Efficient Privacy-Preserving Similarity Range Query With Quadsector Tree in eHealthcare

Yandong Zheng, Rongxing Lu, *Fellow, IEEE*, Yunguo Guan, Jun Shao, and Hui Zhu, *Senior Member, IEEE*

Abstract—As a consequence of advance in the Internet of Things (IoT) and big data technology, smart eHealthcare has emerged and greatly enabled patients to enjoy high-quality healthcare services in disease prediction, clinical decision making and healthcare surveillance. Meanwhile, in order to support the dramatic increase of healthcare data, healthcare centers often outsource the on-premises data to a powerful cloud and deploy the cloud server to manage the data. However, since the healthcare data usually contain some sensitive information and also the cloud server is not fully trusted, healthcare centers need to encrypt the data before outsourcing them to the cloud. Unfortunately, data encryption inevitably hinders some advanced applications of the data like the similarity range query in cloud. Although many studies on similarity range query over encrypted data have been reported, most of them still have some limitations in security, efficiency and practicality. Aiming at this challenge, in this article, we propose a new efficient privacy-preserving similarity range query (EPSim) scheme. Specifically, we first present a modified asymmetric scalar-product-preserving encryption (ASPE) scheme and prove it is selectively secure. Then, we introduce a Quadsector tree to represent the data, and employ a filtration condition to design an efficient algorithm for efficient similarity range queries over the Quadsector tree. Finally, we propose our EPSim scheme by integrating the modified ASPE scheme and Quadsector tree. Detailed security analysis indicates that our proposed EPSim scheme is really secure. In addition, extensive performance evaluations are conducted, and the results also demonstrate it is efficient and practical.

Index Terms—Similarity range query, eHealthcare, outsourced encrypted data, Quadsector tree, Hilbert exclusion condition

1 Introduction

The smart eHealthcare, which benefits from the advanced artificial intelligence [1] and fast-growing Internet of Things technologies [2], has improved healthcare quality in disease prediction, clinical decision making and healthcare surveillance. As a consequence, patients can enjoy better healthcare services, and the healthcare data in healthcare centers have a dramatic increase. As reported by BIS research [3], the global big data in healthcare market are estimated to grow over $68.75 billion by the end of 2025.

In order to support the rapid increase in healthcare data and mine potential applications of these data, healthcare centers should be equipped with abundant storage space and powerful computing capabilities. However, the IT facilities in healthcare centers are incompetent with such huge volumes of healthcare data. A popular option for the healthcare centers is to outsource their on-premises data to a powerful and flexible cloud, and deploy the cloud server to manage the data [4], [5]. Nevertheless, the healthcare data usually contain some sensitive information on patients, and the cloud server is not fully trusted. In such case, healthcare centers tend to encrypt the data before outsourcing them to the cloud. Although the data encryption technique can protect the privacy of the data, it inevitably hinders the healthcare centers to take advantage of advanced applications like data analysis and artificial intelligence over the data. Among these applications, the similarity range query has been considered as the most basic and critical one [6].

In the eHealthcare field, the similarity range query is to search previous patients who are similar to a current patient as shown in Fig. 1. The information of these similar patients can help doctors make more accurate clinic diagnosis, and their treatment history can assist doctors to create a better treatment plan for the current patient. In this work, each patient's information is represented to be a multi-dimensional data record, and the similarity between two data records is measured by euclidean distance. Suppose that $\mathcal{X} = \{\mathbf{x}_i | i = 1, 2, \ldots, n\}$ is a healthcare dataset with n data records and each record $\mathbf{x}_i$ has an identity id_i for $i = 1, 2, \ldots, n$. Then, given a query record $\mathbf{q}$ and a query range τ, the similarity range query is to "*return the identities of data records whose euclidean distance to* $\mathbf{q}$ *is equal to or less than* τ, *i.e.,* $\{\mathrm{id}_i | d(\mathbf{q}, \mathbf{x}_i) \leq \tau; \mathbf{x}_i \in \mathcal{X}\}$", where $d(\cdot, \cdot)$ denotes the euclidean distance.

Due to its importance in practice, the problem of similarity query over outsourced encrypted data has received considerable attention, and various schemes have been proposed. However, existing schemes still have some limitations in security, efficiency, accuracy, and practicality. Specifically, some of the existing schemes [7] only have a weak security. Also, many

- *Yandong Zheng, Rongxing Lu, and Yunguo Guan are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada. E-mail: {yzheng8, rlu1, yguan4}@unb.ca.*
- *Jun Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China. E-mail: chn.junshao@gmail.com.*
- *Hui Zhu is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China. E-mail: zhuhui@xidian.edu.cn.*

Manuscript received 4 May 2020; revised 6 Mar. 2021; accepted 12 May 2021. Date of publication 18 May 2021; date of current version 7 Oct. 2022. (Corresponding author: Rongxing Lu.)
Digital Object Identifier no. 10.1109/TSC.2021.3081350

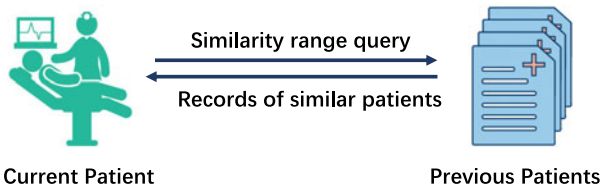


Fig. 1. Example of similarity range query in eHealthcare.

of the existing schemes are not as efficient as expected, since the time-consuming public key encryption operations are required [6], [8], [9], [10], [11], or the computational cost is linear to the size of the dataset [12]. To obtain high efficiency, many schemes fall back on the approximate similarity query [13], [14], [15], [16], [17], [18], [19], or two cloud servers model [20], [21]. Although the approximate similarity query schemes can efficiently support similarity queries over high-dimensional datasets, they cannot provide accurate query results. Two cloud servers model is impractical in some real scenarios because it will inevitably increase the running cost of the system.

Aiming at the above challenges, in this paper, we propose a new efficient, privacy-preserving, and practical similarity range query (EPSim) scheme over encrypted data in cloud. Specifically, our contributions of this paper lie in the following three-fold.

- First, we propose a modified asymmetric scalar-product-preserving encryption (ASPE) scheme, which is original from the scheme in [12]. The modified ASPE scheme allows the data owner to share a part of the secret key with the query user, who can use such part to generate new similarity range queries as he/she wants. Hence, we obtain efficiency without resorting the online data owner and two cloud servers model. Furthermore, we prove the proposed scheme is selectively secure.
- Second, we introduce a Quadsector tree to represent data records, which makes the computational complexity of query processing sublinear to the size of the dataset. Meanwhile, we employ a filtration condition (i.e., Hilbert exclusion condition in Section 3.2) to design an algorithm for similarity range queries over the Quadsector tree.
- Third, by integrating the modified ASPE scheme and Quadsector tree, we design our novel similarity range query scheme. The detailed security analysis and extensive experimental results show that our proposed scheme is secure, efficient, and practical for the eHealthcare scenario.

The remainder of this paper is organized as follows. In Section 2, we introduce our system model, security model and design goals. Then, we describe some preliminaries in Section 3. In Section 4, we present our scheme, followed by security analysis and performance evaluation in Sections 5 and 6, respectively. In Section 7, we present some related work. Finally, we draw our conclusion in Section 8.

2 Models and Design Goals

In this section, we formalize our system model, security model, and identify our design goals.

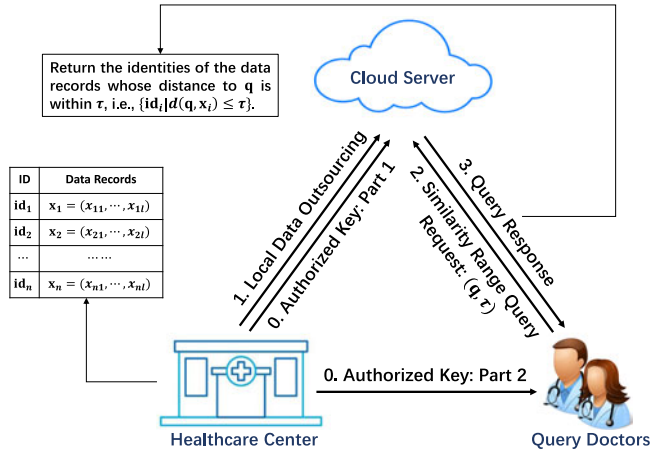


Fig. 2. System model under consideration.

2.1 System Model

In our system model, we consider a typical cloud-assisted similarity range query model, which involves three types of entities, i.e., a healthcare center (HC), a cloud server, and multiple query doctors $\mathcal{U} = \{U_1, U_2, \ldots\}$ as shown in Fig. 2.

• Healthcare Center (HC): The HC (i.e., data owner) has collected a healthcare dataset with huge volumes of electronic health records from patients, which can be denoted by $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Each $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{il})$ is an l-dimensional data record and it has a unique identity id_i for $i = 1, 2, \ldots, n$. In order to make full use of these data, the HC is willing to offer similarity range query service to the doctors. However, since the HC only has limited storage space and computing capability, it chooses to outsource the dataset $\mathcal{X}$ to a powerful cloud and deploy the cloud server to offer similarity range query service to the doctors. Meanwhile, in order to preserve the privacy of the sensitive information in the dataset $\mathcal{X}$, the HC encrypts $\mathcal{X}$ before outsourcing it to the cloud.

• Cloud Server: The cloud server has rich storage space and powerful computing capability. It is responsible for storing the encrypted dataset $\mathcal{X}$ outsourced by the HC and offering similarity range query service to query doctors. In specific, on receiving a query request $(\mathbf{q}, \tau)$ from a query doctor, the cloud server will search on the encrypted dataset $\mathcal{X}$ and find out the data records whose distance to the query record $\mathbf{q}$ is equal to or less than τ, i.e., $\{\mathbf{x}_i | d(\mathbf{q}, \mathbf{x}_i) \leq \tau\}$. Finally, the cloud server returns the identities of these data records, i.e., $\{\mathrm{id}_i | d(\mathbf{q}, \mathbf{x}_i) \leq \tau\}$ to the query doctor.

• Query Doctors $\mathcal{U} = \{U_1, U_2, \ldots\}$: In the system model, there are a set of query doctors (i.e., query users), denoted by $\mathcal{U} = \{U_1, U_2, \ldots\}$. Each query doctor U_p can enjoy the similarity range query service from the cloud server. Meanwhile, when U_p registers in the system, he/she must be authorized by the HC. As shown in Fig. 2, the HC authorizes query doctors by respectively distributing authorized keys to query doctors and the cloud server.

2.2 Security Model

In our security model, the HC is considered to be *trusted* and it is responsible for bootstrapping the whole scheme in the system initialization phase. For the cloud server, it is considered to be *honest-but-curious*, i.e., it will sincerely follow the protocol to store the encrypted dataset $\mathcal{X}$ and offer similarity range

query service to query doctors. However, it may be curious about the plaintext of the data records in the dataset $\mathcal{X}$. Meanwhile, when processing similarity range query requests, the cloud server may be curious about the plaintext of the query requests and the data records satisfying the query condition, i.e., $\{\mathbf{x}_i | d(\mathbf{q}, \mathbf{x}_i) \leq \tau\}$. For the query doctors, they are considered to be *honest-but-curious*. That is, they honestly follow the protocol to launch similarity range query requests to the cloud server, but may be curious about the plaintext of other doctors' query requests and query results. The *honest-but-curious* assumption of the cloud server and query doctors is reasonable because the penalty of dishonest behaviors for the involving cloud server and query doctors is high, including losing the trust of the healthcare center and being prosecuted. In addition, we assume that there is no collusion between the cloud server and query doctors. Similar to the *honest-but-curious* assumption, the non-collusive assumption between the cloud server and query doctors is reasonable due to the high penalty. Note that there may be other passive or active attacks, such as DoS attack and data pollution attack. Since this work focuses on privacy preservation, those attacks are beyond the scope of this paper and will be discussed in our future work.

2.3 Design Goals

Our design goal is to achieve an efficient, privacy-preserving, and practical similarity range query scheme over encrypted data in cloud. In specific, the following objectives should be satisfied.

- *Privacy preservation:* Our proposed scheme should preserve the privacy of the healthcare dataset, query requests as well as query results in our considered system model.
- *Efficiency:* To achieve the privacy preservation requirement, additional computational cost will be inevitably incurred. Therefore, in the proposed scheme, we also aim to improve the efficiency of similarity range query.
- *Practicality:* The proposed scheme should also be practical in real scenarios. Specifically, it should be built upon a single cloud server.

3 Preliminary

In this section, we will introduce an asymmetric scalar-product-preserving encryption (ASPE) scheme [12] and a Hilbert exclusion condition, which are preliminaries of our proposed scheme.

3.1 ASPE Scheme

ASPE scheme [12] was designed for k nearest neighbors query. In ASPE scheme, there are two kinds of data records, i.e., database records and query records. Without loss of generality, both database records and query records are regarded as l-dimensional row vectors. They are encrypted in different ways. Specifically, ASPE scheme $\Pi_{\mathsf{ASPE}} = (\mathsf{AspeSetup}, \mathsf{AspeEnc}, \mathsf{AspeTokenGen})$ is defined as follows.

- AspeSetup: The setup algorithm inputs the parameter l and outputs a random invertible matrix $\mathbf{M} \in \mathbb{R}^{l \times l}$ as the secret key, where $\mathbb{R}$ is the real domain.
- AspeEnc: The encryption algorithm takes the secret key $\mathbf{M}$ as input and encrypts an l-dimensional database record $\mathbf{x}_i$ as $\mathrm{CT}_{\mathbf{x}_i} = \mathbf{x}_i\mathbf{M}$.
- AspeTokenGen: The query token generation algorithm takes the secret key $\mathbf{M}$ as input and generates a query token for an l-dimensional query record $\mathbf{q}$ as $\mathrm{TK}_{\mathbf{q}} = r_q\mathbf{q}(\mathbf{M}^{-1})^T$, where r_q is a random positive number and $\mathbf{M}^{-1}$ is the inverse matrix of $\mathbf{M}$.

The ASPE scheme satisfies the scalar-product-preserving property, which refers that the order of scalar products between a query record and database records over ciphertexts is the same as that over plaintexts. Specifically, let $\mathrm{CT}_{\mathbf{x}_1}$ and $\mathrm{CT}_{\mathbf{x}_2}$ be the ciphertexts of $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Let $\mathrm{TK}_{\mathbf{q}}$ be the query token of $\mathbf{q}$. Then, we have

$$\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}} \geq \mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}} \Leftrightarrow \mathbf{x}_1 \circ \mathbf{q} \geq \mathbf{x}_2 \circ \mathbf{q}.$$

where "$\circ$" denotes the scalar product operation.

Correctness. The scalar-product-preserving property is correct because

$$\begin{aligned}
&\mathrm{CT}_{\mathbf{x}_1} \circ \mathrm{TK}_{\mathbf{q}} \geq \mathrm{CT}_{\mathbf{x}_2} \circ \mathrm{TK}_{\mathbf{q}} \\
\Leftrightarrow\ &\mathrm{CT}_{\mathbf{x}_1}\mathrm{TK}_{\mathbf{q}}^T \geq \mathrm{CT}_{\mathbf{x}_2}\mathrm{TK}_{\mathbf{q}}^T \\
\Leftrightarrow\ &(\mathbf{x}_1\mathbf{M})(r_q\mathbf{q}(\mathbf{M}^{-1})^T)^T \geq (\mathbf{x}_2\mathbf{M})(r_q\mathbf{q}(\mathbf{M}^{-1})^T)^T \\
\Leftrightarrow\ &r_q\mathbf{x}_1\mathbf{M}\mathbf{M}^{-1}\mathbf{q}^T \geq r_q\mathbf{x}_2\mathbf{M}\mathbf{M}^{-1}\mathbf{q}^T \\
\Leftrightarrow\ &r_q\mathbf{x}_1\mathbf{q}^T \geq r_q\mathbf{x}_2\mathbf{q}^T \\
\Leftrightarrow\ &r_q(\mathbf{x}_1 \circ \mathbf{q}) \geq r_q(\mathbf{x}_2 \circ \mathbf{q}) \\
\Leftrightarrow\ &\mathbf{x}_1 \circ \mathbf{q} \geq \mathbf{x}_2 \circ \mathbf{q} \because r_q > 0.
\end{aligned}$$

3.2 Hilbert Exclusion Condition

Hilbert exclusion condition was proposed in [22] and it can be used as a filtration condition in our scheme. In the following, we will introduce the detailed Hilbert exclusion condition. Suppose that $\mathbf{k}_1$ and $\mathbf{k}_2$ denote two l-dimensional data records, and $(\mathbf{q}, \tau)$ denotes a similarity range query request, where $\mathbf{q}$ is an l-dimensional query record and τ is a query range. Meanwhile, $\mathbf{k}_1$, $\mathbf{k}_2$ and $\mathbf{q}$ satisfy that $d(\mathbf{k}_1, \mathbf{q}) > d(\mathbf{k}_2, \mathbf{q})$. Then, we have the following theorem.

Theorem 1. *The condition* $\frac{d(\mathbf{k}_1,\mathbf{q})^2 - d(\mathbf{k}_2,\mathbf{q})^2}{2d(\mathbf{k}_1,\mathbf{k}_2)} > \tau$ *implies that* $d(\mathbf{x}_i, \mathbf{k}_2) < d(\mathbf{x}_i, \mathbf{k}_1)$ *for all* $\mathbf{x}_i$ *such that* $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$. *In other words, all data records satisfying the query condition are closer to* $\mathbf{k}_2$ *than* $\mathbf{k}_1$.

Proof. As shown in Fig. 3, let $V_{\mathbf{k}_1,\mathbf{k}_2}$ denote the perpendicular bisection plane of the line segment $\mathbf{k}_1\mathbf{k}_2$, and the plane $V_{\mathbf{k}_1,\mathbf{k}_2}$ can be represented to be an equation

$$\left(\mathbf{x} - \frac{\mathbf{k}_1 + \mathbf{k}_2}{2}\right) \circ (\mathbf{k}_1 - \mathbf{k}_2) = 0.$$

Then, the distance between $\mathbf{q}$ and $V_{\mathbf{k}_1,\mathbf{k}_2}$ is

$$\begin{aligned}
&d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) \\
&= \left|\left(\mathbf{q} - \frac{\mathbf{k}_1 + \mathbf{k}_2}{2}\right) \circ \frac{\mathbf{k}_1 - \mathbf{k}_2}{d(\mathbf{k}_1, \mathbf{k}_2)}\right| \\
&= \left|\frac{d(\mathbf{k}_2, \mathbf{q})^2 - d(\mathbf{k}_1, \mathbf{q})^2}{2d(\mathbf{k}_1, \mathbf{k}_2)}\right| \\
&= \frac{d(\mathbf{k}_1, \mathbf{q})^2 - d(\mathbf{k}_2, \mathbf{q})^2}{2d(\mathbf{k}_1, \mathbf{k}_2)} \\
&\quad (\because d(\mathbf{k}_1, \mathbf{q}) > d(\mathbf{k}_2, \mathbf{q}) \text{ as shown in Fig. 3}).
\end{aligned}$$

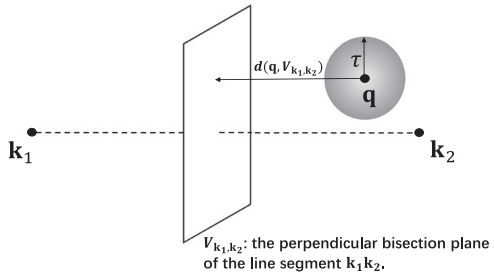


Fig. 3. Illustration for Hilber exclusion condition.

Thus, when $\frac{d(\mathbf{k}_1,\mathbf{q})^2 - d(\mathbf{k}_2,\mathbf{q})^2}{2d(\mathbf{k}_1,\mathbf{k}_2)} > \tau$, we have $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) > \tau$. In this case, it is easy to deduce that $d(\mathbf{x}_i, \mathbf{k}_2) < d(\mathbf{x}_i, \mathbf{k}_1)$ for all $\mathbf{x}_i$ such that $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ as shown in Fig. 3. □

4 Our Proposed Scheme

In this section, we present our EPSim scheme. Before delving into the details, we first introduce a modified ASPE scheme and a tree data structure, called Quadsector tree, which are important building blocks of our proposed scheme.

4.1 The Modified ASPE Scheme

The modified ASPE scheme is based on ASPE scheme [12]. Different from ASPE scheme, the modified ASPE scheme assumes that all values in database records and query records are integers so that we can add some numbers into the ciphertexts and query tokens for better security. If not, we can transform them into integers by scaling. For example, a vector $(0.1, 0.12)$ can be scaled to an integer vector $(10, 12)$ by enlarging 100 times. Since the modified ASPE scheme is only concerned about the sign of the scalar product between data records and query records (introduced in the following context), the scaling transformation does not affect the correctness of the modified ASPE scheme. Specifically, the modified ASPE scheme $\Pi_{\mathsf{MASPE}} = (\mathsf{MAspeSetup}, \mathsf{MAspeEnc}, \mathsf{MAspeTokenGen})$ can be defined as follows.

- MAspeSetup: The setup algorithm inputs the parameter l and outputs an invertible matrix $\mathbf{M} \in \mathbb{R}^{(l+4)\times(l+4)}$ as the secret key, where $\mathbb{R}$ is the real domain.
- MAspeEnc: The encryption algorithm inputs $\mathbf{M}$ and an l-dimensional data record $\mathbf{x}_i$. The algorithm first extends $\mathbf{x}_i$ to be an $(l+4)$-dimensional vector $\mathbf{x}'_i = (r_{i,1}\mathbf{x}_i, r_{i,2}, r_{i,3}, r_{i,4}, r_{i,4})$, where $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$ are random real numbers satisfying $r_{i,1} > r_{i,2} > 0$ and $r_{i,1} > r_{i,3} > 0$. Meanwhile, the random numbers $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$ are different for different data records. Then, it encrypts $\mathbf{x}_i$ into a ciphertext $\mathrm{CT}_{\mathbf{x}_i} = \mathbf{x}'_i\mathbf{M}$.
- MAspeTokenGen: The query token generation algorithm inputs $\mathbf{M}$ and an l-dimensional query record $\mathbf{q}$. The algorithm first extends $\mathbf{q}$ to be an $(l+4)$-dimensional vector $\mathbf{q}' = (r_{q,1}\mathbf{q}, -r_{q,2}, -r_{q,3}, r_{q,4}, -r_{q,4})$, where $\{r_{q,1}, r_{q,2}, r_{q,3}, r_{q,4}\}$ are random real numbers satisfying $r_{q,1} > 2 * r_{q,2} > 0$ and $r_{q,1} > 2 * r_{q,3} > 0$. Meanwhile, the random numbers $\{r_{q,1}, r_{q,2}, r_{q,3}, r_{q,4}\}$ are different for different query records. Then, it generates a query token for $\mathbf{q}$ as $\mathrm{TK}_{\mathbf{q}} = \mathbf{q}'(\mathbf{M}^{-1})^T$.

Due to the introduction of random numbers in the encryption and token generation algorithms, the modified ASPE scheme cannot satisfy the scalar-product-preserving property between data records and query records anymore. Instead, it can preserve the sign of scalar product between data records and query records. Specifically, if $\mathrm{CT}_{\mathbf{x}_i}$ is the ciphertext of $\mathbf{x}_i$ and $\mathrm{TK}_{\mathbf{q}}$ is the query token of $\mathbf{q}$, they satisfy

$$\begin{cases} \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0 \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} > 0; & (1) \\ \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0 \Leftrightarrow \mathbf{x}_i \circ \mathbf{q} \leq 0. & (2) \end{cases}$$

Correctness. In the following, we respectively show the correctness of Eqs. (1) and (2).

Theorem 2. *Eq. (1) is correct.*

Proof. We prove the correctness of Eq. (1) by proving the sufficiency and necessity of Eq. (1), respectively.

• Sufficiency: Prove $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} > 0$.

First, we have

$$\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} = \mathrm{CT}_{\mathbf{x}_i}\mathrm{TK}_{\mathbf{q}}^T = \mathbf{x}'_i\mathbf{M}(\mathbf{q}'(\mathbf{M}^{-1})^T)^T = \mathbf{x}'_i\mathbf{q}'^T.$$

Meanwhile, we have

$$\begin{aligned} &\mathbf{x}'_i\mathbf{q}'^T \\ &= (r_{i,1}\mathbf{x}_i, r_{i,2}, r_{i,3}, r_{i,4}, r_{i,4})(r_{q,1}\mathbf{q}, -r_{q,2}, -r_{q,3}, r_{q,4}, -r_{q,4})^T \\ &= r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} + r_{i,4} * r_{q,4} - r_{i,4} * r_{q,4} \\ &= r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3}. \end{aligned}$$

That is, we have

$$\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} = r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3}. \quad (3)$$

When $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0$, we have

$$\begin{aligned} &r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} > 0 \\ \Rightarrow\ &\mathbf{x}_i\mathbf{q}^T > \frac{r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3}}{r_{i,1} * r_{q,1}} > 0 \\ &(\because r_{i,1}, r_{i,2}, r_{i,3}, r_{q,1}, r_{q,2}, r_{q,3} > 0) \\ \Rightarrow\ &\mathbf{x}_i\mathbf{q}^T > 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} > 0. \end{aligned}$$

Thus, from $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0$, we can deduce that $\mathbf{x}_i \circ \mathbf{q} > 0$.

• Necessity: Prove $\mathbf{x}_i \circ \mathbf{q} > 0 \Rightarrow \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0$.

When $\mathbf{x}_i \circ \mathbf{q} > 0$, we have $\mathbf{x}_i\mathbf{q}^T > 0$. Since $\mathbf{x}_i$ and $\mathbf{q}$ are integer vectors, $\mathbf{x}_i\mathbf{q}^T$ is an integer. Then, from $\mathbf{x}_i \circ \mathbf{q} > 0$, we can deduce that

$$\begin{aligned} &\mathbf{x}_i\mathbf{q}^T \geq 1 \\ \Rightarrow\ &r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T \geq r_{i,1} * r_{q,1} \because r_{i,1}, r_{q,1} > 0 \\ \Rightarrow\ &r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} \\ &\geq r_{i,1} * r_{q,1} - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3}. \end{aligned} \quad (4)$$

Since $r_{i,1} > r_{i,2} > 0$, $r_{i,1} > r_{i,3} > 0$, $r_{q,1} > 2 * r_{q,2} > 0$ and $r_{q,1} > 2 * r_{q,3} > 0$, we have

$$\begin{cases} r_{i,1} * r_{q,1} > r_{i,2} * 2 * r_{q,2} \\ r_{i,1} * r_{q,1} > r_{i,3} * 2 * r_{q,3} \end{cases} \Rightarrow \begin{cases} \frac{1}{2}r_{i,1} * r_{q,1} > r_{i,2} * r_{q,2} \\ \frac{1}{2}r_{i,1} * r_{q,1} > r_{i,3} * r_{q,3}. \end{cases}$$

Thus, we can deduce that

$$\frac{1}{2}r_{i,1} * r_{q,1} + \frac{1}{2}r_{i,1} * r_{q,1} > r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3} \quad (5)$$
$$\Rightarrow r_{i,1} * r_{q,1} > r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3}$$

$$\Rightarrow r_{i,1} * r_{q,1} - r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3} > 0. \quad (6)$$

By combining Eqs. (4) and (6), we can deduce that

$$r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} > 0$$
$$\Rightarrow \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0 \because \text{Eq. (3)}.$$

Thus, from $\mathbf{x}_i \circ \mathbf{q} > 0$, we have $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} > 0$. Hence, Eq. (1) is correct. □

Theorem 3. *Eq. (2) is correct.*

Proof. We prove the correctness of Eq. (2) by proving the sufficiency and necessity of Eq. (2), respectively.

• Sufficiency: Prove $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0 \Rightarrow \mathbf{x}_i \circ \mathbf{q} \leq 0$.

According to Eq. (3), when $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$, we have

$$r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} < 0$$
$$\Rightarrow \mathbf{x}_i\mathbf{q}^T < \frac{r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3}}{r_{i,1} * r_{q,1}} \because r_{i,1}, r_{q,1} > 0$$
$$\Rightarrow \mathbf{x}_i\mathbf{q}^T < \frac{r_{i,2} * r_{q,2} + r_{i,3} * r_{q,3}}{r_{i,1} * r_{q,1}} < 1 \because \text{Eq. (5)}.$$

Since $\mathbf{x}_i$ and $\mathbf{q}$ are integer vectors, $\mathbf{x}_i\mathbf{q}^T$ is an integer. Then, from $\mathbf{x}_i\mathbf{q}^T < 1$, we have $\mathbf{x}_i\mathbf{q}^T \leq 0$. Thus, from $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$, we can deduce that $\mathbf{x}_i \circ \mathbf{q} \leq 0$.

• Necessity: Prove $\mathbf{x}_i \circ \mathbf{q} \leq 0 \Rightarrow \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$.

When $\mathbf{x}_i \circ \mathbf{q} \leq 0$, we have $\mathbf{x}_i\mathbf{q}^T \leq 0$ and can deduce that

$$\mathbf{x}_i\mathbf{q}^T \leq 0$$
$$\Rightarrow r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T \leq 0 \because r_{i,1}, r_{q,1} > 0$$
$$\Rightarrow r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} \leq -r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} < 0 \quad (\because r_{i,2}, r_{q,2}, r_{i,3}, r_{q,3} > 0)$$
$$\Rightarrow r_{i,1} * r_{q,1}\mathbf{x}_i\mathbf{q}^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} < 0$$
$$\Rightarrow \mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0 \because \text{Eq. (3)}.$$

Thus, from $\mathbf{x}_i \circ \mathbf{q} \leq 0$, we have $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$. Therefore, Eq. (2) is correct. □

4.2 Quadsector Tree

The Quadsector tree is a tree data structure and designed for similarity range queries over multi-dimensional data records. In the Quadsector tree, there are two kinds of nodes, i.e., internal nodes and leaf nodes. For each internal node, it has exactly 4 key values $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ and 4 subtrees $\{T_1, T_2, T_3, T_4\}$, where each $\mathbf{k}_j$ is a multi-dimensional data record for $j = 1, 2, 3, 4$. At the same time, the data records in the subtree T_j are closer to $\mathbf{k}_j$ than other key values for $j = 1, 2, 3, 4$, where the distance is measured by the euclidean distance. For each leaf node, it stores data records and the number of stored data records is up to 3. This is because when a leaf node stores 4 data records, it can be further divided. In the following, we show how to build a Quadsector tree T for a given dataset $\mathcal{X}$.

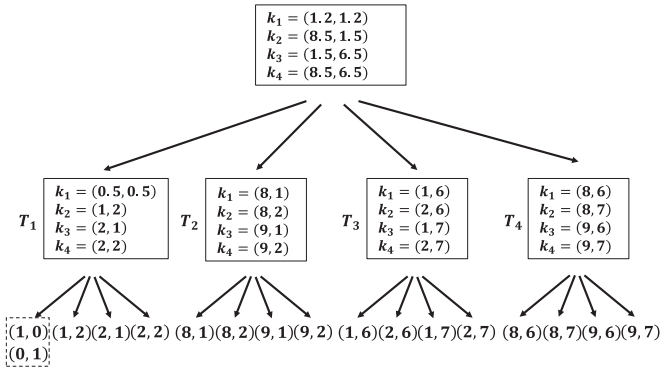


Fig. 4. An example of Quadsector tree for a dataset $\mathcal{X} = \{(1,0), (0,1), (1,2), (2,1), (2,2), (8,1), (9,1), (8,2), (9,2), (1,6), (2,6), (1,7), (2,7), (8,6), (8,7), (9,6), (9,7)\}$.

Step 1. Compute the key values for T's root node. First, divide the dataset $\mathcal{X}$ into 4 subdatasets with K-Means clustering technique [23], and these 4 subdatasets can be represented to be $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4\}$. For $\mathcal{X}_j$, it has a centroid, denoted by $\mathbf{k}_j$. Then, 4 subdatasets have 4 centroids, i.e., $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$. According to the property of K-Means clustering, the data records in $\mathcal{X}_j$ is closer to $\mathbf{k}_j$ than other centroids. Thus, we can use $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ as the key values of the root node.

Step 2. Build the subtrees for T's root node. First, recursively build a subtree T_j for the subdataset $\mathcal{X}_j$, where $j = 1, 2, 3, 4$. Then, use $\{T_1, T_2, T_3, T_4\}$ as the subtrees of T's root node. Meanwhile, T_j is associated with the key value $\mathbf{k}_j$ for $j = 1, 2, 3, 4$.

For a clear description, we give an example to show the concept of Quadsector tree.

Example 1. Assume $\mathcal{X} = \{(1,0), (0,1), (1,2), (2,1), (2, 2), (8,1), (9,1), (8,2), (9,2), (1,6), (2,6), (1,7), (2,7), (8,6), (8,7), (9,6), (9,7)\}$ is a two-dimensional dataset. Then, based on $\mathcal{X}$, a Quadsector tree T can be built as shown in Fig. 4. From this figure, we can see that the root node has 4 key values $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ and 4 subtrees $\{T_1, T_2, T_3, T_4\}$. Each subtree is associated with a subdataset, e.g., T_1 is associated with $\mathcal{X}_1 = \{(1,0), (0,1), (1,2), (2,1), (2,2)\}$. Meanwhile, the key value is the centroid of the corresponding subdataset, e.g., $\mathbf{k}_1 = (1.2, 1.2)$ is the centroid of $\mathcal{X}_1$. In addition, the data records in subtree T_j is closer to $\mathbf{k}_j$ than other key values for $j = 1, 2, 3, 4$.

The Quadsector tree can support efficient similarity range queries and the query algorithm contains two stages, i.e., filtration stage and verification stage, as shown in Algorithm 1. In the filtration stage, the query algorithm finds out candidate data records that are likely to be query result. In the verification stage, the algorithm further verifies all candidate data records and returns the query result.

∗ *Filtration stage:* In the filtration stage, the algorithm searches the Quadsector tree in a depth-first order. Specifically, for a tree node, if it is a leaf node, its key values can be directly added into the candidate set $\mathcal{C}$. If it is an internal node, it contains 4 key values $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ and 4 subtrees

$\{T_1, T_2, T_3, T_4\}$. For each key value $\mathbf{k}_j$, if there exists a key value $\mathbf{k}_{j_1}$ such that

$$\frac{d(\mathbf{k}_j, \mathbf{q})^2 - d(\mathbf{k}_{j_1}, \mathbf{q})^2}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau, \tag{7}$$

for $1 \leq j_1 \leq 4$ and $j_1 \neq j$, the subtree T_j cannot contain the query result and should be pruned. Otherwise, the subtree T_j is added into the stack S and it needs to be searched later.

$*$ *Verification stage:* In the verification stage, the algorithm verifies the data records in the candidate set $\mathcal{C}$. For each $\mathbf{x}_i \in \mathcal{C}$, if $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$, add $\mathbf{x}_i$ into the query result $\mathcal{R}$, i.e, $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$. Otherwise, $\mathbf{x}_i$ does not satisfy the query condition and can be discarded.

Algorithm 1. Similarity Range Query Over Plaintext

Input: The Quadsector tree T and query request $(\mathbf{q}, \tau)$
Output: The query result $\mathcal{R}$.

// Filtration phase
$\mathcal{C} = \emptyset$; // Initialize the candidate set
Stack S $= \emptyset$;
S.$push(T.root)$;
while S is not empty **do**
  $node =$ S.$pop()$;
  if $node$ is a leaf node **then**
    Add $node$'s key values into $\mathcal{C}$
  else
    Let $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ be $node$'s key values;
    for $j = 1, 2, 3, 4$ **do**
      if $\exists\ \mathbf{k}_{j_1}$ satisfies $\frac{d(\mathbf{k}_j, \mathbf{q})^2 - d(\mathbf{k}_{j_1}, \mathbf{q})^2}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau$ **then**
        T_j is pruned;
      else
        S.$push(T_j)$;
// Verification phase
$\mathcal{R} = \emptyset$; // Initialize the query result
for each data record $\mathbf{x}_i$ in $\mathcal{C}$ **do**
  if $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ **then**
    $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$;
return $\mathcal{R}$;

Correctness. The similarity range query algorithm is correct when the filtration inequality Eq. (7) is correct. In the following, we prove that the filtration inequality is correct by Theorem 4.

Theorem 4. *Suppose that* $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ *are 4 key values and* $\{T_1, T_2, T_3, T_4\}$ *are 4 subtrees in an internal node. Then,* T_j *can be pruned when there exists a key value* $\mathbf{k}_{j_1}$ $(j_1 \neq j)$ *such that* $\frac{d(\mathbf{k}_j, \mathbf{q})^2 - d(\mathbf{k}_{j_1}, \mathbf{q})^2}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau$ *for* $j = 1, 2, 3, 4$.

Proof. For the $\mathbf{k}_j$, suppose that there exists a key value $\mathbf{k}_{j_1}$ $(j_1 \neq j)$ such that $\frac{d(\mathbf{k}_j, \mathbf{q})^2 - d(\mathbf{k}_{j_1}, \mathbf{q})^2}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau$. Then, according to Theorem 1, we can deduce that

$$d(\mathbf{x}_i, \mathbf{k}_{j_1}) < d(\mathbf{x}_i, \mathbf{k}_j), \tag{8}$$

for all $\mathbf{x}_i$ such that $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$. In this case, the data records within τ of $\mathbf{q}$ cannot be stored in the subtree T_j. This is because if $\mathbf{x}_i$ is stored in T_j, $d(\mathbf{x}_i, \mathbf{k}_j)$ should be equal to or less than $d(\mathbf{x}_i, \mathbf{k}_{j_1})$, i.e., $d(\mathbf{x}_i, \mathbf{k}_j) \leq d(\mathbf{x}_i, \mathbf{k}_{j_1})$, which contradicts to Eq. (8). Thus, T_j can be pruned. □

4.3 Description of Our EPSim Scheme

In this subsection, we introduce our proposed EPSim scheme, which consists of three phases, i.e., system initialization, local data outsourcing, and similarity range query.

4.3.1 System Initialization

In the system initialization phase, the HC is responsible for bootstrapping the whole system. It first generates a random invertible matrix $\mathbf{M} \in \mathbb{R}^{(l+9)\times(l+9)}$ as the secret key, where $\mathbb{R}$ denotes the real domain. Meanwhile, it randomly chooses a public key encryption algorithm $E(\cdot)$, and initializes a pair of public key and secret key (pk, sk) for the chosen algorithm. After that, the public key pk is published and the secret key sk is sent to the cloud server. When a query doctor U_p registers in the system, the HC randomly chooses two matrices $\mathbf{M}_{p1}$ and $\mathbf{M}_{p2}$ such that $(\mathbf{M}_{p1}^{-1})^T * \mathbf{M}_{p2} = (\mathbf{M}^{-1})^T$. Then, the HC authorizes U_p by respectively distributing $\mathbf{M}_{p1}$ and $\mathbf{M}_{p2}$ to U_p and the cloud server.

4.3.2 Local Data Outsourcing

The HC has a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Each $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{il})$ is an l-dimensional data record and it has a unique identity id_i for $i = 1, 2, \ldots, n$. In the local data outsourcing phase, the HC encrypts the dataset $\mathcal{X}$ and outsources it to the cloud as the following steps.

Step 1. Build a Quadsector tree T for the dataset $\mathcal{X}$. In T, each internal node has 4 key values $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$ and 4 subtrees $\{T_1, T_2, T_3, T_4\}$. Meanwhile, the subtree T_j stores the data records who are closer to $\mathbf{k}_j$ than other key values for $j = 1, 2, 3, 4$. For the leaf nodes, each of them contains up to 3 data records.

Step 2. Encrypt the internal nodes of T. For each internal node, it has 4 key values $\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$. For each $\mathbf{k}_j$, the HC constructs three $(l+5)$-dimensional vectors $\{\mathbf{k}_{j,j_1} | j_1 = 1, 2, 3, 4; j_1 \neq j\}$ for $j = 1, 2, 3, 4$. Each $\mathbf{k}_{j,j_1}$ is constructed based on key values $\mathbf{k}_j$ and $\mathbf{k}_{j_1}$ as

$$\mathbf{k}_{j,j_1} = (||\mathbf{k}_j||_2^2 - ||\mathbf{k}_{j_1}||_2^2, \mathbf{k}_j - \mathbf{k}_{j_1}, d(\mathbf{k}_j, \mathbf{k}_{j_1}), 0.5, -0.5, 0), \tag{9}$$

where $||\cdot||_2^2$ is the square of euclidean norm, e.g., $||\mathbf{k}_j||_2^2 = \mathbf{k}_j \circ \mathbf{k}_j$. Then, the HC extends $\mathbf{k}_{j,j_1}$ to be an $(l+9)$-dimensional vector as $\mathbf{k}'_{j,j_1} = (r_{j,j_1,1}\mathbf{k}_{j,j_1}, r_{j,j_1,2}, r_{j,j_1,3}, r_{j,j_1,4}, r_{j,j_1,4})$, where $\{r_{j,j_1,1}, r_{j,j_1,2}, r_{j,j_1,3}, r_{j,j_1,4}\}$ are random numbers satisfying $r_{j,j_1,1} > r_{j,j_1,2} > 0$ and $r_{j,j_1,1} > r_{j,j_1,3} > 0$. Meanwhile, the random numbers $\{r_{j,j_1,1}, r_{j,j_1,2}, r_{j,j_1,3}, r_{j,j_1,4}\}$ are different for different $\mathbf{k}_{j,j_1}$'s. Furthermore, the HC encrypts $\mathbf{k}_{j,j_1}$ as

$$\mathrm{CT}_{\mathbf{k}_{j,j_1}} = \mathsf{MAspeEnc}(\mathbf{k}_{j,j_1}, \mathbf{M}) = \mathbf{k}'_{j,j_1}\mathbf{M}.$$

Thus, each internal node will be encrypted to be $\{\mathrm{CT}_{\mathbf{k}_{j,j_1}} | j, j_1 = 1, 2, 3, 4; j_1 \neq j\}$. Meanwhile, the internal node has 4 subtrees $\{T_1, T_2, T_3, T_4\}$.

Step 3. Encrypt the leaf nodes of T. For each leaf node $LNode$, it has up to 3 data records. For each $\mathbf{x}_i \in LNode$, the HC constructs an $(l+5)$-dimensional vector $\mathbf{x}'_i$ as

$$\mathbf{x}'_i = (||\mathbf{x}_i||_2^2, \mathbf{x}_i, 0, 0.5, 0.5, -1). \tag{10}$$

Then, the HC extends $\mathbf{x}'_i$ to be an $(l+9)$-dimensional vector $\mathbf{x}''_i = (r_{i,1}\mathbf{x}'_i, r_{i,2}, r_{i,3}, r_{i,4}, r_{i,4})$, where $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$ are

random real numbers satisfying $r_{i,1} > r_{i,2} > 0$ and $r_{i,1} > r_{i,3} > 0$. Meanwhile, $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$ are different for different data records. Furthermore, the HC encrypts $\mathbf{x}_i$ as

$$\mathrm{CT}_{\mathbf{x}_i} = \mathsf{MAspeEnc}(\mathbf{x}_i'', \mathbf{M}) = \mathbf{x}_i''\mathbf{M}.$$

In this case, each leaf node $LNode$ will be encrypted to be $\{(\mathrm{id}_i, \mathrm{CT}_{\mathbf{x}_i}) | \mathbf{x}_i \in LNode\}$. After this step, the Quadsector tree T is encrypted to be T_{cipher}.

Step 4. Finally, the HC outsources the encrypted Quadsector tree T_{cipher} to the cloud server.

4.3.3 Similarity Range Query

Upon receiving the encrypted Quadsector tree T_{cipher}, the cloud server can provide the similarity range query service to query doctors. In specific, the query doctor U_p can launch a similarity range query request $(\mathbf{q}, \tau)$ to the cloud server as the following steps.

Step 1. Based on $(\mathbf{q}, \tau)$, the query doctor U_p constructs an $(l+5)$-dimensional vector $\mathbf{q}'$ as

$$\mathbf{q}' = (1, -2\mathbf{q}, -2\tau, ||\mathbf{q}||_2^2, ||\mathbf{q}||_2^2, \tau^2). \tag{11}$$

Then, U_p extends $\mathbf{q}'$ to be an $(l+9)$-dimensional vector $\mathbf{q}'' = (r_{q,1}\mathbf{q}', -r_{q,2}, -r_{q,3}, r_{q,4}, -r_{q,4})$, where $\{r_{q,1}, r_{q,2}, r_{q,3}, r_{q,4}\}$ are random real numbers satisfying $r_{q,1} > 2 * r_{q,2} > 0$ and $r_{q,1} > 2 * r_{q,3} > 0$. Meanwhile, $\{r_{q,1}, r_{q,2}, r_{q,3}, r_{q,4}\}$ are different for different query requests. Furthermore, with the secret key $\mathbf{M}_{p1}$, U_p uses the modified ASPE scheme to generate a query token $\mathrm{TK}_{\mathbf{q}}$ for $\mathbf{q}$ as

$$\mathrm{TK}_{\mathbf{q}} = \mathsf{MAspeTokenGen}(\mathbf{q}'', \mathbf{M}_{p1}) = \mathbf{q}''(\mathbf{M}_{p1}^{-1})^T.$$

In addition, U_p chooses a random access key ak and uses the public key encryption algorithm to encrypt ak as $E_{pk}(ak)$. Then, U_p sends a similarity range query request together with $\{E_{pk}(ak), \mathrm{TK}_{\mathbf{q}}\}$ to the cloud server.

Step 2. On receiving the query request, the cloud server first checks whether U_p is authorized or not. If not, the cloud server will reject the query request. Otherwise, the cloud server processes the query request as the following steps.

Step 3. The cloud server uses the authorized key $\mathbf{M}_{p2}$ to update the query token $\mathrm{TK}_{\mathbf{q}}$ as

$$\mathrm{TK}_{\mathbf{q}} = \mathrm{TK}_{\mathbf{q}}\mathbf{M}_{p2} = \mathbf{q}''(\mathbf{M}_{p1}^{-1})^T\mathbf{M}_{p2} = \mathbf{q}''(\mathbf{M}^{-1})^T.$$

After updating, the query token $\mathrm{TK}_{\mathbf{q}}$ is converted to a token generated by the secret key $\mathbf{M}$.

Step 4. With the query token $\mathrm{TK}_{\mathbf{q}}$, the cloud server searches on the encrypted Quadsector tree T_{cipher} to obtain the query result. Similar to the query algorithm over plaintext in Algorithm 1, the query algorithm over encrypted data contains two stages, i.e., filtration stage and verification stage, as shown in Algorithm 2.

$*$ *Filtration stage:* The filtration stage in Algorithm 2 is similar to that in Algorithm 1. Different from Algorithm 1, in Algorithm 2, each internal node contains ciphertexts $\{\mathrm{CT}_{\mathbf{k}_{j,j_1}} | j, j_1 = 1, 2, 3, 4; j_1 \neq j\}$ and 4 subtrees $\{T_1, T_2, T_3, T_4\}$. Meanwhile, the filtration inequality in Eq. (7) is replaced by the inequality $\mathrm{CT}_{\mathbf{k}_{j,j_1}} \circ \mathrm{TK}_{\mathbf{q}} > 0$.

Correctness of the Filtration Inequality Replacement. The inequality $\mathrm{CT}_{\mathbf{k}_{j,j_1}} \circ \mathrm{TK}_{\mathbf{q}} > 0$ is equivalent to Eq. (7). First, we have

$$\begin{aligned}
&\mathrm{CT}_{\mathbf{k}_{j,j_1}} \circ \mathrm{TK}_{\mathbf{q}} > 0 \\
\Leftrightarrow\ &\mathrm{CT}_{\mathbf{k}_{j,j_1}} \mathrm{TK}_{\mathbf{q}}^T > 0 \\
\Leftrightarrow\ &(\mathbf{k}'_{j,j_1}\mathbf{M})(\mathbf{q}''(\mathbf{M}^{-1})^T)^T > 0 \\
\Leftrightarrow\ &\mathbf{k}'_{j,j_1}\mathbf{q}''^T > 0 \\
\Leftrightarrow\ &r_{j,j_1,1} * r_{q,1}\mathbf{k}_{j,j_1}\mathbf{q}'^T - r_{j,j_1,2} * r_{q,2} - r_{j,j_1,3} * r_{q,3} > 0 \\
\Leftrightarrow\ &\mathbf{k}_{j,j_1}\mathbf{q}'^T > 0 \because \text{based on the modified ASPE.}
\end{aligned}$$

From $\mathbf{k}_{j,j_1}\mathbf{q}'^T > 0$, we can deduce that

$$||\mathbf{k}_j||_2^2 - ||\mathbf{k}_{j_1}||_2^2 - 2\mathbf{q} \circ (\mathbf{k}_j - \mathbf{k}_{j_1}) - 2\tau d(\mathbf{k}_j, \mathbf{k}_{j_1}) > 0.$$

Furthermore, we can deduce that

$$\begin{aligned}
&\frac{||\mathbf{k}_j||_2^2 - ||\mathbf{k}_{j_1}||_2^2 - 2\mathbf{q}(\mathbf{k}_j - \mathbf{k}_{j_1})}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau \\
\Leftrightarrow\ &\frac{(||\mathbf{k}_j||_2^2 - 2\mathbf{q}\mathbf{k}_j + ||\mathbf{q}||_2^2) - (||\mathbf{k}_{j_1}||_2^2 - 2\mathbf{q}\mathbf{k}_{j_1} + ||\mathbf{q}||_2^2)}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau \\
\Leftrightarrow\ &\frac{d(\mathbf{k}_j, \mathbf{q})^2 - d(\mathbf{k}_{j_1}, \mathbf{q})^2}{2d(\mathbf{k}_j, \mathbf{k}_{j_1})} > \tau.
\end{aligned}$$

That is, Eq. (7) holds. Thus, the inequality $\mathrm{CT}_{\mathbf{k}_{j,j_1}} \circ \mathrm{TK}_{\mathbf{q}} > 0$ is equivalent to Eq. (7).

Algorithm 2. Similarity Range Query Over Ciphertext

Input: The encrypted Quadsector tree T_{cipher} and query token $\mathrm{TK}_{\mathbf{q}}$
Output: The query result $\mathcal{R}$.

```
// Filtration phase
C = ∅; // Initialize the candidate set
Stack S = ∅;
S.push(T_cipher.root);
while S is not empty do
    node = S.pop();
    if node is a leaf node then
        Add node's encrypted data records into C
    else if node is an internal node then
        for j = 1, 2, 3, 4 do
            if ∃ j_1 satisfies CT_{k_{j,j_1}} ∘ TK_q > 0 then
                T_j is pruned;
            else
                S.push(T_j);
// Verification phase
R = ∅; // Initialize the query result
for each ciphertext CT_{x_i} in C do
    if CT_{x_i} ∘ TK_q < 0 then
        R = R ∪ {id_i};
return R;
```

$*$ *Verification stage:* The verification stage in Algorithm 2 is similar to that in Algorithm 1. Different from Algorithm 1, the verification inequality $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ is replaced by inequality $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$. In addition, when the verification inequality holds, the algorithm adds id_i instead of $\mathbf{x}_i$ to the query result $\mathcal{R}$, i.e., $\mathcal{R} = \mathcal{R} \cup \{\mathrm{id}_i\}$.

Correctness of the Verification Inequality Replacement. The inequality $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$ is equivalent to the inequality $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$. First, we have

$$\begin{aligned}
&\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0 \\
\Leftrightarrow\ &\mathrm{CT}_{\mathbf{x}_i} \mathrm{TK}_{\mathbf{q}}^T < 0 \\
\Leftrightarrow\ &(\mathbf{x}_i'' \mathbf{M})(\mathbf{q}''(\mathbf{M}^{-1})^T)^T < 0 \\
\Leftrightarrow\ &\mathbf{x}_i'' \mathbf{q}''^T < 0 \\
\Leftrightarrow\ &r_{i,1} * r_{q,1} \mathbf{x}_i' \mathbf{q}'^T - r_{i,2} * r_{q,2} - r_{i,3} * r_{q,3} < 0 \\
\Leftrightarrow\ &\mathbf{x}_i' \mathbf{q}'^T \leq 0 \because \text{based on the modified ASPE.}
\end{aligned}$$

From $\mathbf{x}_i' \mathbf{q}'^T \leq 0$, we can deduce that

$$||\mathbf{x}_i||_2^2 - 2\mathbf{x}_i \circ \mathbf{q} + ||\mathbf{q}||^2 - \tau^2 \leq 0.$$

Furthermore, we have

$$d(\mathbf{x}_i, \mathbf{q})^2 - \tau^2 \leq 0 \Leftrightarrow\ d(\mathbf{x}_i, \mathbf{q}) \leq \tau.$$

Thus, the inequality $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}} < 0$ is equivalent to the inequality $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$.

After executing Algorithm 2, the cloud server obtains the query result $\mathcal{R}$.

Step 5. The cloud server recovers the access key ak by decrypting the ciphertext $E_{pk}(ak)$. Then, it uses ak to encrypt the query result as $AES_{ak}(\mathcal{R})$, and returns the encrypted query result $AES_{ak}(\mathcal{R})$ to U_p.

Step 6. On receiving the encrypted query result $AES_{ak}(\mathcal{R})$, the query doctor U_p uses ak to recover the query result $\mathcal{R}$.

5 Security Analysis

In this section, we analyze the security of the modified ASPE scheme and our proposed EPSim scheme.

5.1 Security of The Modified ASPE Scheme

Similiar as [24], we prove that the modified ASPE scheme is selectively secure in the real/ideal simulation setting. Before formally proving the security, we first define the leakage in the modified ASPE scheme. Given a data record $\mathbf{x}_i$ and a query record $\mathbf{q}$, the leakage is the sign of scalar product between $\mathrm{CT}_{\mathbf{x}_i}$ and $\mathrm{TK}_{\mathbf{q}}$, i.e., $\mathcal{L} = \mathrm{sign}(\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}})$. With the leakage, we respectively define the real experiment and ideal experiment.

Real Experiment. The real experiment involves a challenger and a PPT (probabilistic polynomial-time) adversary $\mathcal{A}$, and they interact as follows.

- *Setup*: In the setup algorithm, the adversary $\mathcal{A}$ chooses a random data record $\mathbf{x}$, and sends it to the challenger. Then, the challenger executes MAspeSetup algorithm to generate a secret key $\mathbf{M}$.
- *Query phase 1*: The adversary $\mathcal{A}$ adaptively chooses p_1 query records $\{\mathbf{q}_j | 1 \leq j \leq \mathbf{q}_{p_1}\}$ and sends them to the challenger, where p_1 is a polynomial number. Then, the challenger generates a token for each $\mathbf{q}_j$ as $\mathrm{TK}_{\mathbf{q}_j} = \mathsf{MAspeTokenGen}(\mathbf{M}, \mathbf{q}_j)$ for $1 \leq j \leq p_1$, and returns these tokens to $\mathcal{A}$.
- *Challenge phase*: The challenger encrypts $\mathbf{x}_i$ as $\mathrm{CT}_{\mathbf{x}_i} = \mathsf{MAspeEnc}(\mathbf{M}, \mathbf{x}_i)$ and returns $\mathrm{CT}_{\mathbf{x}_i}$ to $\mathcal{A}$.
- *Query phase 2*: Similar to *Query phase 1*, the adversary $\mathcal{A}$ adaptively chooses another $(p_2 - p_1)$ query records $\{\mathbf{q}_j | p_1 < j \leq p_2\}$ and obtains the query tokens $\{\mathrm{TK}_{\mathbf{q}_j} | p_1 < j \leq p_2\}$ from the challenger, where p_2 is a polynomial number.

Ideal Experiment. The ideal experiment involves a simulator with leakage $\mathcal{L}$ and a PPT adversary $\mathcal{A}$, and they interact as follows.

- *Setup*: In the setup algorithm, the adversary $\mathcal{A}$ chooses a random data record $\mathbf{x}_i$, and sends it to the simulator. Then, the simulator chooses a random vector $\mathrm{CT}_{\mathbf{x}_i}$ as the ciphertext of $\mathbf{x}_i$.
- *Query phase 1*: The adversary $\mathcal{A}$ adaptively chooses p_1 query records $\{\mathbf{q}_j | 1 \leq j \leq \mathbf{q}_{p_1}\}$ and sends them to the simulator, where p_1 is a polynomial number. For each $\mathbf{q}_j$, the simulator has a leakage $\mathcal{L} = \mathrm{sign}(\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}_j})$, and it can use $\mathcal{L}$ to construct a query token $\mathrm{TK}_{\mathbf{q}_j}$ for $\mathbf{q}_j$ as
  1) If $\mathrm{sign}(\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}_j}) > 0$, choose a random vector $\mathrm{TK}_{\mathbf{q}_j}$ such that $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}_j} > 0$.
  2) If $\mathrm{sign}(\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}_j}) < 0$, choose a random vector $\mathrm{TK}_{\mathbf{q}_j}$ such that $\mathrm{CT}_{\mathbf{x}_i} \circ \mathrm{TK}_{\mathbf{q}_j} < 0$.

  Then, the simulator returns these query tokens to $\mathcal{A}$.
  Challenge phase: The simulator returns $\mathrm{CT}_{\mathbf{x}_i}$ to $\mathcal{A}$.
  Query phase 2: Similar to *Query phase 1*, the adversary $\mathcal{A}$ adaptively chooses another $(p_2 - p_1)$ query records $\{\mathbf{q}_j | p_1 < j \leq p_2\}$ and obtains the query tokens $\{\mathrm{TK}_{\mathbf{q}_j} | p_1 < j \leq p_2\}$ from the simulator, where p_2 is a polynomial number.

Definition 1 (Security of the modified ASPE scheme). *The modified ASPE scheme is selectively secure with the leakage $\mathcal{L}$* iff *for all PPT adversaries issuing polynomial numbers of query token generations, there exists a PPT simulator such that the probability that a PPT distinguisher $\mathcal{D}$ can distinguish the real and ideal experiments is negligible, i.e.,* $|\Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Real}}) = 1] - \Pr[\mathcal{D}(\mathrm{View}_{\mathcal{A},\mathrm{Ideal},\mathcal{L}}) = 1]|$ *is negligible.*

In our real experiment and ideal experiment, the view of a PPT distinguisher $\mathcal{D}$ is $\{\mathrm{CT}_{\mathbf{x}_i}, \{\mathrm{TK}_{\mathbf{q}_j}\}_{1 \leq j \leq p_2}\}$. In the following, we show that $\mathcal{D}$ cannot distinguish the real experiment and ideal experiment. First, in the real experiment, $\mathrm{CT}_{\mathbf{x}_i}$ and $\mathrm{TK}_{\mathbf{q}_j}$ are generated according to the modified ASPE scheme. In the modified ASPE scheme, the encryption algorithm contains random numbers $\{r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}\}$. The query token generation algorithm contains random numbers $\{r_{q,1}, r_{q,2}, r_{q,3}, r_{q,4}\}$. These random numbers make the ciphertexts and query tokens look like random vectors. Second, in the ideal experiment, the ciphertext and query tokens are randomly chosen based on the leakage, so they are also random vectors. Thus, the view of a distinguisher $\mathcal{D}$ is random ciphertexts and query tokens in both real and ideal experiments. In this case, the probability that $\mathcal{D}$ can distinguish them is negligible. Thus, the modified ASPE scheme is selectively secure, which also means that the modified ASPE scheme is semantically secure.

5.2 Security of Our EPSim Scheme

In this subsection, we analyze the security of our EPSim scheme. In specific, we will show that our scheme is privacy-preserving under our security model, i.e., (i) the dataset stored in the cloud is privacy-preserving; (ii) query requests are privacy-preserving; (iii) query results are privacy-preserving.

• *The dataset stored in the cloud is privacy-preserving.* In our security model, the cloud server is considered to be *honest-but-curious*. It may attempt to deduce the plaintext of the dataset when storing dataset. Before outsourced to the cloud, the dataset has been encrypted to be a Quadsector tree. In the encrypted tree T_{cipher}, each internal node has ciphertexts $\{\mathrm{CT}_{\mathbf{k}_{j,j_1}}|j, j_1 = 1, 2, 3, 4; j_1 \neq j\}$. Each ciphertext is encrypted by the modified ASPE scheme and the secret key is $\mathbf{M}$. Since the modified ASPE scheme is selectively secure and the cloud server only has part of the secret key $\mathbf{M}$, it is difficult for the cloud server to learn about the plaintext of these ciphertexts without the complete $\mathbf{M}$. For the leaf nodes, each of them is also encrypted by the modified ASPE scheme. Then, the security of the modified ASPE scheme can guarantee that the cloud server has no idea on the plaintext of these data records. Thus, the dataset in the cloud is privacy-preserving.

• *The query requests are privacy-preserving.* The query requests involve the private information of the query doctors, so they should be kept secret from the cloud server and other query doctors. In our scheme, when a query doctor U_p launches a query request $(\mathbf{q}, \tau)$, he/she uses the modified ASPE scheme to generate a query token $\mathrm{TK}_{\mathbf{q}}$. In order to obtain the query request $(\mathbf{q}, \tau)$, the cloud server and other query doctors may attempt to deduce $(\mathbf{q}, \tau)$ from the query token $\mathrm{TK}_{\mathbf{q}}$. However, the query token is generated by U_p's authorized key $\mathbf{M}_{p1}$. For the cloud server, it only has the random authorized key $\mathbf{M}_{p2}$, but has no idea on $\mathbf{M}_{p1}$. For other query doctors, their authorized keys are randomly chosen and are different from $\mathbf{M}_{p1}$. In this case, both the cloud server and other query doctors have no idea on the authorized key $\mathbf{M}_{p1}$. Then, the security of the modified ASPE scheme can guarantee that it is difficult for them to obtain the plaintext of the query request from $\mathrm{TK}_{\mathbf{q}}$. Therefore, the query requests are privacy-preserving.

• *The query results are privacy-preserving.* The query result $\mathcal{R}$ contains the identities of data records satisfying the query condition. As described in our security model, the query doctors may be curious about the query results of other query doctors. However, in our scheme, for U_p's query result $\mathcal{R}$, the cloud server first encrypts it with the access key ak, i.e., $AES_{ak}(\mathcal{R})$. Since ak is chosen by U_p and shared with the cloud server by a public key encryption algorithm, other query doctors have no idea on it. Then, the security of AES technique can guarantee that other query doctors cannot obtain the query result $\mathcal{R}$ without ak.

For the cloud server, it may be curious about the plaintext of the data records corresponding to these identities. Since the data records corresponding to the query results are encrypted with the modified ASPE scheme, the security of the modified ASPE scheme can guarantee that the cloud sever has no idea on the plaintext of these data records. Therefore, the query results are privacy-preserving.

6 Performance Evaluation

In this section, we first evaluate the performance of our EPSim scheme from both theoretical and experimental aspects. Since we aim at improving the similar range query efficiency, we mainly focus on evaluating the computational cost of similarity range queries. Meanwhile, since the ASPE scheme [12] is considered as the most efficient similarity range query scheme with accurate query results, to validate the efficiency of our EPSim scheme, we compare it with the ASPE scheme. Then, we show that our EPSim scheme is practical.

6.1 Theoretical Analysis

Suppose that $\mathcal{X}$ is a dataset with n l-dimensional data records and is encrypted into an encrypted Quadsector tree, denoted by T_{cipher}. Since T_{cipher} is a Quadtree, its height is about $\log_4 n$. Meanwhile, the ciphertext of each T_{cipher}'s internal node contains 12 $(l+9)$-dimensional vectors, and the ciphertext of each T_{cipher}'s leaf node is an $(l+9)$-dimensional vector. When performing similarity range queries over T_{cipher}, the computational cost depends on two factors, i.e., (i) the number of internal nodes and leaf nodes that need to be searched; and (ii) the computational cost of searching internal nodes and leaf nodes. Given a similarity query $(\mathbf{q}, \tau)$, on the one hand, the number of internal nodes and leaf nodes that need to be searched is closely related to the size of the query result. Without loss of generality, suppose that $\alpha_{\mathbf{q},\tau}$ denotes the size of the query $(\mathbf{q}, \tau)$'s query result. Then, the number of internal nodes and leaf nodes that need to be searched is about $\alpha_{\mathbf{q},\tau} * \log_4 n$. On the other hand, since the ciphertext of each T_{cipher}'s internal node contains 12 $(l+9)$-dimensional vectors and the ciphertext of each T_{cipher}'s leaf node is an $(l+9)$-dimensional vector, the computational cost of searching internal nodes and leaf nodes is $12(l+9)$ and $(l+9)$, respectively. Thus, the overall computational cost of a similarity query is approximately $O(\alpha_{\mathbf{q},\tau} * (l+9) * \log_4 n)$. In this case, the computational cost of similarity queries is affected by three parameters, i.e., τ, l, and n.

• The size of dataset n: When the size of the dataset (i.e., n) increases, the height of Quadsector tree (i.e., $\log_4 n$) will become larger. Hence, the computational cost of similarity range queries will increase.

• The query range τ: When τ increases, the size of the query result (i.e., $\alpha_{\mathbf{q},\tau}$) will increase. As a result, the computational cost of similarity range queries will correspondingly increase. Meanwhile, we have the following observation.

Observation 1. *Let $\mathcal{X}$ be dataset and the data records in each dimension are in the domain of $[L, R]$. Then, in most cases, when the query range τ is larger than half of the domain, i.e., $\tau > \frac{R-L}{2}$, almost no subtrees can be pruned during the process of similarity range queries. Then, the similarity range queries will become less efficient.*

We take two-dimensional dataset as an example to show that Observation 1 is correct. From Hilbert exclusion condition, we can infer that one subtree can be pruned *iff* $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) > \tau$. Specifically, when $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) > \tau$ and $d(\mathbf{k}_1, \mathbf{q}) > d(\mathbf{k}_2, \mathbf{q})$, the query result will be closer to $\mathbf{k}_2$ than $\mathbf{k}_1$ and the data records in $\mathbf{k}_1$'s subtree can be pruned. When $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) > \tau$ and $d(\mathbf{k}_1, \mathbf{q}) < d(\mathbf{k}_2, \mathbf{q})$. The data records in $\mathbf{k}_2$'s subtree can be pruned. That is, whether a subtree can be pruned depends on the relationship between $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2})$ and τ.

Meanwhile, when $\tau > \frac{R-L}{2}$, $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2})$ and τ almost cannot satisfy Hilbert exclusion condition and almost no subtrees can be pruned. As shown in Fig. 5b, when $V_{\mathbf{k}_1,\mathbf{k}_2}$ is in

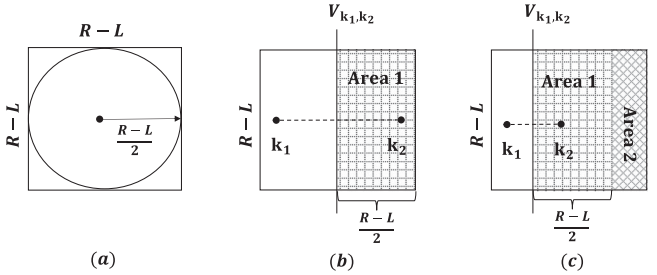


Fig. 5. Example to show the correctness of Observation 1.

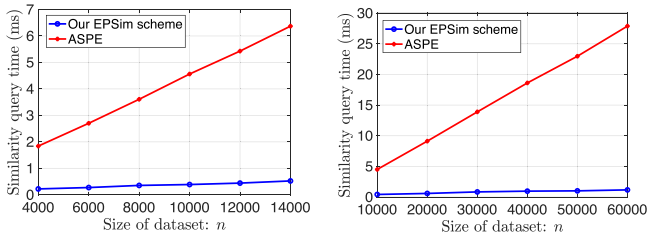


Fig. 6. Computational cost versus the size of dataset n.

the middle of the whole area, $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2})$ is equal to or less than τ for any query record $\mathbf{q}$. Thus, Hilbert exclusion condition cannot be satisfied. As shown in Fig. 5c, when $V_{\mathbf{k}_1,\mathbf{k}_2}$ is on the left side of the whole area, $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2})$ is equal to or less than τ for any $\mathbf{q}$ in Area 1. In contrast, for any $\mathbf{q}$ in Area 2, $d(\mathbf{q}, V_{\mathbf{k}_1,\mathbf{k}_2}) > \frac{R-L}{2}$ holds. Thus, when $\mathbf{q}$ falls into Area 2, the subtree related to $\mathbf{k}_1$ can be pruned. However, the number of data records falling into Area 2 is small. This is because $\mathbf{k}_1$ and $\mathbf{k}_2$ should be symmetrical about the line $V_{\mathbf{k}_1,\mathbf{k}_2}$. When $V_{\mathbf{k}_1,\mathbf{k}_2}$ is on the leaf side of the whole area, $\mathbf{k}_2$ should be near to $V_{\mathbf{k}_1,\mathbf{k}_2}$. Meanwhile, since $\mathbf{k}_2$ is the centroid of data records in $\mathbf{k}_2$'s subtree. Then, most data records in $\mathbf{k}_2$'s subtree should be near to $\mathbf{k}_2$. Correspondingly, most query records should fall into Area 1, so they cannot satisfy the Hilbert exclusion condition. Hence, almost no subtrees can be pruned even if $V_{\mathbf{k}_1,\mathbf{k}_2}$ is on the left side of the whole area. Likewise, when $V_{\mathbf{k}_1,\mathbf{k}_2}$ is on the right of the whole area, the subtrees also cannot be pruned. Thus, Observation 1 is correct.

• The dimension of data records l: The computational cost of similarity range queries increases with the increase of l. When l increases, the computational cost of computing the scalar product between encrypted data records and query tokens will increase.

6.2 Experimental Analysis

In this subsection, we experimentally evaluate the performance of our EPSim scheme and compare it the ASPE scheme. We implemented our scheme and the ASPE scheme in Java and conducted experiments on a machine with Apple M1 CPU, 16 GB RAM, and macOS Big Sur operating system. Since both our scheme and the ASPE scheme can use public key encryption algorithm $E(\cdot)$ to protect the privacy of the query results, the computational cost from $E(\cdot)$ is not considered in our experiments. The evaluated datasets include a real dataset and a synthetic dataset. The real dataset is an EEG dataset [25]. It consists of 14,980 data records and 14 attributes. The synthetic dataset is randomly generated and all values fall within the range of $[0, 20]$. Since the computational cost of similarity range query is affected by parameters n, τ, and l, we evaluate the impact of these parameters, respectively. In our evaluation, each experiment is conducted 1000 times, and the average result is reported.

• *The size of dataset n:* In Fig. 6a, we plot the computational cost of our scheme and the ASPE scheme varying with n on EEG dataset. In this experiment, we set $l = 5$, $\tau = 6$, and n ranges from 4,000 to 14,000. From this figure, we can see that the computational cost of our scheme and the ASPE scheme increases with n but our scheme is more efficient than the ASPE scheme. For example, when $n = 14000$, the average computational cost of our scheme is about 0.524 ms, while that of the ASPE scheme is about 6.37 ms.

In Fig. 6b, we plot the computational cost of our scheme and the ASPE scheme varying with n on a synthetic dataset. In this experiment, we set $l = 5$, $\tau = 3$, and n ranges from 10,000 to 60,000. From this figure, we can see that the computational cost of our scheme and the ASPE scheme increases with n but the increasing rate of our scheme is lower than that of the ASPE scheme. In addition, our scheme is much more efficient than the ASPE scheme. For example, when $n = 60000$, the average computational cost of our scheme is about 1.165 ms, while that of the ASPE scheme is about 27.88 ms.

• *The query range τ:* In Fig. 7a, we plot the computational cost of our scheme and the ASPE scheme varying with τ on EEG dataset. In this experiment, we set $n = 14000$, $l = 5$, and τ ranges from 3 to 10. From this figure, we can see that the computational cost of the ASPE scheme is almost unchanged. This is because the similarity range query in the ASPE scheme is achieved by traversing all data records and the query efficiency is not related to the query range τ. In our scheme, the computational cost increases with the query range τ. This is because when τ increases, more data records can satisfy the query condition. That is, the size of the query result will increase. In this case, the computational cost of similarity queries will correspondingly increase. In addition, we can see that our scheme is much more efficient than the ASPE scheme.

In Fig. 7b, we plot the computational cost of our scheme and the ASPE scheme varying with τ on a synthetic dataset. In this experiment, we set $n = 60000$, $l = 5$, and τ ranges from 2 to 4. This experiment also demonstrates that the computational cost of our scheme increases with τ and is more efficient than the ASPE scheme.

• *The dimension of data records l:* In Fig. 8a, we plot the computational cost of our scheme and the ASPE scheme

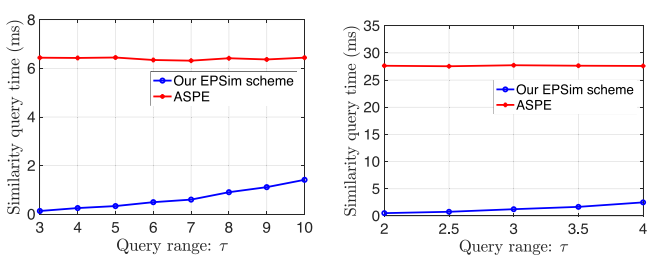
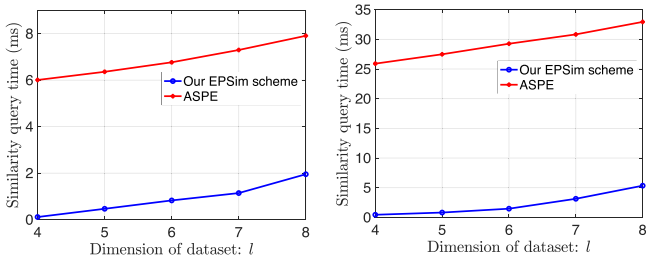


Fig. 7. Computational cost versus the query range τ.



(a) EEG dataset (b) Synthetic dataset

Fig. 8. Computational cost versus the data dimension l.

varying with l on EEG dataset. In this experiment, we set $n = 14000$, and l is in the range of $\{4, 5, 6, 7, 8\}$. In order to make query results contain 4 to 10 data records, the query range τ is correspondingly set to be $\{3, 6, 8, 10, 12\}$. From this figure, we can see that the computational cost of the ASPE scheme and our scheme linearly increases as l becomes larger. Meanwhile, we can see that our scheme is more efficient than the ASPE scheme.

In Fig. 8b, we plot the computational cost of our scheme and the ASPE scheme varying with l on a synthetic dataset. In this experiment, we set $n = 60000$ and l is in the range of $\{4, 5, 6, 7, 8\}$. To make the query result contain 4 to 6 data records, the query range τ is correspondingly set to be $\{1.5, 2.5, 3.5, 4.5, 5.5\}$. From Fig. 8b, we can see that this experiment also shows the computational cost of our scheme increases as l becomes larger. Meanwhile, our scheme is much more efficient than the ASPE scheme in query processing.

6.3 Practicality

In this subsection, we show that our scheme is practical as defined in Section 2.3. On the one hand, when a query doctor U_p registers in the system, the HC authorizes U_p through a random secret key $\mathbf{M}_{p1}$. Thus, the HC does not have to share the secret key $\mathbf{M}$ with query doctors. On the other hand, the HC does not have to stay online. Meanwhile, our scheme is built under a single cloud server. Therefore, our scheme is practical.

7 Related Work

The similarity query is to search the data records who are similar to a specific query record, where the similarity between two data records are measured by various distance metrics such as euclidean distance. Meanwhile, with the popularity of cloud service, the similarity query over outsourced encrypted data has attracted considerable attention from academia and industria [26], and many privacy-preserving similarity query schemes were proposed.

Oliveira *et al.* [7] introduced a distance-preserving transformation technique to encrypt the data. In the proposed scheme, the distance over encrypted data records is the same as that over plaintext data records. With the distance-preserving property, it is easy to perform the similarity range query over encrypted data. However, Wong *et al.* [12] showed that the distance-preserving transformation could not resist level-2 attack (defined in [12]). In the level-2 attack, when an attacker knows the encrypted dataset and several plaintext data records, the whole dataset can be recovered.

Some similarity query schemes [6], [8], [9], [10], [12] are inefficient. For the schemes [6], [8], [9], [10], they employed public key encryption techniques, e.g., Paillier encryption technique, to achieve similarity queries. However, the time-consuming public key encryption techniques make these schemes inefficient. For the scheme [12], it is an asymmetric scalar-product preserving encryption (ASPE) scheme [12] and was proposed to achieve similarity queries based on matrix encryption. Although, compared with the public key encryption based schemes [6], [8], [9], [10], the ASPE scheme is more efficient and is considered as the most efficient similarity query scheme with accurate query results. However, the ASPE scheme performs similarity queries by traversing each data record, so the computational complexity of query processing is linear to the size of the dataset. Thus, it is also inefficient.

Some efficient similarity query schemes [20], [21], [27], [28], [29] were proposed, but they are impractical in some real scenarios. Zhu *et al.* [27], [28] designed two similarity range query schemes. In these schemes, interactive protocols between the data owner and query users are designed such that the query users can obtain the query result without the secret key. In [29], Zhu *et al.* adopted Paillier homomorphic encryption technique in the process of query encryption such that the query users can launch queries without the secret key. The schemes [27], [28], [29] can avoid the secret key sharing with query users but they require that data owners (i.e., healthcare centers in our scheme) stay online.

To further improve the similarity query efficiency, two schemes [20], [21] were proposed by combining garbled circuits technique [30], SSE, and LSH, but these two schemes have another limitation, i.e., they were built under the model of two cloud servers. However, in order to reduce the cloud bills, the data owners may prefer to deploy a single cloud server in some real scenarios. Thus, the schemes [20], [21] are also impractical. In addition, to balance data security, query efficiency and query result accuracy, some approximate privacy-preserving similarity query schemes were proposed [13], [14], [15], [16], [17], [18], [19], but they are not applicable to our scenario.

8 Conclusion

In this paper, we have proposed an efficient and privacy-preserving similarity range query scheme. Specifically, we first presented a Quadsector tree to represent dataset, and employed an efficient filtration condition, i.e., Hilbert exclusion condition, to design an efficient similarity query algorithm. At the same time, based on the ASPE scheme, we constructed a modified ASPE scheme and proved that it is selectively secure. Finally, we proposed our scheme by employing the modified ASPE scheme to encrypt and search the Quadsector tree. In addition, security analysis and performance evaluation show that our scheme can well preserve the privacy of dataset, query requests and query results, and it is indeed efficient and practical in similarity range queries. In our future work, we will explore other

data structures to represent the dataset and construct more efficient similarity range query scheme.

Acknowledgments

This work was supported in part by the NSERC Discovery under Grant 04009, in part by the Natural Science Foundation of Shaanxi Province under Grant 2019ZDLGY12-02, in part by the ZJNSF under Grant LZ18F020003, and in part by the NSFC under Grants U1709217 and 61972304.

References

[1] Z. Ahmed, K. Mohamed, S. Zeeshan, and X. Dong, "Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine," *Database*, vol. 2020, 2020, Art. no. baaa010.

[2] R. Lu, "A new communication-efficient privacy-preserving range query scheme in fog-enhanced IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2497–2505, Apr. 2019.

[3] BIS Research. Accessed: May 14, 2021. [Online]. Available: https://www.prnewswire.com/news-releases/global-big-data-in-healthcare-market-to-reach-6875-billion-by-2025-reports-bis-research-678151823.html

[4] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, and K. R. Choo, "Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data," *Inf. Sci.*, vol. 498, pp. 91–105, 2019.

[5] Y. Zheng, R. Lu, X. Yang, and J. Shao, "Achieving efficient and privacy-preserving top-k query over vertically distributed data sources," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[6] W. Wu, J. Liu, H. Rong, H. Wang, and M. Xian, "Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database," *IEEE Access*, vol. 6, pp. 41771–41784, 2018.

[7] S. R. M. Oliveira and O. R. Zaïane, "Privacy preserving clustering by data transformation," in *Proc. XVIII Simpósio Brasileiro de Bancos de Dados*, 2003, pp. 304–318.

[8] S. Rane and P. T. Boufounos, "Privacy-preserving nearest neighbor methods: Comparing signals without revealing them," *IEEE Signal Process. Mag.*, vol. 30, no. 2, pp. 18–28, Mar. 2013.

[9] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 664–675.

[10] M. Kesarwani *et al.*, "Efficient secure k-nearest neighbours over encrypted data," in *Proc. 21st Int. Conf. Extending Database Technol.*, 2018, pp. 564–575.

[11] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-NN query for outsourced ehealthcare data," *J. Med. Syst.*, vol. 43, no. 5, 2019, Art. no. 123.

[12] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[13] M. L. Yiu, I. Assent, C. S. Jensen, and P. Kalnis, "Outsourced similarity search on metric data assets," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 338–352, Feb. 2012.

[14] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 1156–1167.

[15] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. 29th IEEE Int. Conf. Data Eng.*, 2013, pp. 733–744.

[16] H. Xu, S. Guo, and K. Chen, "Building confidential and efficient query services in the cloud with RASP data perturbation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 322–335, Feb. 2014.

[17] X. Yuan, H. Cui, X. Wang, and C. Wang, "Enabling privacy-assured similarity retrieval over millions of encrypted records," in *Proc. 20th Eur. Symp. Res. Comput. Secur.*, 2015, pp. 40–60.

[18] X. Yuan, X. Wang, C. Wang, C. Yu, and S. Nutanong, "Privacy-preserving similarity joins over encrypted data," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2763–2775, Nov. 2017.

[19] X. Lei, A. X. Liu, R. Li, and G. Tu, "SecEQP: A secure and efficient scheme for SkNN query problem over encrypted geodata on cloud," in *Proc. 35th IEEE Int. Conf. Data Eng.*, 2019, pp. 662–673.

[20] Y. Zheng, H. Cui, C. Wang, and J. Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1285–1298, Jun. 2017.

[21] H. Cui, X. Yuan, Y. Zheng, and W. Cong, "Towards encrypted in-network storage services with secure near-duplicate detection," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2018.2850333.

[22] R. C. H. Connor, F. A. Cardillo, L. Vadicamo, and F. Rabitti, "Hilbert exclusion: Improved metric search through finite isometric embeddings," *ACM Trans. Inf. Syst.*, vol. 35, no. 3, 2017, Art. no. 17.

[23] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, "Approximate kernel k-means: Solution to large scale kernel clustering," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 895–903.

[24] S. Lai *et al.*, "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 745–762.

[25] EEG eye state data set. Accessed: May 14, 2021. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State

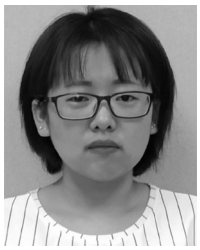
[26] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2021.3061611.

[27] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN computation on encrypted cloud data without sharing key with query users," in *Proc. Int. Workshop Secur. Cloud Comput.*, 2013, pp. 55–60.

[28] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN query on encrypted cloud database without key-sharing," *Int. J. Electron. Secur. Digit. Forensics*, vol. 5, no. 3/4, pp. 201–217, 2013.

[29] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-NN query over encrypted cloud data with key confidentiality," *J. Parallel Distrib. Comput.*, vol. 89, pp. 1–12, 2016.

[30] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *Proc. 27th Annu. Symp. Found. Comput. Sci.*, 1986, pp. 162–167.



Yandong Zheng received the MS degree from the Department of Computer Science, Beihang University, China, in 2017. She is currently working toward the PhD degree with the Faculty of Computer Science, University of New Brunswick, Canada. Her research interests include cloud computing security, big data privacy, and applied privacy.

Rongxing Lu (Fellow, IEEE) received the first PhD degree from Shanghai Jiao Tong University, China, in 2006 and the second PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He is currently an associate professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. From April 2013 to August 2016, he was an assistant professor with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore. From May 2012 to April 2013, he was a postdoctoral fellow with the University of Waterloo. He has authored or coauthored extensively in his areas of expertise, which include applied cryptography, privacy enhancing technologies, and IoT-big data security and privacy. He was the recipient of the most prestigious Governor General's Gold Medal in 2012 and the 8th IEEE Communications Society Asia Pacific Outstanding Young Researcher Award in 2013. He has an H-index of 76 from Google Scholar as of May 2021. He was also the recipient of nine best student paper awards from some reputable journals and conferences and the 2016–2017 Excellence in Teaching Award from FCS, UNB. He is currently the vice-chair of various conferences of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee).

Yunguo Guan is currently working toward the PhD degree with the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Jun Shao received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008. From 2008 to 2010, he was a postdoctoral fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA. He is currently a professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Hui Zhu (Senior Member, IEEE) received the BSc degree from Xidian University, Xi'an, China, in 2003, the MSc degree from Wuhan University, Wuhan, China, in 2005, and the PhD degree from Xidian University in 2009. He was a research fellow with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013. Since 2016, he has been a professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.