Efficient and Privacy-Preserving Similarity Query With Access Control in eHealthcare

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

***Abstract*— Similarity queries, giving a way to disease diagnosis based on similar patients, have wide applications in eHealthcare and are essentially demanded to be processed under fine-grained access policies due to the high sensitivity of healthcare data. One efficient and flexible way to implement such queries is to outsource healthcare data and the corresponding query services to a powerful cloud. Nevertheless, considering data privacy, healthcare data are usually outsourced in an encrypted form and required to be accessed in a privacy-preserving way. In the past years, many schemes have been proposed for privacy-preserving similarity queries. However, none of them is applicable to achieve data access control and access pattern privacy preservation. Aiming at this challenge, we propose an efficient and access pattern privacy-preserving similarity range query scheme with access control (named EPSim-AC). In our proposed scheme, we first design a novel tree structure, called k-d-PB tree, to index healthcare data and introduce an efficient k-d-PB tree based similarity query algorithm with access control. Second, to balance the search efficiency and access pattern privacy of k-d-PB tree, we also define a weakened access pattern privacy, called k-d-PB tree's β-access pattern unlinkability. After that, we preserve the privacy of k-d-PB tree based similarity queries with access control through a symmetric homomorphic encryption scheme and present our detailed EPSim-AC scheme. Finally, we analyze the security of our scheme and also conduct extensive experiments to evaluate its performance. The results demonstrate that our scheme can guarantee k-d-PB tree's β-access pattern unlinkability and has high efficiency.**

***Index Terms*— Similarity query, access control, access pattern privacy, eHealthcare, k-d-PB tree.**

I. Introduction

THE rapid growth of aging population, the boom of wireless body area networks, and the digitization of healthcare systems have jointly facilitated the generation of a massive amount of healthcare data in healthcare centers [1], and the generated data have been further utilized to provide high-quality healthcare query services to doctors. Among these services, similarity queries, which search on a healthcare dataset for historical patients similar to the current patient, are highly regarded [2], [3]. Due to privacy concerns, healthcare centers usually authorize each doctor to access a small proportion of data. For example, an ophthalmologist is only authorized to access ophthalmic patients' data. As a result, similarity queries should be implemented under the restraint of access control. Specifically, suppose that $\mathbf{x}_i$ is a healthcare record with an access policy $\mathbf{a}_i$, and $(\mathbf{q}, \tau)$ is a similarity query request with an attribute vector $\mathbf{v}_j$, where $\mathbf{q}$ is a query record and τ is a query range. Then, $\mathbf{x}_i$ satisfies the query request $(\mathbf{q}, \tau)$ *iff* the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{q}$ is within τ, i.e., the distance $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$, and $\mathbf{v}_j$ satisfies the access policy $\mathbf{a}_i$, i.e., $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$, where the access control is realized by the Hidden Vector based access structure and its match relationship is denoted by "$\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$" (defined in Subsection III-A).

To efficiently implement similarity queries with access control, healthcare centers are inclined to outsource their data to a computationally powerful cloud and depute the cloud server to provide the corresponding query services to doctors. Considering data privacy, datasets are usually outsourced in an encrypted form and are required to be accessed in a privacy-preserving way because leaking datasets' access pattern and even datasets' update information may result in the leakage of entire datasets [4]–[6]. In this work, we focus on the similarity query over a static dataset, and as a result, it is free of the datasets' update based attacks in [6]. However, it may suffer from access pattern based attacks in [4], [5]. Although various privacy-preserving similarity query schemes [2], [7]–[19] have been proposed, none of them can simultaneously preserve the access pattern privacy and achieve data access control. Specifically, schemes in [2], [7]–[15] are not access pattern privacy-preserving, and all of them [2], [7]–[19] cannot achieve data access control. The reason for lacking work on privacy-preserving similarity queries with access control and access pattern privacy is that there are two challenges lying in designing such a scheme as follows: *Challenge I: How to efficiently achieve similarity queries with access control?* Existing researches on similarity queries and access control are independently proposed, i.e., many schemes [2], [7]–[19] were proposed for privacy-preserving similarity queries, and many were proposed for data access control [20], [21]. However, there is no scheme specifically designed for efficiently processing similarity queries and

Manuscript received July 1, 2021; revised December 24, 2021; accepted January 31, 2022. Date of publication February 16, 2022; date of current version March 9, 2022. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Discovery Grant 04009, in part by the Zhejiang Natural Science Foundation (ZJNSF) under Grant LZ18F020003, in part by the NSFC under Grant 61972304, and in part by the NSF of Shaanxi Province under Grant 2019ZDLGY12-02. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: Rongxing Lu.)*

Yandong Zheng, Rongxing Lu, Yunguo Guan, and Songnian Zhang are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: yzheng8@unb.ca; rlu1@unb.ca; yguan4@unb.ca; songnian.zhang@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).

Digital Object Identifier 10.1109/TIFS.2022.3152395

access control. A straightforward way to achieve similarity queries with access control is to integrate a similarity query scheme with an access control scheme. That is, when a similarity query request is coming, we first use a similarity query scheme to search on the dataset for a set of records satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ and then verify whether each such $\mathbf{x}_i$ satisfies $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) \stackrel{?}{=} 1$. Nevertheless, such an integration is inefficient because the access control is only used for verifying the returned records from similarity query scheme and remained to be fully exploited for query efficiency improvement. *Challenge II: How to balance the access pattern privacy and efficiency of similarity queries?* Private Information Retrieval (PIR) [22] and Oblivious Random Access Machine (ORAM) [23] are two typical techniques designed for access pattern privacy preservation but they are respectively inefficient in the computational cost and communication overhead, which will inevitably degrade the efficiency of similarity queries.

Thus, it is still challenging to achieve privacy-preserving similarity queries with access control and access pattern privacy. To address these challenges, we propose an efficient and privacy-preserving similarity range query scheme with access control (EPSim-AC) over an encrypted healthcare dataset, which can (i) efficiently achieve similarity queries with access control; and (ii) well balance the access pattern privacy and query efficiency of similarity queries. In our scheme, we develop two strategies to tackle with *Challenge I* and *Challenge II* as follows.

- ***Strategy I:*** We design a new tree structure, called k-d-PB tree, to index the healthcare center's dataset. Since k-d-PB tree involves not only healthcare data records but also their attributes, it can jointly exploit similarity query criteria and access control requirements for query efficiency improvement. Then, we design an efficient k-d-PB tree based similarity range query algorithm with access control.
- ***Strategy II:*** To well balance the access pattern privacy and efficiency of similarity queries, we define a kind of weakened access pattern privacy, called k-d-PB tree's β-access pattern unlinkability for $\beta > 1$, in which a k-d-PB tree path is indistinguishably accessed with $(\beta - 1)$ tree paths. The reason why we consider β-access pattern unlinkability is that hiding the access pattern of each tree path among all tree paths is unnecessary and outrageously expensive [24] while β-access pattern unlinkability can be achieved at a relatively low cost. Although k-d-PB tree's β-access pattern unlinkability is weaker than its full access pattern, it is secure enough when the k-d-PB tree is large. The introduction of k-d-PB tree's β-access pattern unlinkability can facilitate the balance of access pattern privacy and similarity query efficiency.

Specifically, the main contributions of this work are threefold as follows.

- First, we design a k-d-PB tree and an efficient k-d-PB tree based similarity range query algorithm to process similarity queries with access control.
- Second, we define the k-d-PB tree's β-access pattern unlinkability to facilitate the balance of similarity query efficiency and access pattern privacy.
- Third, we employ a symmetric homomorphic encryption (SHE) scheme to preserve the privacy of k-d-PB tree based similarity query algorithm and further propose our EPSim-AC scheme. In addition, we analyze the security of our scheme and conduct experiments to evaluate its performance. The results demonstrate that our scheme can guarantee k-d-PB tree's β-access pattern unlinkability and has high efficiency.

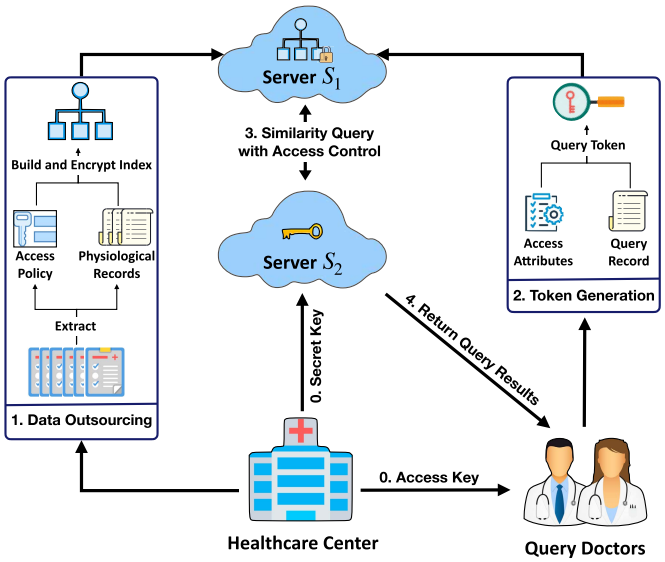


Fig. 1. System model under consideration.

The remainder of this paper is organized as follows. In Section II, we introduce our system model, security model, and design goal. Then, we describe some preliminaries in Section III. In Section IV, we present our scheme, followed by security analysis and performance evaluation in Section V and Section VI, respectively. In Section VII, we present some related work. Finally, we draw our conclusion in Section VIII.

II. Models and Design Goal

In this section, we introduce our system model, security model, and identify our design goal.

A. System Model

In the system model, we consider an access pattern privacy-preserving similarity query scheme with access control in the cloud, which contains three types of entities, i.e., a healthcare center, two cloud servers $\{S_1, S_2\}$, and multiple query doctors $\{U_1, U_2, \cdots\}$, as shown in Fig. 1.

- Healthcare Center (HC): HC has a healthcare dataset with a huge volume of healthcare records collected from historical patients. Each record is a d-dimensional vector $\mathbf{x}_i$ and is associated with an l-dimensional access policy vector $\mathbf{a}_i$. Doctors can have access to $\mathbf{x}_i$ *iff* their access attributes satisfy the access policy $\mathbf{a}_i$. Without loss of generality, let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{a}_i) | i = 1, 2, \cdots, n\}$ denote the healthcare dataset. To take full advantage of $\mathcal{X}$'s value, HC uses $\mathcal{X}$ to offer an efficient similarity query service with access control to doctors. Considering the constraint of computational capability and storage space, HC outsources $\mathcal{X}$ to a cloud and delegates the corresponding cloud servers to offer similarity query services to doctors. To guarantee the data only be shared with qualified

doctors and improve query efficiency, HC builds an index tree T for $\mathcal{X}$. Since the healthcare data contain some sensitive information, HC encrypts T into an encrypted tree $E(\mathrm{T})$ and outsources it to the cloud.

• Query Doctors $\{U_1, U_2, \cdots\}$: In our system model, there exist many query doctors, denoted by $\{U_1, U_2, \cdots\}$. Each U_j is associated with an attribute vector $\mathbf{v}_j$, which determines U_j's access right to the outsourced healthcare dataset $E(\mathrm{T})$. That is, U_j only has access to $E(\mathrm{T})$'s data records whose access policies can be satisfied by $\mathbf{v}_j$. Specifically, when U_j launches a similarity query $(\mathbf{q}, \tau)$ to the cloud servers, the returned query result will be all records satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ and $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$, where $\mathbf{q}$ is a d-dimensional query record and τ is a query range.

• Two Cloud Servers $\{S_1, S_2\}$: In our system model, there are two cloud servers, denoted by S_1 and S_2. Both of them have powerful computing resources and abundant storage spaces. S_1 and S_2 respectively store the encrypted index tree $E(\mathrm{T})$ of the healthcare data and the secret key that is used to encrypt the index tree. On receiving similarity query requests from doctors, S_1 and S_2 can cooperate together on searching $E(\mathrm{T})$ for the query results and returning them to query doctors.

B. Security Model

In our security model, HC is assumed to be *trusted* because it is the initiator of the entire system. Query doctors are assumed to be *honest*, namely, they will honestly follow our scheme to launch similarity query requests to cloud servers. The honest assumption is reasonable because doctors in our scheme have been authorized by the HC to enjoy the query services. Their any misbehavior may result in serious penalties and even the revocation of authorization from the HC. For cloud servers, both S_1 and S_2 are considered to be *honest-but-curious*. That is, they will honestly follow the protocol to store the outsourced healthcare data for HC and offer the similarity query service with access control to doctors. However, they may be curious about some private information, including the access pattern of healthcare dataset and plaintexts of healthcare data records, query requests and query results. In addition, we assume that there is no collusion between S_1 and S_2. The non-collusive assumption is reasonable because different cloud servers may have a conflict of interest. Note that there may be other active attacks in the system. Since this work focuses more on privacy preservation, those attacks are beyond the scope of this paper and will be discussed in our future work.

C. Design Goal

In this work, our goal is to design an efficient and privacy-preserving similarity query scheme with access control over cloud, and the following objectives should be achieved.

• *Privacy preservation:* The privacy of the healthcare dataset, similarity query requests, and query results should be preserved. Meanwhile, we aim to hide k-d-PB tree's β-access pattern privacy from cloud servers.

• *Efficiency:* Since the privacy preservation objective inevitably incurs the additional computational cost, we also aim to improve the query efficiency as much as possible.

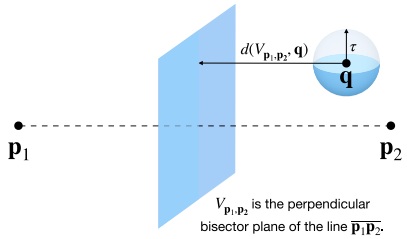


Fig. 2. Explanation of Hilbert exclusion condition.

III. Preliminaries

In this section, we will review some preliminaries, including hidden vector based access structure, Hilbert exclusion condition, and a symmetric encryption (SHE) technique.

A. Hidden Vector-Based Access Structure

Hidden vector based access structure is a kind of attribute based access control and is commonly used to implement fine-grained access control for healthcare data [25]. Let Σ denote all attributes in the system and Σ^l denote the domain of l-dimensional attribute vectors. Let "$*$" denote the *don't care* attribute and $\{\Sigma \cup *\}^l$ denote the domain of l-dimensional vectors over $\{\Sigma \cup *\}$. Without loss of generality, we assume that all attribute values in Σ are positive integers. An access policy is denoted as $\mathbf{a} = \{a_1, a_2, \cdots, a_l\} \in \{\Sigma \cup *\}^l$. An attribute vector is denoted as $\mathbf{v} = (v_1, v_2, \cdots, v_l) \in \Sigma^l$. Let $\mathsf{HV.Match}(\mathbf{a}, \mathbf{v})$ denote the match relationship between $\mathbf{a}$ and $\mathbf{v}$, where

$$\mathsf{HV.Match}(\mathbf{a}, \mathbf{v}) = \begin{cases} 1 & a_i = v_i \text{ or } a_i = "*" \text{ for } 1 \leq i \leq l; \\ 0 & \text{Otherwise.} \end{cases}$$

That is, $\mathbf{v}$ satisfies the access policy $\mathbf{a}$ *iff* v_i has the same value as a_i for all $a_i \neq "*"$.

B. Hilbert Exclusion Condition

Hilbert exclusion condition was proposed in [26] and can be employed to speed up the query efficiency of similarity queries as shown in Theorem 1.

Theorem 1 (Hilbert Exclusion Condition): Let $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{q}$ denote three d-dimensional data records satisfying $d(\mathbf{p}_1, \mathbf{q}) > d(\mathbf{p}_2, \mathbf{q})$, where $d(\cdot, \cdot)$ denotes the Euclidean distance. If they satisfy the inequality $\frac{d(\mathbf{p}_1,\mathbf{q})^2 - d(\mathbf{p}_2,\mathbf{q})^2}{2d(\mathbf{p}_1,\mathbf{p}_2)} > \tau$, any record $\mathbf{x}_i$ satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ is closer to $\mathbf{p}_2$ than $\mathbf{p}_1$, i.e., $d(\mathbf{x}_i, \mathbf{p}_1) > d(\mathbf{x}_i, \mathbf{p}_2)$.

In the following, we prove the correctness of Theorem 1.

Proof: As shown in Fig. 2, given two records $\mathbf{p}_1$ and $\mathbf{p}_2$, the perpendicular bisector plane of the line $\overline{\mathbf{p}_1\mathbf{p}_2}$ can be represented as $V_{\mathbf{p}_1,\mathbf{p}_2} : (\mathbf{x} - \frac{\mathbf{p}_1+\mathbf{p}_2}{2})^T(\mathbf{p}_1 - \mathbf{p}_2) = 0$. Then, the distance between $\mathbf{q}$ and $V_{\mathbf{p}_1,\mathbf{p}_2}$ will be

$$\begin{aligned} & d(V_{\mathbf{p}_1,\mathbf{p}_2}, \mathbf{q}) \\ &= \left| \frac{1}{d(\mathbf{p}_1, \mathbf{p}_2)} * (\mathbf{q} - \frac{\mathbf{p}_1 + \mathbf{p}_2}{2})^T (\mathbf{p}_1 - \mathbf{p}_2) \right| \\ &= \left| \frac{1}{d(\mathbf{p}_1, \mathbf{p}_2)} * \frac{2\mathbf{q}^T(\mathbf{p}_1 - \mathbf{p}_2) - (\|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2)}{2} \right| \\ &= \left| \frac{d(\mathbf{p}_2, \mathbf{q})^2 - d(\mathbf{p}_1, \mathbf{q})^2}{2d(\mathbf{p}_1, \mathbf{p}_2)} \right|. \end{aligned}$$

Since $d(\mathbf{p}_1, \mathbf{q}) > d(\mathbf{p}_2, \mathbf{q})$ as described in Theorem 1, we can further deduce that $d(V_{\mathbf{p}_1,\mathbf{p}_2}, \mathbf{q}) = \frac{d(\mathbf{p}_1,\mathbf{q})^2 - d(\mathbf{p}_2,\mathbf{q})^2}{2d(\mathbf{p}_1,\mathbf{p}_2)}$. In this case, when $\frac{d(\mathbf{p}_1,\mathbf{q})^2 - d(\mathbf{p}_2,\mathbf{q})^2}{2d(\mathbf{p}_1,\mathbf{p}_2)} > \tau$, we have $d(V_{\mathbf{p}_1,\mathbf{p}_2}, \mathbf{q}) > \tau$. According to Fig. 2, we can easily deduce that any record $\mathbf{x}_i$ satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ is closer to $\mathbf{p}_2$ than $\mathbf{p}_1$, i.e., $d(\mathbf{x}_i, \mathbf{p}_1) > d(\mathbf{x}_i, \mathbf{p}_2)$.

C. SHE Technique

SHE technique is a symmetric homomorphic encryption technique, which was proposed in [27] and has been proved to be semantically secure in [28]. It can support leveled homomorphic addition and multiplication. Specifically, the SHE technique $\Pi_{\mathsf{SHE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows.

- $\mathsf{KeyGen}(k_0, k_1, k_2)$: Given three security parameters $\{k_0, k_1, k_2\}$ with $k_1 \ll k_2 < k_0$, the key generation algorithm first selects two prime numbers p, q such that $|p| = |q| = k_0$, and let $\mathtt{N} = pq$. Then, it chooses a random number $\mathtt{L}$ such that $|\mathtt{L}| = k_2$. Finally, it outputs the public key $pk = (k_0, k_1, k_2, \mathtt{N})$, the secret key $sk = (p, \mathtt{L})$, and the basic message space $\mathcal{M} = \{m | m \in [-2^{k_1-1}, 2^{k_1-1})\}$.
- $\mathsf{Enc}(m, sk, pk)$: A message $m \in \mathcal{M}$ can be encrypted by pk and sk as $E(m) = (r\mathtt{L} + m)(1 + r'p) \mod \mathtt{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.
- $\mathsf{Dec}(sk, E(m))$: On input sk and $E(m)$, the decryption algorithm first computes $m' = (E(m) \mod p) \mod \mathtt{L} = (m + \mathtt{L}) \mod \mathtt{L}$. Then, if $m' < \frac{\mathtt{L}}{2}$, it means that $m \geq 0$, and $m = m'$. If $m' > \frac{\mathtt{L}}{2}$, it means that $m < 0$, and $m = m' - \mathtt{L}$.

The SHE technique satisfies the following homomorphic addition and multiplication properties: (i) *Homomorphic addition-I:* $E(m_1) + E(m_2) \rightarrow E(m_1 + m_2)$; (ii) *Homomorphic multiplication-I:* $E(m_1) * E(m_2) \rightarrow E(m_1 * m_2)$; (iii) *Homomorphic addition-II:* $E(m_1) + m_2 \rightarrow E(m_1 + m_2)$; and (iv) *Homomorphic multiplication-II:* when $m_2 > 0$, $E(m_1) * m_2 \rightarrow E(m_1 * m_2)$.

In addition, with the homomorphic properties, the SHE technique can be transformed to a public key encryption scheme, as discussed in [29]. Let $E(0)_1$ and $E(0)_2$ denote two ciphertexts of 0, which are generated by encrypting 0 twice. Then, if we regard $\{E(0)_1, E(0)_2\}$ as the public key, we can encrypt a message m as

$$E(m) \leftarrow (m + r_1 * E(0)_1 + r_2 * E(0)_2) \mod \mathtt{N}, \qquad (1)$$

where $r_1, r_2 \in \{0,1\}^{k_2}$. As proved in [29], the public key version of the SHE technique is also semantically secure.

Sign Computation Protocol: Based on the SHE technique, we have proposed a privacy-preserving sign computation protocol in [28]. This protocol is run between two non-collusive cloud servers (i.e., S_1 and S_2 in our scheme). S_1 holds two ciphertexts $E(m)$ and $E(-1)$, and S_2 holds a secret key sk, where $m \in \mathcal{M}$. They cooperate on running the sign computation protocol such that S_2 obtains the sign of m while keeping the plaintext of m secret from both S_1 and S_2, where the sign of m is defined as

$$\text{sign}(m) = \begin{cases} 1 & m > 0; \\ -1 & m \leq 0. \end{cases}$$

The sign computation protocol has two steps as follows.

Step 1: S_1 first chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$ and uses the homomorphic properties to compute

$$(r_1 * E(m) + r_2 * E(-1)) \mod \mathtt{N} \rightarrow E(r_1 m - r_2).$$

Then, S_1 sends $E(r_1 m - r_2)$ to S_2.

Step 2: On receiving $E(r_1 m - r_2)$, S_2 uses sk to recover the value $r_1 m - r_2$ as $\mathsf{Dec}(sk, E(r_1 m - r_2)) \rightarrow r_1 m - r_2$. Based on the sign of $r_1 m - r_2$, S_2 sets the sign of m as

$$\text{sign}(m) = \begin{cases} 1 & r_1 m - r_2 > 0; \\ -1 & r_1 m - r_2 < 0. \end{cases}$$

IV. Our Proposed Scheme

In this section, we present our EPSim-AC scheme. Before delving into the details, we first introduce a k-d-PB tree and the corresponding similarity range query algorithm with access control over the k-d-PB tree. Then, we define k-d-PB tree's β-access pattern privacy considered in this work.

A. k-d-PB Tree

k-d-PB tree is designed by integrating the key idea of k-d tree and partition-based (PB) tree, and can efficiently support similarity queries with access control. Before introducing the k-d-PB tree in detail, we first recall the key idea of k-d tree and PB tree. Both of them are used for representing multi-dimensional data records and built by recursively cutting or partitioning a dataset into subdatasets. Meanwhile, each of their internal nodes has two key values, a left subtree T_l, and a right subtree T_r. Differently, the key values are chosen in different ways. For the k-d tree, the key values include a cutting dimension `cd` and a cutting value `val`, where `cd` is randomly chosen among all dimensions and `val` is the medium value of all data records in the `cd`-th dimension. Data records in the left subtree T_l (resp. right subtree T_r) are less than or equal to (resp. larger than) `val` in the `cd`-th dimension. For the PB tree, the key values are two pivot records, denoted by $\mathbf{p}_1$ and $\mathbf{p}_2$, that are chosen from all data records. Data records in the left subtree T_l (resp. right subtree T_r) are closer or have an equal distance (resp. farther) to $\mathbf{p}_1$ than $\mathbf{p}_2$. Based on the idea of the k-d tree and PB tree, we design a k-d-PB tree.

Let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{a}_i) | i = 1, 2, \cdots, n\}$ denote a healthcare dataset. Each $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d})$ is a d-dimensional healthcare data record. Each $\mathbf{a}_i = (a_{i,1}, a_{i,2}, \cdots, a_{i,l}) \in \{\Sigma \cup *\}^l$ is an l-dimensional hidden vector based access policy. Then, $\mathcal{X}$ can be represented to a k-d-PB tree T, as shown in Alg. 1. Each internal node of T has two key values, a left subtree, and a right subtree. The key values can be chosen based on either data records or access policy. We take the root node as an example to show the way to choose key values.

Case 1: Choose key values of the root node based on data records. In this case, the key values choosing way is the same as that of PB tree. First, we choose two data records $\mathbf{p}_1, \mathbf{p}_2 \in \{\mathbf{x}_i\}_{i=1}^n$ as pivots. Based on $\mathbf{p}_1$ and $\mathbf{p}_2$, we divide $\mathcal{X}$ into two subdatasets $\mathcal{X}_1$ and $\mathcal{X}_2$. For each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}_1$ (resp. $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}_2$), $\mathbf{x}_i$ is closer or has an equal distance (resp. farther) to $\mathbf{p}_1$ than $\mathbf{p}_2$. As a result, the key values of the root node are $\mathbf{p}_1$ and $\mathbf{p}_2$.

Case 2: Choose key values of root node based on access policy. In this case, the key values choosing way is the same as that of k-d tree. First, we choose a cutting dimension cd from all l dimensions. Then, we compute the median value of all access policy vectors $\{\mathbf{a}_i\}_{i=1}^n$ in the cd-th dimension as the cutting value val. Note that the values of $\{\mathbf{a}_i\}_{i=1}^n$ are in the domain of $\{\Sigma \cup *\}$. To avoid the effect of "*", when computing val, we only consider the data records whose values in the cd-th dimension are not "$*$". Based on cd and val, we divide the dataset $\mathcal{X}$ into two subdatasets $\mathcal{X}_1$ and $\mathcal{X}_2$. For each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}_1$, the value of $\mathbf{a}_i$ is less than or equal to val, or is "*" in the cd-th dimension. That is, $a_{i,\mathtt{cd}} \leq \mathtt{val}$ or $a_{i,\mathtt{cd}} =$"*". For each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}_2$, the value of $\mathbf{a}_i$ is larger than val or is "*" in the cd-th dimension. That is, $a_{i,\mathtt{cd}} > \mathtt{val}$ or $a_{i,\mathtt{cd}} =$"*". As a result, the key values of the root node are cd and val.

After determining key values of the root node, we continue to build the left subtree T_l and right subtree T_r of the root node based on the subdatasets $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively.

Algorithm 1 TreeBuilding(Dataset $\mathcal{X}$)

Set $\mathcal{X}_1 = \emptyset$;
Set $\mathcal{X}_2 = \emptyset$;
if use data records as key values **then**
  Choose two pivots $\{\mathbf{p}_1, \mathbf{p}_2\}$;
  $key_1 = \mathbf{p}_1$; $key_2 = \mathbf{p}_2$;
  for each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}$ **do**
    if $d(\mathbf{x}_i, \mathbf{p}_1) \leq d(\mathbf{x}_i, \mathbf{p}_2)$ **then**
      $\mathcal{X}_1 = \mathcal{X}_1 \cup \{(\mathbf{x}_i, \mathbf{a}_i)\}$;
    else
      $\mathcal{X}_2 = \mathcal{X}_2 \cup \{(\mathbf{x}_i, \mathbf{a}_i)\}$;
else if use access policy as key values **then**
  Choose a cutting dimension cd;
  $\mathtt{val} = \text{median}\{a_{i,\mathtt{cd}} | a_{i,\mathtt{cd}} \neq "*" \text{ for } i = 1, 2, \cdots, n\}$;
  $key_1 = \mathtt{cd}$; $key_2 = \mathtt{val}$;
  for each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{X}$ **do**
    if $a_{i,\mathtt{cd}} \leq \mathtt{val}$ || $a_{i,\mathtt{cd}} =$ "$*$" **then**
      $\mathcal{X}_1 = \mathcal{X}_1 \cup \{(\mathbf{x}_i, \mathbf{a}_i)\}$;
    else if $a_{i,\mathtt{cd}} > \mathtt{val}$ || $a_{i,\mathtt{cd}} =$ "$*$" **then**
      $\mathcal{X}_2 = \mathcal{X}_2 \cup \{(\mathbf{x}_i, \mathbf{a}_i)\}$;
$\mathrm{T}_l = \text{TreeBuilding}(\mathcal{X}_1)$;
$\mathrm{T}_r = \text{TreeBuilding}(\mathcal{X}_2)$;
return Root$(key_1, key_2, \mathrm{T}_l, \mathrm{T}_r)$;

For better understanding, in the following, we give an example to show the structure of the k-d-PB tree.

Example 1: Let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{a}_i)\}_{i=1}^7$ be a dataset with 7 records, where

$$\begin{cases} \mathbf{x}_1 = (2,3); \mathbf{a}_1 = (2,5); \mathbf{x}_2 = (3,1); \mathbf{a}_2 = (3,2); \\ \mathbf{x}_3 = (7,8); \mathbf{a}_3 = (1,2); \mathbf{x}_4 = (8,9); \mathbf{a}_4 = (*,1); \\ \mathbf{x}_5 = (1,1); \mathbf{a}_5 = (4,1); \mathbf{x}_6 = (3,4); \mathbf{a}_6 = (6,4); \\ \mathbf{x}_7 = (8,8); \mathbf{a}_7 = (5,3). \end{cases}$$

We can build a k-d-PB tree for the dataset $\mathcal{X}$, as shown in Fig. 3. As depicted in the figure, the key values of the root node are chosen based on the access policy, where $\mathtt{cd} = 1$ and $\mathtt{val} = 3$. Based on the key values, all records with $a_{i,\mathtt{cd}} \leq \mathtt{val}$ or $a_{i,\mathtt{cd}} =$"*" are put in the left subtree, and all records with $a_{i,\mathtt{cd}} > \mathtt{val}$ or $a_{i,\mathtt{cd}} =$"*" are put in the right subtree. In the second layer of the tree, the key values are chosen based on data records, where the key values of the left one are $\{\mathbf{p}_1 = (2,3), \mathbf{p}_2 = (8,9)\}$, and those of the right one are $\{\mathbf{p}_1 = (1,1), \mathbf{p}_2 = (8,9)\}$. Based on these key values, the records can be further divided into two subdatasets based on their distance to the pivots, as shown in Fig. 3. After that, we can recursively build subtrees for the subdatasets. Due to page limitation, we omit the details here.

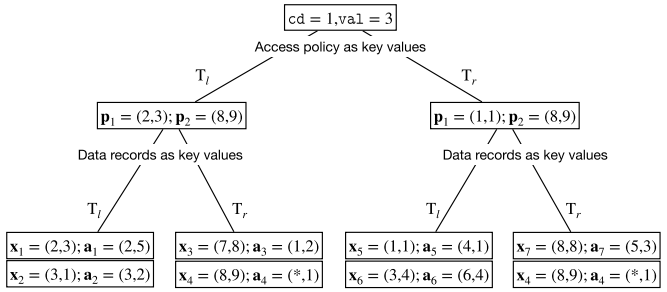


Fig. 3. Example of k-d-PB tree building.

B. Similarity Query Algorithm With Access Control

Based on k-d-PB tree, we design an efficient similarity range query algorithm with access control. Let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{a}_i) | i = 1, 2, \cdots, n\}$ be a healthcare dataset and have been represented to a k-d-PB tree T. Let $(\mathbf{q}, \tau, \mathbf{v}_j)$ be a query request, where $\mathbf{q}$ is a query data record, τ is a query range, and $\mathbf{v}_j \in \Sigma$ is U_j's attribute vector. Then, we can run the similarity query algorithm to search on T for all data records satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ and $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$. As shown in Alg. 2, the similarity range query algorithm has two stages, i.e., filtration and verification.

• *Filtration stage:* In the filtration stage, the searcher searches on $\mathcal{X}$ for a candidate set $\mathcal{C}$ containing data records that may satisfy the query request as follows.

(1) When the searched node is a leaf node with the data record $(\mathbf{x}_i, \mathbf{a}_i)$, the searcher directly adds $(\mathbf{x}_i, \mathbf{a}_i)$ into the candidate set $\mathcal{C}$, i.e., $\mathcal{C} = \mathcal{C} \cup \{(\mathbf{x}_i, \mathbf{a}_i)\}$.

(2) When the searched node $node$ is an internal node. Based on $node$'s key values, there are two cases.

Case 1: The key values are pivots $(\mathbf{p}_1, \mathbf{p}_2)$. If $(\mathbf{p}_1, \mathbf{p}_2)$ and $(\mathbf{q}, \tau)$ do not satisfy Hilbert Exclusion Condition, i.e.,

$$\frac{d(\mathbf{p}_1, \mathbf{q})^2 - d(\mathbf{p}_2, \mathbf{q})^2}{2d(\mathbf{p}_1, \mathbf{p}_2)} \leq \tau,$$

data records satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ may fall in T_l. Then, $node.\mathrm{T}_l$ cannot be pruned and needs to be searched. Similarly, if $(\mathbf{p}_1, \mathbf{p}_2)$ and $(\mathbf{q}, \tau)$ do not satisfy that

$$\frac{d(\mathbf{p}_2, \mathbf{q})^2 - d(\mathbf{p}_1, \mathbf{q})^2}{2d(\mathbf{p}_1, \mathbf{p}_2)} < \tau,$$

data records satisfying $d(\mathbf{x}_i, \mathbf{q}) \leq \tau$ may fall within $node.\mathrm{T}_r$. Then, $node.\mathrm{T}_r$ needs to be searched.

Case 2: The key values of $node$ are (cd, val). When $v_{j,\mathtt{cd}} \leq$ val, all records satisfying $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$

are in $node.\mathrm{T}_l$. Then, only $node.\mathrm{T}_l$ needs to be searched later. Otherwise, when $v_{j,\mathrm{cd}} >$ val, all records satisfying $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$ are in $node.\mathrm{T}_r$. Then, only $node.\mathrm{T}_r$ needs to be searched later.

Algorithm 2 SimQuery(Tree T, Query $(\mathbf{q}, \tau, \mathbf{v}_j)$)

```
// Filtration stage
C = FILTRATION(T.root, (q, τ, v_j));
// Verification stage
for each (x_i, a_i) ∈ C do
    if d(x_i, q) ≤ τ and HV.Match(a_i, v_j) = 1 then
        R = R ∪ {x_i};
return R;
function FILTRATION(Node node, Query (q, τ, v_j))
    if node is a leaf node with (x_i, a_i) then
        C = C ∪ {(x_i, a_i)}
    else if node is an internal node then
        // Case 1: the key values are (p_1, p_2)
        if (d(p_1,q)^2 − d(p_2,q)^2)/(2d(p_1,p_2)) ≤ τ then
            FILTRATION(node.T_l, (q, τ, v_j));
        if (d(p_2,q)^2 − d(p_1,q)^2)/(2d(p_1,p_2)) < τ
            FILTRATION(node.T_r, (q, τ, v_j));
        // Case 2: the key values are (cd, val)
        if v_{j,cd} ≤ val then
            FILTRATION(node.T_l, (q, τ, v_j));
        else
            FILTRATION(node.T_r, (q, τ, v_j));
```

• *Verification stage:* In the verification stage, the searcher verifies whether each $(\mathbf{x}_i, \mathbf{a}_i) \in \mathcal{C}$ satisfies that $d(\mathbf{x}_i, \mathbf{q}) \le \tau$ and $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$. If yes, add $\mathbf{x}_i$ into the query result, i.e., $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$.

Thus, the basic operations of the similarity range query scheme with access control include

(1) $\frac{d(\mathbf{p}_1,\mathbf{q})^2 - d(\mathbf{p}_2,\mathbf{q})^2}{2d(\mathbf{p}_1,\mathbf{p}_2)} \le \tau$ and $v_{j,\mathrm{cd}} \le \mathtt{val}$;

(2) $\frac{d(\mathbf{p}_2,\mathbf{q})^2 - d(\mathbf{p}_1,\mathbf{q})^2}{2d(\mathbf{p}_1,\mathbf{p}_2)} < \tau$ and $v_{j,\mathrm{cd}} > \mathtt{val}$;

(3) $d(\mathbf{x}_i, \mathbf{q}) \le \tau$ and $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$.

Remarks: From the similarity query algorithm, we can observe that the query efficiency of our scheme depends on the filtration effect. To enhance the filtration effect, we can optimize the structure of k-d-PB tree. When constructing an internal node, although the key values of each internal node can be chosen based on either data records or access policy, we will construct it based on both data records and access policy, and choose the one that has a better filtration effect with a high probability. Specifically, when we use the data records as the key values, we need to consider the probability that two pivots $(\mathbf{p}_1, \mathbf{p}_2)$ and a query request $(\mathbf{q}, \tau)$ satisfy the Hilbert exclusion condition. If the probability is high, we can choose data records as the key values. Similarly, when we use the access policy as the key values, either the left subtree and right subtree can be pruned in similarity queries, as shown in Alg. 1. Thus, we need to consider that whether the number of records in the left subtree and right subtree are approximately equal. If yes, we have potential to obtain a good filtration effect and can choose the access policy as the key values.

C. Access Pattern Privacy of k-d-PB Tree

When a k-d-PB tree T is accessed, we demand to consider its access pattern privacy. Since it is widely believed that hiding the access pattern of each tree path among all tree paths is unnecessary and outrageously expensive, especially when the tree is large [24], in this paper, we define a weakened access pattern privacy of k-d-PB tree, called β-access pattern unlinkability, as shown in Definition 1.

Definition 1 (k-d-PB Tree's β-Access Pattern Unlinkability): A k-d-PB tree is accessed in a way satisfying β-access pattern unlinkability if a tree path is indistinguishably accessed with $(\beta - 1)$ tree paths for $\beta > 1$.

D. Main Idea of Our EPSim-AC Scheme

To well preserve the privacy of k-d-tree based similarity queries with access control, we first design a way to represent k-d-PB tree and query requests. Then, based on the designed representation, we present the main idea of our EPSim-AC scheme.

Representation of k-d-PB Tree and Query Requests: We represent k-d-PB tree T and query requests as follows.

• *Representation of* T*'s internal nodes.* For each internal node, based on its key values, we represent it to two $(l+d+3)$-dimensional vectors $\mathbf{u}_L$ and $\mathbf{u}_R$, which can be constructed as follows.

Case 1: When the key values are pivots $(\mathbf{p}_1, \mathbf{p}_2)$, we construct $\mathbf{u}_L$ and $\mathbf{u}_R$ as

$$\begin{cases} \mathbf{u}_L = (\mathbf{0}_{l+1}, -d(\mathbf{p}_1, \mathbf{p}_2), \|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2, \mathbf{p}_2 - \mathbf{p}_1); \\ \mathbf{u}_R = (\mathbf{0}_{l+1}, d(\mathbf{p}_1, \mathbf{p}_2), \|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2, \mathbf{p}_2 - \mathbf{p}_1); \end{cases}$$

where $\mathbf{0}_{l+1}$ is an $(l+1)$-dimensional zero vector.

Case 2: When the key values are (cd, val), we construct $\mathbf{u}_L$ and $\mathbf{u}_R$ as $\mathbf{u}_L = \mathbf{u}_R = (\mathbf{0}_{\mathrm{cd}-1}, 1, \mathbf{0}_{l-\mathrm{cd}}, -\mathrm{val}, \mathbf{0}_{d+2})$, where $\mathbf{0}_{\mathrm{cd}-1}$, $\mathbf{0}_{l-\mathrm{cd}}$, and $\mathbf{0}_{d+2}$ are respectively $(\mathrm{cd}-1)$-, $(l-\mathrm{cd})$-, and $(d+2)$-dimensional zero vectors.

• *Representation of* T*'s leaf nodes.* For each leaf node with the record $(\mathbf{x}_i, \mathbf{a}_i)$, we first replace all "*" attribute values in $\mathbf{a}_i$ to 0 and then represent $(\mathbf{x}_i, \mathbf{a}_i)$ to a $(2l+d+3)$-dimensional vector $\mathbf{z}_i = (\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \mathbf{z}_{i,3})$, where

$$\begin{cases} \mathbf{z}_{i,1} = (\mathbf{x}_i, \|\mathbf{x}_i\|^2, 1); \\ \mathbf{z}_{i,2} = (a_{i,1}, a_{i,2}, \cdots, a_{i,l}); \\ \mathbf{z}_{i,3} = (-2a_{i,1}^2, -2a_{i,2}^2, \cdots, -2a_{i,l}^2, \sum_{w=1}^{l} a_{i,w}^3). \end{cases} \tag{2}$$

• *Representation of query requests.* For a query request $(\mathbf{q}, \tau, \mathbf{v}_j)$, we first represent it to an $(l+d+3)$-dimensional filtration vector

$$\mathbf{t}_1 = (\mathbf{v}_j, 1, 2\tau, 1, 2\mathbf{q}), \tag{3}$$

and a $(2l+d+3)$-dimensional verification vector $\mathbf{t}_2 = (\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \mathbf{t}_{2,3})$, where

$$\begin{cases} \mathbf{t}_{2,1} = (-2\mathbf{q}, 1, \|\mathbf{q}\|^2 - \tau^2); \\ \mathbf{t}_{2,2} = \delta^2(v_{j,1}^2, v_{j,2}^2, \cdots, v_{j,l}^2); \\ \mathbf{t}_{2,3} = \delta^2(v_{j,1}, v_{j,2}, \cdots, v_{j,l}, 1), \end{cases} \tag{4}$$

and $\delta > \tau$.

Based on the representation of k-d-PB tree and query requests, we have two inferences.

Inference I: For an internal node and a query request, whatever key values are in the internal node, we have

$$\begin{cases} node.\mathrm{T}_l \text{ needs to be searched } \Leftrightarrow \mathbf{u}_L \circ \mathbf{t}_1 \leq 0; \\ node.\mathrm{T}_r \text{ needs to be searched } \Leftrightarrow \mathbf{u}_R \circ \mathbf{t}_1 > 0. \end{cases}$$

Next, we prove the correctness of Inference I.

Proof: We prove the correctness of Inference I based on key values of internal nodes.

Case 1: If the key values are pivots $(\mathbf{p}_1, \mathbf{p}_2)$, we have

$$\begin{aligned} &\mathbf{u}_L \circ \mathbf{t}_1 \leq 0 \\ &\Leftrightarrow (\mathbf{0}_{l+1}, -d(\mathbf{p}_1, \mathbf{p}_2), \|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2, \mathbf{p}_2 - \mathbf{p}_1) \\ &\quad \circ (\mathbf{v}_j, 1, 2\tau, 1, 2\mathbf{q}) \leq 0; \\ &\Leftrightarrow -2\tau d(\mathbf{p}_1, \mathbf{p}_2) + \|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2 + 2(\mathbf{p}_2 - \mathbf{p}_1) \circ \mathbf{q} \leq 0; \\ &\Leftrightarrow \frac{\|\mathbf{p}_1\|^2 - \|\mathbf{p}_2\|^2 - 2(\mathbf{p}_1 - \mathbf{p}_2) \circ \mathbf{q}}{2d(\mathbf{p}_1, \mathbf{p}_2)} \leq \tau; \\ &\Leftrightarrow \frac{d(\mathbf{p}_1, \mathbf{q})^2 - d(\mathbf{p}_2, \mathbf{q})^2}{2d(\mathbf{p}_1, \mathbf{p}_2)} \leq \tau; \\ &\Leftrightarrow node.\mathrm{T}_l \text{ needs to be searched.} \end{aligned}$$

Similarly, we have

$$\begin{aligned} \mathbf{u}_R \circ \mathbf{t}_1 > 0 &\Leftrightarrow \frac{d(\mathbf{p}_2, \mathbf{q})^2 - d(\mathbf{p}_1, \mathbf{q})^2}{2d(\mathbf{p}_1, \mathbf{p}_2)} < \tau; \\ &\Leftrightarrow node.\mathrm{T}_r \text{ needs to be searched.} \end{aligned}$$

Case 2: If the key values are (cd, val), we have

$$\begin{aligned} &\mathbf{u}_L \circ \mathbf{t}_1 \leq 0; \\ &\Leftrightarrow (\mathbf{0}_{\mathrm{cd}-1}, 1, \mathbf{0}_{l-\mathrm{cd}}, -\mathrm{val}, \mathbf{0}_{d+2}) \circ (\mathbf{v}_j, 1, 2\tau, 1, 2\mathbf{q}) \leq 0; \\ &\Leftrightarrow v_{j,\mathrm{cd}} - \mathrm{val} \leq 0 \Leftrightarrow v_{j,\mathrm{cd}} \leq \mathrm{val}; \\ &\Leftrightarrow node.\mathrm{T}_l \text{ needs to be searched.} \end{aligned}$$

Similarly, we have

$$\begin{aligned} \mathbf{u}_R \circ \mathbf{t}_1 > 0 &\Leftrightarrow v_{j,\mathrm{cd}} > \mathrm{val}; \\ &\Leftrightarrow node.\mathrm{T}_r \text{ needs to be searched.} \end{aligned}$$

Therefore, Inference I holds. □

Inference II: For a leaf node and a query request, we have

$$\mathbf{z}_i \circ \mathbf{t}_2 \leq 0 \Leftrightarrow \{d(\mathbf{x}_i, \mathbf{q}) \leq \tau \text{ and } \mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1.\}$$

Next, we prove the correctness of Inference II.

Proof: First, we have

$$\begin{aligned} &\mathbf{z}_i \circ \mathbf{t}_2 \leq 0 \\ &\Leftrightarrow (\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \mathbf{z}_{i,3}) \circ (\mathbf{t}_{2,1}, \mathbf{t}_{2,2}, \mathbf{t}_{2,3}) \leq 0; \\ &\Leftrightarrow \mathbf{z}_{i,1} \circ \mathbf{t}_{2,1} + \mathbf{z}_{i,2} \circ \mathbf{t}_{2,2} + \mathbf{z}_{i,3} \circ \mathbf{t}_{2,3} \leq 0; \\ &\Leftrightarrow \delta^2 (\sum_{w=1}^{l} (a_{i,w} * v_{j,w}^2) + \sum_{w=1}^{l} (-2a_{i,w}^2 * v_{j,w}) + \sum_{w=1}^{l} a_{i,w}^3) \\ &\quad + \|\mathbf{x}_i\|^2 - 2 * \mathbf{x}_i \circ \mathbf{q} + \|\mathbf{q}\|^2 - \tau^2 \leq 0; \\ &\Leftrightarrow \delta^2 (\sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2) + (\mathbf{x}_i - \mathbf{q})^2 - \tau^2 \leq 0. \end{aligned} \quad (5)$$

Since all values $\{a_{i,1}, a_{i,2}, \cdots, a_{i,l}\}$ and $\{v_{j,1}, v_{j,2}, \cdots, v_{j,l}\}$ are non-negative integers, we have $\delta^2 \sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2 \geq 0$. Since $\delta > \tau$, when $\delta^2 \sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2 \geq 1$, we have $\delta^2 (\sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2) - \tau^2 > 0$ and furthermore $\delta^2 (\sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2) + (\mathbf{x}_i - \mathbf{q})^2 - \tau^2 > 0$, which contradicts with Eq. (5). Hence, we can deduce that

$$\delta^2 \sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2 = 0 \Rightarrow \sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2 = 0.$$

After that, we have $a_{i,w} = 0$ or $a_{i,w} = v_{j,w}$ for $w = 1, 2, \cdots, l$. We further have $a_{i,w} = \text{“}*\text{”}$ or $a_{i,w} = v_{j,w}$, i.e., $\mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1$. Based on Eq. (5), when $\sum_{w=1}^{l} a_{i,w} (v_{j,w} - a_{i,w})^2 = 0$, we have $(\mathbf{x}_i - \mathbf{q})^2 - \tau^2 \leq 0 \Rightarrow d(\mathbf{x}_i, \mathbf{q}) \leq \tau^2$. Thus, when $\mathbf{z}_i \circ \mathbf{t}_2 \leq 0$, we have

$$d(\mathbf{x}_i, \mathbf{q}) \leq \tau \text{ and } \mathsf{HV.Match}(\mathbf{a}_i, \mathbf{v}_j) = 1.$$

Therefore, Inference II holds. □

With Inference I and Inference II, the basic operations of the similarity query algorithm with access control become

$$\begin{cases} \text{Internal node: } \mathbf{u}_L \circ \mathbf{t}_1 \leq 0 \text{ and } \mathbf{u}_R \circ \mathbf{t}_1 > 0; \\ \text{Leaf node: } \mathbf{z}_i \circ \mathbf{t}_2 \leq 0. \end{cases}$$

To make the operations $\mathbf{u}_L \circ \mathbf{t}_1 \leq 0$ and $\mathbf{u}_R \circ \mathbf{t}_1 > 0$ indistinguishable, we attach a label l_L to $\mathbf{u}_L$ and a label l_R to $\mathbf{u}_R$, and set $l_L = -1$ and $l_R = 1$. In this case, we have

$$\begin{cases} \mathbf{u}_L \circ \mathbf{t}_1 \leq 0 \Leftrightarrow \mathrm{sign}(\mathbf{u}_L \circ \mathbf{t}_1) = l_L; \\ \mathbf{u}_R \circ \mathbf{t}_1 > 0 \Leftrightarrow \mathrm{sign}(\mathbf{u}_R \circ \mathbf{t}_1) = l_R. \end{cases}$$

The operations on $\mathbf{u}_L$ and $\mathbf{u}_R$ become an equality test.

Main Idea of Our EPSim-AC Scheme: Based on the representation of k-d-PB tree and query requests, we introduce the main idea of our scheme. Let T be a k-d-PB tree, whose key values in each internal node are $\{\mathbf{u}_L, l_L, \mathbf{u}_R, l_R\}$ and that in each leaf node is $\mathbf{z}_i$. Given a query request $\{\mathbf{t}_1, \mathbf{t}_2\}$, we introduce a similarity query algorithm to search on T for the query result. To preserve β-access pattern unlinkability later, in our similarity query algorithm, we hierarchically search on T for the query result. Further, to speed up query efficiency, we associate a flag with each internal node, which denotes whether the node has been pruned or not. Specifically, if $node.\mathtt{flag} = 0$, it means that $node$ has been pruned. If $node.\mathtt{flag} = 1$, it means that $node$ has not been pruned. Especially, at the beginning of the algorithm, we set $\mathrm{T}.root.\mathtt{flag} = 1$ denoting the root node is not pruned. Next, we introduce the searching algorithm, which has two stages, i.e., filtration and verification.

• *Filtration stage:* In the filtration stage, we hierarchically search on T for a candidate set $\mathcal{C}$. Without loss of generality, let $\mathcal{N} = [node_1, node_2, \cdots, node_{|\mathcal{N}|}]$ denote an array of internal nodes to be searched in the i-th layer of T. Each $node_j \in \mathcal{N}$ has key values $\{\mathbf{u}_{j,L}, l_{j,L}, \mathbf{u}_{j,R}, l_{j,R}\}$ and is associated with a flag $node_j.\mathtt{flag}$. Based on $\mathcal{N}$, we can determine the node array to be searched in the next layer of T as the following steps.

Step 1: For each $node_j \in \mathcal{N}$, we can deduce that

$$node_j.\mathrm{T}_l.\mathtt{flag} = \begin{cases} 1 & \text{if } \begin{cases} \mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1) = l_{j,L}; \\ node_j.\mathtt{flag} = 1; \end{cases} \\ 0 & \text{Otherwise.} \end{cases} \quad (6)$$

$$node_j.\mathrm{T}_r.\mathtt{flag} = \begin{cases} 1 & \text{if } \begin{cases} \mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1) = l_{j,R}; \\ node_j.\mathtt{flag} = 1; \end{cases} \\ 0 & \text{Otherwise.} \end{cases} \quad (7)$$

That is, $node_j.\mathrm{T}_l.\mathtt{flag} = 1$ (i.e., $node_j.\mathrm{T}_l$ needs to be searched) *iff* $\mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1) = l_{j,L}$ and its parent node $node_j$ is not pruned, i.e., $node_j.\mathtt{flag} = 1$. Meanwhile, $node_j.\mathrm{T}_r.\mathtt{flag} = 1$ (i.e., $node_j.\mathrm{T}_r$ needs to be searched) *iff* $\mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1) = l_{j,L}$ and its parent node $node_j$ is not pruned, i.e., $node_j.\mathtt{flag} = 1$. Otherwise, we set $node_j.\mathrm{T}_l.\mathtt{flag} = 0$ and $node_j.\mathrm{T}_r.\mathtt{flag} = 0$. Then, the nodes need to be searched in the next layer will be

$$\begin{aligned} \mathcal{N}_{next,1} = &\{node_j.\mathrm{T}_l | node_j.\mathrm{T}_l.\mathtt{flag} = 1; node_j \in \mathcal{N}\} \\ &\cup \{node_j.\mathrm{T}_r | node_j.\mathrm{T}_r.\mathtt{flag} = 1; node_j \in \mathcal{N}\}. \end{aligned}$$

In contrast, the nodes do not need to be searched will be

$$\begin{aligned} \mathcal{N}_{next,0} = &\{node_j.\mathrm{T}_l | node_j.\mathrm{T}_l.\mathtt{flag} = 0; node_j \in \mathcal{N}\} \\ &\cup \{node_j.\mathrm{T}_r | node_j.\mathrm{T}_r.\mathtt{flag} = 0; node_j \in \mathcal{N}\}. \end{aligned}$$

Step 2: To guarantee k-d-PB tree's β-access pattern unlinkability, we will randomly choose $|\mathcal{N}_{next,1}| * (\beta - 1)$ nodes from $\mathcal{N}_{next,0}$ and add them into $\mathcal{N}_{next,1}$. Note that when the number of nodes in $\mathcal{N}_{next,0}$ is smaller than $|\mathcal{N}_{next,1}| * (\beta - 1)$, we will add all nodes in $\mathcal{N}_{next,0}$ into $\mathcal{N}_{next,1}$.

• *Verification stage:* For each record $\mathbf{z}_i \in \mathcal{C}$, we verify whether it satisfies the query request by checking $\mathbf{z}_i \circ \mathbf{t}_2 \overset{?}{\leq} 0$.

E. Our EPSim-AC Scheme

In this subsection, we present our EPSim-AC scheme. Before presenting our scheme, we introduce two default calculation methods in the following description of our scheme.

(1) We use SHE technique to encrypt an integer vector $\mathbf{x} = (x_1, x_2, \cdots, x_d)$ into a ciphertext vector $E(\mathbf{x})$ as $E(\mathbf{x}) = (E(x_1), E(x_2), \cdots, E(x_d))$. Especially, S_1 will not be assigned a secret key sk in our EPSim-AC scheme, so it encrypts vectors using public SHE technique in Eq. (1). S_2 will be assigned an sk, so it encrypts vectors using the original SHE technique.

(2) To prevent S_2 from knowing the structure of k-d-PB tree, when S_1 accesses each layer of k-d-tree, it first permutates the nodes to be accessed. After it receives the result related to these nodes from S_2, it uses the same permutation method to permute the result again.

Specifically, our EPSim-AC scheme contains three phases, i.e., system initialization, data outsourcing, and query processing. Details are shown as follows.

• ***System Initialization:*** In the system initialization phase, HC is responsible for initializing the system. First, it chooses three security parameters (k_0, k_1, k_2) and generates a pair of SHE's public key and secret key as $(pk, sk) \longleftarrow \mathsf{KeyGen}(k_0, k_1, k_2)$. Second, it generates three ciphertexts $\{E(0)_1, E(0)_2, E(-1)\}$. Then, it generates a hash-based message authentication codes (HMAC), i.e., $H_K(\cdot)$, where K is a secret key. Finally, it publishes $\{pk, E(0)_1, E(0)_2, E(-1)\}$, distributes $H_K(\cdot)$ to S_1 and users, and distributes sk to S_2.

• ***Data Outsourcing:*** In the data outsourcing phase, HC outsources its dataset to the cloud server S_1. Specifically, let $\mathcal{X} = \{(\mathbf{x}_i, \mathbf{a}_i) | i = 1, 2, \cdots, n\}$ be a healthcare dataset. The HC outsources it to S_1 as the following steps.

Step 1: HC represents $\mathcal{X}$ to be a k-d-PB tree T. Then, it encrypts T into $E(T)$. Specifically, for each internal node, HC first represents it to a four-tuple $\{\mathbf{u}_L, l_L, \mathbf{u}_R, l_R\}$ and further encrypts it into a ciphertext $\{E(\mathbf{u}_L), E(l_L), E(\mathbf{u}_R), E(l_R)\}$. For each leaf node, HC first represents it to $\mathbf{z}_i$ and then encrypts $\mathbf{z}_i$ into a ciphertext $E(\mathbf{z}_i)$. After that, T will be encrypted into an encrypted k-d-PB tree, denoted by $E(\mathrm{T})$.

Step 2: HC further randomly permutates the left and right subtrees of each internal node and sends the permutated $E(\mathrm{T})$ to the cloud server S_1.

• ***Query Processing:*** On receiving $E(\mathrm{T})$, S_1 can offer similarity range query service with access control to doctors with the help of S_2. Specifically, a doctor U_j with the identity id_j can launch a query request $(\mathbf{q}, \tau, \mathbf{v}_j)$ to S_1 as follows.

Step 1: U_j generates a query token for $(\mathbf{q}, \tau, \mathbf{v}_j)$. Specifically, it represents $(\mathbf{q}, \tau, \mathbf{v}_j)$ to two vectors $\{\mathbf{t}_1, \mathbf{t}_2\}$ and encrypts them into ciphertexts $\{E(\mathbf{t}_1), E(\mathbf{t}_2)\}$. Then, it chooses a session key ssk and encrypts it into a ciphertext $\mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk)$. Finally, U_j sends a query request with the query token $\{E(\mathbf{t}_1), E(\mathbf{t}_2), \mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk), \mathrm{id}_j\}$ to S_1.

Step 2: On receiving $\{E(\mathbf{t}_1), E(\mathbf{t}_2), \mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk), \mathrm{id}_j\}$, S_1 searches on $E(\mathrm{T})$ to find out the query result. The searching process has a filtration stage and a verification stage. The former filters out candidate data records that possibly satisfy the query request, and the latter verifies whether each candidate record satisfies the query request.

Filtration Stage: In the filtration stage, two cloud servers hierarchically search on $E(\mathrm{T})$ for the candidate records, as shown in Alg. 3. Without loss of generality, let $\mathcal{N} = [node_1, node_2, \cdots, node_{|\mathcal{N}|}]$ denote an array of internal nodes in the i-th layer of $E(\mathrm{T})$ to be searched. Meanwhile, each $node_j \in \mathcal{N}$ is related to an encrypted flag

$$E(node_j.\mathtt{flag}) = \begin{cases} E(1) & node_j \text{ is not pruned;} \\ E(0) & node_j \text{ is pruned.} \end{cases}$$

Then, S_1 and S_2 cooperate on searching $\mathcal{N}$ as follows.

Step 1: For each $node_j \in \mathcal{N}$ with $\{E(\mathbf{u}_L), E(l_L), E(\mathbf{u}_R), E(l_R)\}$, S_1 does the following computation.

(1) S_1 uses SHE's homomorphic properties to compute $E(\mathbf{u}_L \circ \mathbf{t}_1)$ and $E(\mathbf{u}_R \circ \mathbf{t}_1)$. Then, it runs sign computation protocol with S_2 such that S_2 obtains $\mathrm{sign}(\mathbf{u}_L \circ \mathbf{t}_1)$ and $\mathrm{sign}(\mathbf{u}_R \circ \mathbf{t}_1)$. After that, S_2 sends $E(\mathrm{sign}(\mathbf{u}_L \circ \mathbf{t}_1))$ and $E(\mathrm{sign}(\mathbf{u}_R \circ \mathbf{t}_1))$ to S_1.

(2) For each $node_j \in \mathcal{N}$, S_1 uses SHE's homomorphic properties to compute

$$\begin{aligned} &E(s_{j,L}) \\ &= E(r_{j,L}(\mathrm{sign}(\mathbf{u}_L \circ \mathbf{t}_1) - l_L) + r'_{j,L}(node_j.\mathtt{flag} - 1)); \end{aligned} \quad (8)$$

$$\begin{aligned} &E(s_{j,R}) \\ &= E(r_{j,R}(\mathrm{sign}(\mathbf{u}_R \circ \mathbf{t}_1) - l_R) + r'_{j,R}(node_j.\mathtt{flag} - 1)), \end{aligned} \quad (9)$$

where $\{r_{j,L}, r'_{j,L}, r_{j,R}, r'_{j,R}\} \in \mathcal{M}$ are non-zero. It is worth noting that since $\{r_{j,L}, r'_{j,L}\}$ are non-zero, we have

$$s_{j,L}=0 \Leftrightarrow \begin{cases} \text{sign}(\mathbf{u}_L \circ \mathbf{t}_1)=l_L; \\ node_j.\texttt{flag}=1. \end{cases} \Leftrightarrow node_j.\text{T}_l.\texttt{flag}=1. \tag{10}$$

and

$$s_{j,R}=0 \Leftrightarrow \begin{cases} \text{sign}(\mathbf{u}_R \circ \mathbf{t}_1)=l_R; \\ node_j.\texttt{flag}=1. \end{cases} \Leftrightarrow node_j.\text{T}_r.\texttt{flag}=1. \tag{11}$$

Then, based on $\{E(s_{j,L}), E(s_{j,R})\}_{node_j \in \mathcal{N}}$, S_1 constructs an array A with $2|\mathcal{N}|$ elements, where

$$A[2*j-1]=E(s_{j,L}) \quad \text{and } A[2*j]=E(s_{j,R})$$

for $1 \le j \le |\mathcal{N}|$. Finally, it sends A to S_2.

Step 2: On receiving A, S_2 decrypts each $A[k]$ as $m_k \leftarrow \mathsf{Dec}(sk, A[k])$ for $1 \le k \le |A|$ and constructs two sets $\{\mathcal{B}_0, \mathcal{B}_1\}$ as

$$\begin{cases} \mathcal{B}_0 = \{(k, E(m_k))|m_k \neq 0; 1 \le k \le 2|\mathcal{N}|\}; \\ \mathcal{B}_1 = \{(k, E(m_k))|m_k = 0; 1 \le k \le 2|\mathcal{N}|\}. \end{cases} \tag{12}$$

which respectively denote indices to be and not to be searched in the next layer. This is because when $m_k = 0$, we have

$$\begin{cases} s_{j,L}=0 \Rightarrow node_j.\text{T}_l.\texttt{flag}=1 \quad k=2j-1; \\ s_{j,R}=0 \Rightarrow node_j.\text{T}_r.\texttt{flag}=1 \quad k=2j. \end{cases}$$

That is, the node with index k needs to be searched in the next layer. Similarly, when $m_k \neq 0$, the node with index k will not be searched in the next layer.

In addition, to guarantee k-d-PB tree's β-access pattern unlinkability, S_2 will randomly choose a subset $\mathcal{B}'_0$ from $\mathcal{B}_0$ such that $|\mathcal{B}'_0| = |\mathcal{B}_1| * (\beta - 1)$. Especially, when $|\mathcal{B}'_0| < |\mathcal{B}_1| * (\beta - 1)$, S_2 sets $\mathcal{B}'_0 = \mathcal{B}_0$. Then, it sends $\mathcal{B}_1 \cup \mathcal{B}'_0$ to S_1.

Step 3: On receiving $\mathcal{B}_1 \cup \mathcal{B}'_0$, S_2 constructs the nodes to be searched in the next layer. Specifically, for each $k \in \mathcal{B}_1 \cup \mathcal{B}'_0$, if $k = 2j-1$, $node_j.\text{T}_l$ needs to be searched and $E(m_{2j-1})$ is the flag of $node_j.\text{T}_l$. If $k = 2j$, $node_j.\text{T}_r$ needs to be searched and $E(m_{2j})$ is the flag of $node_j.\text{T}_r$. That is, the nodes to be searched in the next layer are

$$\begin{aligned} \mathcal{N}_{next} = &\{(node_j.\text{T}_l, E(m_{2j-1}))|2j-1 \in \mathcal{B}_1 \cup \mathcal{B}'_0\} \\ &\cup \{(node_j.\text{T}_r, E(m_{2j}))|2j \in \mathcal{B}_1 \cup \mathcal{B}'_0\}. \end{aligned}$$

Then, S_1 organizes $\mathcal{N}_{next}$ to a node array and sets $\mathcal{N} = \mathcal{N}_{next}$, which will be searched in the next layer.

Step 4: S_1 continues to run ***Step 1***-***Step 3*** to search $\mathcal{N}$ until $E(\text{T})$'s last layer is searched.

Verification Stage: In the verification stage, two servers verify whether each candidate record with the ciphertext $E(\mathbf{z}_i)$ in $\mathcal{C}$ satisfies the query request or not as follows.

Step 1: S_1 uses SHE's homomorphic properties to compute $E(\mathbf{z}_i \circ \mathbf{t}_2)$ and runs the sign computation protocol with S_2 such that S_2 obtains $\text{sign}(\mathbf{z}_i \circ \mathbf{t}_2)$.

Step 2: S_1 chooses a d-dimensional random vector $\mathbf{r}_i = (r_{i,1}, r_{i,2}, \cdots, r_{i,d})$, where $r_{i,j} \in \{0,1\}^{k_1}$. Then, it extracts the d-dimensional data record $E(\mathbf{x}_i)$ from $E(\mathbf{z}_i)$, i.e., values in the first d dimensions of $E(\mathbf{z}_i)$. Furthermore, it computes $E(\mathbf{x}_i + \mathbf{r}_i) \rightarrow E(\mathbf{y}_i)$. Then, it computes $H_K(\text{id}_j)$ and decrypts $\text{AES}_{H_K(\text{id}_j)}(ssk)$ to recover the session key ssk. After that, it uses ssk to encrypt $\mathbf{r}_i$ as $\text{AES}_{ssk}(\mathbf{r}_i)$. Finally, S_1 sends $\{E(\mathbf{y}_i), \text{AES}_{ssk}(\mathbf{r}_i)\}$ to the server S_2.

Step 3: On obtaining $\text{sign}(\mathbf{z}_i \circ \mathbf{t}_2)$ and $\{E(\mathbf{y}_i), \text{AES}_{ssk}(\mathbf{r}_i)\}$, if $\text{sign}(\mathbf{z}_i \circ \mathbf{t}_2) = -1$, S_2 uses sk to recover $\mathbf{y}_i$ from $E(\mathbf{y}_i)$ and returns $\{\mathbf{y}_i, \text{AES}_{ssk}(\mathbf{r}_i)\}$ to U_j.

Step 4: On receiving $\{\mathbf{y}_i, \text{AES}_{ssk}(\mathbf{r}_i)\}$, U_j uses ssk to recover $\mathbf{r}_i$ from $\text{AES}_{ssk}(\mathbf{r}_i)$ and recovers $\mathbf{x}_i$ as $\mathbf{x}_i = \mathbf{y}_i - \mathbf{r}_i$.

Algorithm 3 Filtration(Tree $E(\text{T})$, Token $E(\mathbf{t}_1)$)

Set $\mathcal{C} = \emptyset$; // Candidate query result
Array $\mathcal{N} = [E(\text{T}).root]$; // Nodes to be searched at start
Array A; // an array of length $2|\mathcal{N}|$
for $i = 1, 2, \cdots, E(\text{T}).\texttt{height}$ **do**
  Server S_1:
  for each $\mathcal{N}[j]$ with $\{E(\mathbf{u}_L), E(l_L), E(\mathbf{u}_R), E(l_R)\}$ **do**
    Compute $E(\mathbf{u}_L \circ \mathbf{t}_1)$ and $E(\mathbf{u}_R \circ \mathbf{t}_1)$;
    Compute $E(\text{sign}(\mathbf{u}_L \circ \mathbf{t}_1))$ and $E(\text{sign}(\mathbf{u}_R \circ \mathbf{t}_1))$;
    Compute $E(s_{j,L})$ and $E(r_{j,R})$ as Eq. (8) and Eq. (9);
    $\mathcal{A}[2*j-1] = E(s_{j,L})$; $\mathcal{A}[2*j] = E(s_{j,R})$;
  Server S_1 **to** S_2: $S_1 \xrightarrow{A} S_2$;
  Server S_2:
  for $k = 1, 2, \cdots, 2|\mathcal{N}|$ **do**
    $m_k \leftarrow \mathsf{Dec}(sk, A[k])$;
  $\mathcal{B}_0 = \{(k, E(m_k))|m_k \neq 0; 1 \le k \le 2|\mathcal{N}|\}$;
  $\mathcal{B}_1 = \{(k, E(m_k))|m_k = 0; 1 \le k \le 2|\mathcal{N}|\}$;
  Choose a set $\mathcal{B}'_0$ from $\mathcal{B}_0$ such that $|\mathcal{B}'_0| = |\mathcal{B}_1| * (\beta - 1)$;
  Server S_2 **to** S_1: $S_2 \xrightarrow{\mathcal{B}_1 \cup \mathcal{B}'_0} S_1$;
  Server S_1:
  $\mathcal{N}_{next} = \{(node_j.\text{T}_l, E(m_{2j-1}))|2j-1 \in \mathcal{B}_1 \cup \mathcal{B}'_0\}$;
    $\cup \{(node_j.\text{T}_r, E(m_{2j}))|2j \in \mathcal{B}_1 \cup \mathcal{B}'_0\}$;
  $\mathcal{N} = \mathcal{N}_{next}$;
return $\mathcal{R}$;

Algorithm 4 Verification(Candidate set $\mathcal{C}$, Token $E(\mathbf{t}_2)$)

Set $\mathcal{R} = \emptyset$; // Query result
for each $E(\mathbf{z}_i) \in \mathcal{C}$ **do**
  Server S_1:
  Compute $E(\mathbf{z}_i \circ \mathbf{t}_2)$;
  Run the sign computation protocol $\rightarrow S_2 : \text{sign}(\mathbf{z}_i \circ \mathbf{t}_2)$;
  $E(\mathbf{y}_i) \leftarrow E(\mathbf{x}_i + \mathbf{r}_i)$;
  Compute $H_K(\text{id}_j)$;
  $ssk \leftarrow$ decrypt $\text{AES}_{H_K(\text{id}_j)}(ssk)$;
  Compute $\text{AES}_{ssk}(\mathbf{r}_i)$;
  Server S_1 **to** S_2: $S_1 \xrightarrow{E(\mathbf{y}_i), \text{AES}_{ssk}(\mathbf{r}_i)} S_2$;
  Server S_2:
  if $\text{sign}(\mathbf{z}_i \circ \mathbf{t}_2) == -1$ **then**
    $\mathbf{y}_i \leftarrow E(\mathbf{y}_i)$;
    Server S_2 **to** U_j: $S_2 \xrightarrow{\mathbf{y}_i, \text{AES}_{ssk}(\mathbf{r}_i)} U_j$;
    User U_j:
    $\mathbf{r}_i \leftarrow \text{AES}_{ssk}(\mathbf{r}_i)$; $\mathbf{x}_i = \mathbf{y}_i - \mathbf{r}_i$; $\mathcal{R} = \mathcal{R} \cup \{\mathbf{x}_i\}$;
return $\mathcal{R}$;

V. Security Analysis

In this section, we analyze the security of our scheme. As described in our design goal, we aim to preserve the data privacy of healthcare dataset, similarity query requests, and query results, and protect the access pattern of healthcare data from two honest-but-curious cloud servers. Next, we show that our scheme satisfies our design goal.

Theorem 2: Healthcare dataset, query requests, and query results can be kept secret from server S_1.

Proof: We first list the view of S_1 and show why S_1 cannot obtain any information on the plaintext of healthcare dataset, query request, and query results from its views.

- ***View 1:*** encrypted k-d-PB tree $E(\mathrm{T})$. Since $E(\mathrm{T})$ is encrypted by the SHE technique that is proved to be the semantically secure [28], S_1 cannot obtain any information about the healthcare dataset from $E(\mathrm{T})$.
- ***View 2:*** query token $\{E(\mathbf{t}_1), E(\mathbf{t}_2), \mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk)\}$. Since $E(\mathbf{t}_1)$ and $E(\mathbf{t}_2)$ are ciphertexts of SHE technique, the security of SHE technique can guarantee that S_1 cannot deduce any information about $\mathbf{t}_1$ and $\mathbf{t}_2$. Meanwhile, $\mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk)$ is the ciphertext of ssk that can be legitimately accessed by S_1. Thus, S_1 cannot obtain any information about query requests from $\{E(\mathbf{t}_1), E(\mathbf{t}_2), \mathrm{AES}_{H_K(\mathrm{id}_j)}(ssk)\}$.
- ***View 3:*** When searching $E(\mathrm{T})$'s each layer $\mathcal{N} = [node_1, node_2, \cdots, node_{|\mathcal{N}|}]$, S_1 can view each $node_j$'s $E(\mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1))$ and $E(\mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1))$, and the nodes need to be searched in the next layer, i.e.,

$$\mathcal{N}_{next} = \{(node_j.\mathrm{T}_l, E(m_{2j-1}))|2j-1 \in \mathcal{B}_1 \cup \mathcal{B}'_0\} \cup \{(node_j.\mathrm{T}_r, E(m_{2j}))|2j \in \mathcal{B}_1 \cup \mathcal{B}'_0\}.$$

Since each $node_j$'s $E(\mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1))$ and $E(\mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1))$ are SHE technique's ciphertexts, the security of SHE technique can ensure that S_1 has no idea on the plaintext of $\{\mathbf{u}_{j,L}, \mathbf{u}_{j,R}, \mathbf{t}_1\}$. For the nodes in $\mathcal{N}_{next}$, S_1 may try to deduce whether an internal node's left or right subtree satisfies the query request and further deduce the range of the query request. However, on the one hand, to guarantee k-d-PB tree's β-access pattern unlinkability, $\mathcal{N}_{next}$ contains some nodes that do not satisfy the query request but are searched in the next layer. On the other hand, since $E(\mathrm{T})$'s left and right subtrees have been permutated, S_1 cannot match the left and right subtrees of $E(\mathrm{T})$ to that of the original tree T. Hence, S_1 cannot know which node's left or right subtree really satisfies the query request and cannot do the further inference. Thus, healthcare dataset, query request, and query results can be kept secret from S_1. □

Theorem 3: Healthcare dataset, query requests, and query results can be kept secret from server S_2.

Proof: We list the view of S_2 and show why S_2 cannot obtain any information on the plaintext of healthcare dataset, query request, and query results from its view as follows.

- ***View 1:*** When searching $E(\mathrm{T})$'s each layer $\mathcal{N} = [node_1, node_2, \cdots, node_{|\mathcal{N}|}]$, S_2 can view $\{\mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1), \mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1)\}$ and $\{\mathcal{B}_0, \mathcal{B}_1\}$ (i.e., indices to be and not to be searched in the next layer).

$$\begin{cases} \mathcal{B}_0 = \{(k, E(m_k))|m_k \neq 0;\ 1 \leq k \leq 2|\mathcal{N}|\}; \\ \mathcal{B}_1 = \{(k, E(m_k))|m_k = 0;\ 1 \leq k \leq 2|\mathcal{N}|\}. \end{cases}$$

S_2 obtains each $node_j$'s $\{\mathrm{sign}(\mathbf{u}_{j,L} \circ \mathbf{t}_1), \mathrm{sign}(\mathbf{u}_{j,R} \circ \mathbf{t}_1)\}$ through the sign computation protocol. We have proved that the sign computation protocol is privacy-preserving in [28] and can guarantee that S_2 cannot obtain any information about $\{\mathbf{u}_{j,L}, \mathbf{u}_{j,R}, \mathbf{t}_1\}$. For $\{\mathcal{B}_1, \mathcal{B}_0\}$, they have been permutated, S_2 cannot match $\mathcal{B}_0[k]$ or $\mathcal{B}_1[k]$ to the node in the original node array $\mathcal{N}$. In this case, S_2 only knows the number of internal nodes to be or not to be searched in the next layer but does not know which nodes they are.

- ***View 2:*** When verifying each $E(\mathbf{z}_i)$, S_2 can view $\mathrm{sign}(\mathbf{z}_i \circ \mathbf{t}_2)$, $\mathbf{y}_i$, and $\mathrm{AES}_{ssk}(\mathbf{r}_i)$. Since $\mathrm{sign}(\mathbf{z}_i \circ \mathbf{t}_2)$ is obtained through the sign computation protocol, the security of the sign computation protocol can guarantee that S_2 has no idea on $\{\mathbf{z}_i, \mathbf{t}_2\}$. For $\mathbf{y}_i$, since it is in the form of $\mathbf{y}_i = \mathbf{x}_i + \mathbf{r}_i$, the unknownness of $\mathbf{r}_i$ can guarantee that S_2 has no idea on $\mathbf{x}_i$. For $\mathrm{AES}_{ssk}(\mathbf{r}_i)$, the security of AES can guarantee that S_2 cannot recover $\mathbf{r}_i$ from the ciphertext $\mathrm{AES}_{ssk}(\mathbf{r}_i)$ without the secret key ssk. Thus, S_2 cannot obtain any information about data records, query requests, and query results. □

Theorem 4: Our EPSim-AC scheme satisfies k-d-PB tree's β-access pattern unlinkability.

Proof: In our scheme, S_1 stores a k-d-PB tree $E(\mathrm{T})$, and only it has a chance to learn about the access pattern of $E(\mathrm{T})$. However, it cannot break k-d-PB tree's β-access pattern unlinkability. Specifically, in each query, the number of accessed tree paths depends on that of candidate data records (that may satisfy the query request). Without loss of generality, we suppose that there are γ candidate data records, and all of them are located in the last layer of k-d-PB tree. Based on the description of our scheme, when S_1 determines which nodes need to be accessed in the last layer, S_2 will randomly add $\gamma(\beta - 1)$ nodes (not to be searched) to the set of nodes (to be searched). On the one hand, when the tree is small, the number of all $E(\mathrm{T})$'s data records may be less than $\beta * \gamma$. In this case, as described in our scheme, S_2 will add all nodes (not to be searched) to the set of nodes (to be searched). Our scheme will satisfy the full access pattern privacy, let alone β-access pattern unlinkability. On the other hand, when the tree is large, the number of candidate data records will be much smaller than that of all $E(\mathrm{T})$'s data records. S_2 will randomly add $\gamma(\beta - 1)$ nodes (not to be searched) to the set of nodes (to be searched). In this case, γ candidate data records will be accessed with other $\gamma(\beta - 1)$ data records. Since each node corresponds to a tree path, it means that γ tree paths will be accessed with $\gamma(\beta - 1)$ tree paths. Then, one tree path is indistinguishably accessed with $(\beta - 1)$ on average. Thus, our similarity query scheme with access control satisfies k-d-PB tree's β-access pattern unlinkability. □

For a clear description, we give an example to show the probability that S_1 can correctly figure out the access pattern of a query in Example 2.

Example 2: Suppose that a k-d-PB tree is large and the query range is small. The number of candidate data records is γ, and there will be $\gamma(\beta - 1)$ data records (not to be searched) to be accessed with γ candidate records. In this case, S_1 can correctly figure out tree paths of γ candidate records among all tree paths of $\gamma * \beta$ with the probability

$prob = \frac{1}{\binom{\gamma*\beta}{\gamma}}$. When $\beta = 5$ and $\gamma = 10$, we can deduce that

$$prob = \frac{1}{\binom{\gamma*\beta}{\gamma}} = \frac{1}{\binom{50}{10}} = 9.73 \times 10^{-11}.$$

The probability is pretty small, so our EPSim-AC scheme can well preserve the access pattern privacy. □

VI. Performance Evaluation

In this section, we evaluate the computational cost of our EPSim-AC scheme and compare it with an existing similarity query scheme, i.e., SkNN in [16]. The reason why we compare our scheme with SkNN is that it can achieve similarity queries with access pattern privacy and can be adapted to process similarity queries with access control.

SkNN is a secure kNN query scheme and was constructed based on the Pailler homomorphic encryption technique under the model of two servers (i.e., S_1 and S_2). Its main idea is to encrypt each data record $\mathbf{x}_i$ into a ciphertext vector $\mathsf{Paillier.Enc}(\mathbf{x}_i)$ using the Paillier technique and outsource them to S_1. When a user intends to launch a query request with a query record $\mathbf{q}$, it encrypts $\mathbf{q}$ into a ciphertext vector $\mathsf{Paillier.Enc}(\mathbf{q})$ using Paillier technique and sends it to S_1. On receiving $\mathbf{q}$, S_1 linearly checks each $\mathbf{x}_i$ and determines whether $\mathbf{x}_i$ is a kNN of $\mathbf{q}$ with the help of S_2. SkNN can be adapted to process similarity range queries with access control. Specifically, same as our scheme, each $(\mathbf{x}_i, \mathbf{a}_i)$ is represented to a vector $\mathbf{z}_i$ as Eq. (2), and each query request $(\mathbf{q}, \tau, \mathbf{v}_j)$ is represented as a vector $\mathbf{t}_2$ as Eq. (3). Then, $\mathbf{z}_i$ and $\mathbf{t}_2$ are encrypted into two ciphertext vectors $\mathsf{Paillier.Enc}(\mathbf{z}_i)$ and $\mathsf{Paillier.Enc}(\mathbf{t}_2)$. To verify whether $(\mathbf{x}_i, \mathbf{a}_i)$ satisfies the query request $(\mathbf{q}, \tau, \mathbf{v}_j)$, S_1 determines whether $\mathbf{z}_i \circ \mathbf{t}_2 \overset{?}{\leq} 0$.

(1) S_1 uses SkNN's secure multiplication protocol and homomorphic property to calculate $\mathsf{Paillier.Enc}(\mathbf{z}_i \circ \mathbf{t}_2)$. Same as our scheme, S_1 chooses two random numbers $\{r_1, r_2\} \in \{0, 1\}^\lambda$ satisfying $r_1 > r_2 > 0$, where λ is a parameter and much less than Paillier technique's security parameter. Then, S_1 calculates $\mathsf{Paillier.Enc}(r_1(\mathbf{z}_i \circ \mathbf{t}_2) - r_2)$.

(2) S_1 extracts $\mathsf{Paillier.Enc}(\mathbf{x}_i)$ from the first d dimensions of $\mathbf{z}_i$. Then, it chooses a random vector $\mathbf{r}_i$ with the same length as $\mathbf{x}_i$ and computes $\mathsf{Paillier.Enc}(\mathbf{r}_i + \mathbf{x}_i)$. Meanwhile, it encrypts $\mathbf{r}_i$ as $\mathrm{AES}_{ssk}(\mathbf{r}_i)$. After that, S_1 sends $\{\mathsf{Paillier.Enc}(r_1(\mathbf{z}_i \circ \mathbf{t}_2) - r_2), \mathsf{Paillier.Enc}(\mathbf{r}_i + \mathbf{x}_i), \mathrm{AES}_{ssk}(\mathbf{r}_i)\}$ to S_2.

(3) On receiving them, S_2 decrypts $\mathsf{Paillier.Enc}(r_1(\mathbf{z}_i \circ \mathbf{t}_2) - r_2)$ to recover a plaintext $m = r_1(\mathbf{z}_i \circ \mathbf{t}_2) - r_2$. If length$(m) \approx \mathtt{N}$, it means that $\mathbf{z}_i \circ \mathbf{t}_2 \leq 0$, where $\mathtt{N}$ is the modulus of Paillier technique. In this case, S_1 recovers $\mathbf{r}_i + \mathbf{x}_i$ by decrypting $\mathsf{Paillier.Enc}(\mathbf{r}_i + \mathbf{x}_i)$ and returns $\{\mathbf{r}_i + \mathbf{x}_i, \mathrm{AES}_{ssk}(\mathbf{r}_i)\}$ to the query user.

(4) The query user recovers $\mathbf{r}_i$ by decrypting $\mathrm{AES}_{ssk}(\mathbf{r}_i)$ and recovers the query result $\mathbf{x}_i$ as $\mathbf{x}_i = \mathbf{r}_i + \mathbf{x}_i - \mathbf{r}_i$.

We implemented our scheme and adapted SkNN in Java and conducted experiments on a machine with Apple M1 chip, 16 GB RAM, and macOS Big Sur operating system. The parameters of SHE technique are set as $k_0 = 1024$, $k_1 = 40$, and $k_2 = 160$. The security parameter of Paillier technique is set as $\kappa = 512$. Since it is difficult to find a real dataset simultaneously containing data records and attribute values, we evaluated the performance of our scheme on a real medical EEG dataset [30] containing 14980 14-dimensional records and without attribute values. Meanwhile, we appended 4 dimensions to each record as attribute values, and each value is randomly chosen from $\{2, 4, *\}$. In addition, we evaluated the performance of our scheme on a synthetic dataset with 25000 12-dimensional records, where the first 8 dimensions in each record are data values and the last 4 dimensions are attribute values. Each data value is in the range of $[0, 70]$, and each attribute value is chosen from $\{2, 4, *\}$. Each experiment is evaluated 100 times and the average result is reported.

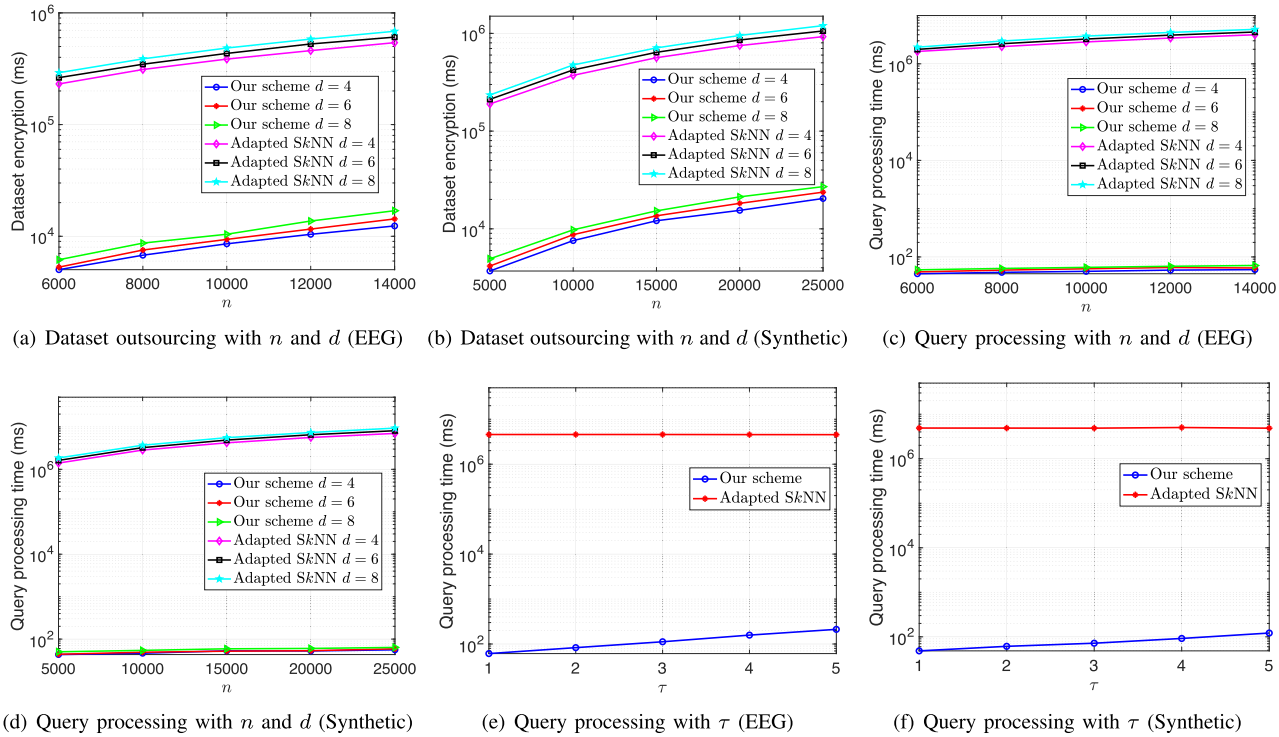
A. Computational Cost of Dataset Outsourcing

In our scheme, the computational cost of dataset outsourcing comes from building and encrypting a k-d-PB tree for the healthcare dataset $\mathcal{X}$, which is related to three parameters: (i) n: the size of the dataset; (ii) d: the dimension of data records; and (iii) l: the dimension of attribute values. Since l has the same effect on the performance of data outsourcing as d. In our experiment, we fix l to 4 and focus on evaluating the computational cost of data outsourcing with n and d.

In Figs. 4(a) and 4(b), we plot the computational cost of our scheme and adapted SkNN's dataset outsourcing varying with n and d over EEG dataset and a synthetic dataset. In this experiment, the parameters are set based on datasets. For EEG dataset, we set $n \in \{6000, 8000, 10000, 12000, 14000\}$ and $d \in \{4, 6, 8\}$. For the synthetic dataset, we set $n \in \{5000, 10000, 15000, 20000, 25000\}$ and $d \in \{4, 6, 8\}$. From Figs. 4(a) and 4(b), we can see that the computational cost of our scheme and adapted SkNN's dataset outsourcing increases with n and d, but our scheme is much more efficient than adapted SkNN. For instance, outsourcing 14000 8-dimensional EEG data records only takes 16939 ms in our scheme while that takes 684516 ms in adapted SkNN.

B. Computational Cost of Query Token Generation

In our scheme, the computational cost of query token generation comes from representing the query request $(\mathbf{q}, \tau, \mathbf{v}_j)$ to two vectors $\{\mathbf{t}_1, \mathbf{t}_2\}$ and encrypting $\{\mathbf{t}_1, \mathbf{t}_2\}$ into ciphertexts. It is closely associated with two parameters: (i) d: the dimension of query record; and (ii) l: the dimension of attribute values. Same as the evaluation of computational cost of dataset outsourcing, we fix l to be 4 and focus on evaluating the computational cost of query token generation varying with d. Since the computational cost of query token generation is not affected by the distribution of the dataset, we only conducted the performance evaluation of query token generation on EEG dataset. In TABLE I, we list the computational cost of query token generation with d, where $d \in \{4, 6, 8\}$. From TABLE I, we can see that the computational cost of query token generation grows with d. This is because when d increases, the dimension of $\mathbf{t}_1$ and $\mathbf{t}_2$ will correspondingly increase as shown in Eq. (3) and Eq. (4). The computational cost of encrypting $\mathbf{t}_1$ and $\mathbf{t}_2$ will become larger. Hence, the query user needs to take more computational cost to generate the query token. Nevertheless, our scheme is still more than 300 times faster than that of the adapted SkNN.



(a) Dataset outsourcing with n and d (EEG) (b) Dataset outsourcing with n and d (Synthetic) (c) Query processing with n and d (EEG)

(d) Query processing with n and d (Synthetic) (e) Query processing with τ (EEG) (f) Query processing with τ (Synthetic)

Fig. 4. Performance evaluation.

TABLE I
COMPUTATIONAL COST OF QUERY TOKEN GENERATION

Dimension d	4	6	8
Our scheme (ms)	0.103	0.119	0.133
Adapted S*k*NN (ms)	38.479	42.977	48.596

TABLE II
COMPUTATIONAL COST OF QUERY RESULT RECOVERY

Dimension d	4	6	8
Our scheme (μs)	3.462	3.503	3.605

C. Computational Cost of Query Processing

In our scheme, the computational cost of query processing comes from searching encrypted k-d-PB tree for query results, which is affected by four parameters: (i) n: the size of the dataset; (ii) d: the dimension of data records; and (iii) l: the dimension of attribute values; and (iv) τ: query range. In our evaluation, we fix l to be 4 and focus on evaluating the performance of our scheme varying with n, d, and τ.

In Figs. 4(c) and 4(d), we plot the computational cost of our scheme and adapted SkNN's query processing varying with n and d over EEG dataset and a synthetic dataset. In this experiment, to guarantee the size of each query result is the same, we set $\tau = 1$. For other parameters, we set them based on datasets. For EEG dataset, we set $n \in \{6000, 8000, 10000, 12000, 14000\}$ and $d \in \{4, 6, 8\}$. For the synthetic dataset, we set $n \in \{5000, 10000, 15000, 20000, 25000\}$ and $d \in \{4, 6, 8\}$. From Figs. 4(c) and 4(d), we can see that the computational cost of our scheme and adapted SkNN's query processing increases with n and d but our scheme is much efficient than adapted SkNN. For instance, processing a query over EEG dataset with 14000 8-dimensional data records takes 66.46 ms in our scheme while that takes 5169080 ms in adapted SkNN.

In Figs. 4(e) and 4(f), we plot the computational cost of our scheme and adapted SkNN's query processing varying with τ over EEG dataset and a synthetic dataset. In this experiment, we set the parameters of EEG dataset as: $n = 14000$, $d = 6$, and $\tau \in \{1, 2, 3, 4, 5\}$. We set the parameters of the synthetic dataset as $n = 15000$, $d = 6$, and $\tau \in \{1, 2, 3, 4, 5\}$. From Figs. 4(e) and 4(f), we can see that the computational cost of our scheme and adapted SkNN's query processing increases with τ. This is because the size of query results increases with the growth of τ. Meanwhile, our scheme is much more efficient than adapted SkNN.

D. Computational Cost of Query Result Recovery

In our scheme, the computational cost of query result recovery is from decrypting $\mathbf{r}_i$ from the ciphertext $\text{AES}_{ssk}(\mathbf{r}_i)$, which is related to the dimension of data records, i.e., d. Since the computational cost of query result recovery in adapted SkNN is the same as that of our scheme, we only conduct the evaluation for our scheme. As shown in TABLE II, we evaluate the computational cost of recovering one $\mathbf{r}_i$ from the corresponding ciphertext $\text{AES}_{ssk}(\mathbf{r}_i)$ varying with d. From TABLE II, we can see that the computational cost increases with d but is pretty efficient. For instance, recovering a 6-dimensional $\mathbf{r}_i$ only takes 3.503 μs.

VII. RELATED WORK

Privacy-preserving similarity query has been extensively studied, and various schemes have been proposed. In this section, we review some of them closely related to our work.

Some privacy-preserving similarity query schemes were designed based on the matrix encryption technique. In 2009, Wong *et al.* [7] proposed an asymmetric scalar-product-preserving encryption (ASPE) scheme and utilized it to construct an efficient kNN query scheme. Based on this scheme, Cao *et al.* [8] proposed a privacy-preserving multi-keyword ranked search scheme, and Wang *et al.* [9] proposed a privacy-preserving multi-keyword fuzzy search scheme. Since the schemes [7]–[9] encrypt data through matrix encryption, they are efficient. However, they have a weak security and cannot resist against known-plaintext attacks, as proved in [31]. To improve security of these schemes, Zhang *et al.* [10] integrated ASPE scheme with Paillier homomorphic encryption technique to design a similarity query scheme but the proposed scheme is secure under a strong assumption that the cloud server has no idea on any plaintext query vector. Recently, Zheng *et al.* [11] proposed a modified ASPE scheme to enhance the security of matrix encryption by introducing more random numbers into ciphertexts and presented a Quadsector tree structure to improve query efficiency. However, these schemes do not consider either access pattern privacy or access control.

Some privacy-preserving similarity query schemes were designed based on public key encryption techniques. In 2013, Rane and Boufounos [12] constructed a secure similarity query scheme using homomorphic encryption techniques. Since the proposed scheme was designed under the model of client and server, the computational cost at the client side is high. To reduce the computational cost of the client side, Zheng *et al.* [2] leveraged a k-d-tree structure to represent data and further proposed an efficient similarity query scheme under the model of two non-collusive servers. However, the scheme in [2] did not take the privacy of access pattern into consideration. To preserve access pattern privacy, Elmehdwi *et al.* [16] used homomorphic encryption technique to design an access-pattern privacy-preserving similarity query scheme. Wu *et al.* [17] integrated Paillier encryption technique and ElGamal technique to propose a secure kNN classification scheme with access pattern privacy. Guan *et al.* [18] proposed an access-pattern privacy-preserving kNN query scheme based on a 2-2 threshold Paillier homomorphic encryption technique. Cui *et al.* [19] leveraged Voronoi diagram to represent dataset and further employed Paillier homomorphic encryption technique to propose an access pattern privacy-preserving similarity query scheme. However, these access pattern privacy-preserving schemes [16]–[19] are computationally expensive and do not take access control into consideration.

Some privacy-preserving approximate similarity query schemes were proposed. Kuzu *et al.* [13] constructed a similarity query scheme using employed locality-sensitive hashing (LSH). Yuan *et al.* [14] constructed a similarity query scheme using symmetric searchable encryption and LSH. Lei *et al.* [15] constructed a location-based similarity query scheme by coding nearby regions but this scheme can only applicable to similarity queries over two-dimensional data. In addition, all of these approximate similarity query schemes can only return approximate query results and do not consider either access pattern privacy or access control.

Different from the above schemes, we design an efficient and privacy-preserving similarity query scheme supporting access control, which has not only high efficiency but also access pattern privacy.

VIII. Conclusion

In this paper, we have proposed an efficient and privacy-preserving similarity query scheme with access control for encrypted healthcare data, which not only preserves k-d-PB tree's β-access pattern unlinkability but also has high efficiency. In this scheme, we first introduced a k-d-PB tree to represent healthcare dataset and designed an efficient k-d-PB tree based similarity query algorithm with access control. Second, we defined a kind of weakened access pattern privacy, called k-d-PB tree's β-access pattern unlinkability. Then, we preserved the privacy of similarity queries with access control through SHE technique and proposed our EPSim-AC scheme. In our future work, we will explore to implement similarity queries with access control by designing new data structures and privacy-preserving protocols.

References

[1] Y. Zheng and R. Lu, "Efficient privacy-preserving similarity range query based on pre-computed distances in eHealthcare," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[2] 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, pp. 123:1–123:13, May 2019.

[3] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient privacy-preserving similarity range query with Quadsector tree in eHealthcare," *IEEE Trans. Services Comput.*, early access, May 18, 2021, doi: 10.1109/TSC.2021.3081350.

[4] M. Lacharité, B. Minaud, and K. G. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in *Proc. IEEE Symp. Secur. Privacy (SP)*. San Francisco, CA, USA: IEEE Computer Society, May 2018, pp. 297–314.

[5] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Denver, CO, USA, Oct. 2015, pp. 668–679.

[6] J. Ning, G. S. Poh, X. Huang, R. Deng, S. Cao, and E.-C. Chang, "Update recovery attacks on encrypted database within two updates using range queries leakage," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 12, 2020, doi: 10.1109/TDSC.2020.3015997.

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

[8] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[9] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 2112–2120.

[10] Z. Zhang, K. Wang, C. Lin, and W. Lin, "Secure top-k inner product retrieval," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 77–86.

[11] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient privacy-preserving similarity range query with Quadsector tree in eHealthcare," *IEEE Trans. Services Comput.*, early access, May 18, 2021, doi: 10.1109/TSC.2021.3081350.

[12] 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.

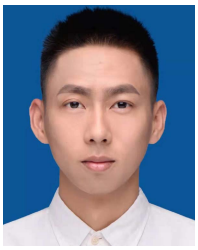
[13] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. ICDE*, Apr. 2012, pp. 1156–1167.

[14] X. Yuan, H. Cui, X. Wang, and C. Wang, "Enabling privacy-assured similarity retrieval over millions of encrypted records," in *Proc. ESORICS*, 2015, pp. 40–60.

[15] X. Lei, A. X. Liu, R. Li, and G.-H. Tu, "SecEQP: A secure and efficient scheme for SkNN query problem over encrypted geodata on cloud," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 662–673.
[16] 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.*, Mar. 2014, pp. 664–675.
[17] 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.
[18] Y. Guan, R. Lu, Y. Zheng, J. Shao, and G. Wei, "Toward oblivious location-based k-nearest neighbor query in smart cities," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14219–14231, Sep. 2021, doi: 10.1109/JIOT.2021.3068859.
[19] N. Cui, X. Yang, B. Wang, J. Li, and G. Wang, "SVkNN: Efficient secure and verifiable k-nearest neighbor query on the cloud platform*," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 253–264.
[20] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.
[21] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 94–105, May 2018.
[22] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proc. 39th Annu. Symp. Found. Comput. Sci.*, Milwaukee, WI, USA, Oct. 1995, pp. 41–50.
[23] E. Stefanov *et al.*, "Path ORAM: An extremely simple oblivious RAM protocol," *J. ACM*, vol. 65, no. 4, pp. 18:1–18:26, Apr. 2018.
[24] Z. Zhang, K. Wang, W. Lin, A. W.-C. Fu, and R. C.-W. Wong, "Practical access pattern privacy by combining PIR and oblivious shuffle," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1331–1340.
[25] E. Mrema and V. Kumar, "Fine grained attribute based access control of healthcare data," in *Proc. 12th Int. Symp. Med. Inf. Commun. Technol. (ISMICT)*, Mar. 2018, pp. 1–5.
[26] 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, pp. 17:1–17:27, 2017.
[27] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving O($\log^3$n) communication-efficient privacy-preserving range query in fog-based IoT," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5220–5232, Jun. 2020.
[28] 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.*, early access, Feb. 23, 2021, doi: 10.1109/TDSC.2021.3061611.
[29] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Toward privacy-preserving cybertwin-based spatiotemporal keyword query for ITS in 6G era," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16243–16255, Nov. 2021.
[30] *EEG Data Set*. Accessed: Jun. 1, 2021. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State
[31] W. Lin, K. Wang, Z. Zhang, and H. Chen, "Revisiting security risks of asymmetric scalar product preserving encryption and its variants," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1116–1125.

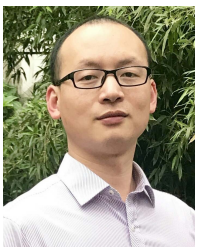
Yandong Zheng received the M.S. degree from the Department of Computer Science, Beihang University, China, in 2017. She is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Canada. Her research interests include cloud computing security, big data privacy,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

Rongxing Lu (Fellow, IEEE) is an Associate Professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. His research interests include applied cryptography, privacy enhancing technologies, and the IoT-big data security and privacy. He was the Winner of the 2016–2017 Excellence in Teaching Award from the FCS, UNB. He was the recipient of nine best (student) paper awards from some reputable journals and conferences. Currently, he serves as the Chair for the IEEE ComSoc Communications and Information Security Technical Committee (CIS-TC), and the Founding Co-Chair for the IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC).



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

Songnian Zhang received the M.S. degree from Xidian University, China, in 2016. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include cloud computing security, big data query, and query privacy.



Jun Shao (Member, IEEE) received the Ph.D. degree from the Department of Computer and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Post-Doctoral Fellow with the School of Information Sciences and Technology, The Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include networks security and applied cryptography.

Hui Zhu (Senior Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, China, in 2005, and the Ph.D. 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.