi-keyword rank searchable encryption (MRSE) for keyword match queries. Yin et al. [6] introduced a secure multi-owner MRSE scheme. Specifically, a secure single keyword search scheme and a conjunctive MRSE scheme for multiple data owners are constructed. Yang et al. [7] presented a multiuser MRSE scheme based on the K-nearest neighbor searchable encryption (KNN-SE) algorithm. Unfortunately, Yang's scheme is limited to one given language and cannot meet the requirements of semantic search. To solve these problems, Guan et al. [8] constructed a cross-lingual MRSE scheme, which can eliminate the language barriers in searchable encryption. These MRSE models can perform keyword match queries in a privacy-preserving way, but the keyword match query is coarse-grained and unsuitable for fine-grained similar EMR queries. For keyword range queries, Wang et al. [9] and Zhu et al. [10] both presented a circular range query scheme. These schemes can support secure range queries over encrypted spatial data within a given circular range. Xu et al. [11] constructed a geometric range query scheme that supports access control while searching over encrypted spatial data. Wang et al. [12] introduced two Boolean range query schemes to achieve keyword queries over encrypted spatial data. However, these keyword range query schemes are primarily targeted at spatial data and cannot be used to perform EMR queries.

Similarity query over encrypted data: Recently, some schemes have been proposed to enable similarity searching over encrypted data. Elmehdwi et al. [13] and Rane et al. [14] both presented a similarity query scheme based on homomorphic encryption, which can achieve high security. However, their schemes are inefficient in large-scale ehealthcare systems due to the high cost of homomorphic encryption. Lin et al. [15] and Wang et al. [16] used locality-sensitive hashing (LSH) to construct a similarity query scheme. However, in their schemes, the search accuracy is not sufficient. Wang et al. [17] applied wildcard-based techniques to achieve a similarity query focusing on system security and usability. However, each keyword must generate a fuzzy set, which can lead to higher storage overhead. Recently, Jin et al. [18] proposed an efficient and privacy-preserving similarity range query scheme over encrypted genomic sequences based on the Paillier cryptosystem, which is inefficient in large-scale ehealthcare systems. Song et al. [19] designed a geometric range query scheme and a multidimensional spatial keyword similarity search scheme with access control. Unfortunately, their schemes primarily focus on spatial keyword search, which is unsuitable for similar EMR queries.

1.2. Contributions of this study

This paper presents a privacy-preserving similar EMR query scheme that can efficiently help patients find similar EMRs in a large-scale eHealthcare system. The primary contributions of this study are summarized below:

- Unlike the existing EMR query schemes, we focus on helping patients find similar EMRs as a reference. Specifically, the proposed scheme consists of a coarse-grained query and a fine-grained query. The coarse-grained query first finds a collection of EMRs corresponding to the patient's set of health symptom keywords. Then, the fine-grained query can find similar EMRs that meet the threshold.
- We also focus on the challenges of efficiency and privacy in large-scale ehealthcare systems. In the proposed scheme, we use enhanced ASPE to implement EMR querying in an efficient and privacy-preserving way. To capture the coarse-grained query, a novel privacy-preserving health symptom keyword match query based on a binary decision tree is designed. Also, the performance analysis shows that the matching method designed with a tree is markedly more efficient than the method without a tree. To realize the fine-grained query, we efficiently calculate the squared Euclidean distances between the patient's health data vector and the health data vectors in the matching collection of EMRs based on enhanced ASPE. Unlike other similarity calculation methods such as [20,21], the output only shows whether the distance is within the threshold.
- We implement the proposed scheme based on real-world and simulated data. The experimental results show that the proposed scheme is highly efficient for a large-scale ehealthcare system. For example, a coarse-grained query can be performed within 0.1 ms, and a fine-grained query over 100,000 health data only requires approximately 1.2 s on a personal computer.

Table 1
List of notations.

Notation	Definition
σ	secret key generated by the healthcare center
d	maximum number of keywords allowed in the set of health symptom keywords
l	number of health symptom keywords presented by the patient, $l \le d$
t	size of health symptom keyword dictionary
n	dimension of health data vector
L_i	health symptom keyword vector of ith keyword in the dictionary
A_j	health data vector of jth EMR
B_c	coarse-grained query vector of patient c
Q_c	fine-grained query vector of patient c
Θ_c	similarity threshold of patient c
$\alpha_i L_i^*$	ciphertext for L_i
$\beta_j e A_j^*$	ciphertext for A_j
$r_c B_c^*$	trapdoor for the coarse-grained query
$R_c e Q_c^*$	trapdoor for the fine-grained query
T	binary decision tree for the coarse-grained query
C_k	collection of EMRs corresponding to the same health symptom keyword set
E_c	search result of patient c

2. Models and design goals

In this section, we introduce the system and the threat models and elaborate on the design goals for the proposed scheme. Table 1 shows a list of notations used in this paper.

2.1. System model

Fig. 1 shows the system model, which primarily involves three entities, the health care center, cloud server, and patients.

- *The healthcare center* is the data owner, which can provide encrypted data (i.e., encrypted EMRs, health symptom keywords, health data, and the coarse-grained query tree) and secret keys.

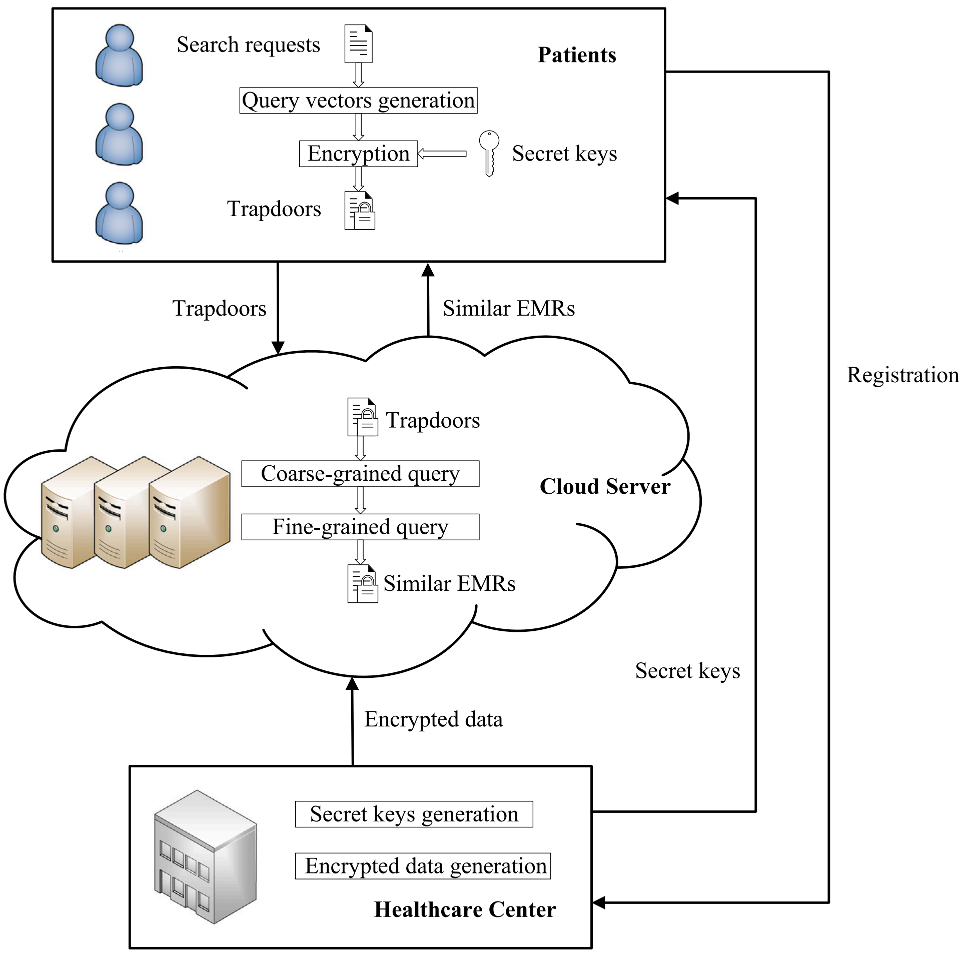


Fig. 1. System model and entity interactions.

Specifically, the healthcare center encrypts the desensitized EMRs using the AES encryption algorithm and generates secret keys to encrypt the health symptom keywords and health data extracted from EMRs. Then, a binary decision tree for a coarse-grained query is constructed using these data. Finally, the health care center stores the encrypted data in the cloud server and transmits secret keys to newly registered patients.

- *The cloud server* has a virtually unbounded storage space and computation abilities, which can provide data storage, query, and computing services for healthcare centers or patients.
- *Patients* are data users and want to obtain EMRs similar to their own as references. After receiving the secret keys from the health care center, a patient uses the secret keys to generate two trapdoors for the coarse-grained and fine-grained queries, which are then forwarded to the cloud server. After the cloud server returns the search result, the patient decrypts the encrypted EMRs using the symmetric key.

2.2. Threat model

The threat model associated with each entity in the above system model is defined in this study.

- *Healthcare centers* are considered fully trusted and cannot be compromised.
- *The cloud server* is regarded as an honest-but-curious entity that can provide reliable services but also tries to understand private information about encrypted data and trapdoors [22,23]. In addition, we follow similar assumptions as in [24,25]. The cloud server cannot collude with patients. Some methods [26,27] have been used to resist collusion attacks, and the proposed scheme can be extended based on these schemes.
- *Patients* are also considered fully trusted, which has been assumed in many previous studies of SE [12,28]. They perform their operations precisely. The secret keys are not disclosed to other entities either actively or passively.

Based on this information, we consider the known sample model [29], where in addition to the encrypted data and trapdoors, the cloud server still has some samples of health symptom keywords, health data, and a trapdoor in plaintext. However, the cloud server cannot learn the corresponding encrypted value.

2.3. Design goals

Given the above system and threat models, we aim to construct an efficient and privacy-preserving similar EMR query scheme for large-scale ehealthcare systems. Particularly, the requirements listed below should be met.

- *Privacy protection of EMR:* Considering the privacy of health care data, the EMRs should be encrypted using the AES encryption algorithm before being stored in the cloud server. For each EMR, access is restricted to the health care center and authorized patients.
- *Privacy protection of health symptom keywords, health data, and trapdoors:* Health symptom keywords, health data, and trapdoors can help the cloud server with efficient matching in the proposed

scheme. However, all of these are related to health care data. Considering the sensitivity of health care data, the patient would suffer heavy losses, and the reputation of the health care center would be damaged if privacy is compromised. Thus, it is essential to protect the privacy of the health symptom keywords, health data, and trapdoor.

- *Unlinkability of trapdoor:* For security, the same search request should generate different trapdoors each time to prevent the cloud server from learning the relationship between trapdoors. Otherwise, the cloud server can perform statistical analysis and learn the keywords with background knowledge. Therefore, the encryption function for the search request should be randomized rather than deterministic.
- *Support EMRs updating:* In an ehealthcare system, frequent update operations are common; thus, the proposed scheme should support adding and removing EMRs over the cloud server.
- *Efficient for a large-scale health care system:* Because a large-scale ehealthcare system usually contains a large number of EMRs, the query method in the proposed scheme must be efficient so that the query time is acceptable.

3. Preliminaries

This section introduces the building blocks required in this study before describing the proposed scheme in detail.

3.1. Asymmetric scalar-product-preserving encryption

Asymmetric Scalar-Product-Preserving Encryption (ASPE) is an efficient encryption technique proposed by Wong et al. [30] to calculate the inner product of two vectors in a privacy-preserving way. In ASPE, the cloud server can calculate the inner product $Enc(R) \cdot Enc(Q) = R \cdot Q$ without disclosing the plaintext information of R and Q. The encrypted vectors $Enc(R)$ and $Enc(Q)$ are generated by random splitting and matrix multiplication. The enhanced ASPE [31] can achieve a higher security level. In this study, we use the enhanced ASPE [31] to perform EMR querying in an efficient and privacy-preserving way.

3.2. Polynomial expression

Similar to [28], we use polynomials to represent the set of health symptom keywords. Based on the fundamental theorem of algebra, we have:

$$b_d\left(x-x_1\right)\left(x-x_2\right)\cdots\left(x-x_d\right)=b_0+b_1x+b_2x^2 + \cdots + b_dx^d\left(b_d\neq 0\right). \tag{1}$$

The set of zero roots of the polynomial is the health symptom keyword set $\{x_1, x_2, \dots, x_d\}$, and the polynomial can be represented as a $(d+1)$-dimensional vector $B = (b_0, b_1, \dots, b_d)$ by coefficients, where d is the maximum number of keywords allowed in the health symptom keyword set. The single health symptom keyword x_i can also be represented as a $(d+1)$-dimensional vector $L_i = (1, x_i, x_i^2, \dots, x_i^d)$. If $L_i \cdot B = b_0 + b_1x_i + b_2x_i^2 + \cdots + b_dx_i^d = 0$, the single health symptom keyword x_i belongs to the health symptom keyword set $\{x_1, x_2, \dots, x_d\}$.

4. Proposed scheme

In this section, the proposed scheme is described in detail. First, we provide an overview of the different steps involved. Then, we detail each process along with the algorithms used in them.

4.1. Overview

The proposed scheme aims to help patients find similar EMRs in a large-scale eHealthcare system as a reference. Specifically, when patients register in the system, they obtain secret keys from the health care center. The patient then uses the secret keys to generate two trapdoors for the coarse-grained and fine-grained queries. The trapdoors are then forwarded to the cloud server. The cloud server will find EMRs similar to the patient's by performing a coarse-grained and fine-grained query. Finally, the patient uses the symmetric key to decrypt the encrypted EMRs as a reference. The entire process is efficient and privacy-preserving. To achieve a coarse-grained query, we build a binary decision tree to find the collection of EMRs corresponding to the patient's health symptom keyword set, which is more efficient than the coarse-grained query method without a tree in Section 6.3. We use the enhanced ASPE for fine-grained queries to find similar EMRs that meet the threshold, achieving a higher security level at a lower cost. The scheme consists of six parts as follows:

- **Initialization:** In this phase, the health care center will generate secret keys and distribute them to newly registered patients through a secure channel.
- **GenIndex:** In this phase, the health care center will encrypt the desensitized EMRs using the AES encryption algorithm and use the secret keys to encrypt the health symptom keywords and health data vectors extracted from EMRs. Then, a binary decision tree for the coarse-grained query is constructed. Finally, the tree and the above-encrypted data are stored in the cloud server.
- **GenTrapdoor:** In this phase, after receiving the secret keys from the health care center, the patient uses secret keys to generate two trapdoors for the coarse-grained and fine-grained query, where the trapdoor for the coarse-grained query represents the patient's set of health symptom keywords, and the trapdoor for the fine-grained query represents the patient's health data vector and the threshold. Then, the trapdoors will be forwarded to the cloud server.
- **Coarse-grained query:** In this phase, with the trapdoors from the patient, the cloud server performs the coarse-grained query efficiently to find the collection of EMRs corresponding to the patient's health symptom keyword set, which can narrow down the scope of the fine-grained query.
- **Fine-grained query:** In this phase, the cloud server performs the fine-grained query efficiently to find similar EMRs that meet the threshold set by the patient. Then, the search result will be sent to the patient.
- **EMR update:** If the healthcare center wants to add or remove an EMR, the cloud server finds the collection of EMRs corresponding to the health symptom keyword set of the updated data by a coarse-grained query. Then, the cloud server effectively executes add or remove data operations in the EMR collection to update data.

4.2. Details of the proposed scheme

(1) Initialization

$\sigma \leftarrow KeyGen(1^\lambda)$: The healthcare center randomly generates the secret key $\sigma = (sk, S_c, S_f, M_{c1}, M_{c2}, M_{f1}, M_{f2})$, where sk is the symmetric key used to encrypt EMRs. S_c, M_{c1}, and M_{c2} are a $(d+1)$-dimensional binary vector and two $(d+1)\times(d+1)$-dimensional invertible matrices used to split and encrypt vectors in the coarse-grained query, respectively. S_f, M_{f1} and M_{f2} are an $(n+3)$-dimensional binary vector and two $(n+3)\times(n+3)$-dimensional invertible matrices, respectively. These are used to split and encrypt individual vectors in a fine-grained query. Then, the newly generated secret key σ will be transmitted to newly registered patients through a secure channel.

(2) GenIndex

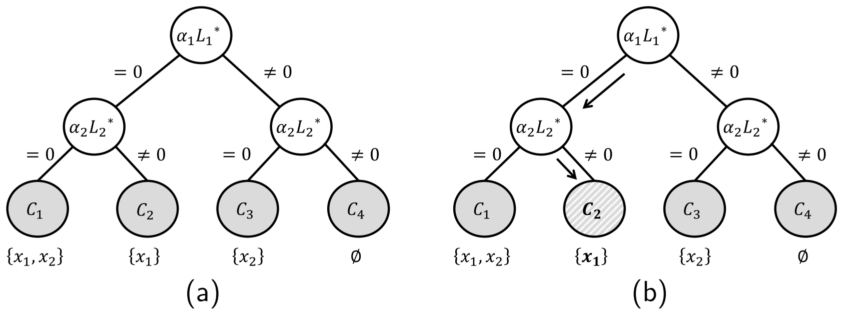


Fig. 2. The binary decision tree for coarse-grained query. (a) An example of the coarse-grained query tree; (b) an example of the coarse-grained query.

$\left(T, \alpha_i L_i^*, \beta_j eA_j^*\right) \leftarrow GenIndex\left(\sigma, L_i, A_j\right)$: EMRs are encrypted in this phase before being outsourced to the cloud server. Following this, the health care center will use secret keys to encrypt the health symptom keywords and health data vectors extracted from EMRs.

Specifically, for the coarse-grained query, the health care center has the health symptom keyword vector $L_i = \left(1, x_i, x_i^2, \ldots, x_i^d\right)$, $(i = 1, 2, \ldots, t)$, where t is the size of the health symptom keyword dictionary (e.g., 20) and x_i is a positive integer that represents the ith keyword in the dictionary. For each vector L_i, a binary vector S_c is used to split and encrypt it into invertible matrices M_{c1} and M_{c2}. If $S_c[m] = 0, (m = 0, 1, \ldots, d)$, $L_{ia}[m]$ and $L_{ib}[m]$ will be set as $L_{ia}[m] = L_{ib}[m] = L_i[m]$; otherwise, $L_{ia}[m]$ and $L_{ib}[m]$ will be randomly set so that $L_{ia}[m] + L_{ib}[m] = L_i[m]$. Then, L_{ia} and L_{ib} are transformed into square matrices as:

$$L_{ia}{}' = \begin{pmatrix} L_{ia}[0] & 0 & \cdots & 0 \\ 0 & L_{ia}[1] & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & L_{ia}[d] \end{pmatrix}, \tag{2}$$

$$L_{ib}{}' = \begin{pmatrix} L_{ib}[0] & 0 & \cdots & 0 \\ 0 & L_{ib}[1] & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & L_{ib}[d] \end{pmatrix}. \tag{3}$$

For each vector L_i, a random nonzero integer α_i and two $(d+1) \times (d+1)$-dimensional lower triangular matrices C_{ia} and C_{ib} with diagonal entries set to 1 are generated. Finally, the encrypted health symptom keywords are denoted as $\alpha_i L_i^* = \alpha_i \left(M_{c1} C_{ia} L_{ia}{}' M_{c2}, M_{c1} C_{ib} L_{ib}{}' M_{c2}\right)$.

For the fine-grained query, the health care center has the n-dimensional health data vector $A_j = \left(a_1, a_2, \ldots, a_n\right)$, where j represents the corresponding EMR number and n is the dimension of the collected health data vector (e.g., 18). For each vector A_j, the health care center expands it into an $(n+3)$-dimensional vector $eA_j = \left(1, -2a_1, -2a_2, \ldots, -2a_n, \sum_{i=1}^{n} a_i^2, -1\right)$. Then, the binary vector S_f and invertible matrices M_{f1} and M_{f2} are used to split and encrypt the vector eA_j. The specific process is similar to the treatment of vector L_i. Then, eA_{ja} and eA_{jb} are transformed into square matrices $eA_{ja}{}'$ and $eA_{jb}{}'$ in the same form as Eqs. (2) and (3). Then, for each vector eA_j, a random positive integer β_j and two $(n+3) \times (n+3)$-dimensional lower triangular matrices C_{ja}, C_{jb} with diagonal entries set to 1 are generated. Finally, the vector eA_j is encrypted as $\beta_j eA_j^* = \beta_j \left(M_{f1} C_{ja} eA_{ja}{}' M_{f2}, M_{f1} C_{jb} eA_{jb}{}' M_{f2}\right)$.

After obtaining the ciphertexts, a binary decision tree T with a depth of $t+1$ for a coarse-grained query is constructed. Specifically, the ith encrypted health symptom keyword $\alpha_i L_i^*$ is stored in the non-leaf node of the ith layer, and each leaf node points to a collection of EMRs $C_k, \left(k = 1, 2, \ldots, 2^t\right)$ corresponding to the same health symptom keyword set. To prevent the cloud server from guessing the underlying keywords by referring to the frequency of the health symptom keyword set, each collection of EMRs will be filled with dummy EMRs so that its size is the same as that of the maximum collection. The keywords in dummy EMRs are not dictionary words and thus have no impact on the query result. In the coarse-grained query stage, a trapdoor will be matched from the root node until it reaches a leaf node, which points to the collection of EMRs corresponding to the patient's health symptom keyword set. When the trapdoor is successfully matched with a non-leaf node, it will visit the left child node of this node. Otherwise, it will visit the right child node of this node. A simple example of the coarse-grained query tree is shown in Fig. 2(a).

After this process is completed, the tree T and the encrypted data $\alpha_i L_i^*, \beta_j eA_j^*$ are stored in the cloud server.

Remarks. In practice, the encrypted health symptom keywords stored in the non-leaf node of the ith layer should be random to protect security and privacy. However, for convenience, we still use the previous statement for the rest of this study.

(3) GenTrapdoor

$\left(r_c B_c^*, R_c eQ_c^*\right) \leftarrow GenTrapdoor\left(\sigma, B_c, Q_c\right)$: When a patient c registers in the system, it will receive the secret key σ provided by the health care center and uses this key to encrypt its search request.

For the coarse-grained query, patient c specifies its health symptom keyword set as $\left\{X_1, X_2, \ldots, X_l\right\}$ consisting of l keywords. We assume that different integers represent the keywords. To make the number of keywords in the patient's health symptom keyword set consistent, $d-l$ dummy keywords $\left\{X_{l+1}, \ldots, X_d\right\}$ are randomly generated and added to $\left\{X_1, X_2, \ldots, X_l\right\}$. Because every dummy keyword is not a dictionary word, the matching results are not affected. Then, a polynomial function to represent the set of health symptom keywords is constructed as:

$$b_d\left(x - X_1\right)\left(x - X_2\right)\cdots\left(x - X_d\right) = b_0 + b_1 x + b_2 x^2 + \cdots + b_d x^d, \tag{4}$$

where b_d is a nonzero integer. Then, the coarse-grained query vector can be constructed as $B_c = \left(b_0, b_1, \ldots, b_d\right)$ by the coefficients of the polynomial function. Then, the binary vector S_c and invertible matrices M_{c1}, M_{c2} will be used to split and encrypt B_c, respectively. However, unlike *GenIndex*, if $S_c[p] = 0, (p = 0, 1, \ldots, d)$, the patient randomly sets $B_{ca}[p]$ and $B_{cb}[p]$ such that $B_{ca}[p] + B_{cb}[p] = B_c[p]$. If $S_c[p] = 1$, $B_{ca}[p], B_{cb}[p]$ will be set as $B_{ca}[p] = B_{cb}[p] = B_c[p]$. Then, B_{ca} and B_{cb} are transformed into square matrices $B_{ca}{}', B_{cb}{}'$ in the same form as Eqs. (2) and (3). Then, patient c randomly generates a nonzero integer r_c and two $(d+1) \times (d+1)$-dimensional lower triangular matrices C_{ca} and C_{cb} with diagonal entries set to 1. Finally, the trapdoor for a coarse-grained query can be denoted as $r_c B_c^* = r_c\left(M_{c2}^{-1} B_{ca}{}' C_{ca} M_{c1}^{-1}, M_{c2}^{-1} B_{cb}{}' C_{cb} M_{c1}^{-1}\right)$.

For the fine-grained query, patient c first sets a similarity threshold Θ_c and then collects its own health data to construct an n-dimensional health data vector $Q_c = \left(q_1, q_2, \ldots, q_n\right)$, which will be expanded to the $(n+3)$-dimensional vector $eQ_c = \left(\sum_{i=1}^{n} q_i^2, q_1, q_2, \ldots, q_n, 1, \Theta_c\right)$. Then, the patient uses a binary vector S_f and invertible matrices M_{f1} and M_{f2} to encrypt the vector eQ_c. The specific process is similar to the treatment of vector B_c. Then, eQ_{ca} and eQ_{cb} are transformed into square matrices $eQ_{ca}{}'$ and $eQ_{cb}{}'$ in the same form as

Eqs. (2) and (3). Patient c then randomly generates a positive integer R_c, and two $(n+3)\times(n+3)$-dimensional lower triangular matrices C_{fa} and C_{fb} with diagonal entries set to 1 are generated. Finally, the trapdoor for the fine-grained query can be denoted as $R_c eQ_c^* = R_c\left(M_{f2}^{-1}eQ_{ca}{}'C_{fa}M_{f1}^{-1}, M_{f2}^{-1}eQ_{cb}{}'C_{fb}M_{f1}^{-1}\right)$.

After completing this process, the trapdoors for the coarse-grained and fine-grained queries are forwarded to the cloud server.

(4) Coarse-grained query

$C_k \leftarrow Coarse-GrainedQuery\left(T, \alpha_i L_i^*, r_c B_c^*\right)$: After receiving trapdoors from patient c, the cloud server first uses $r_c B_c^*$ to execute the coarse-grained query. As Fig. 2(b) shows, trapdoor $r_c B_c^*$ is matched from the root node of tree T. We assume that the currently visited non-leaf node is at layer i and $\otimes$ is a ciphertext matching symbol. The matching results between the encrypted health symptom keywords $\alpha_i L_i^*$ stored in layer i's non-leaf node and $r_c B_c^*$ are calculated as:

$$\begin{aligned}
Result &= tr\left[\alpha_i L_i^* \otimes r_c B_c^*\right] \\
&= tr\ \left[\ \alpha_i\left(M_{c1}C_{ia}L_{ia}{}'M_{c2}, M_{c1}C_{ib}L_{ib}{}'M_{c2}\right)\otimes r_c \right.\\
&\quad \left.\left(M_{c2}^{-1}B_{ca}{}'C_{ca}M_{c1}^{-1}, M_{c2}^{-1}B_{cb}{}'C_{cb}M_{c1}^{-1}\right)\ \right] \\
&= \alpha_i r_c * tr\left[C_{ia}L_{ia}{}'B_{ca}{}'C_{ca} + C_{ib}L_{ib}{}'B_{cb}{}'C_{cb}\right] \\
&= \alpha_i r_c * tr\left[L_{ia}{}'B_{ca}{}'\right] + \alpha_i r_c * tr\left[L_{ib}{}'B_{cb}{}'\right] \\
&= \alpha_i r_c\left(L_i\cdot B_c\right) \\
&= \alpha_i r_c\left(b_0 + b_1 x_i + b_2 x_i^2 + \cdots + b_d x_i^d\right),
\end{aligned} \tag{5}$$

where $tr[X]$ is the trace of X. If the matching result is 0, meaning that patient c's health symptom keyword set contains keyword x_i, then the left child node of the current node is visited. Otherwise, the right child node of the current node is visited. When a leaf node is visited, the coarse-grained query is over. The leaf node points to the collection of EMRs C_k corresponding to patient c's health symptom keyword set, which will be used as the scope of the fine-grained query. The concrete coarse-grained query process is listed in Algorithm 1.

(5) Fine-grained query

$E_c \leftarrow Fine-GrainedQuery\left(C_k, \beta_j eA_j^*, R_c eQ_c^*\right)$: After finding the collection of EMRs C_k, the cloud server performs the fine-grained query. Specifically, for the encrypted health data $\beta_j eA_j^*$ corresponding to each EMR in C_k, the similarity of $\beta_j eA_j^*$ and patient c's trapdoor $R_c eQ_c^*$ is calculated as:

$$\begin{aligned}
Similarity &= tr\left[\beta_j eA_j^* \otimes R_c eQ_c^*\right] \\
&= tr\ \left[\ \beta_j\left(M_{f1}C_{ja}eA_{ja}{}'M_{f2}, M_{f1}C_{jb}eA_{jb}{}'M_{f2}\right)\otimes \right.\\
&\quad \left. R_c\left(M_{f2}^{-1}eQ_{ca}{}'C_{fa}M_{f1}^{-1}, M_{f2}^{-1}eQ_{cb}{}'C_{fb}M_{f1}^{-1}\right)\ \right] \\
&= \beta_j R_c * tr\left[C_{ja}eA_{ja}{}'eQ_{ca}{}'C_{fa} + C_{jb}eA_{jb}{}'eQ_{cb}{}'C_{fb}\right] \\
&= \beta_j R_c * tr\left[eA_{ja}{}'eQ_{ca}{}'\right] + \beta_j R_c * tr\left[eA_{jb}{}'eQ_{cb}{}'\right] \\
&= \beta_j R_c\left(eA_j\cdot eQ_c\right) \\
&= \beta_j R_c\left[\sum_{i=1}^{n}\left(q_i - a_i\right)^2 - \Theta_c\right].
\end{aligned} \tag{6}$$

The similarity represents the result of the squared Euclidean distance, which is often used as a similarity measure [20,32,33], between vectors A_j and Q_c minus threshold Θ_c. By comparing the inner product with 0, the cloud server can find the EMRs that meet the similarity threshold, which will be constructed as a set E_c. In the end, the search result E_c will be sent to patient c, who can use symmetric key sk to decrypt the encrypted EMRs as a reference. The concrete fine-grained query process is listed in Algorithm 2.

(6) EMR update

We consider two EMR update operations: adding and removing EMRs. When a healthcare center wants to add or remove an EMR over the cloud server, it will use the coarse-grained query to update data. Specifically, if the health care center wants to add a new EMR, the desensitized EMR will be encrypted using the AES encryption algorithm first. Then, similar to *GenIndex* and *GenTrapdoor*, the encrypted health data and the trapdoor for coarse-grained query corresponding to the

Algorithm 1 Coarse-Grained Query

```
Require: The binary decision tree $T$ for coarse-grained query, the encrypted health symptom keywords $\alpha_i L_i^*$ stored in the non-leaf node of the $i$-th layer, and the patient $c$'s trapdoor for coarse-grained query $r_c B_c^*$.
Ensure: The collection of EMRs $C_k$ corresponding to the patient $c$'s health symptom keyword set.
1: Generate a pointer $P$ to the root node of the tree $T$.
2: for $i = 1$ to $t$ do
3:     Compute $Result = tr[\alpha_i L_i^* \otimes r_c B_c^*]$.
4:     if $Result == 0$ then
5:         $P = P \rightarrow leftchild.$
6:     else
7:         $P = P \rightarrow rightchild.$
8:     end if
9: end for
10: return The EMR collection $C_k$ corresponding to the leaf node pointed to by pointer $P$.
```

Algorithm 2 Fine-Grained Query

```
Require: The collection of EMRs $C_k$ corresponding to patient $c$'s health symptom keyword set, encrypted health data $\beta_j eA_j^*$, and patient $c$'s trapdoor for fine-grained query $R_c eQ_c^*$.
Ensure: The EMR set $E_c$ meets the similarity threshold of the patient $c$.
1: Initializes an empty set $E_c$.
2: for the encrypted health data $\beta_j eA_j^*$ corresponding to each EMR in $C_k$ do
3:     Compute $Similarity = tr[\beta_j eA_j^* \otimes R_c eQ_c^*]$.
4:     if $Similarity < 0$ then
5:         Add the corresponding EMR of vector $\beta_j eA_j^*$ to set the $E_c$.
6:     end if
7: end for
8: return The search result $E_c$.
```

EMR will be constructed. These data will be sent to the cloud server, which can find a collection of EMRs corresponding to the health symptom keyword set of the updated data by the coarse-grained query. Then, the new EMR and its corresponding encrypted health data will be added to the collection. Similarly, after the EMR collection is found, an old EMR can be removed with the corresponding EMR number.

5. Security analysis

The primary security goals of the proposed scheme, as described earlier in Section 2.3 are privacy protection of EMR, health symptom keywords, health data and trapdoor, and unlinkability of the trapdoor. Because EMRs can be encrypted using the AES encryption algorithm, which has been proven to be secure, the privacy protection of EMRs can be guaranteed. Therefore, in this section, we primarily consider the privacy protection of health symptom keywords, health data and trapdoors and the unlinkability of trapdoors in detail.

5.1. Security definitions

Similar to the searchable encryption schemes in [11,12,34], the information leak during the query process is described using the following parameters:

- **Size Pattern:** The cloud server is aware of the total number of indices (i.e., health symptom keywords and health data vectors) stored and the total number of trapdoors patients sent.

- **Access Pattern:** The cloud server can learn the identifier of each encrypted data returned against a specific query.
- **Search Pattern:** The cloud server can reveal if different trapdoors query specific encrypted data.

The work in [35,36] can be applied to hide the *Access Patterns* and *Search Patterns*, respectively. Thus, we do not consider them in this study.

5.2. Security analysis

Theorem 1. *The proposed scheme achieves privacy protection of health symptom keywords, health data, and trapdoors in the known sample model.*

Proof. In the coarse-grained query, we assume that the cloud server has encrypted health symptom keywords and some plaintexts. The cloud server also cannot learn the mapping relationship between plaintexts and ciphertexts. For the ciphertext $\alpha_i L_i^*$, without the vector S_c, the cloud server cannot obtain the matrices L_{ia}' and L_{ib}'; therefore, L_{ia}' and L_{ib}' are two unknown $(d+1)\times(d+1)$-dimensional matrices. For the linear equation $\alpha_i L_i^* = \alpha_i\left(M_{c1}C_{ia}L_{ia}'M_{c2}, M_{c1}\ C_{ib}L_{ib}'M_{c2}\right)$, where M_{c1} and M_{c2} are both unknown $(d+1)\times(d+1)$-dimensional matrices and C_{ia} and C_{ib} are two unknown $(d+1)\times(d+1)$-dimensional lower triangular matrices with diagonal entries set to 1, there are $2(d+1)$ unknowns in L_{ia}' and L_{ib}', $2(d+1)^2$ unknowns in M_{c1} and M_{c2}, $d(d+1)$ unknowns in C_{ia} and C_{ib}, and an unknown in α_i. Because there are only $2(d+1)^2$ equations, which are fewer than the number of unknowns, it is difficult for the cloud server to solve them. In addition, we also assume that the cloud server has trapdoors for coarse-grained queries and some plaintexts, and the cloud server cannot learn the mapping relationship between the plaintexts and ciphertexts. For the ciphertext $r_c B_c^*$, there are $2(d+1)$ unknowns in B_{ca}', B_{cb}', $2(d+1)^2$ unknowns in M_{c1}, M_{c2}, $d(d+1)$ unknowns in C_{ca}, C_{cb}, and an unknown in r_c. For $r_c B_c^* = r_c\left(M_{c2}^{-1}B_{ca}'C_{ca}M_{c1}^{-1}, M_{c2}^{-1}B_{cb}'C_{cb}M_{c1}^{-1}\right)$, because there are only $2(d+1)^2$ equations, it is also difficult to solve it.

In the fine-grained query, privacy protection for health data and trapdoors can be proven in a similar way to the process described above. Thus, to avoid repetition, we skip the proof. The privacy protection of health symptom keywords, health data, and trapdoors is proven. □

Theorem 2. *The proposed scheme achieves trapdoor unlinkability.*

Proof. In the coarse-grained query, the search request is augmented with dummy keywords. We assume that $D_0, D_1 \subset D$ are two dummy keyword sets injected into the same health symptom keyword set $\left\{X_1, X_2, \ldots, X_l\right\}$. Thus, $\Pr\left[D_0 = D_1\right] = \frac{1}{C_{N_D}^{d-l}}$, where N_D is the size of dummy keyword dictionary D. Given two same coarse-grained query vectors $B_0 = B_1 = \left(b_0, b_1, \ldots, b_d\right)$, for each dimension p, where $S_c[p] = 0$, $B_0[p]$ and $B_1[p]$ are set to two random values. We assume that $\Pr\left[\left\{B_{0a}[p], B_{0b}[p]\right\} = \left\{B_{1a}[p], B_{1b}[p]\right\}\right] = \varepsilon$ for any dimension p where $S_c[p] = 0$; thus, we have $\Pr\left[\left\{B_{0a}', B_{0b}'\right\} = \left\{B_{1a}', B_{1b}'\right\}\right] = \varepsilon^{s_0}$ where s_0 is the number of zeros in the vector S_c. We also have $\Pr\left[\left\{B_{0a}', B_{0b}'\right\} = \left\{B_{1a}', B_{1b}'\right\}\right] = \frac{1}{C_{N_D}^{d-l}} \cdot \varepsilon^{s_0}$ if $\left\{B_{0a}', B_{0b}'\right\}$ and $\left\{B_{1a}', B_{1b}'\right\}$ are both constructed with the same health symptom keyword set $\left\{X_1, X_2, \ldots, X_l\right\}$. For random lower triangular matrices $\left\{C_{0a}, C_{0b}\right\}$ and $\left\{C_{1a}, C_{1b}\right\}$, we assume that $\Pr\left[\left\{C_{0a}[i][j], C_{0b}[i][j]\right\} = \left\{C_{1a}[i][j]\ , C_{1b}[i][j]\right\}\right] = \eta$ for any elements below the diagonal, we have $\Pr\left[\left\{C_{0a}, C_{0b}\right\} = \left\{C_{1a}, C_{1b}\right\}\right] = \eta^{\frac{d(d+1)}{2}}$. because $\varepsilon \to 0$, $\eta \to 0$ and r_0, r_1 are both random nonzero integers where $\Pr\left[r_0 = r_1\right] \to 0$, then $\Pr\left[r_0 B_0^* = r_1 B_1^*\right] = \Pr\left[r_0\left(M_{c2}^{-1}B_{0a}'C_{0a}M_{c1}^{-1}, M_{c2}^{-1}\ B_{0b}'C_{0b}M_{c1}^{-1}\right) = r_1\left(M_{c2}^{-1}B_{1a}'C_{1a}M_{c1}^{-1}, M_{c2}^{-1}B_{1b}'C_{1b}M_{c1}^{-1}\right)\right] \to 0$.

In the fine-grained query, the trapdoor's unlinkability can be proven in a similar way to the process described above; thus, to avoid repetition, we skip this proof. Thus, the unlinkability of the trapdoor is proven. □

Table 2
Time cost of different phases of the fine-grained query.

Index building	Trapdoor generating	Query
137.083 ms	7.574 ms	1.201 ms

6. Performance analysis

To verify the efficiency of the proposed scheme in a large-scale ehealthcare system, we measure the computational cost associated with the three phases: *Index building, Trapdoor generating*, and *Query*. The scheme has been implemented in Python, running on a 1.6 GHz i5 system with 12 GB of RAM.

6.1. Experiments based on real-world data

In this experimental analysis, we verify the practicability of the proposed scheme using real-world data. For the coarse-grained query, we use the Symptoms Corpus in Kaggle,[1] which describes specific symptoms. The corpus contains 20 symptoms: bitter taste , cough, dizziness, fever, inappetence, perspiration, sneezing, sore throat, spasm, vomiting, arthralgia, asthenia, chill, diarrhea, fever, headache, restlessness, stomach ache, stop words, and thirst. With reference to this content, the health symptom keyword dictionary size is 20. Experimental details for the coarse-grained query are described in Section 6.2.

For the fine-grained query, we verify the practicability using the Indian Liver Patient Dataset (ILPD)[2]. This dataset consists of 583 instances, and each instance contains 10 variables: age, gender, total bilirubin, direct bilirubin, total proteins, albumin, A/G ratio, SGPT , SGOT, and Alkphos. This dataset also contains 416 liver patient records and 167 non-liver patient records.[3] We chose the liver patient records as the health data vectors corresponding to the same health symptom for the experiment, and each vector contains 10 dimensions. We present the time cost of different phases for the fine-grained query using real data in Table 2, which is efficient in the real world because the time required is in milliseconds for several hundred records.

Based on this analysis, the proposed scheme is feasible in real medical scenarios. To demonstrate the efficiency of the proposed scheme in a large-scale ehealthcare system, we also experimented with simulated data.

6.2. Experiments based on simulated data

We randomly generate simulated data to evaluate the computational overhead of the proposed scheme in detail.

Index building: As shown in Section 4.2, the healthcare center constructs and encrypts health symptom keywords and health data vectors extracted from EMRs. Then, a binary decision tree for the coarse-grained query is constructed. Fig. 3(a) shows the computational cost of *Index building* for the coarse-grained query. Even if the number of d is different, the fluctuation in the time cost is small. The computational cost of *Index building* for coarse-grained queries increases with increasing t. Thus, the computational cost of *Index building* for a coarse-grained query is primarily affected by the size of the health symptom keyword dictionary t. Fig. 3(b) shows the computational cost of *Index building* for the fine-grained query. The computational cost of *Index building* for fine-grained queries increases as the number of dimensions or the amount of health data increases. In general, the computational

[1] [Online]. Available: https://www.kaggle.com/banabasave/symptoms-data

[2] [Online]. Available: https://archive.ics.uci.edu/ml/index.php

[3] Two of the 416 liver patient records have missing values and were thus removed from the measurements.

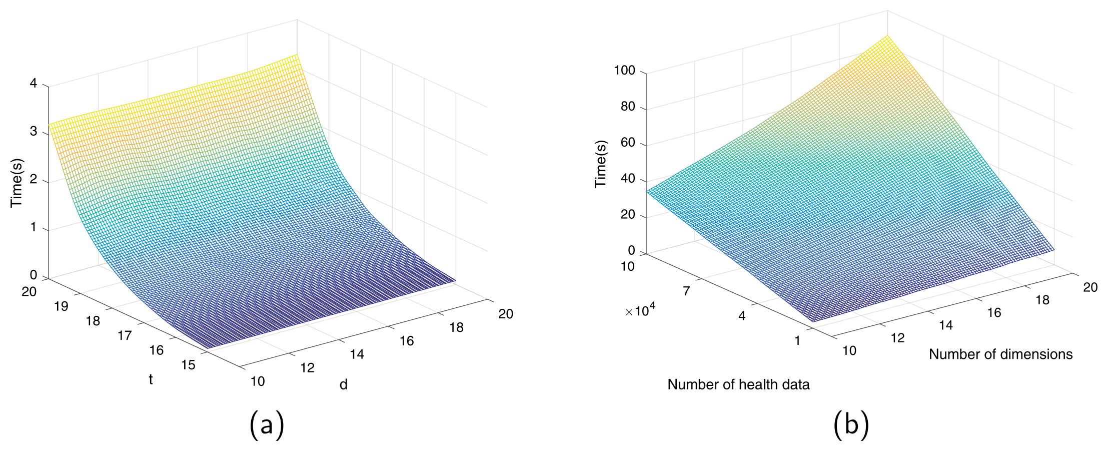


Fig. 3. Computational cost of *index building*. (a) for coarse-grained query; (b) for fine-grained query.

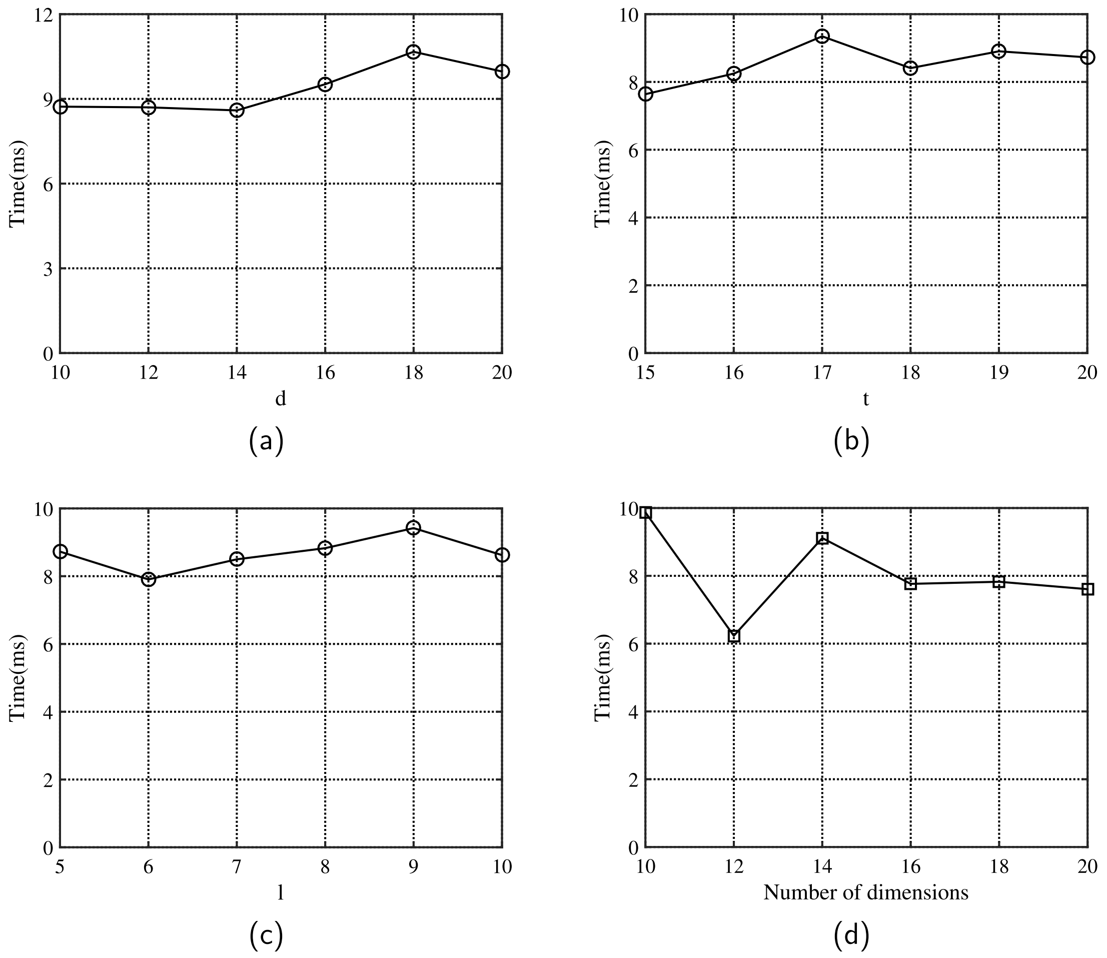


Fig. 4. Computational cost of *Trapdoor generation*. (a) Different numbers of d for coarse-grained queries when $t = 20$ and $l = 5$; (b) different numbers of t for coarse-grained queries when $d = 10$ and $l = 5$; (c) different numbers of l for coarse-grained queries when $t = 20$ and $d = 10$; (d) different numbers of dimensions for fine-grained queries.

overhead of *Index building* is acceptable, and the proposed scheme can achieve efficient index encryption.

Trapdoor generation: As shown in Section 4.2, after receiving the secret keys from the health care center, the patient uses them to generate two trapdoors for the coarse-grained and fine-grained queries. Then, the trapdoors will be forwarded to the cloud server for the query. Figs. 4(a)–4(c) show that the computational cost of *Trapdoor generation* for coarse-grained queries is insensitive to the maximum number of keywords allowed in the set of health symptom keywords d, the size of the health symptom keyword dictionary t, or the number of health symptom keywords l. In Fig. 4(d), the computational cost of *Trapdoor generation* for fine-grained queries is insensitive to the number of dimensions. Thus, Fig. 4 shows that the proposed scheme is efficient on the patient's side.

Query: As discussed in Section 4.2, after receiving the trapdoors from the patient, the cloud server performs coarse-grained and fine-grained queries to find the collection of EMRs corresponding to the patient's health symptom keyword set and the similar EMRs that meet

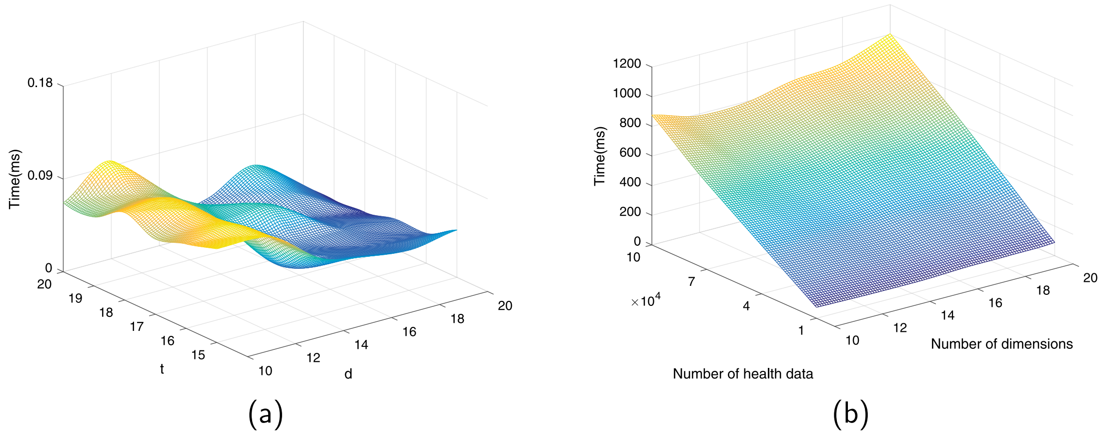


Fig. 5. Computational cost of *Query*. (a) for coarse-grained query; (b) for fine-grained query.

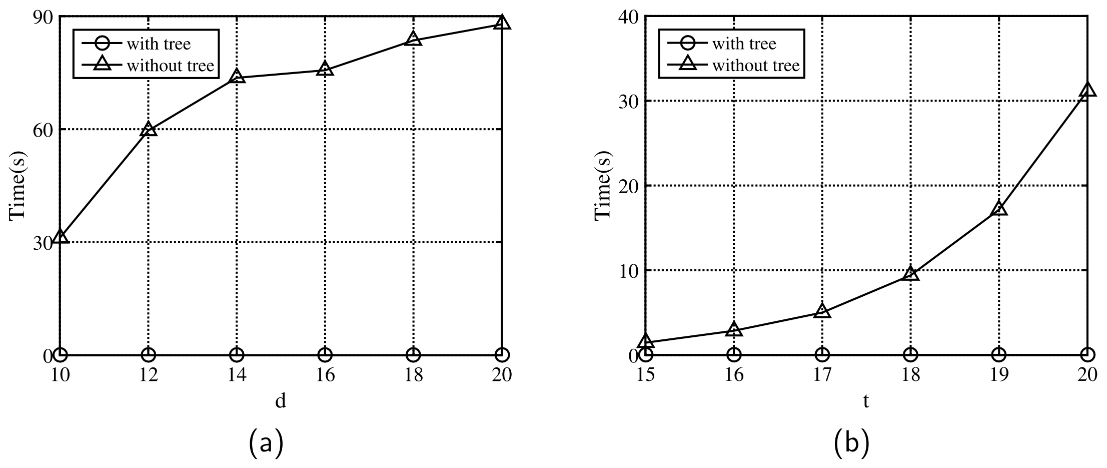


Fig. 6. Computational cost of *Query*. (a) Different numbers of *d* for coarse-grained queries when *t* = 20; (b) different numbers of *t* for coarse-grained queries when *d* = 10.

the threshold set, respectively. Fig. 5(a) shows that the coarse-grained query in the proposed scheme is efficient, and its running time is less than 0.1 ms. Fig. 5(b) shows that as the number of health data increases, the computational cost of *Query* for fine-grained query grows linearly. However, the fine-grained query in the proposed scheme is also efficient. When the number of health data is 100,000 and the number of dimensions is 20, the time cost of a fine-grained query is approximately 1 , 003 ms. The time cost will be even lower in a cloud server with strong computing power. Additionally, the computational cost for fine-grained queries is not sensitive to the number of dimensions. In general, the computational overhead of *Query* is acceptable, and the proposed scheme for a large-scale ehealthcare system is efficient.

6.3. Discussion

In the proposed scheme, we propose an efficient and privacy-preserving coarse-grained query method based on a binary decision tree to match the same set of health symptom keywords. Similar to [28], we also design a coarse-grained query method without a tree. Specifically, the proposed method uses the trapdoor for the coarse-grained query to match the set of health symptom keywords for each EMR's set. Fig. 6 shows the time cost of *Query* for the coarse-grained query with and without the tree. The coarse-grained query with a tree is far more efficient than without a tree, and the running time is less than 0.1 ms, which shows that the proposed coarse-grained query method is efficient with the help of a binary decision tree. In addition to the targeted similar EMR query, the proposed coarse-grained query method can be applied to other practical keyword match query applications (e.g., task recommendation and medical service recommendation). In the future, we believe this method can help improve the matching efficiency in other fields.

7. Conclusion

In this paper, to help a patient find similar EMRs in large-scale health care systems as a reference, we construct a privacy-preserving similar EMR query scheme. We use enhanced ASPE to construct coarse-grained and fine-grained queries to realize the scheme. In the coarse-grained query, we build a binary decision tree to find a collection of EMRs that corresponds to the patient's health symptom keyword set to narrow the scope of the fine-grained query. In the fine-grained query, we find similar EMRs that meet the threshold through similarity calculation. A detailed security analysis shows that the scheme is secure. We also apply and evaluate the proposed scheme using real-world and simulated data, showing that the proposed scheme is efficient for large-scale health care systems. In the future, we plan to work on designing more efficient and privacy-preserving electronic medical records query schemes.

CRediT authorship contribution statement

Chang Xu: Conceptualization, Methodology. **Zijian Chan:** Software, Writing – original draft. **Liehuang Zhu:** Supervision. **Rongxing Lu:** Methodology, Resources. **Yunguo Guan:** Formal analysis. **Kashif Sharif:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant Nos. 61972037, U1804263).

References

[1] S. Chenthara, K. Ahmed, H. Wang, F. Whittaker, Security and privacy-preserving challenges of e-health solutions in cloud computing, IEEE Access 7 (2019) 74361–74382.

[2] P. Li, S. Guo, T. Miyazaki, M. Xie, J. Hu, W. Zhuang, Privacy-preserving access to big data in the cloud, IEEE Cloud Comput. 3 (5) (2016) 34–42.

[3] R. Zhang, R. Xue, L. Liu, Searchable encryption for healthcare clouds: A survey, IEEE Trans. Serv. Comput. 11 (6) (2018) 978–996.

[4] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000, 2000, pp. 44–55.

[5] J. Domingo-Ferrer, O. Farràs, J. Ribes-González, D. Sánchez, Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges, Comput. Commun. 140–141 (2019) 38–60.

[6] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, K. Li, Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners, Future Gener. Comput. Syst. (ISSN: 0167-739X) 100 (2019) 689–700.

[7] Y. Yang, X. Liu, R.H. Deng, Multi-user multi-keyword rank search over encrypted data in arbitrary language, IEEE Trans. Dependable Secure Comput. 17 (2) (2020) 320–334.

[8] Z. Guan, X. Liu, L. Wu, J. Wu, R. Xu, J. Zhang, Y. Li, Cross-lingual multi-keyword rank search with semantic extension over encrypted data, Inform. Sci. (ISSN: 0020-0255) 514 (2020) 523–540.

[9] B. Wang, M. Li, H. Wang, H. Li, Circular range search on encrypted spatial data, in: 2015 IEEE 35th International Conference on Distributed Computing Systems, 2015, pp. 794–795.

[10] H. Zhu, R. Lu, C. Huang, L. Chen, H. Li, An efficient privacy-preserving location-based services query scheme in outsourced cloud, IEEE Trans. Veh. Technol. 65 (9) (2016) 7729–7739.

[11] G. Xu, H. Li, Y. Dai, K. Yang, X. Lin, Enabling efficient and geometric range query with access control over encrypted spatial data, IEEE Trans. Inf. Forensics Secur. 14 (4) (2019) 870–885.

[12] X. Wang, J. Ma, X. Liu, R.H. Deng, Y. Miao, D. Zhu, Z. Ma, Search Me in the Dark: Privacy-preserving boolean range query over encrypted spatial data, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 2253–2262.

[13] Y. Elmehdwi, B.K. Samanthula, W. Jiang, Secure k-nearest neighbor query over encrypted data in outsourced environments, in: 2014 IEEE 30th International Conference on Data Engineering, 2014, pp. 664–675.

[14] S. Rane, P.T. Boufounos, Privacy-preserving nearest neighbor methods: comparing signals without revealing them, IEEE Signal Process. Mag. 30 (2) (2013) 18–28.

[15] W. Lin, H. Cui, B. Li, C. Wang, Privacy-preserving similarity search with efficient updates in distributed key-value stores, IEEE Trans. Parallel Distrib. Syst. 32 (5) (2021) 1072–1084.

[16] B. Wang, S. Yu, W. Lou, Y.T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 2112–2120.

[17] C. Wang, K. Ren, S. Yu, K.M.R. Urs, Achieving usable and privacy-assured similarity search over outsourced cloud data, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 451–459.

[18] J. Jin, Y. Zheng, P. Xiong, EPSim-GS: Efficient and privacy-preserving similarity range query over genomic sequences, in: 2021 18th International Conference on Privacy, Security and Trust, 2021, pp. 1–10.

[19] F. Song, Z. Qin, L. Xue, J. Zhang, X. Lin, X. Shen, Privacy-preserving keyword similarity search over encrypted spatial data in cloud computing, IEEE Internet Things J. 9 (8) (2022) 6184–6198.

[20] C. Huang, R. Lu, H. Zhu, J. Shao, X. Lin, FSSR: Fine-grained EHRs sharing via similarity-based recommendation in cloud-assisted ehealthcare system, in: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, 2016, pp. 95–106.

[21] W. Tang, J. Ren, Y. Zhang, Enabling trusted and privacy-preserving healthcare services in social media health networks, IEEE Trans. Multimed. 21 (3) (2019) 579–590.

[22] C. Zhang, L. Zhu, C. Xu, X. Liu, K. Sharif, Reliable and privacy-preserving truth discovery for mobile crowdsensing systems, IEEE Trans. Dependable Secure Comput. 18 (3) (2021) 1245–1260.

[23] C. Zhang, L. Zhu, C. Xu, K. Sharif, K. Ding, X. Liu, X. Du, M. Guizani, TPPR: A trust-based and privacy-preserving platoon recommendation scheme in VANET, IEEE Trans. Serv. Comput. 15 (2) (2022) 806–818, http://dx.doi.org/10.1109/TSC.2019.2961992.

[24] C. Xu, N. Wang, L. Zhu, K. Sharif, C. Zhang, Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system, IEEE Internet Things J. 6 (5) (2019) 8345–8356.

[25] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, K.-K. Raymond Choo, Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data, Inform. Sci. 498 (2019) 91–105.

[26] K. Cheng, Y. Shen, Y. Wang, L. Wang, J. Ma, X. Jiang, C. Su, Strongly secure and efficient range queries in cloud databases under multiple keys, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2494–2502.

[27] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, A. van Moorsel, Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 211–227.

[28] J. Shu, X. Jia, K. Yang, H. Wang, Privacy-preserving task recommendation services for crowdsourcing, IEEE Trans. Serv. Comput. 14 (1) (2021) 235–247.

[29] K. Liu, C. Giannella, H. Kargupta, An attacker's view of distance preserving maps for privacy preserving data mining, in: Knowledge Discovery in Databases: PKDD 2006, 2006, pp. 297–308.

[30] W.K. Wong, D.W.-l. Cheung, B. Kao, N. Mamoulis, Secure KNN computation on encrypted databases, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, 2009, pp. 139–152.

[31] S. Hu, M. Li, Q. Wang, S.S.M. Chow, M. Du, Outsourced biometric identification with privacy, IEEE Trans. Inf. Forensics Secur. 13 (10) (2018) 2448–2463.

[32] C. Xu, J. Wang, L. Zhu, C. Zhang, K. Sharif, PPMR: A privacy-preserving online medical service recommendation scheme in ehealthcare system, IEEE Internet Things J. 6 (3) (2019) 5665–5673.

[33] K. Zhou, J. Ren, PassBio: Privacy-preserving user-centric biometric authentication, IEEE Trans. Inf. Forensics Secur. 13 (12) (2018) 3050–3063.

[34] B. Wang, M. Li, L. Xiong, FastGeo: Efficient geometric range queries on encrypted spatial data, IEEE Trans. Dependable Secure Comput. 16 (2) (2019) 245–258.

[35] M. Islam, M. Kuzu, M. Kantarcioglu, Access pattern disclosure on searchable encryption: Ramification, attack and mitigation, in: 19th Annual Network and Distributed System Security Symposium, 2012.

[36] C. Liu, L. Zhu, M. Wang, Y. an Tan, Search pattern leakage in searchable encryption: Attacks and new construction, Inform. Sci. 265 (2014) 176–188.