Efficient and Privacy-Preserving Similarity Range Query Over Encrypted Time Series Data

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

Abstract—Similarity query over time series data plays a significant role in various applications, such as signal processing, speech recognition, and disease diagnosis. Meanwhile, driven by the reliable and flexible cloud services, encrypted time series data are often outsourced to the cloud, and as a result, the similarity query over encrypted time series data has recently attracted considerable attention. Nevertheless, existing solutions still have issues in supporting similarity queries over time series data with different lengths, query accuracy and query efficiency. To address these issues, in this article, we propose a new efficient and privacy-preserving similarity range query scheme, where the time warp edit distance (TWED) is used as the similarity metric. Specifically, we first organize time series data into a kd-tree by leveraging TWED's triangle inequality, and design an efficient similarity range query algorithm for the kd-tree. Second, based on a symmetric homomorphic encryption technique, we carefully devise a suite of privacy-preserving protocols to provide a security guarantee for kd-tree based similarity range queries. After that, by using the similarity range query algorithm and these protocols, we propose our privacy-preserving similarity range query scheme, in which we elaborate on two strategies to make our scheme resist against the cloud inference attack. Finally, we analyze the security of our scheme and conduct extensive experiments to evaluate its performance, and the results indicate that our proposed scheme is indeed privacy-preserving and efficient.

Index Terms—Time series data, time warp edit distance, similarity range query, triangle inequality, kd-tree

1 Introduction

Similarity query over time series data, which aims to identify samples that are similar to the sample of interest, has a significant number of applications in various areas, such as signal processing, speech recognition, and disease diagnosis [1]. For example, the similarity query over time series ECG data can be used to detect Premature Ventricular Contraction disease [2]. Undoubtedly, the wide application has made the similarity query over time series data popular. Nevertheless, as the rapid development of the Internet of Things (IoT), tremendous volumes of data are generated with high velocity [3]. As reported in [4], the total amount of data will reach 149 zettabytes by 2024. Such tremendous volumes of data will seriously affect the efficiency of the similarity query. To improve query efficiency, many data owners migrate their data to the computationally powerful cloud and delegate the cloud server to perform similarity queries. However, since the data are private assets of the data owners and the cloud server is not fully trusted, exposing the plaintext data to the cloud server may inflict severe economic loss to the data owners. For addressing the privacy issue, data owners usually leverage encryption techniques to encrypt the data before outsourcing them to the cloud, yet a consensus has emerged that encryption techniques will hinder the cloud server to perform the similarity queries over the outsourced time series data.

To tackle the dilemma of similarity queries over encrypted time series data, various schemes [2], [5], [6], [7], [8], [9], [10], [11] were proposed. Based on the similarity metric, existing schemes can be divided into three categories, i.e., L_p-norm based similarity query schemes [5], [6], [7], [8], [9], [10], [11], edit distance based similarity query schemes [12], [13], [14], and dynamic time warping (DTW) based similarity query schemes [2]. However, L_p-based schemes [5], [7], [8], [9], [10] cannot support similarity queries over time series data with different lengths. Although edit distance and DTW distance based schemes [2], [12], [13], [14] can support similarity queries over time series data with different lengths, the schemes [2], [12] can only return approximate query results and the schemes [13], [14] suffer from the linear search efficiency. Hence, existing schemes still have issues in supporting similarity queries over time series data with different lengths, query accuracy and query efficiency.

To solve the above challenges, in this paper, we propose an efficient and privacy-preserving similarity range query scheme for time series data, where time warp edit distance (TWED) (defined in Section 3.1) is deployed as the similarity metric. Similar to edit distance and DTW distance, TWED can support similarity queries over time series data with different lengths. Meanwhile, TWED computation involves the absolute value computation and minimum value computation as described in Section 3.1. Although these basic operations can be supported by fully homomorphic encryption

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

Manuscript received 7 Aug. 2020; revised 13 Dec. 2020; accepted 13 Feb. 2021. Date of publication 23 Feb. 2021; date of current version 9 July 2022.
(Corresponding author: Rongxing Lu.)
Digital Object Identifier no. 10.1109/TDSC.2021.3061611

techniques [15] in a non-interactive and privacy-preserving way, the consequent computational cost is too high. Thus, we consider achieving TWED computation and the corresponding similarity range queries by using an efficient symmetric homomorphic encryption (SHE) technique [16] in a two-server model. The key idea is that we first organize time series data into a kd-tree by leveraging TWED's triangle inequality, and design an efficient similarity range query algorithm for the kd-tree. Then, we focus on designing privacy-preserving techniques to preserve the privacy of kd-tree based similarity range queries. Specifically, the main contributions of this paper are three folds as follows.

- First, based on an SHE technique [16], we carefully devise a suite of privacy-preserving protocols including privacy-preserving sign computation protocol, minimum value computation protocol, absolute value computation protocol, and bootstrapping protocol.
- Second, based on the kd-tree based similarity range query algorithm and the above-mentioned protocols, we propose our privacy-preserving similarity range query scheme in a two-server model, where server 1 and server 2 respectively store encrypted data and the corresponding secret keys. Meanwhile, server 1 needs to be granted the encrypted data comparison capability. Note that, in such a model, server 1 may launch the *cloud inference attack* (defined in Section 2.2). That is, when a server has a collection of encrypted data, the capability to compare encrypted data, as well as the capability to construct the ciphertext for any plaintext, it can infer the plaintexts of all encrypted data by comparing the encrypted data with some ciphertexts having known plaintexts. To resist against the cloud inference attack, we elaborate on the following two strategies.

(1) We design different encryption methods for kd-tree's internal nodes and leaf nodes such that each of them does not allow server 1 to simultaneously have the capability to compare encrypted data and the capability to construct the ciphertext for any plaintext.

(2) Based on the above constrained encryption methods, we carefully design different search methods for the kd-tree's internal nodes and leaf nodes such that server 1 and server 2 can accomplish similarity range queries in a privacy-preserving way.

- Third, we analyze the security of our scheme and conduct experiments to evaluate its performance. The results show that our scheme is privacy-preserving and can resist against the cloud inference attack. Also, it is efficient in terms of computational costs and communication overhead. Especially, the security of our scheme depends on the security of the SHE technique [16] but the SHE technique was only proved to be known-plaintext attacks (KPA) secure. In this work, as one of our contributions, we prove that the SHE technique is chosen-plaintext attacks (CPA) secure, and it can provide a strong security guarantee for our scheme.

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

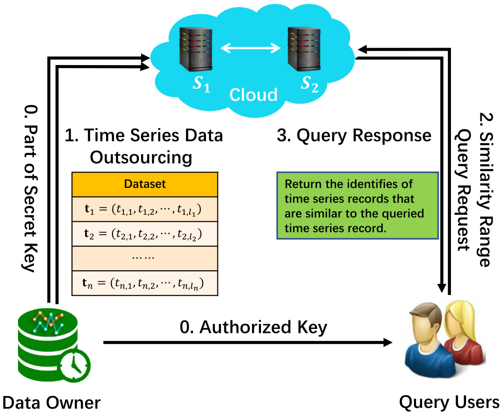

S_1
Cloud
S_2
0. Part of Secret Key
1. Time Series Data Outsourcing
Dataset
$\mathbf{t}_1 = (t_{1,1}, t_{1,2}, \cdots, t_{1,l_1})$
$\mathbf{t}_2 = (t_{2,1}, t_{2,2}, \cdots, t_{2,l_2})$
……
$\mathbf{t}_n = (t_{n,1}, t_{n,2}, \cdots, t_{n,l_n})$
3. Query Response
Return the identifies of time series records that are similar to the queried time series record.
2. Similarity Range Query Request
0. Authorized Key
Data Owner
Query Users

Fig. 1. System model under consideration.

2 Models and Design Goals

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

2.1 System Model

In our system model, we consider a cloud-assisted similarity range query model over time series data, which involves a data owner, a cloud with two servers $\{S_1, S_2\}$, and multiple query users $\mathcal{U} = \{U_1, U_2, \ldots\}$, as shown in Fig. 1.

- Data Owner: The data owner has a time series dataset with n time series records, i.e., $\mathcal{T} = \{\mathbf{t}_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,l_i})| i = 1, 2, \ldots, n\}$, where l_i is the length of $\mathbf{t}_i$ for $i = 1, 2, \ldots, n$. Each $\mathbf{t}_i \in \mathcal{T}$ has a unique identity id_i for $i = 1, 2, \ldots, n$. Meanwhile, we assume that all values in the dataset are integers. If they are not integers originally, we can easily convert them into integers. To make full use of these data, the data owner is willing to offer a similarity range query service to some users. Since the data owner only has limited computing capability and storage space, it outsources the dataset $\mathcal{T}$ to a cloud, and employs the cloud to offer the similarity range query service to users. To preserve the privacy of the data, the data owner encrypts the dataset $\mathcal{T}$ before outsourcing them to the cloud.
- Cloud with Two Servers $\{S_1, S_2\}$: The cloud has two servers $\{S_1, S_2\}$, and both of them have powerful computing capability and abundant storage space. The cloud is responsible for storing the encrypted time series data for the data owner and offering the similarity range query service to query users. Suppose that $(\mathbf{q}, \delta)$ is a query request, where $\mathbf{q}$ is a time series record and δ is a distance threshold. Then, on receiving $(\mathbf{q}, \delta)$, the cloud will search the encrypted dataset $E(\mathcal{T})$ to find out the time series records whose distance to $\mathbf{q}$ is equal to or less than δ, i.e., $\{\mathbf{t}_i | d(\mathbf{q}, \mathbf{t}_i) \leq \delta\}$, where $d(\cdot, \cdot)$ denotes the TWED distance (defined in Section 3.1). Finally, the cloud will return the identities of these records to the query user, i.e., $\{\mathrm{id}_i | d(\mathbf{q}, \mathbf{t}_i) \leq \delta\}$.
- Query Users $\mathcal{U} = \{U_1, U_2, \ldots\}$: The system has a set of query users $\mathcal{U} = \{U_1, U_2, \ldots\}$. When they register in the system, the data owner authorizes them with an authorized key as shown in Fig. 1. After the authorization, the query users can enjoy the similarity range query service.

2.2 Security Model

In our security model, the data owner is *trusted* as it is the initiator of the entire system. For the query users, they are assumed to be *honest* and are not allowed to collude with S_1. That is, they will faithfully launch the similarity query requests to the cloud servers. As for the cloud servers $\{S_1, S_2\}$, both of them are considered to be *honest-but-curious*. They will faithfully store the encrypted data for the data owner and offer the similarity query service for query users. However, they may be curious about some private information including the plaintexts of the time series data, query requests, as well as query results. Moreover, we assume that there is no collusion between S_1 and S_2 because different cloud service providers may have conflicts of interest [10]. In addition, S_1 is responsible for storing the encrypted data. To obtain the plaintexts of the dataset, S_1 may launch *cloud inference attack*, in which a cloud server seeks to infer the plaintexts of the encrypted data. In this attack, the cloud server is assumed to have a collection of encrypted data, the capability to compare these encrypted data, and the capability to construct the ciphertext for any plaintext. Then, it can launch the cloud inference attack to infer the plaintext m of a ciphertext $E(m)$ as follows.

(1) The cloud server chooses a random number a. Then, it constructs the ciphertext of $i * a$, i.e., $E(i * a)$, and compares $E(m)$ with $E(i * a)$ for $i = \{0, 1, 2, \ldots\}$ until it finds an i satisfying $i * a \le m < (i + 1) * a$.

(2) The cloud server continues to construct $E(i * a + j)$ and compare $E(m)$ with $E(i * a + j)$ for $j = \{0, 1, \ldots, a - 1\}$ until it finds a j satisfying $m = i * a + j$. In this case, the cloud server obtains the plaintext $m = i * a + j$.

It is worth noting that it is easy for the cloud server to obtain the ciphertext of any plaintext data when the encryption technique is a public key encryption technique or a symmetric encryption with homomorphic properties. For the public key encryption technique, the cloud server can encrypt any data with the public key. For the symmetric encryption with homomorphic properties, when the cloud server obtains a ciphertext $E(1)$, it can use homomorphic properties to generate $E(m)$ for any plaintext m.

2.3 Design Goals

In this work, our goal is to design an efficient and privacy-preserving similarity range query scheme for time series data. Specifically, the following objectives should be satisfied.

- *Privacy preservation:* The basic requirement of our proposed scheme is privacy preservation. We aim to not only preserve the privacy of the outsourced time series dataset, query requests, as well as query results, but also resist against the cloud inference attack.
- *Efficiency:* We also aim to minimize the computational cost of similarity range queries and improve query efficiency as much as possible.

3 Preliminaries

In this section, we define the TWED-based similarity range query, and briefly review kd-tree and the SHE technique.

3.1 TWED-Based Similarity Range Query

In this subsection, we formally define time warp edit distance, and then introduce the concept of TWED-based similarity range query. TWED was proposed in [17] and it is a distance metric for time series data. In [17], the authors experimentally proved that TWED is quite effective compared with Edit Distance, DTW, Longest Common Subsequence (LCSS), and Edit Distance with Real Penalty (ERP). Formally, TWED can be defined as follows.

Definition 1 (Time warp edit distance). *Suppose that* $\mathbf{t} = (t_1, \ldots, t_{l_t})$ *and* $\mathbf{q} = (q_1, \ldots, q_{l_q})$ *are two time series records, where* l_t *and* l_q *are respectively the lengths of* $\mathbf{t}$ *and* $\mathbf{q}$. *Meanwhile, let* $\mathbf{t}|_1^i = (t_1, \ldots, t_i)$, *and* $\mathbf{q}|_1^j = (q_1, \ldots, q_j)$ *be the subsequences of* $\mathbf{t}$ *and* $\mathbf{q}$. *Then, the TWED between* $\mathbf{t}|_1^i$ *and* $\mathbf{q}|_1^j$ *can be computed as follows.*

$$d(\mathbf{t}|_1^i, \mathbf{q}|_1^j) = \min \begin{cases} d(\mathbf{t}|_1^{i-1}, \mathbf{q}|_1^j) + \Gamma(t_i \rightarrow \Lambda) \\ d(\mathbf{t}|_1^{i-1}, \mathbf{q}|_1^{j-1}) + \Gamma(t_i \rightarrow q_j) \\ d(\mathbf{t}|_1^{i}, \mathbf{q}|_1^{j-1}) + \Gamma(\Lambda \rightarrow q_j) \end{cases},$$

with

$$\begin{aligned} \Gamma(t_i \rightarrow \Lambda) &= |t_i - t_{i-1}| + \lambda \\ \Gamma(t_i \rightarrow q_j) &= |t_i - q_j| + |t_{i-1} - q_{j-1}| \\ \Gamma(\Lambda \rightarrow q_j) &= |q_j - q_{j-1}| + \lambda, \end{aligned}$$

where λ *is a mismatch penalty parameter. Then, the distance between* $\mathbf{t}$ *and* $\mathbf{q}$ *will be* $d(\mathbf{t}, \mathbf{q}) = d(\mathbf{t}|_1^{l_t}, \mathbf{q}|_1^{l_q})$.

TWED is a metric [17] and satisfies the triangle inequality as proved in [18]. Specifically, given three time series records $\mathbf{t}$, $\mathbf{p}$ and $\mathbf{q}$, we have $|d(\mathbf{t}, \mathbf{p}) - d(\mathbf{q}, \mathbf{p})| \le d(\mathbf{t}, \mathbf{q})$.

Algorithm 1. TWED(Record $\mathbf{t}$, Record $\mathbf{q}$)

```
Initialize an $l_t \times l_q$ matrix D
$\mathrm{D}[1,1] = |t_1 - q_1|$;
for $i = 2$ to $l_t$ do
    $\mathrm{D}[i,1] = \mathrm{D}[i-1,1] + |t_i - t_{i-1}| + \lambda$
for $j = 2$ to $l_q$ do
    $\mathrm{D}[1,j] = \mathrm{D}[1,j-1] + |q_j - q_{j-1}| + \lambda$
for $i = 2$ to $l_t$ do
    for $j = 2$ to $l_q$ do
        $d_1 = \mathrm{D}[i-1,j] + |t_i - t_{i-1}| + \lambda$.
        $d_2 = \mathrm{D}[i-1,j-1] + |t_i - q_j| + |t_{i-1} - q_{j-1}|$.
        $d_3 = \mathrm{D}[i,j-1] + |q_j - q_{j-1}| + \lambda$.
        $\mathrm{D}[i,j] = \min\{d_1, d_2, d_3\}$.
return $\mathrm{D}[l_t, l_q]$
```

TEWD Computation Algorithm. TWED between two series records can be computed by a dynamic programming algorithm. As shown in Algorithm 1, to compute $d(\mathbf{t}, \mathbf{q})$, the dynamic programming algorithm will incrementally compute a distance matrix $\mathrm{D}_{l_t \times l_q}$ step by step, where each $\mathrm{D}[i,j]$ denotes the TWED between the subsequences $\mathbf{t}|_1^i$ and $\mathbf{q}|_1^j$ for $i = 1, 2, \ldots, l_t$ and $j = 1, 2, \ldots, l_q$. Thus, based on the definition, we have $d(\mathbf{t}, \mathbf{q}) = \mathrm{D}[l_t, l_q]$.

Remark. To facilitate the description, the TWED considered in this work is slightly different from that in [17]. Specifically, we remove the time elasticity parameter when computing TWED. Actually, our proposed scheme can be easily extended to support the TWED-based similarity range query with time elasticity.

Based on TWED, we can define the TWED-based similarity range query. Suppose that $\mathcal{T} = \{\mathbf{t}_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,l_i}) | i = 1, 2, \ldots, n\}$ is a time series dataset, and each $\mathbf{t}_i$ has a unique identity id_i. Then, the TWED-based similarity range query can be formally defined as follows.

Definition 2 (TWED-based similarity range query). *Let $(\mathbf{q}, \delta)$ be a query request, where $\mathbf{q} = (q_1, q_2, \ldots, q_{l_q})$ is a query record and δ is a distance threshold. Then, the TWED-based similarity range query is to search the dataset $\mathcal{T}$ for the time series records, whose distance to $\mathbf{q}$ is equal to or less than δ, i.e., $\{\mathrm{id}_i | d(\mathbf{q}, \mathbf{t}_i) \leq \delta\}$.*

3.2 kd-Tree

kd-tree [19] is a tree-based index and is designed to organize multi-dimensional data. It has two types of nodes, i.e., internal nodes and leaf nodes. Each internal node has a cutting dimension cd, a cutting value val, a left subtree T_l and a right subtree T_r. Each leaf node contains a data record. The kd-tree is built by recursively cutting the dataset. Specifically, when building an internal node, we need to first choose a cutting dimension cd among all dimensions and use the median of all data records' values in the cdth dimension as the cutting value val. Then, based on cd and val, we cut the dataset into two subdatasets. One subdataset contains the data records whose values in the cdth dimension are less than val, which is used to build a left subtree T_l. The other subdataset contains the data records whose values in the cdth dimension are equal to or larger than val, which is used to build a right subtree T_r. Furthermore, we can use $\{cd, val, \mathrm{T}_l, \mathrm{T}_r\}$ to build an internal node.

Range Queries Over kd-Tree. The kd-tree can efficiently support range queries. Suppose that $\mathcal{V} = \{\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,k}) |\ i = 1, 2, \ldots, n\}$ is a k-dimensional dataset, and it is represented to be a kd-tree T. Let $Q = ([v_1^l, v_1^r], [v_2^l, v_2^r], \ldots, [v_k^l, v_k^r])$ denote a range query, where v_i^l and v_i^r are respectively the lower bound and upper bound of the ith query range for $i = 1, 2, \ldots, k$. Then, the searcher can efficiently search the kd-tree T to find out the data records within Q. The search algorithm includes two stages, i.e., filtration and verification as shown in Algorithm 2.

Algorithm 2. RangeQuery(Tree T, Query Range Q)

```
// Filtration stage
$\mathcal{C} = \emptyset$; // Initialize the candidate result
$\mathcal{C} = \mathrm{Filtration}(\mathrm{T}.root, Q)$;
// Verification stage
for each $\mathbf{v}_i \in \mathcal{C}$ do
    if $\mathbf{v}_i \in Q$ then
        $Result = Result \cup \{\mathbf{v}_i\}$;
return $Result$;
```

• *Filtration stage:* In the filtration stage, the searcher recursively searches T for a set of candidate records $\mathcal{C}$ that are probably within Q, as shown in Algorithm 3. When the searched node is a leaf node, the searcher adds the node's data record into the candidate set $\mathcal{C}$. When the searched node is an internal node with $\{cd, val, \mathrm{T}_l, \mathrm{T}_r\}$, the searcher needs to check the relationship between the query range in the cdth dimension (i.e., $[v_{cd}^l, v_{cd}^r]$) and val. Based on their relationship, the searcher determines whether to search the left subtree and right subtree as follows.

Algorithm 3. Filtration(Node $node$, Query Range Q)

```
if $node$ is a leaf node then
    Add $node$'s data record into $\mathcal{C}$;
else
    if $v_{cd}^l < val$ then
        Filtration($node.\mathrm{T}_l$, $Q$);
    if $v_{cd}^r \geq val$ then
        Filtration($node.\mathrm{T}_r$, $Q$);
return $\mathcal{C}$;
```

(1) When $v_{cd}^l < val$, the searcher continues to search the left subtree. This is because all data records' values in the left subtree are less than val in the cdth dimension. Then, "$v_{cd}^l < val$" means that Q interacts with the left subtree, so the left subtree needs to be searched.

(2) When $v_{cd}^r \geq val$, the searcher continues to search the right subtree. This is because all data records' values in the right subtree are equal to or larger than val in the cdth dimension. Then, "$v_{cd}^r \geq val$" means that Q interacts with the right subtree, so the right subtree needs to be searched.

• *Verification stage:* In the verification stage, the searcher further verifies whether each candidate data record in $\mathcal{C}$ is within Q or not. As shown in Algorithm 2, if $\mathbf{v}_i \in \mathcal{C}$ is within Q, the searcher adds $\mathbf{v}_i$ into the query result, i.e., $Result = Result \cup \{\mathbf{v}_i\}$.

3.3 The SHE Technique

The SHE technique is a symmetric homomorphic encryption technique and can efficiently support homomorphic addition and multiplication [16]. The SHE technique $\Pi_{\mathsf{SHE}} = (\mathsf{SHEKeyGen}, \mathsf{SHEEnc}, \mathsf{SHEDec})$ can be defined as follows.

• $\mathsf{SHEKeyGen}(k_0, k_1, k_2)$: Given security parameters $\{k_0, k_1, k_2\}$ satisfying $k_1 \ll k_2 < k_0$, the key generation algorithm first chooses two large prime numbers p, q with $|p| = |q| = k_0$, and sets $\mathcal{N} = pq$. Then, it selects a random number $\mathcal{L}$ with $|\mathcal{L}| = k_2$. Finally, the algorithm outputs the public key $pk = (k_0, k_1, k_2, \mathcal{N})$, the secret key $sk = (p, \mathcal{L})$, and the message space $\mathcal{M} = \{m | m \in (-2^{k_1}, 2^{k_1})\}$.

• $\mathsf{SHEEnc}(sk, m)$: With the secret key sk, a message $m \in \mathcal{M}$ is encrypted as $E(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.

• $\mathsf{SHEDec}(sk, E(m))$: Given sk and $E(m)$, the decryption algorithm recovers a message m' as $m' = (E(m) \bmod p) \bmod \mathcal{L} = (m + \mathcal{L}) \bmod\ \mathcal{L}$. If $m' < \frac{\mathcal{L}}{2}$, it means that $m \geq 0$, and we have $m = m'$. Otherwise, if $m' > \frac{\mathcal{L}}{2}$, it means that $m < 0$, and we have $m = m' - \mathcal{L}$.

The SHE technique satisfies the homomorphic addition and multiplication properties as follows.

- *Homomorphic addition-I:* Two ciphertexts $E(m_1)$ and $E(m_2)$ satisfy that $E(m_1) + E(m_2) \rightarrow E(m_1 + m_2)$.
- *Homomorphic multiplication-I:* Two ciphertexts $E(m_1)$ and $E(m_2)$ satisfy $E(m_1) * E(m_2) \rightarrow E(m_1 * m_2)$.
- *Homomorphic addition-II:* A ciphertext $E(m_1)$ and a plaintext m_2 satisfy $E(m_1) + m_2 \rightarrow E(m_1 + m_2)$.

- *Homomorphic multiplication-II:* A ciphertext $E(m_1)$ and a plaintext m_2 satisfy $E(m_1) * m_2 \to E(m_1 * m_2)$.

Discussion About SHE's Multiplicative Depth. The SHE technique is a leveled fully homomorphic encryption. It supports an unbounded number of homomorphic addition-Is, addition-IIs and multiplication-IIs. However, it can only support a limited number of homomorphic multiplication-Is. This is because homomorphic multiplication-Is will increase the random number in the ciphertext and too many homomorphic multiplication-Is will result in the incorrectness of decryption. Suppose that $E(m_1) = (r_1\mathcal{L} + m_1)(1 + r_1'p) \bmod \mathcal{N}$ and $E(m_2) = (r_2\mathcal{L} + m_2)(1 + r_2'p) \bmod \mathcal{N}$, where $r_1, r_2 \in \{0,1\}^{k_2}$. Then, we have

$$\begin{aligned} E(m_1 * m_2) &= E(m_1) * E(m_2) \\ &= (r_1\mathcal{L} + m_1)(1 + r_1'p)(r_2\mathcal{L} + m_2)(1 + r_2'p) \bmod \mathcal{N} \\ &= m_1 * m_2 + r_1r_2\mathcal{L}^2 + r_1\mathcal{L}m_1 + r_2\mathcal{L}m_2 + \Delta * p, \end{aligned}$$

where "Δ" denotes the coefficient of p. Then, we consider decrypting the ciphertext $E(m_1 * m_2)$ as

$$\begin{aligned} &(E(m_1 * m_2) \bmod p) \bmod \mathcal{L} \\ &= ((m_1 * m_2 + r_1r_2\mathcal{L}^2 + r_1\mathcal{L}m_1 + r_2\mathcal{L}m_2 + \Delta * p) \bmod p) \bmod \mathcal{L} \\ &= ((m_1 * m_2 + r_1r_2\mathcal{L}^2 + r_1\mathcal{L}m_1 + r_2\mathcal{L}m_2) \bmod p) \bmod \mathcal{L}. \end{aligned}$$

In this case, the decryption is correct *iff* $m_1 * m_2 < \mathcal{L}$, $m_1 * m_2 + r_1r_2\mathcal{L}^2 + r_1\mathcal{L}m_1 + r_2\mathcal{L}m_2 < p$. Since $m_1, m_2 \in \{0,1\}^{k_1}$, $r_1, r_2, \mathcal{L} \in \{0,1\}^{k_2}$ and $k_1 \ll k_2$, we have $m_1 * m_2 + r_1\mathcal{L}m_1 + r_2\mathcal{L}m_2 \ll r_1r_2\mathcal{L}^2$. Thus, we just need to guarantee that $r_1r_2\mathcal{L}^2 < p$. Since the length of $r_1r_2\mathcal{L}^2$ is at most $4k_2$ and the length of p is k_0, if $4k_2 < k_0$, we can guarantee that $r_1r_2\mathcal{L}^2 < p$. Similarly, when the required multiplicative depth is θ (i.e., the maximal number of consecutive multiplications), the parameters should satisfy that $2(\theta+1)k_2 < k_0$. In other words, given the parameters $\{k_0, k_2\}$, the maximum multiplicative depth of the SHE technique is $\theta = \lfloor \frac{k_0}{2k_2} - 1 \rfloor$.

Remark 1. SHE's homomorphic properties enable us to encrypt data with ciphertexts. Given $\{E(0), E(1), E(-1)\}$, a message $m \in \mathcal{M}$ can be encrypted as

$$E(m) = \begin{cases} m * E(1) + r * E(0) & m \geq 0 \\ |m| * E(-1) + r * E(0) & m < 0, \end{cases} \tag{1}$$

where $r \in \{0,1\}^{k_2}$ is a random number for further randomizing the ciphertext $E(m)$.

Remark 2. The SHE technique can resist against KPA as proved in [16]. If we deploy the SHE technique as the encryption technique, server S_1 can obtain the ciphertexts of any plaintexts with homomorphic properties. In this case, S_1 has the capability of launching CPA. To keep the data secret from S_1, we demand to deploy a homomorphic encryption scheme, which is semantically secure under CPA. Luckily, we find that the SHE technique can be proved to be CPA-secure. The detailed proof is shown in Section 5.1.

4 Our Proposed Scheme

In this section, we first introduce an efficient similarity range query algorithm for kd-tree and present some privacy-preserving protocols. Then, based on these building blocks, we present an efficient and privacy-preserving similarity range query scheme for time series data.

4.1 Efficient Similarity Range Query Algorithm

In this subsection, we introduce an efficient similarity range query algorithm for kd-tree. First, we discuss on how to organize time series data into a kd-tree, and then describe the query algorithm in detail.

Organize Time Series Data into a kd-Tree. Suppose that $\mathcal{T} = \{\mathbf{t}_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,l_i}) | i = 1, 2, \ldots, n\}$ is a time series dataset. To build a kd-tree for $\mathcal{T}$, we first choose k time series records $\mathcal{P} = \{\mathbf{p}_j | \mathbf{p}_j = (p_{j,1}, p_{j,2}, \ldots, p_{j,l_j}) | j = 1, 2, \ldots, k\}$ as pivots from the dataset $\mathcal{T}$, where l_j is the length of $\mathbf{p}_j$ for $j = 1, 2, \ldots, k$. Then, based on $\mathcal{T}$ and $\mathcal{P}$, we build a kd-tree T as follows.

- *Step 1*: For each $\mathbf{t}_i \in \mathcal{T}$, we compute its TWEDs with k pivots as $\mathbf{v}_i = (d(\mathbf{t}_i, \mathbf{p}_1), \ldots, d(\mathbf{t}_i, \mathbf{p}_k))$ for $i = 1, 2, \ldots, n$. Then, these distances can form a k-dimensional distance dataset $\mathcal{V} = \{\mathbf{v}_i = (d(\mathbf{t}_i, \mathbf{p}_1), \ldots, d(\mathbf{t}_i, \mathbf{p}_k)) | i = 1, \ldots, n\}$.
- *Step 2*: We build a kd-tree T for $\mathcal{V}$. Then, we replace the distance vectors in T's leaf nodes with original time series records. Specifically, if the distance vector in a leaf node is $\mathbf{v}_i$, we replace $\mathbf{v}_i$ with the record $\mathbf{t}_i$. The reason for this replacement is that, in our follow-up algorithm, the internal nodes of T can only assist the searcher to complete filtration. To achieve accurate similarity queries, the searcher needs to conduct further verification using original time series data.

An Efficient Similarity Range Query Algorithm for kd-Tree. With the kd-tree T, we design an efficient similarity range query algorithm, as shown in Algorithm 4. Given a query request $(\mathbf{q}, \tau)$, the searcher can run this algorithm to obtain the query result $\{\mathbf{t}_i | d(\mathbf{q}, \mathbf{t}_i) \leq \delta\}$. This algorithm contains a filtration stage and a verification stage.

- *Filtration stage:* In the filtration stage, the searcher uses $(\mathbf{q}, \tau)$ to construct a kd-tree based range query Q. Then, the searcher performs Q over the kd-tree T to filter out a set of candidate records that are possible in the query result.

Specifically, the searcher first computes the distances between $\mathbf{q}$ and each pivot $\mathbf{p}_j$, i.e., $d(\mathbf{q}, \mathbf{p}_j)$ for $j = 1, 2, \ldots, k$. Then, according to TWED's triangle inequality, we have $|d(\mathbf{q}, \mathbf{p}_j) - d(\mathbf{t}_i, \mathbf{p}_j)| \leq d(\mathbf{q}, \mathbf{t}_i)$. That is, $|d(\mathbf{q}, \mathbf{p}_j) - d(\mathbf{t}_i, \mathbf{p}_j)|$ is a lower bound of $d(\mathbf{q}, \mathbf{t}_i)$. Then, if $\mathbf{t}_i$ satisfies $d(\mathbf{q}, \mathbf{t}_i) \leq \delta$, the inequality $|d(\mathbf{q}, \mathbf{p}_j) - d(\mathbf{t}_i, \mathbf{p}_j)| \leq \delta$ must hold. Furthermore, we have $d(\mathbf{t}_i, \mathbf{p}_j) \in [d(\mathbf{q}, \mathbf{p}_j) - \delta, d(\mathbf{q}, \mathbf{p}_j) + \delta]$ for $j = 1, 2, \ldots, k$. Without loss of generality, let $[v_j^l, v_j^r]$ denote the range $[d(\mathbf{q}, \mathbf{p}_j) - \delta, d(\mathbf{q}, \mathbf{p}_j) + \delta]$ for $j = 1, 2, \ldots, k$. In this case, we can obtain a query range

$$Q = ([v_1^l, v_1^r], [v_2^l, v_2^r], \ldots, [v_k^l, v_k^r]). \tag{2}$$

When $\mathbf{t}_i$ satisfies $d(\mathbf{q}, \mathbf{t}_i) \leq \delta$, we have

$$(d(\mathbf{t}_i, \mathbf{p}_1), d(\mathbf{t}_i, \mathbf{p}_2), \ldots, d(\mathbf{t}_i, \mathbf{p}_k)) \in Q. \tag{3}$$

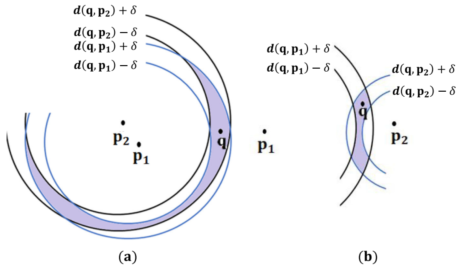

$d(\mathbf{q},\mathbf{p}_2)+\delta$
$d(\mathbf{q},\mathbf{p}_2)-\delta$
$d(\mathbf{q},\mathbf{p}_1)+\delta$
$d(\mathbf{q},\mathbf{p}_1)-\delta$
$\mathbf{p}_2$
$\mathbf{p}_1$
$\mathbf{q}$
(a)
$d(\mathbf{q},\mathbf{p}_1)+\delta$
$d(\mathbf{q},\mathbf{p}_1)-\delta$
$d(\mathbf{q},\mathbf{p}_2)+\delta$
$d(\mathbf{q},\mathbf{p}_2)-\delta$
$\mathbf{q}$
$\mathbf{p}_1$
$\mathbf{p}_2$
(b)

Fig. 2. Example of pivots choosing.

Since $(d(\mathbf{t}_i,\mathbf{p}_1), d(\mathbf{t}_i,\mathbf{p}_2),\ldots,d(\mathbf{t}_i,\mathbf{p}_k))$ has been represented to be a kd-tree T, the searcher can perform the range query Q over T to obtain a set of candidate records $\mathcal{C}$ that are possibly in the query result as Algorithm 3.

- *Verification stage:* In the verification stage, the searcher verifies whether each $\mathbf{t}_i \in \mathcal{C}$ satisfies $d(\mathbf{q},\mathbf{t}_i) \leq \delta$ or not. If $d(\mathbf{q},\mathbf{t}_i) \leq \delta$, the searcher adds $\mathbf{t}_i$ into the query result, i.e., $Result = Result \cup \{\mathbf{t}_i\}$.

Algorithm 4. SimQuery(Tree T, Query Request $(\mathbf{q},\delta)$)

```
Represent $(\mathbf{q},\delta)$ to be $Q$
// Filtration stage
$\mathcal{C} = \emptyset$; // Initialize the candidate result
$\mathcal{C} = \text{Filtration}(\text{T}.root, Q)$;
// Verification stage
for each $\mathbf{t}_i \in \mathcal{C}$ do
    if $d(\mathbf{q},\mathbf{t}_i) \leq \delta$ then
        $Result = Result \cup \{\mathbf{t}_i\}$;
return $Result$;
```

How to Choose Pivots for the Dataset $\mathcal{T}$? Since the similarity query efficiency heavily depends on the choice of pivots, we discuss how to choose pivots. As described above, the query range for the jth dimension is $[v_j^l, v_j^r] = [d(\mathbf{q},\mathbf{p}_j)-\delta, d(\mathbf{q},\mathbf{p}_j)+\delta]$ for $j = 1,2,\ldots,n$. These query ranges are used to filter the data records that are possibly in the query result. Only the record $\mathbf{t}_i$ satisfies that $d(\mathbf{t}_i,\mathbf{p}_j) \in [d(\mathbf{q},\mathbf{p}_j)-\delta, d(\mathbf{q},\mathbf{p}_j)+\delta]$ for all $i = 1,2,\ldots,k$, can it be in the query result. To improve query efficiency, we should reduce the number of data records that can simultaneously satisfy $d(\mathbf{t}_i,\mathbf{p}_j) \in [d(\mathbf{q},\mathbf{p}_j)-\delta, d(\mathbf{q},\mathbf{p}_j)+\delta]$ for all $i = 1,2,\ldots,k$. Based on this idea, we should choose pivots that are far away from each other.

For better understanding, we compare the difference of choosing two pivots with small distance and choosing two pivots with large distance, as shown in Fig. 2. Without loss of generality, we suppose that the dataset in Fig. 2 is a two-dimensional dataset, $\mathbf{q}$ is a query record and $\{\mathbf{p}_1,\mathbf{p}_2\}$ are two pivots. We draw four arcs, i.e., (i) $d(\mathbf{t}_i,\mathbf{p}_1) = d(\mathbf{q},\mathbf{p}_1)-\delta$; (ii) $d(\mathbf{t}_i,\mathbf{p}_1) = d(\mathbf{q},\mathbf{p}_1)+\delta$; (iii) $d(\mathbf{t}_i,\mathbf{p}_2) = d(\mathbf{q},\mathbf{p}_2)-\delta$; and (iv) $d(\mathbf{t}_i,\mathbf{p}_2) = d(\mathbf{q},\mathbf{p}_2)+\delta$. In this case, the data records falling in the shaded area simultaneously satisfy $d(\mathbf{t}_i,\mathbf{p}_1) \in [d(\mathbf{q},\mathbf{p}_1)-\delta, d(\mathbf{q},\mathbf{p}_1)+\delta]$ and $d(\mathbf{t}_i,\mathbf{p}_2) \in [d(\mathbf{q},\mathbf{p}_2)-\delta, d(\mathbf{q},\mathbf{p}_2)+\delta]$, and they can be in the query result. We can see that the shaded area in Fig. 2a with two near pivots is larger than that in Fig. 2b with two far away pivots. This is because when the two pivots are too near, they have the same filtration effect and the overall filtration effect will be weakened. Thus, the chosen pivots should be far away from each other and we can use the following method to choose pivots.

First, we randomly choose a data record as the first pivot. Then, we traverse the dataset and choose the data record that has the largest distance with the first pivot as the second pivot. In a similar way, we can choose more pivots. For the number of pivots, we show that 4 to 10 pivots can get a good filtration effect in the performance evaluation in Section 6. In addition, the chosen pivots can be either the data records of the raw time series or sub-series. However, for convenience, we usually use raw time series as the pivots.

4.2 Privacy-Preserving Protocols

Based on the SHE technique, we introduce four privacy-preserving protocols including a sign computation protocol, a minimum value computation protocol, an absolute value computation protocol, and a bootstrapping protocol. In these protocols, all parameters, encryption algorithms, and decryption algorithms are based on the SHE technique.

Protocol 1: Sign Computation Protocol. The privacy-preserving sign computation protocol is to compute the encrypted sign of a message based on its ciphertext. Specifically, given a ciphertext $E(m)$, when $m < 0$, the sign of m is -1. Meanwhile, the encrypted sign of m will be $E(-1)$, i.e., $E(sign(m)) = E(-1)$. Otherwise, when $m \geq 0$, $E(sign(m)) = E(1)$. This protocol involves cloud server S_1 and cloud server S_2. S_1 holds a ciphertext $E(m)$, where $m \in \mathcal{M}$ ($\mathcal{M} = \{m | m \in (-2^{k_1}, 2^{k_1})\}$), and S_2 holds the corresponding secret key sk. Then, S_1 and S_2 can obtain $E(sign(m))$ while preserving the privacy of m as follows.

- *Step 1:* S_1 chooses two random numbers $r_1, r_2 \in \{0,1\}^{k_1}$ satisfying $r_1 > r_2 > 0$.
- *Step 2:* S_1 computes $E(x) = E(r_1 * m + r_2)$ with SHE's homomorphic properties, and sends $E(x)$ to S_2.
- *Step 3:* On receiving $E(x)$, S_2 uses sk to recover x. Then, based on the sign of x, S_2 deduces that $sign(m) = -1$ if $x < 0$ and $sign(m) = 1$ otherwise. Finally, S_2 returns $E(sign(m))$ to S_1.

Correctness. The sign computation protocol is correct *iff* "$m < 0 \Leftrightarrow sign(m) = -1$" and "$m \geq 0 \Leftrightarrow sign(m) = 1$". Since "$m < 0 \Leftrightarrow sign(m) = -1$" and "$m \geq 0 \Leftrightarrow sign(m) = 1$" are equivalent to each other, we only prove the equivalence "$m < 0 \Leftrightarrow sign(m) = -1$" holds.

Proof. We prove the equivalence "$m < 0 \Leftrightarrow sign(m) = -1$" by respectively proving its sufficiency and necessity.

(1) "$\Rightarrow$": In the sign computation protocol, S_1 will compute $E(x) = E(r_1 * m + r_2)$ and send it to S_2. After receiving $E(x)$, S_2 recovers a plaintext $x = r_1 * m + r_2$ by decryption. Due to $r_1 > r_2 > 0$, when $m < 0$, $r_1 * m + r_2 < 0$ holds, i.e., $x < 0$, S_2 will return $sign(m) = -1$ to S_1.

(2) "$\Leftarrow$": When $sign(m) = -1$, it means that $x < 0$. Then, we have $r_1 * m + r_2 < 0$. Furthermore, we have $m < 0$.

Therefore, the equivalence "$m < 0 \Leftrightarrow sign(m) = -1$" holds, and the sign computation protocol is correct. □

Protocol 2: Minimum Value Computation Protocol. The privacy-preserving minimum value computation protocol is to compute $E(\min\{m_1, m_2\})$ based on $E(m_1)$ and $E(m_2)$. This protocol involves S_1 and S_2. S_1 holds ciphertexts $\{E(m_1), E(m_2), E(1), E(-1)\}$, and S_2 holds the secret key

sk. Then, S_1 and S_2 can obtain $E(\min\{m_1, m_2\})$ while preserving the privacy of m_1 and m_2 as follows.

- *Step 1:* S_1 utilizes SHE's homomorphic properties to compute $E(m) = E(m_1 - m_2) = E(m_1) + E(m_2) * E(-1)$. Then, S_1 computes $E(x) = E(r_1 * m + r_2)$ with SHE's homomorphic properties and sends $E(x)$ to S_2, where $r_1, r_2 \in \{0,1\}^{k_1}$ are two random numbers satisfying $r_1 > r_2 > 0$.
- *Step 2:* On receiving $E(x)$, S_2 uses sk to recover x and returns an encrypted value $E(b)$ to S_1, where $E(b) = E(1)$ if $x < 0$ and $E(b) = E(0)$ otherwise.
- *Step 3:* With $E(b)$, S_1 computes $E(\min\{m_1, m_2\}) = E(m_1 - m_2) * E(b) + E(m_2)$.

Correctness. The minimum value computation protocol is correct *iff* $E(\min\{m_1, m_2\}) = E(m_1 - m_2) * E(b) + E(m_2)$. According to SHE's homomorphic properties, it is equivalent to prove $E(\min\{m_1, m_2\}) = E((m_1 - m_2) * b + m_2)$. That is, $E(\min\{m_1, m_2\})$ should be $E(m_1)$ when $m_1 < m_2$ and $E(m_2)$ otherwise.

Proof. When $m_1 < m_2$, we have $m = m_1 - m_2 < 0$ and can deduce that $x = r_1 * m + r_2 < 0$. Then, $b = 1$ and $E(\min\{m_1, m_2\}) = E((m_1 - m_2) * b + m_2) = E(m_1)$. Otherwise, when $m_1 \geq m_2$, we have $b = 0$ and $E(\min\{m_1, m_2\}) = E(m_2)$. Thus, the minimum value computation protocol is correct. □

Protocol 3: Absolute Value Computation Protocol. The privacy-preserving absolute value computation protocol is to compute $E(|m_1 - m_2|)$ based on $E(m_1)$ and $E(m_2)$. This protocol involves S_1 and S_2. S_1 holds $\{E(m_1), E(m_2)\}$, and S_2 holds the secret key sk. Then, S_1 can obtain $E(|m_1 - m_2|)$ with the help of S_2 while preserving the privacy of m_1 and m_2. The absolute value computation protocol has three steps and the first step is the same as that of the minimum value computation protocol. Thus, we only introduce step 2 and step 3 as follows.

- *Step 1:* Same as the step 1 of the minimum value computation protocol.
- *Step 2:* On receiving $E(x)$, S_2 uses sk to recover the plaintext x. Then, based on the sign of x, S_2 returns an encrypted value $E(b)$ to S_1, where $E(b) = E(-1)$ if $x < 0$ and $E(b) = E(1)$ otherwise.
- *Step 3:* On receiving $E(b)$, S_1 computes the $E(|m_1 - m_2|)$ as $E(|m_1 - m_2|) = E(b) * E(m_1 - m_2)$.

Correctness. The absolute value computation protocol is correct *iff* $E(|m_1 - m_2|) = E(b) * E(m_1 - m_2)$. According to SHE's homomorphic properties, it is equivalent to prove $E(|m_1 - m_2|) = E(b * (m_1 - m_2))$. That is, $|m_1 - m_2|$ should be $m_2 - m_1$ when $m_1 < m_2$ and $m_1 - m_2$ otherwise.

Proof. When $m_1 < m_2$, we have $m = m_1 - m_2 < 0$ and $x = r_1 * m + r_2 < 0$. Then, we have $b = -1$ and $E(|m_1 - m_2|) = E(b * (m_1 - m_2)) = E(m_2 - m_1)$. Similarly, when $m_1 \geq m_2$, we have $E(|m_1 - m_2|) = E(m_1 - m_2)$. Thus, the absolute value computation protocol is correct. □

Protocol 4: Bootstrapping Protocol. In the SHE technique, the number of homomorphic multiplications supported by the SHE technique is affected by the security parameters [16]. To make the SHE technique support an infinite number of homomorphic multiplications, we design a privacy-preserving bootstrapping protocol, which can convert a ciphertext to be a new ciphertext with smaller random numbers. This protocol involves S_1 and S_2. S_1 holds $\{E(m), E(1), E(-1)\}$ and S_2 holds a secret key sk. Then, they can bootstrap the ciphertext $E(m)$ while preserving the privacy of m as follows.

- *Step 1:* S_1 chooses a random number $r_1 \in \mathcal{M}$, and then computes $E(m + r_1)$ as

$$E(m + r_1) = \begin{cases} E(m) + |r_1| * E(-1) & r_1 < 0 \\ E(m) + r_1 * E(1) & r_1 \geq 0. \end{cases}$$

S_1 sends $E(m + r_1)$ to S_2.

- *Step 2:* On receiving $E(m + r_1)$, S_2 decrypts $E(m + r_1)$ to obtain a plaintext m'. Then, S_2 uses sk to encrypt the plaintext $E(m')$ and returns it to S_1.
- *Step 3:* On receiving $E(m')$, S_1 generates a new ciphertext $E(m)$ as

$$E(m)^{new} = \begin{cases} E(m') + |r_1| * E(1) & r_1 < 0 \\ E(m') + r_1 * E(-1) & r_1 \geq 0. \end{cases}$$

Correctness. The bootstrapping protocol is correct *iff* $E(m)^{new}$ is a ciphertext of m. When $r_1 < 0$, we have $E(m)^{new} = E(m') + |r_1| * E(1) = E(m + r_1) + |r_1| * E(1) = E(m + r_1 + |r_1|)$. Since $r_1 < 0$, we can deduce that $E(m)^{new} = E(m)$. Similarly, when $r_1 \geq 0$, the equality $E(m)^{new} = E(m)$ also holds.

4.3 Description of Our Scheme

In this subsection, we present our similarity range query scheme based on the similarity range query algorithm in Algorithm 4. In this algorithm, the search process contains a filtration stage and a verification stage. In the filtration stage, the searcher processes the range query Q based on the data in kd-tree's internal nodes. In the verification stage, the searcher only processes the similarity range query $(\mathbf{q}, \delta)$ based on the candidate data records in kd-tree's leaf nodes.

To resist the cloud inference attack, we consider to use different encryption methods to encrypt the data used in the filtration stage and the data used in the verification stage. Each encryption method should satisfy that S_1 cannot have the capability to compare encrypted data and the capability to construct the ciphertext for any plaintext at the same time. Specifically, we encrypt Q and kd-tree's internal nodes with an invertible matrix such that S_1 cannot construct the ciphertext for any plaintext. Meanwhile, we encrypt $(\mathbf{q}, \delta)$ and kd-tree's leaf nodes with the SHE technique such that S_1 cannot compare any encrypted data.

Specifically, our similarity range query scheme consists of three phases, i.e., system initialization, local data outsourcing, and similarity range query processing.

4.3.1 System Initialization

The data owner initializes the entire system. First, the data owner generates an invertible matrix $\mathrm{A} \in \mathbb{R}^{(2k+4)\times(2k+4)}$ for the encryption of kd-tree's internal nodes, where $\mathbb{R}$ is the real domain. Then, given security parameters (k_0, k_1, k_2), the data owner calls $\mathsf{SHEKeyGen}(k_0, k_1, k_2)$ to generate a pair of public key and secret key, i.e., $\{pk, sk\}$, for the encryption of kd-tree's leaf nodes. After that, the data owner publishes pk, and sends sk to S_2. Meanwhile, it generates two ciphertexts $\{E(-1), E(1)\}$ and sends them to S_1. Furthermore, to build a kd-tree for his/her dataset, the data owner chooses k pivots $\mathcal{P} = \{\mathbf{p}_j | \mathbf{p}_j = (p_{j,1}, p_{j,2}, \ldots, p_{j,l_j}) | j = 1, 2, \ldots, k\}$ from the dataset $\mathcal{T}$. When a query user registers in the system, the data owner will generate a set of ciphertexts $\{E(0), E(1), E(-1)\}$. Then, the data owner authorizes the query user with these ciphertexts and $\{\mathrm{A}^{-1}, \mathcal{P}\}$, where A^{-1} is the inverse of A.

4.3.2 Local Data Outsourcing

Suppose that $\mathcal{T} = \{\mathbf{t}_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,l_i}) | i = 1, 2, \ldots, n\}$ is a dataset, and each $\mathbf{t}_i \in \mathcal{T}$ has a unique identity id_i. Let $\mathcal{P} = \{\mathbf{p}_j | \mathbf{p}_j = (p_{j,1}, p_{j,2}, \ldots, p_{j,l_j}) | j = 1, 2, \ldots, k\}$ be k pivots chosen in the system initialization phase. The data owner outsources the dataset to the server S_1 as follows.

- *Step 1*: Based on $\mathcal{T}$ and $\mathcal{P}$, the data owner organizes the dataset $\mathcal{T}$ into a kd-tree T as described in Section 4.1.
- *Step 2*: The data owner encrypts T's internal nodes. For each node with $\{cd, val, \mathrm{T}_l, \mathrm{T}_r\}$, the data owner uses the matrix A to encrypt it as follows.

(1) Construct two $(2k+2)$-dimensional vectors $\mathbf{u}_L = (u_{L,1}, \ldots, u_{L,2k+2})$ and $\mathbf{u}_R = (u_{R,1}, \ldots, u_{R,2k+2})$ as

$$u_{L,i} = \begin{cases} 1 & i = 2*cd-1 \\ -val & i = 2*k+1 \\ 1 & i = 2*k+2 \\ 0 & \text{Otherwise} \end{cases}, u_{R,i} = \begin{cases} -1 & i = 2*cd \\ val & i = 2*k+1 \\ 1 & i = 2*k+2 \\ 0 & \text{Otherwise.} \end{cases}$$

(2) Encrypt the vectors $\mathbf{u}_L$ and $\mathbf{u}_R$ as

$$\begin{cases} CT_{\mathbf{u}_L} = r_L(\mathbf{u}_L, r'_L, r'_L)\mathrm{A} \\ CT_{\mathbf{u}_R} = r_R(\mathbf{u}_R, r'_R, r'_R)\mathrm{A}, \end{cases} \tag{4}$$

where $r'_L, r'_R \in \mathbb{R}$ are two random real numbers, and $r_L, r_R \in \mathbb{R}^+$ are two positive real random numbers. Then, the internal node is encrypted to be $CT_{node} = \{(CT_{\mathbf{u}_L}, node.\mathrm{T}_l), (CT_{\mathbf{u}_R}, node.\mathrm{T}_r)\}$, where $node.\mathrm{T}_l$ and $node.\mathrm{T}_r$ are respectively $node$'s left subtree and right subtree. Meanwhile, to preserve the privacy of the data, the left subtree and right subtree are randomly permutated such that they are indistinguishable.

- *Step 3*: The data owner encrypts T's leaf nodes with SHEEnc(sk, m) algorithm. Each leaf node has a time series record $\mathbf{t}_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,l_i})$ and an identity id_i. The data owner will use $\{pk, sk\}$ to encrypt them as $\{E(\mathbf{t}_i), E(\mathrm{id}_i)\}$, where $E(\mathbf{t}_i) = (E(t_{i,1}), E(t_{i,2}), \ldots, E(t_{i,l_i}))$.
- *Step 4*: Finally, the data owner outsources the encrypted kd-tree, denoted by $E(\mathrm{T})$, to the server S_1.

4.3.3 Similarity Range Query Processing

On receiving $E(\mathrm{T})$, the cloud can offer similarity range query service to query users. Specifically, the query user can launch a query request $(\mathbf{q}, \delta)$ as the following steps.

- *Step 1*: Based on $(\mathbf{q}, \delta)$, the query user uses A^{-1} to generate a filtration token for the filtration stage of the similarity range query.

(1) The query user uses $(\mathbf{q}, \delta)$ to construct a range query $Q = ([v_1^l, v_1^r], [v_2^l, v_2^r], \ldots, [v_k^l, v_k^r])$ as described in Section 4.1, where $[v_j^l, v_j^r] = [d(\mathbf{q}, \mathbf{p}_j) - \delta, d(\mathbf{q}, \mathbf{p}_j) + \delta]$ for $j = 1, \ldots, k$.

(2) The query user chooses two positive random real numbers $r_1, r_2 \in \mathbb{R}^+$ satisfying $r_1 > r_2 > 0$, and constructs a $(2k+2)$-dimensional vector $\mathbf{z} = (z_1, \ldots, z_{2k+2})$ as

$$\begin{cases} z_{2i-1} = r_1 * v_i^l & 1 \le i \le k \\ z_{2i} = r_1 * v_i^r & 1 \le i \le k \\ z_{2k+1} = r_1;\ z_{2k+2} = r_2. \end{cases}$$

(3) The query user encrypts $\mathbf{z}$ as $\mathrm{T}K_{\mathbf{z}} = (\mathbf{z}, r_q, -r_q)(\mathrm{A}^{-1})^T$, where $r_q \in \mathbb{R}$ is a random number.

- *Step 2*: Based on $(\mathbf{q}, \delta)$, the query user uses $\{E(0), E(1), E(-1)\}$ to generate the verification token $(E(\mathbf{q}), E(\delta))$, where $E(\mathbf{q}) = (E(q_1), \ldots, E(q_{l_q}))$. Specifically, q_i and δ are encrypted as Eq. (1) for $j = 1, 2, \ldots, l_q$.
- *Step 3*: The query user chooses a session key ssk. Then, it sends a similarity range query request with tokens $\{\mathrm{T}K_{\mathbf{z}}, E(\mathbf{q}), E(\delta)\}$ and ssk to S_1 via a secure channel.
- *Step 4*: On receiving $\{\mathrm{T}K_{\mathbf{z}}, E(\mathbf{q}), E(\delta)\}$, S_1 and S_2 cooperate to search $E(\mathrm{T})$ for the query result. The search process contains a filtration stage and a verification stage.

Filtration Stage. In the filtration stage, S_1 searches $E(\mathrm{T})$ to obtain a set of candidate time series records that are possibly in the query result, as shown in Algorithm 5. The filtration algorithm over the encrypted kd-tree is similar to that over the plaintext kd-tree. Differently, the conditions "$v_{cd}^l < val$" and "$v_{cd}^r \ge val$" are replaced with "$CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} < 0$" and "$CT_{\mathbf{u}_R} \circ \mathrm{T}K_{\mathbf{z}} \ge 0$".

Algorithm 5. FiltrationCipher(Node $E(node)$, Token $\mathrm{T}K_{\mathbf{z}}$)

```
if $node$ is a leaf node then
  Add $node$'s encrypted data into $\mathcal{C}$;
else
  if $CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} < 0$ then
    FiltrationCipher($node.\mathrm{T}_l$, $\mathrm{T}K_{\mathbf{z}}$);
  if $CT_{\mathbf{u}_R} \circ \mathrm{T}K_{\mathbf{z}} \ge 0$ then
    FiltrationCipher($node.\mathrm{T}_r$, $\mathrm{T}K_{\mathbf{z}}$);
return $\mathcal{C}$;
```

Correctness. The filtration algorithm is correct *iff*

$$\begin{cases} CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} < 0 \Leftrightarrow v_{cd}^l < val \\ CT_{\mathbf{u}_R} \circ \mathrm{T}K_{\mathbf{z}} \ge 0 \Leftrightarrow v_{cd}^r \ge val. \end{cases}$$

Proof. First, we have

$$\begin{aligned} CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} &= (r_L(\mathbf{u}_L, r'_L, r'_L)\mathrm{A}) \circ ((\mathbf{z}, r_q, -r_q)(\mathrm{A}^{-1})^T) \\ &= r_L(\mathbf{u}_L, r'_L, r'_L)(\mathbf{z}, r_q, -r_q)^T \\ &= r_L\left(\sum_{i=1}^{2k+2}(u_{L,i} * z_i) + r'_L * r_q - r'_L * r_q\right) \\ &= r_L(r_1(v_{cd}^l - val) + r_2). \end{aligned}$$

When $CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} < 0$, we have $r_L(r_1(v_{cd}^l - val) + r_2) < 0$. Since $r_L > 0$, we can deduce that $r_1(v_{cd}^l - val) + r_2 < 0$. Moreover, since $r_1 > 0$, $r_2 > 0$ and $r_1 > r_2$, we have $v_{cd}^l < val$. In contrast, when $v_{cd}^l < val$, it is easy to deduce that $CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} < 0$. Thus, $CT_{\mathbf{u}_L} \circ \mathrm{T}K_{\mathbf{z}} <$

$0 \Leftrightarrow v^l_{cd} < val$. Similarly, we can prove $\mathrm{C}T_{\mathbf{u}_R} \circ \mathrm{T}K_{\mathbf{z}} \geq 0 \Leftrightarrow v^r_{cd} \geq val$. □

After the filtration stage, S_1 can obtain a set of records $\mathcal{C} = \{E(\mathbf{t}_i), E(\mathrm{id}_i) | \mathbf{t}_i\ may\, satisfy\ \ d(\mathbf{t}_i, \mathbf{q}) \leq \delta\}$.

Verification Stage. In this stage, S_1 verifies whether each $\{E(\mathbf{t}_i), E(\mathrm{id}_i)\}$ satisfies $d(\mathbf{t}_i, \mathbf{q}) \leq \delta$ or not as follows.

(1) S_1 and S_2 compute $E(d(\mathbf{t}_i, \mathbf{q}))$. The TWED computation algorithm over encrypted records is similar to that over plaintext records in Algorithm 1. Differently, since both $\mathbf{t}_i$ and $\mathbf{q}$ are encrypted, the algorithm will compute an encrypted distance matrix. Based on Algorithm 1, computing each element of the encrypted distance matrix requires three types of basic operations over encrypted data, i.e., addition, absolute value computation, and minimum value computation, which can be respectively completed by the homomorphic property of the SHE technique, our privacy-preserving absolute value computation protocol and minimum value computation protocol. In addition, when finishing one element computation, the cloud server will run the bootstrapping protocol to convert the element to be a new ciphertext with smaller random numbers. Finally, S_1 can obtain the encrypted distance $E(d(\mathbf{t}_i, \mathbf{q}))$.

(2) S_1 computes $E(\delta - d(\mathbf{t}_i, \mathbf{q})) = E(\delta) + E(-1) * E(d(\mathbf{t}_i, \mathbf{q}))$ and runs the sign computation protocol to obtain the encrypted sign of $\delta - d(\mathbf{t}_i, \mathbf{q})$, i.e., $E(\mathrm{sign}(\delta - d(\mathbf{t}_i, \mathbf{q})))$. For a easy description, we let $E(b_i)$ denote $E(\mathrm{sign}(\delta - d(\mathbf{t}_i, \mathbf{q})))$. Then, we have

$$E(b_i) = \begin{cases} E(-1) & \delta < d(\mathbf{t}, \mathbf{q}) \\ E(1) & \delta \geq d(\mathbf{t}, \mathbf{q}). \end{cases}$$

(3) S_1 computes $E(b_i * \mathrm{id}_i) = E(b_i) * E(\mathrm{id}_i)$.

After the verification stage, S_1 can obtain a collection of encrypted identities, i.e., $E(\mathcal{ID}) = \{E(b_i * \mathrm{id}_i) | \{E(\mathbf{t}_i), E(\mathrm{id}_i)\} \in \mathcal{C}\}$.

- *Step 5*: Based on $E(\mathcal{ID})$, S_1 and S_2 cooperate together to return the query result to the query user. To preserve the privacy of id_i, S_1 and S_2 will respectively return a part of information about id_i to the query user. Meanwhile, it is only when $d(\mathbf{t}_i, \mathbf{q}) \leq \delta$, i.e., $E(b_i) = E(1)$, id_i needs to be returned to the query user. This is because only when $d(\mathbf{q}, \mathbf{t}_i) \leq \delta$, $\mathbf{t}_i$ satisfies the query condition. Based on this idea, in the following, we take $E(b_i * \mathrm{id}_i)$ as an example to show how to return the correct query result to the query user in a privacy-preserving way.

(1) S_1 chooses two random numbers $r_{i,1}, r_{i,2} \in \{0,1\}^{k_1}$ such that $r_{i,1} > r_{i,2}$, and S_1 computes two ciphertexts as

$$\begin{cases} AES_{ssk}(r_{i,1} || r_{i,2}) \\ E(x) = E(b_i * \mathrm{id}_i * r_{i,1} + r_{i,2}). \end{cases}$$

Then, S_1 sends them to S_2.

(2) On receiving $AES_{ssk}(r_{i,1} || r_{i,2})$ and $E(x) = E(b_i * \mathrm{id}_i * r_{i,1} + r_{i,2})$, S_2 first recovers x by decryption. Based on the sign of x, S_2 obtains $\mathrm{sign}(b_i * \mathrm{id}_i)$. If $\mathrm{sign}(b_i * \mathrm{id}_i) = 1$, we have $E(b_i) = E(1)$ due to $\mathrm{id}_i \in \{0,1\}^{k_1}$. In this case, id_i needs to be returned. Specifically, S_2 will return $\{AES_{ssk}(r_{i,1} || r_{i,2}), x\}$ to the query user via a secure channel, where $AES_{ssk}(r_{i,1} || r_{i,2})$ is computed by S_1 and x is computed S_2.

(3) On receiving $\{AES_{ssk}(r_{i,1} || r_{i,2}), x\}$, the query user first uses ssk to recover $r_{i,1}$ and $r_{i,2}$, and computes id_i as

$$\mathrm{id}_i = \frac{x - r_{i,2}}{r_{i,1}}. \quad (5)$$

Similarly, S_1 and S_2 returns all identifies of data records satisfying the query condition to the query user, i.e., $\{\mathrm{id}_i | d(\mathbf{q}, \mathbf{t}_i) \leq \delta\}$.

Correctness. In the following, we show Eq. (5) is correct.

Proof. Since $sign(b_i * \mathrm{id}_i) = 1$, we have $b_i = 1$ and $b_i * \mathrm{id}_i * r_{i,1} + r_{i,2} > 0$. Then, we can deduce that $x = \mathrm{id}_i * r_{i,1} + r_{i,2}$. Thus, we have $\frac{x - r_{i,2}}{r_{i,1}} = \mathrm{id}_i$, and Eq. (5) is correct. □

5 Security Analysis

In this section, we analyze the security of our similarity range query scheme. Since our scheme is based on the SHE technique and the SHE technique is required to be CPA-secure as discussed in Section 3.3, we first prove that the SHE technique is semantically secure against CPA, and then prove the security of our scheme. Finally, we show that our scheme can resist against the cloud inference attack.

5.1 Security of SHE Technique

We prove that SHE technique is semantically secure against CPA by reducing the security of the SHE technique to an $(\mathcal{L}, p)$-based decision assumption. In the following, we first introduce $(\mathcal{L}, p)$-based decision problem and $(\mathcal{L}, p)$-based decision assumption. Then, we prove that the SHE technique is semantically secure against CPA under the $(\mathcal{L}, p)$-based decision assumption.

According to the SHE technique, in the following description, let k_2 and k_0 be two security parameters satisfying $k_2 < k_0$. Let p and q be two large primes with $|p| = |q| = k_0$. Let $\mathcal{N} = pq$ and $Z_{\mathcal{N}}$ is a group of integers modulo $\mathcal{N}$. Let $\mathcal{L}$ be a random number with $|\mathcal{L}| = k_2$.

Before introducing the $(\mathcal{L}, p)$-based decision problem, we first introduce three basic lemmas used in it.

Lemma 1. *Any valid ciphertext in the SHE technique is in the form of* $E(m) = (m + \alpha\mathcal{L} + \beta p) \bmod \mathcal{N}$, *where* $\alpha, \beta \in Z_{\mathcal{N}}$ *and* $\alpha\mathcal{L} < p$.

Proof. First, for the ciphertext without being applied homomorphic operations, it is $E(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$. We have

$$\begin{aligned} E(m) &= (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N} \\ &= (m + r\mathcal{L} + (rr'\mathcal{L} + mr')p) \bmod \mathcal{N}. \end{aligned}$$

Let $\alpha = r \bmod \mathcal{N}$ and $\beta = (rr'\mathcal{L} + mr') \bmod \mathcal{N}$. Then, any $E(m)$ can be represented as $E(m) = (m + \alpha\mathcal{L} + \beta p) \bmod \mathcal{N}$. Second, for the ciphertext being applied homomorphic operations, it is easy to deduce that the ciphertext is also in the form of $E(m) = (m + \alpha\mathcal{L} + \beta p) \bmod \mathcal{N}$, only if the resulting $m < \mathcal{L}$ and $\alpha\mathcal{L} < p$.

In the following, we show the reason why the ciphertext $E(m) = (m + \alpha\mathcal{L} + \beta p) \bmod \mathcal{N}$ should satisfy $\alpha\mathcal{L} < p$. This is because decrypting m is to compute

$$\begin{aligned} &(E(m) \bmod p) \bmod \mathcal{L} \\ &= (((m + \alpha\mathcal{L} + \beta p) \bmod \mathcal{N}) \bmod p) \bmod \mathcal{L} \\ &= ((m + \alpha\mathcal{L}) \bmod p) \bmod \mathcal{L}. \end{aligned}$$

The decrypted message is correct *iff* $(m+\alpha\mathcal{L}) \bmod p = m+\alpha\mathcal{L}$, i.e., $m+\alpha\mathcal{L} < p$. Since $m \in (-2^{k_1}, 2^{k_1})$, $|p| = k_0$, and $k_1 \ll k_0$, we can deduce that $m \ll p$. Then, we can obtain that $m+\alpha\mathcal{L} < p$ is approximately equivalent to $\alpha\mathcal{L} < p$. That is, a valid ciphertext $E(m) = (m+\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$ should satisfy $\alpha\mathcal{L} < p$. □

Based on the form of the ciphertext, we can construct a set $\mathbb{S}$ as $\mathbb{S} : \{x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} | \alpha, \beta \in Z_{\mathcal{N}}, \alpha\mathcal{L} < p\}$. We can observe that the ciphertext $E(m) = (m+\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$ can be represented to be $E(m) = m + x$, where $x \in \mathbb{S}$. Next, we show that for each $x \in \mathbb{S}$, its representation is unique.

Lemma 2. *For any* $x \in \mathbb{S}$ *(* $\mathbb{S} : \{x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} | \alpha, \beta \in Z_{\mathcal{N}}, \alpha\mathcal{L} < p\}$*), the representation* $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$ *is unique.*

Proof. We prove Lemma 2 by contradiction. Suppose that there are two pairs (α_1, β_1) and (α_2, β_2) such that $x = (\alpha_1\mathcal{L}+\beta_1 p) \bmod \mathcal{N} = (\alpha_2\mathcal{L}+\beta_2 p) \bmod \mathcal{N}$. Then, we have

$$\begin{aligned}&(\alpha_1\mathcal{L}+\beta_1 p) \bmod \mathcal{N} = (\alpha_2\mathcal{L}+\beta_2 p) \bmod \mathcal{N}\\ \Leftrightarrow\ &((\alpha_1-\alpha_2)\mathcal{L}+(\beta_1-\beta_2)p) \bmod \mathcal{N} = 0\\ \Leftrightarrow\ &(\alpha_1-\alpha_2)\mathcal{L} \bmod p = 0 \Leftrightarrow |\alpha_1-\alpha_2|\mathcal{L} \bmod p = 0.\end{aligned}$$

Since $\alpha_1\mathcal{L} < p$ and $\alpha_2\mathcal{L} < p$, we have $|\alpha_1-\alpha_2|\mathcal{L} < p$ and $|\alpha_1-\alpha_2|\mathcal{L} \bmod p = |\alpha_1-\alpha_2|\mathcal{L} = 0$. Hence, $\alpha_1 = \alpha_2$ and $\alpha_1\mathcal{L} = \alpha_2\mathcal{L}$. Moreover, we can deduce that $\beta_1 p \bmod \mathcal{N} = \beta_2 p \bmod \mathcal{N} = (x - \alpha_1\mathcal{L}) \bmod \mathcal{N}$. Thus, $\beta_1 = \beta_2$. As a result, any $x \in \mathbb{S}$ can be uniquely represented as $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$, where $\alpha, \beta \in Z_{\mathcal{N}}, \alpha\mathcal{L} < p$. □

In the following, we show that any $x \in Z_{\mathcal{N}}$ can be represented as $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$. Different from the form of a valid ciphertext, the representation of x does not require $\alpha\mathcal{L} < p$. The details are as follows.

Lemma 3. *For any* $x \in Z_{\mathcal{N}}$*, there exist* $\alpha, \beta \in Z_{\mathcal{N}}$ *such that* $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$.

Proof. Since $|\mathcal{L}| = k_2$, $|p| = k_0$, and $k_2 < k_0$, we have $\mathcal{L} < p$. Since p is a prime, we have $\gcd(\mathcal{L}, p) = 1$. According to the Extended euclidean algorithm, there exist two numbers γ and ψ such that $\gamma\mathcal{L} + \psi p = 1$, where γ and ψ have opposite signs. For any $x \in Z_{\mathcal{N}}$, it can be represented as

$$x = x * 1 = x * (\gamma\mathcal{L} + \psi p) = x\gamma\mathcal{L} + x\psi p.$$

Let $\alpha = x\gamma \bmod \mathcal{N}$ and $\beta = x\psi \bmod \mathcal{N}$. Then, we have

$$(\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} = x\gamma\mathcal{L} + x\psi p \bmod \mathcal{N} = x.$$

As a result, any $x \in Z_{\mathcal{N}}$ can be represented as $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$, and $Z_{\mathcal{N}}$ can be represented as $\{x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} | \alpha, \beta \in Z_{\mathcal{N}}\}$. □

Based on the definition of set $\mathbb{S}$, we can see $\mathbb{S} \subset Z_{\mathcal{N}}$, and thus $Z_{\mathcal{N}}$ can be divided into two sets

$$\begin{cases}\mathbb{S} : \{x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} | \alpha, \beta \in Z_{\mathcal{N}}, \alpha\mathcal{L} < p\}\\ \overline{\mathbb{S}} = Z_{\mathcal{N}} \setminus \mathbb{S} : \{x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N} | \alpha, \beta \in Z_{\mathcal{N}}, \alpha\mathcal{L} \geq p\}.\end{cases}$$

According to the two sets $\mathbb{S}, \overline{\mathbb{S}}$, where $\mathbb{S} \cup \overline{\mathbb{S}} = Z_{\mathcal{N}}$, we define the $(\mathcal{L}, p)$-based decision problem.

Definition 3 ($(\mathcal{L}, p)$-based decision problem). *Given* $(k_0, k_2, \mathcal{N})$*, the* $(\mathcal{L}, p)$*-based decision problem is to determine that a random number* $x \in Z_{\mathcal{N}}$ *is in* $\mathbb{S}$ *or* $\overline{\mathbb{S}}$ *without knowing* $(p, q, \mathcal{L})$.

- *Intractability of* $(\mathcal{L}, p)$*-Based Decision Problem.* The $(\mathcal{L}, p)$-based decision problem is to determine whether a random number $x \in Z_{\mathcal{N}}$ is in $\mathbb{S}$ or $\overline{\mathbb{S}}$ without knowing $(p, q, \mathcal{L})$. It is easy to observe that there is no efficient algorithm to solve the $(\mathcal{L}, p)$-based decision problem when $(p, q, \mathcal{L})$ are unknown. Thus, to solve this decision problem, the distinguisher $\mathcal{B}$ can only attempt to find the values of $\{p, q, \mathcal{L}\}$ in the following two ways.

$*$ Since $\mathcal{N} = pq$, the first way is to factorize $\mathcal{N}$ to obtain p and q. After obtaining p, $\mathcal{B}$ can recover $\mathcal{L}$ when it has a set of values $\{x_i = (\alpha_i\mathcal{L}+\beta_i p) \bmod \mathcal{N} \in \mathbb{S} | i = 1, 2, \ldots, l\}$. By computing each $x_i \bmod p$, $\mathcal{B}$ can obtain a new set of values $\{\alpha_i\mathcal{L} | i = 1, 2, \ldots, l\}$. Then, by computing the great common division of all values in $\{\alpha_i\mathcal{L} | i = 1, 2, \ldots, l\}$, $\mathcal{L}$ can be revealed with a high probability. Once $\mathcal{B}$ knows α, $\mathcal{L}$, the $(\mathcal{L}, p)$-based decision problem can be solved. Therefore, we need to choose proper parameter k_0 for $|p| = |q| = k_0$ to ensure the large integer factoring problem is hard.

$*$ The second way is to exhaust the values of $\{\alpha, \mathcal{L}\}$ and compute $\gcd(x - \alpha\mathcal{L}, \mathcal{N})$. Since $\mathcal{N} = pq$, $\gcd(x - \alpha\mathcal{L}, \mathcal{N})$ is either p or q. Then, there are two cases.

(1) When $\gcd(x - \alpha\mathcal{L}, \mathcal{N}) = p$, $\mathcal{B}$ has the values of α, $\mathcal{L}$ and p. In this case, $\mathcal{B}$ can solve the $(\mathcal{L}, p)$-based decision problem based on the value of α.

(2) When $\gcd(x - \alpha\mathcal{L}, \mathcal{N}) = q$, $\mathcal{B}$ has the values of α, $\mathcal{L}$ and q. Since x can also be represented to be $x = (\alpha_1\mathcal{L} + \beta_1 q) \bmod \mathcal{N}$, the obtained α is actually α_1 rather than α in $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$. To obtain the value of α, $\mathcal{B}$ first computes $p = \frac{\mathcal{N}}{q}$ and further uses Extend euclidean algorithm to compute α and β satisfying $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$. Then, it can solve the $(\mathcal{L}, p)$-based decision problem based on the value of α.

In this way, the computational cost comes from exhausting the values of $\{\alpha, \mathcal{L}\}$. According to Lemma 2, when $x \in \mathbb{S}$, the representation of $x = (\alpha\mathcal{L}+\beta p) \bmod \mathcal{N}$ is unique. Meanwhile, since $\alpha > 0$ and $\mathcal{L}$ is a k_2-bit number, the computational cost is larger than 2^{k_2}.

Summarizing the above two ways, when the security parameters k_0 and k_2 are large enough, e.g., $k_0 = 512$ and $k_2 = 160$, the $(\mathcal{L}, p)$-based decision problem is intractable.

Definition 4 ($(\mathcal{L}, p)$-based Decision Assumption). *$(\mathcal{L}, p)$-based decision problem satisfies $(\mathcal{L}, p)$-based decision assumption if for any polynomial time algorithm, its advantage in solving the $(\mathcal{L}, p)$-based decision problem is a negligible function in* k_0 *and* k_2.

Theorem 1. *The SHE technique is semantically secure against CPA under the* $(\mathcal{L}, p)$*-based decision assumption.*

Proof. Assume that there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that has a non-negligible advantage ε to break the semantic security of the SHE technique. We can construct a distinguisher $\mathcal{B}$, which has access to $\mathcal{A}$ and can have a non-negligible advantage to break the $(\mathcal{L}, p)$-based decision problem. Let $z \in \{0, 1\}$ be a random bit, an $(\mathcal{L}, p)$-based decision instance $(k_0, k_2, \mathcal{N}, x)$ is given to $\mathcal{B}$, where x is randomly chosen from $\mathbb{S}$ if $z=0$, and x is randomly chosen from $\overline{\mathbb{S}}$ if $z=1$. Then, the $(\mathcal{L},p)$-based decision problem is to guess z.

With the $(\mathcal{L},p)$-based decision instance $(k_0,k_2,\mathcal{N},x)$, $\mathcal{B}$ first chooses k_1 such that $k_1 \ll k_2$ and sets $\mathcal{M}=\{m|m\in(-2^{k_1},2^{k_1})\}$ as the message space. Then, $\mathcal{B}$ sends $(k_0,k_1,k_2,\mathcal{N})$ to $\mathcal{A}$.

Upon receiving $(k_0,k_1,k_2,\mathcal{N})$, $\mathcal{A}$ chooses two messages $m_0,m_1\in\mathcal{M}$ and sends them to $\mathcal{B}$. At this time, $\mathcal{B}$ flips a bit $b\in\{0,1\}$, computes $c=m_b+x$ and returns c as a ciphertext back to $\mathcal{A}$.

After receiving c, $\mathcal{A}$ returns $\mathcal{B}$ a bit $b'\in\{0,1\}$ as the guess of b. $\mathcal{B}$ then guesses $z=0$ if $b'=b$. Obviously, when $z=0$, i.e., $x\in\mathbb{S}$, we have $c=m_b+x=m_b+\alpha\mathcal{L}+\beta p \bmod \mathcal{N}$ is a valid ciphertext. In this case, $\mathcal{A}$ can exert his capability and will guess b correctly with the probability $\frac{1}{2}+\varepsilon$. Then, $\Pr[\mathcal{B}\,\text{success}|z=0]=\frac{1}{2}+\varepsilon$. On the other hand, when $z=1$, i.e., $x\in\overline{\mathbb{S}}$, $c=m_b+x=m_b+\alpha\mathcal{L}+\beta p \bmod \mathcal{N}$ is no longer a valid ciphertext. Then, the probability that $\mathcal{A}$ can guess b correctly is only $\frac{1}{2}$. Then, $\Pr[\mathcal{B}\,\text{success}|z=1]=\frac{1}{2}$. Summarizing the above two cases, we have $\Pr[\mathcal{B}\,\text{success}]=\frac{1}{2}(\frac{1}{2}+\varepsilon)+\frac{1}{2}\cdot\frac{1}{2}=\frac{1}{2}+\frac{\varepsilon}{2}$. Since ε is non-negligible, the above result contradicts with the $(\mathcal{L},p)$-based decision assumption. Thus, the SHE technique is semantically secure against CPA. □

5.2 Security of Our Similarity Range Query Scheme

Our similarity range query scheme is a searchable encryption scheme, so we prove that it is selectively secure in the real/ideal world model. Before that, we define the leakage function of our scheme from the perspective of S_1 and S_2.

- $\mathtt{L}(S_1)$: The information leaked to S_1 includes the public key pk, encrypted kd-tree $E(\mathrm{T})$, query token $\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)\}$, access pattern of $E(\mathrm{T})$, and two ciphertexts $\{E(-1),E(1)\}$.
- $\mathtt{L}(S_2)$: The information leaked to S_2 includes public key pk, secret key sk, the number of time series records in the candidate results, and that in the final results.

Based on $\mathtt{L}(S_1)$ and $\mathtt{L}(S_2)$, we can define the real/ideal world as follows.

Real World. In the real world, there are two probabilistic polynomial time adversaries, denoted by $\mathcal{A}_1$ and $\mathcal{A}_2$, and a challenger, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are non-collusive. The challenger interacts with $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows.

- *System initialization phase:* $\mathcal{A}_1$ sends a time series dataset $\mathcal{T}=\{\mathbf{t}_i=(t_{i,1},t_{i,2},\ldots,t_{i,l_i})|i=1,2,\ldots,n\}$ and the identity of each $\mathbf{t}_i$ is id_i to the challenger. The challenger runs the system initialization algorithm in the similarity range query scheme to generate the security keys $\{pk,sk,\mathcal{P},\mathrm{A}\}$. Then, the challenger publishes pk and sends sk to $\mathcal{A}_2$. With the security keys $\{pk,\mathcal{P},\mathrm{A}\}$, the challenger runs the local data outsourcing algorithm to encrypt $\mathcal{T}$ into an encrypted kd-tree $E(\mathrm{T})$.
- *Token generation phase 1:* $\mathcal{A}_1$ submits w_1 similarity queries $\{(\mathbf{q}_i,\delta_i)|1\le i\le w_1\}$ to the challenger, where w_1 is a polynomial number. The challenger runs the token generation algorithm to generate the tokens $\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)|1\le i\le w_1\}$ for these queries and returns them to $\mathcal{A}_1$.
- *Challenge phase:* The challenger returns $E(\mathrm{T})$ to $\mathcal{A}_1$.
- *Token generation phase 2:* $\mathcal{A}_1$ submits w_2-w_1 similarity queries $\{(\mathbf{q}_i,\delta_i)|w_1+1\le i\le w_2\}$ to the challenger and gets the query tokens $\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)|w_1+1\le i\le w_2\}$ from the challenger, where w_2 is a polynomial number.
- *Similarity Range Query Processing:* When $\mathcal{A}_1$ obtains the query tokens $\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)|1\le i\le w_2\}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ cooperatively search on $E(\mathrm{T})$ to obtain the query result of each token $\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)\}$.

In the real world, the view of $\mathcal{A}_1$, i.e., $\mathrm{View}_{\mathcal{A}_1,\mathrm{Real}}$, includes $\{E(\mathrm{T}),\{\mathrm{TK}_{\mathbf{z}_i},E(\mathbf{q}_i),E(\delta_i)\}_{i=1}^{w_2}\}$, access pattern of $E(\mathrm{T})$ and some SHE ciphertexts received in the privacy-preserving protocols. The view of $\mathcal{A}_2$, i.e., $\mathrm{View}_{\mathcal{A}_2,\mathrm{Real}}$, includes the AES ciphertexts in the candidate result and query result, and some plaintexts received in the privacy-preserving protocols.

Ideal World. The ideal world involves PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, and a simulator with the leakage $\mathtt{L}(S_1)$ and $\mathtt{L}(S_2)$. The simulator interacts with $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows.

- *System initialization phase:* $\mathcal{A}_1$ sends the dataset $\mathcal{T}=\{\mathbf{t}_i=(t_{i,1},t_{i,2},\ldots,t_{i,l_i})|i=1,2,\ldots,n\}$ and $\{\mathrm{id}_i|i=1,2,\ldots,n\}$ to the simulator. The simulator first publishes the public key $pk\in\mathtt{L}(S_1)$ and sends $sk\in\mathtt{L}(S_2)$ to S_2. Then, with $\mathtt{L}(S_1)$ and $\mathtt{L}(S_2)$, the simulator generates an encrypted kd-tree $E'(\mathrm{T})$. The main idea to build $E'(\mathrm{T})$ is to replace the values in $E(\mathrm{T})$'s internal nodes with random values generated by $\mathtt{L}(S_1)$ and self-blind the values in $E(\mathrm{T})$'s leaf nodes as follows.

(1) Internal node replacement: For an internal node with the value $\mathrm{C}T_{node}=\{(\mathrm{C}T_{\mathbf{u}_L},node.\mathrm{T}_l),(\mathrm{C}T_{\mathbf{u}_R},node.\mathrm{T}_r)\}$, the simulator replaces $\mathrm{C}T_{\mathbf{u}_L}$ and $\mathrm{C}T_{\mathbf{u}_R}$ with two $(2k+4)$-dimensional random vectors. Then, the node becomes $\mathrm{C}T'_{node}=\{(\mathrm{C}T'_{\mathbf{u}_L},node.\mathrm{T}_l),(\mathrm{C}T'_{\mathbf{u}_R},node.\mathrm{T}_r)\}$.

(2) Leaf node self-blinding: The simulator first uses $\{E(1),E(-1)\}\in\mathtt{L}(S_1)$ to compute $E(0)=E(1)+E(-1)$. Then, for each internal node with $E(\mathbf{t}_i)=(E(t_{i,1}),E(t_{i,2}),\ldots,E(t_{i,l_i}))$ and $E(\mathrm{id}_i)$, the simulator self-blinds each $E(t_{i,j})$ as $E'(t_{i,j})=(E(t_{i,j})+\sum_{\eta,\xi,\sigma}E(0)^{\eta}E(1)^{\xi}E(1)^{\sigma}) \bmod \mathcal{N}$. According to SHE's homomorphic properties, $E'(t_{i,j})$ is also a ciphertext of $t_{i,j}$. Similarly, the simulator self-blinds $E(\mathrm{id}_i)$ into $E'(\mathrm{id}_i)$.

- *Token generation phase 1:* $\mathcal{A}_1$ submits w_1 query requests $\{(\mathbf{q}_i,\delta_i)|1\le i\le w_1\}$ to the simulator. The simulator generates the filtration token and verification token for each $(\mathbf{q}_i,\delta_i)$ as follows.

(1) Filtration token: The simulator generates the filtration token for $(\mathbf{q}_i,\delta_i)$ based on $E(\mathrm{T})$'s access pattern in $\mathtt{L}(S_1)$. Specifically, according to the access pattern, the simulator knows $\{\mathrm{sign}(\mathrm{C}T_{\mathbf{u}_L}\circ\mathrm{T}K_{\mathbf{z}_i}),\mathrm{sign}(\mathrm{C}T_{\mathbf{u}_R}\circ\mathrm{T}K_{\mathbf{z}_i})\}$ for each internal node $\mathrm{C}T_{node}=\{(\mathrm{C}T_{\mathbf{u}_L},node.\mathrm{T}_l),(\mathrm{C}T_{\mathbf{u}_R},node.\mathrm{T}_r)\}$. Suppose that there are γ internal nodes in $E(\mathrm{T})$, i.e., $\mathrm{C}T_{node_j}=\{(\mathrm{C}T_{\mathbf{u}_L,j},\mathrm{C}T_{\mathbf{u}_R,j})|j=1,2,\ldots,\gamma\}$. The simulator generates a 2γ-dimensional vector $\mathbf{r}_i=(r_1,r_2,\ldots,r_{2\gamma-1},r_{2\gamma})$. We set $r_{2i-1}>0$ if $\mathrm{sign}(\mathrm{C}T_{\mathbf{u}_L,j},\mathrm{T}K_{\mathbf{z}_i})>0$ and $r_{2i-1}<0$ otherwise for $i=1,2,\ldots,\gamma$. Similarly, we set $r_{2i}>0$ if $\mathrm{sign}(\mathrm{C}T_{\mathbf{u}_R,j},\mathrm{T}K_{\mathbf{z}_i})>0$ and $r_{2i}<0$ otherwise for $i=1,2,\ldots,\gamma$. Then, the simulator generates a filtration token $\mathrm{T}K'_{\mathbf{z}_i}$ such that

$$\begin{bmatrix} \mathrm{C}T'_{\mathbf{u}_L,1} \\ \mathrm{C}T'_{\mathbf{u}_R,1} \\ \cdots \\ \mathrm{C}T'_{\mathbf{u}_L,2\gamma-1} \\ \mathrm{C}T'_{\mathbf{u}_R,2\gamma} \end{bmatrix} \mathrm{T}K'^{T}_{\mathbf{z}_i}=\mathbf{r}_i.$$

(2) Verification token: The simulator generates the verification token $\{E'(\mathbf{q}_i), E'(\delta_i)\}$ for each $(\mathbf{q}_i, \delta_i)$ by self-blinding $\{E(\mathbf{q}_i), E(\delta_i)\} \in \mathtt{L}(S_1)$.

Finally, the simulator returns $\{(\mathrm{T}K'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i))|\ 1 \leq i \leq w_1\}$ to $\mathcal{A}_1$.

- *Challenge phase:* The challenger returns $E'(\mathrm{T})$ to $\mathcal{A}_1$.
- *Token generation phase 2:* In this phase, $\mathcal{A}_1$ submits $w_2 - w_1$ similarity range query requests $\{(\mathbf{q}_i, \delta_i)|w_1+1 \leq i \leq w_2\}$ to the simulator and gets the query tokens $\{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i)|w_1+1 \leq i \leq w_2\}$ from the simulator.
- *Similarity Range Query Processing:* When $\mathcal{A}_1$ obtains the query tokens $\{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i)|1 \leq i \leq w_2\}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ cooperatively run the similarity query processing algorithm to search on $E'(\mathrm{T})$ and obtain the query result for each token $\{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i)\}$.

In the ideal world, the view of $\mathcal{A}_1$, i.e., $\mathrm{View}_{\mathcal{A}_1,\mathrm{Ideal}}$, includes $\{E'(\mathrm{T}), \{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i)\}_{i=1}^{w_2}\}$, access pattern of $E'(\mathrm{T})$ and some SHE ciphertexts received in the privacy-preserving protocols. The view of $\mathcal{A}_2$, i.e., $\mathrm{View}_{\mathcal{A}_2,\mathrm{Ideal}}$, includes the AES ciphertexts in the candidate result and query result, and some plaintexts received in the privacy-preserving protocols.

Definition 5 (Security of Our Scheme). *Our scheme is selectively secure with the leakage* $\mathtt{L}(S_1)$ *and* $\mathtt{L}(S_2)$ iff *for any two adversary* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *issuing a polynomial number of similarity range queries, there exists a simulator such that the advantage that* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *can distinguish the view of real and ideal experiments is negligible.*

Theorem 2. *Our scheme is selectively secure with the leakage* $\mathtt{L}(S_1)$ *and* $\mathtt{L}(S_2)$ iff *the SHE technique is CPA-secure.*

Proof. According to Definition 5, our scheme is selective secure *iff* $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot distinguish the view of real and ideal experiments.

- $\mathcal{A}_1$ cannot distinguish the view of real and ideal experiments. Specifically, $\mathrm{View}_{\mathcal{A}_1,\mathrm{Real}}$ includes $\{E(\mathrm{T}), \{\mathrm{TK}_{\mathbf{z}_i}, E(\mathbf{q}_i), E(\delta_i)\}_{i=1}^{w_2}\}$, access pattern of $E(\mathrm{T})$, the AES ciphertexts in the candidate result and some SHE ciphertexts received in the privacy-preserving protocols. $\mathrm{View}_{\mathcal{A}_1,\mathrm{Ideal}}$ includes $\{E'(\mathrm{T}), \{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i),\ E'(\delta_i)\}_{i=1}^{w_2}\}$, access pattern of $E'(\mathrm{T})$, the AES ciphertexts in the candidate result and some SHE ciphertexts received in the privacy-preserving protocols. For the encrypted tree $E(\mathrm{T})$ and $E'(\mathrm{T})$, both of them contain internal nodes and leaf nodes. The internal nodes in $E(\mathrm{T})$ are encrypted by the invertible matrix encryption and those in $E'(\mathrm{T})$ are random vectors. Since the invertible matrix encryption is selectively secure as proved in [20], $\mathcal{A}_1$ cannot distinguish the internal nodes of $E(\mathrm{T})$ and $E'(\mathrm{T})$. The leaf nodes of $E'(\mathrm{T})$ are generated by self-blinding those of $E(\mathrm{T})$ and both of them are encrypted by the SHE encryption technique. Since the SHE encryption is CPA-secure, $\mathcal{A}_1$ cannot distinguish the leaf nodes of $E(\mathrm{T})$ and $E'(\mathrm{T})$. Similarly, $\mathcal{A}_1$ also cannot distinguish $\{\mathrm{TK}_{\mathbf{z}_i}, E(\mathbf{q}_i), E(\delta_i)\}_{i=1}^{w_2}$ and $\{\mathrm{TK}'_{\mathbf{z}_i}, E'(\mathbf{q}_i), E'(\delta_i)\}_{i=1}^{w_2}$. For the access pattern of $E(\mathrm{T})$, it is the same as that of $E'(\mathrm{T})$. For the SHE ciphertexts received in the privacy-preserving protocols, the CPA security of the SHE encryption can guarantee that the ciphertexts in the real experiment and those in the ideal experiment are indistinguishable. Thus, $\mathcal{A}_1$ cannot distinguish the real and ideal experiments.
- $\mathcal{A}_2$ cannot distinguish the view of real and ideal experiments. Specifically, $\mathrm{View}_{\mathcal{A}_2,\mathrm{Real}}$ includes the AES ciphertexts in the candidate result and query result, and some plaintexts received in the privacy-preserving protocols. $\mathrm{View}_{\mathcal{A}_2,\mathrm{Ideal}}$ includes the AES ciphertexts in the candidate result and query result, and some plaintexts received in the privacy-preserving protocols. Since both the AES ciphertexts in the real and ideal experiments are the ciphertexts of some random numbers, $\mathcal{A}_2$ cannot distinguish them. For the plaintext received in the privacy-preserving protocols, each plaintext contains at least one random numbers as described in Section 4.2. Then, the plaintexts in both real experiment and ideal experiment are random numbers. Thus, $\mathcal{A}_2$ cannot distinguish $\mathrm{View}_{\mathcal{A}_2,\mathrm{Real}}$ and $\mathcal{A}_2$ $\mathrm{View}_{\mathcal{A}_2,\mathrm{Ideal}}$.

Therefore, $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot distinguish the real and ideal experiments, so our scheme is selectively secure with the leakage $\mathtt{L}(S_1)$ and $\mathtt{L}(S_2)$. □

5.3 Resistance Against Cloud Inference Attack

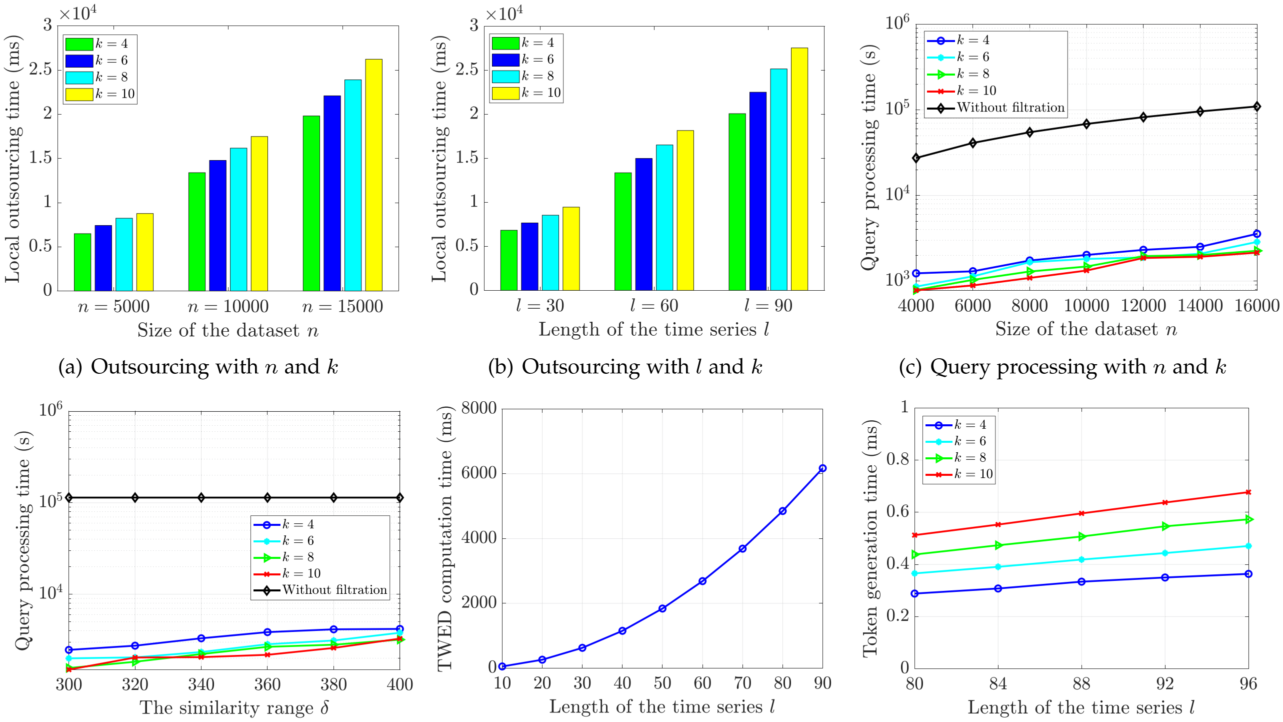
In this subsection, we show that our scheme can resist against the cloud inference attack. Since S_1 stores the encrypted data, only S_1 possibly launches the cloud inference attack. As described in our security model, S_1 can successfully launch the cloud inference attack *iff* S_1 has the capability to compare these encrypted data, and the capability to construct the ciphertext for any plaintext data at the same time. In our scheme, we employ invertible matrix encryption and SHE technique to respectively encrypt the internal nodes and leaf nodes of the kd-tree. The matrix encryption is a symmetric encryption with a secret key A. Since the matrix encryption cannot support homomorphic operations, S_1 cannot construct the ciphertext for any plaintext data. Although the SHE technique can support the homomorphic properties such that S_1 can construct the ciphertext for any plaintext. However, S_1 is not allowed to obtain any plaintext comparison relationship over SHE's ciphertexts. Hence, our scheme can resist the cloud inference attack.

6 Performance Evaluation

We evaluate the performance of our scheme from the aspects of local data outsourcing, similarity range query processing, query token generation and query result recovery.

6.1 Experimental Setting

We implemented our scheme in Java and conducted experiments on a machine with an Intel(R) Core(TM) i7-3770 CPU @3.40 GHz, 16 GB RAM and Windows 10 operating system. In our experiments, we use SHE technique to preserve the data privacy, and the security parameters are set to $k_0 = 2048$, $k_1 = 20$ and $k_2 = 160$. Then, the maximal multiplicative depth of the scheme is $\theta = \frac{k_0}{2k_2} - 1 \approx 5$. Due to the limitation of the multiplicative depth, S_1 and S_2 will call the bootstrapping protocol to bootstrap the ciphertext when computing the TWED. Specifically, after computing the ciphertext of an element $\mathrm{D}[i,j]$ in Algorithm 1, S_1 and S_2 will bootstrap it once. Based on these parameters, we choose a pair of public key and secret key, i.e., (pk, sk). Meanwhile, the session key ssk for the AES algorithm is a 256-bit



(a) Outsourcing with n and k (b) Outsourcing with l and k (c) Query processing with n and k

(d) Query processing with δ and k (e) TWED computation with l (f) Token generation with l and k

Fig. 3. Performance evaluation of our scheme.

random number. We evaluate the performance of our scheme on a real ElectricDevices dataset derived from UCR time series archive [21], which collects behavioral data about how consumers use electricity within the home to help reduce UK's carbon footprint. The similarity range query over this dataset can help to do classification for a given time series and provide carbon emissions suggestions to the user based on the similar time series data. This dataset contains 16,637 time series records in total, and the length of each record is 96. Since the values in the original dataset are real numbers, we convert them into integers by enlarging them 100 times and rounding their corresponding enlarged data. In addition, for each experiment, we conduct multiple times and the average result is reported.

6.2 Experimental Results

In this subsection, we will show the experimental results of local data outsourcing, similarity range query processing, query token generation, and query result recovery.

6.2.1 Local Data Outsourcing

In our scheme, the local data outsourcing phase builds and encrypts the kd-tree. The computational cost is related to three parameters, i.e., the size of the dataset n, the length of time series records l, and the number of pivots k.

- *The parameter n and k:* In Fig. 3a, we plot the computational cost of local data outsourcing versus with n and k. In this experiment, the parameters are set as $n = \{5000, 10000, 15000\}$, $k = \{4, 6, 8, 10\}$, and $l = 96$. From this figure, we can see that as n and k become larger, the computational cost of local data outsourcing will increase. This is because the larger the parameters n and k are, the larger the kd-tree is. Then, the computational cost will correspondingly increase.

- *The parameter l and k:* In Fig. 3b, we plot the computational cost of local data outsourcing varying with l and k. In this experiment, the parameters are set as $l = \{30, 60, 90\}$, $k = \{4, 6, 8, 10\}$, and $n = 16637$. From this figure, we can see that the computational cost of local data outsourcing also increases with the increase of l and k.

6.2.2 Similarity Range Query Processing

Since the similarity range query is processed by S_1 and S_2, we evaluate the computational cost of query processing and the communication overhead between S_1 and S_2.

Computational Cost. As described in Section 4.3, the computational cost of the similarity range query processing is affected by parameters n, l, k, and the similarity range δ.

- *The parameter n and k:* In Fig. 3c, we plot the computational cost of similarity range query processing versus n and k. In this experiment, the parameters are set as $n = \{4000, 6000, \ldots, 16000\}$, $k = \{4, 6, 8, 10\}$, $\delta = 350$, and $l = 96$. From this figure, we can see that the computational cost of query processing increases with the increase of n. Meanwhile, the difference of the computational costs in different k values is not significant. Moreover, to validate the efficiency of our filtration strategy, we plot the computational cost of the query processing without filtration strategy in Fig. 3c. The experiment result shows that our filtration strategy is pretty efficient. For example, when $n = 14000$ and $k = 10$, the computational cost with filtration is 1925 s, while that without filtration is 95,912 s. That is, the filtration strategy reduces about 97.9 percent computational cost.

- *The parameter δ and k:* In Fig. 3d, we plot the computational cost of the similarity query processing varying with δ. In this experiment, we set $\delta = \{300, 320, 340, 360, 380, 400\}$, $n = 16637$, $l = 96$, and $k = \{4, 6, 8, 10\}$. From Fig. 3d, we can

TABLE 1
Communication Overhead of TWED Computation With l

Time series' length l	20	40	60	80
Comm. Overhead (MB)	2.58	10.55	24.02	42.97

see that the computational cost increases as δ becomes larger. This is reasonable because the larger δ results in that more time series records can satisfy the query condition. Therefore, both the filtration stage and verification stage need to take more computational cost. Meanwhile, to validate the efficiency of our filtration strategy, we also plot the computational cost of similarity query processing without filtration in Fig. 3d. We can see that the computational cost with filtration is much less than that without filtration. For example, when $\delta = 380$ and $k = 4$, the computational cost with filtration is about 4137 s, while that without filtration is 113978 s. Thus, the filtration strategy reduces 96.3 percent computational cost, and it is pretty efficient.

- *The parameter l:* As shown in Section 4.3, the parameter l only has impact on the computational cost of TWED computation in the verification stage. Thus, we will evaluate how l affects the computational cost of TWED computation. Specifically, we plot the computational cost of TWED computation varying with l as shown in Fig. 3e. In this experiment, l varies from 10 to 90, and we see that the computational cost of TWED computation grows with the increase of l. Overall, our TWED computation algorithm is efficient. For example, when $l = 80$, computing the TWED between two series records takes 4851 ms.

Communication Overhead. The similarity range queries are processed by servers S_1 and S_2, and the query processing process brings the communication overhead between S_1 and S_2. As described in Section 4.3, the similarity range query contains the filtration phase and the verification phase. Meanwhile, only the verification phase requires the communication between S_1 and S_2. The main communication overhead comes from TWED computation and depends on the parameter l. In Table 1, we show the communication overhead of TWED computation varying with l. From this table, we can see that the communication overhead of TWED computation quadratically increases with l.

6.2.3 Query Token Generation

As described in Section 4.3, the computational cost of query token generation is related to parameters l and k. Thus, in Fig. 3f, we plot the computational of token generation varying with l and k. From this figure, we can see that when l and k become larger, generating a query token takes more computational cost. Overall, our query token generation algorithm is efficient. For example, when $k = 10$ and $l = 96$, generating a query token takes 0.67 ms.

6.2.4 Query Result Recovery

The query result is encrypted by the AES algorithm, so the computational cost of query result recovery depends on the computational cost of AES decryption. Since AES decryption is extremely efficient, our scheme is efficient in query result recovery. Specifically, decrypting a 2048-bit integer only takes 0.076 ms.

7 Related Work

In the literature, various privacy-preserving time series similarity query schemes were proposed. Existing schemes can be divided into three categories, L_p-norm based time series similarity query, edit distance based time series similarity query and DTW-based time series similarity query.

For the L_p-norm based time series similarity query schemes, the similarity between two time series records is measured by L_p-norm distance. Especially, L_2-norm, i.e., euclidean distance, is the most common considered distance. Various privacy-preserving similarity query schemes can be deployed to achieve euclidean distance based time series similarity query. Specifically, Wong *et al.* [5] designed an asymmetric scalar-product-preserving encryption scheme to achieve similarity query over encrypted database. The schemes [6], [7], [8], [9], [10], [11] use homomorphic encryption techniques (e.g., Paillier encryption) to design privacy-preserving similarity query schemes. The scheme [22] was designed for privacy-preserving Jaccard-based similarity query, and it can also support euclidean-based similarity query. In addition, since the euclidean distance computation can be transformed to be inner product computation [5], the function-hiding inner product encryption schemes [23], [24], [25], [26], [27], [28], [29], [30] also can support euclidean distance based similarity query. As a similarity metric, euclidean distance performs well for time series data in some applications. However, euclidean distance and even L_p-norm distance cannot support the similarity query over time series data with different lengths.

The edit distance based time series similarity query schemes [12], [13], [14], [31] can support the similarity query over time series data with different lengths. Specifically, the scheme [12] employed the garble circuit technique to design a privacy-preserving edit distance based similarity query, and it leverages the clustering method to reduce the computational cost of query processing. Although the scheme is efficient, it can only achieves approximate similarity query. The schemes [13], [14] proposed a similarity query algorithm by combining two-party secret sharing, garble circuit and partial homomorphic encryption techniques. However, all of the schemes [13], [14] suffer from the linear search efficiency. The scheme [31] deployed the customized garble circuit to achieve secure edit distance computation, but this scheme did not consider similarity query.

To achieve similarity queries over time series data with different lengths, dynamic time warping distance was proposed in [32]. Meanwhile, some DTW-based similarity query schemes [1], [2] were proposed. The scheme [1] was designed in a client-server mode, and it employed a homomorphic encryption technique to achieve similarity computation. On the one hand, this scheme focuses on DTW-based similarity evaluation, and did not consider similarity query. On the other hand, this scheme is run between the client and the server, the computational cost and communication cost of the client is large. The scheme [2] designed a secure multi-party computation protocol to achieve privacy-preserving DTW-based similarity query. In the proposed protocol [2], the authors used sliding window based DTW as the similarity metric, and Keogh's lower bound as a pruning condition for the DTW-based similarity query. Furthermore, they employed arithmetic sharing and garbled circuits techniques to preserve the privacy of the querying process. Although it can achieve efficient similarity query, it cannot return accurate query results. Therefore, existing similarity range query schemes over time series data still have some limitations.

8 Conclusion

In this paper, we have proposed an efficient and privacy-preserving similarity range query scheme for time series data. Specifically, we first organized time series data into a kd-tree by leveraging TWED's triangle inequality, and designed an efficient similarity range query algorithm for the kd-tree. Then, based on the SHE technique, we introduced a suite of privacy-preserving protocols to provide security guarantee for similarity range queries. Finally, we proposed our similarity range query scheme based on the similarity range query algorithm and our privacy-preserving protocols, in which we elaborate on two strategies to make our scheme resist against the cloud inference attack. Meanwhile, security analysis demonstrated that our scheme is privacy-preserving and can resist against the cloud inference attack. In addition, performance evaluation validated its efficiency.

Acknowledgments

This work was supported in part by NSERC Discovery Grants (04009), LMCRF-S-2020-03, Natural Science Foundation of Shaanxi Province (2019ZDLGY12-02), ZJNSF (LZ18F020003), and NSFC (U1709217, 61972304).

References

[1] H. Zhu, X. Meng, and G. Kollios, "Privacy preserving similarity evaluation of time series data," in *Proc. 17th Int. Conf. Extending Database Technol.*, 2014, pp. 499–510.

[2] X. Liu and X. Yi, "Privacy-preserving collaborative medical time series analysis based on dynamic time warping," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2019, pp. 439–460.

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

[4] Volume of data/information created worldwide from 2010 to 2024. Accessed: Aug. 1, 2020. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

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

[6] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," *Proc. Netw. Distrib. Syst. Secur. Symp.*, vol. 68, 2011.

[7] 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.

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

[9] 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. 41 771–41 784, 2018.

[10] 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, 2019.

[11] A. Salem, P. Berrang, M. Humbert, and M. Backes, "Privacy-preserving similar patient queries for combined biomedical data," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 1, pp. 47–67, 2019.

[12] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 492–503.

[13] K. Cheng, Y. Hou, and L. Wang, "Secure similar sequence query on outsourced genomic data," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 237–251.

[14] T. Schneider and O. Tkachenko, "EPISODE: Efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2019, pp. 315–327.

[15] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 24–43.

[16] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. Ghorbani, "Achieving $O(\log^3 n)$ communication-efficient privacy-preserving range query in fog-based IoT," *IEEE Internet of Things J.*, vol. 7, no. 6, pp. 5220–5232, Jun. 2020.

[17] P. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 306–318, Feb. 2009.

[18] P. Marteau, "Time warp edit distance," *CoRR*, vol. abs/0802.3522, 2008.

[19] J. H. Friedman, F. Baskett, and L. J. Shustek, "An algorithm for finding nearest neighbors," *IEEE Trans. Comput.*, vol. C-24, no. 10, pp. 1000–1006, Oct. 1975.

[20] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.3004442.

[21] Y. Chen *et al.*, "The UCR time series classification archive," Jul. 2015. [Online]. Available: www.cs.ucr.edu/eamonn/time_series_data/

[22] T. T. N. Le and T. V. X. Phuong, "Privacy preserving jaccard similarity by cloud-assisted for classification," *Wireless Pers. Commun.*, vol. 112, pp. 1875–1892, 2020.

[23] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2015, pp. 470–491.

[24] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," in *Proc. 19th IACR Int. Conf. Public-Key Cryptogr.*, 2016, pp. 164–195.

[25] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2018, pp. 544–562.

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

[27] G. Sheng, T. Wen, Q. Guo, and Y. Yin, "Privacy preserving inner product of vectors in cloud computing," *Int. J. Distrib. Sensor Netw.*, vol. 10, 2014, Art. no. 537252.

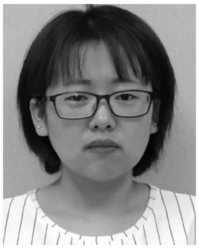
[28] L. Wang, T. Hayashi, Y. Aono, and L. T. Phong, "A generic yet efficient method for secure inner product," in *Proc. Int. Conf. Netw. Syst. Secur.*, 2017, pp. 217–232.

[29] F. Benhamouda, F. Bourse, and H. Lipmaa, "CCA-secure inner-product functional encryption from projective hash functions," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2017, pp. 36–66.

[30] O. Stan, R. Sirdey, C. Gouy-Pailler, P. Blanchart, A. B. Hamida, and M. Zayani, "Privacy-preserving tax calculations in smart cities by means of inner-product functional encryption," in *Proc. 2nd Cyber Secur. Netw. Conf.*, 2018, pp. 1–8.

[31] R. Zhu and Y. Huang, "Efficient and precise secure generalized edit distance and beyond," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2020.2984219.

[32] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. 3rd Int. Conf. Knowl. Discov. Data Mining*, 1994, pp. 359–370.



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

Rongxing Lu (Fellow, IEEE) is currently an associate professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of nine best (student) paper awards from some reputable journals and conferences. Currently, he serves as the vice-chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). He is the Winner of 2016–2017 Excellence in Teaching Award, FCS, UNB.

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

Jun Shao received the PhD degree from the Department of Computer and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a post-doctoral fellow with the School of Information Sciences and Technology, Pennsylvania State University, State College, Pennsylvania, 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 network security and applied cryptography.

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

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.