2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS) | 979-8-3503-2433-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/MASS58611.2023.00070

Towards Efficient Privacy-Preserving Top-k Trajectory Similarity Query

Kelai Yi, Yuefeng Chen, Yuchen Su, Xiong Li*, Hongbo Liu*, Huan Dai, Xiaonan Guo, Yingying Chen

***Abstract*—Similarity search for trajectories, especially the top-k similarity query, has been widely used in different fields, such as personalized travel route recommendation, car pooling, etc. Previous works have studied top-k similarity trajectory query in plaintext, but the increasing attention to privacy protection makes top-k similarity query on trajectory data become a challenge. In this paper, we propose a privacy-preserving top-k similarity query scheme over large-scale trajectory data based on Hilbert curve and homomorphic encryption. Towards this end, we first define a spatio-temporal trajectory similarity measure that supports homomorphic computation under ciphertext based on numerical integration algorithm for discrete trajectory data. A new filter-and-refine strategy for similarity query is also proposed to filter out the dissimilar trajectories based on Hilbert curve and refine the remaining trajectories with a secure average comparison protocol over the encrypted data. Finally, the exact query results can be obtained through Hilbert curve decoding. Our security analysis demonstrates that both locations and identities of the queried trajectories are preserved from the inference attack, and so does the privacy of the query user's trajectory. Meanwhile, extensive experimental results show that the proposed scheme can filter out 95% dissimilar trajectories with over 99% average precision, achieving higher query efficiency than the state-of-the-art techniques.**

***Index Terms*—Trajectory similarity query, Location privacy protection, Hilbert curve, Homomorphic encryption**

I. Introduction

With the rapid development of the mobile sensing and global positioning technologies, location-based services (LBSs) are becoming increasingly popular and important, such as Meituan, Didi, Ctrip, etc. Due to the widespread application of LBSs, massive amount of trajectory and location data has been collected by location service provider. Especially for large-scale trajectory data, their huge social and application values enable wide use in various fields (e.g., traffic and transportation optimizations). As a typical application with respect to large-scale trajectories, the top-k similarity search returns the k most similar trajectories for a given trajectory, which is often used in LBSs such as tourist route design, carpooling, and social network personalized recommendation, etc. However, the trajectory or location in LBSs contains a lot of sensitive information of the user, such as frequently visited places, home addresses, work places, etc. These information can be revealed by data mining or statistical analysis techniques, which brings a great threat to the privacy of users or even national security. For instance, U.S. military bases and patrol routes were leaked by the heat map of the fitness app Strava[1]. Many existing studies on similarity search or top-k similarity search are primarily based on plaintext without taking the privacy issues into accounts [1-3]. Therefore, location privacy protection has been extremely necessary. In order to protect location privacy in LBSs, not only are the privacy-preserving policies [4] needed, but also different technical means should be proposed. Currently, there are already some mature solutions for location privacy protection in LBSs, which can be generally classified as obfuscation-based approaches and cryptography-based approaches. The main idea of obfuscation-based location privacy protection approaches is to hide the user's real location through techniques such as cloaking [5], dummy locations [6], and differential privacy [7], while the cryptography-based approaches preserve location privacy via some cryptographic tools such as space transformation [8], secure multiparty computation (SMC) [9], and private information retrieval (PIR) [10]. However, neither can achieve high accuracy and high efficiency on location privacy protection at the same time.

In trajectory similarity search, only a few works have investigated privacy-preserving trajectory similarity search. Liu et al. [11] first study secure similarity computation of encrypted trajectories based on data packing technique. However, it only considers the similarity calculation between any two encrypted trajectories, which may lead to extremely high search complexity for similarity queries on big data. Teng et al. [12] propose a secure trajectory similarity search scheme based on a bi-directional similarity measurement and improve the search efficiency by secure signature matching. Although [12] can be extended to top-k similarity search on trajectories, it is hard to predetermine a proper threshold for the query request. Considering trajectory clustering, Guan et al. [13] propose a privacy-preserving scheme for trajectory range query of discrete Fréchet distance, which returns all the identities of trajectories that satisfy the distance threshold in the request. But the user cannot identify the exact top-k results from the output identities since they are obtained only by verification. Moreover, the aforementioned methods cannot obtain the top-

*Xiong Li and Hongbo Liu are the corresponding authors. E-mail: {lixiong, hongbo.liu}@uestc.edu.cn.

K. Yi, Y. Chen, Y. Su, X. Li and H. Liu are with the Dept. of CSE, University of Electronic Science and Technology of China, Chengdu 611731, China.

H. Dai is with the Dept. of EIE, Suzhou University of Science and Technology, Suzhou 215009, China.

X. Guo is with the Dept. of IST, George Mason University, Fairfax, VA 22030.

Y. Chen is with WINLAB, Rutgers University, North Brunswick, N.J. 08902-3390.

[1]https://www.wired.com/story/strava-heat-map-military-bases-fitness-trackers-privacy/

2155-6814/23/$31.00 ©2023 IEEE
DOI 10.1109/MASS58611.2023.00070

k trajectories apart from the similarity values [11, 12] or the identities [13], and they just consider the spatial features of the trajectory but ignore the temporal features, which are also important features of the object's moving patterns.

An accurate and efficient top-k query mechanism is needed. But to achieve such efficient privacy-preserving top-k trajectory similarity query, the following challenges should be tackled.

1) Similarity measurement over encrypted trajectory data: Most trajectory similarity measures are based on dynamic programming algorithms, which involve a large number of recursive operations. For the encrypted data, it becomes a difficult and complex task since multiple times of comparisons and branching operations on the ciphertexts are called recursively. Thus, a similarity measure should be carefully designed to evaluate the trajectory similarity in the ciphertext state.

2) Efficient privacy-preserving top-k trajectory similarity search: Intuitively, the top-k similarity query on large-scale encrypted trajectory dataset can be achieved by first computing the similarities between every two trajectories and then sorting them in the ciphertext state, which consumes a huge amount of computational cost and communication overhead. Meanwhile, most of trajectories in the dataset are very dissimilar to the queried one, which leads to many unnecessary operations. Therefore, an efficient privacy-preserving top-k trajectory similarity search scheme that filters before refining is in need.

In order to overcome the above challenges, we propose an efficient Privacy-preserving Top-k Trajectory Similarity Query scheme (PTTSQ), and the contributions are as follow:

- We define a new spatio-temporal similarity measure for discrete trajectories, which facilitates the distance calculation of large-scale encrypted trajectories. Besides, We design a Hilbert-based filtering method to screen out the dissimilar trajectories while preserving privacy.
- We propose a privacy-preserving top-k trajectory similarity query scheme (PTTSQ) over large-scale trajectory data, which enables the user to retrieve the k most similar trajectories while preserving the privacy of the query requests, the query results and the trajectory datasets.
- Extensive experiments are conducted to demonstrate the effectiveness of the proposed PTTSQ, and the results show that most dissimilar trajectories can be filtered out in PTTSQ and it is computationally efficient for top-k queries compared to the state-of-the-art methods.

II. Preliminaries

In this section, we first define a spatio-temporal trajectory similarity measure (DSED) used in PTTSQ. Next, we briefly introduce the relevant technical tools.

A. Discrete Synchronous Euclidean Distance (DSED)

To facilitate the spatio-temporal distance measurement [14] in the ciphertext state, we formally define a new similarity measure for discrete trajectories.

Suppose that $T^A = \{\tau_i^A\}_{i=1}^{l_a} = \{(t_1^A, p_1^A), (t_2^A, p_2^A), ..., (t_{l_a}^A, p_{l_a}^A)\}$ and $T^B = \{\tau_j^B\}_{j=1}^{l_b} = \{(t_1^B, p_1^B), (t_2^B, p_2^B), ..., (t_{l_b}^B, p_{l_b}^B)\}$ are two trajectories of moving objects, where t_i^A, t_j^B are the sampled timestamps, and p_i^A, p_j^B are the corresponding geolocations ($i \in [1, l_a]$, $j \in [1, l_b]$). Then, the distance from a sampling point $\tau_i^A = (t_i^A, p_i^A)$ to another trajectory T^B can be calculated by linear interpolation as follows.

$$Dist(\tau_i^A, T^B) = d(p_i^A, \hat{p}_j^B) = d(p_i^A, \frac{t_{j+1}^B - t_i^A}{t_{j+1}^B - t_i^B} \cdot p_j^B + \frac{t_i^A - t_j^B}{t_{j+1}^B - t_i^B} \cdot p_{j+1}^B), \tag{1}$$

where $d()$ denotes the Euclidean distance, and (t_j^B, p_j^B), (t_{j+1}^B, p_{j+1}^B) are two temporally adjacent points in T^B satisfying $t_j^B \le t_i^A < t_{j+1}^B$. Then, $\langle t_i^A : (p_i^A, \hat{p}_j^B)\rangle$ is called a *synchronous pair* at point τ_i^A. Similarly, we have $\langle t_j^B : (p_j^B, \hat{p}_i^A)\rangle$ at point τ_j^B.

To compute the **D**iscrete **S**ynchronous **E**uclidean **D**istance (DSED), we traverse all the sampling points in T^A and T^B, and obtain $(l_a + l_b - 2)$ synchronous pairs at each recorded timestamp in chronological order. Then, for the u-th synchronous pair, $u \in [1, l_a + l_b - 2]$, we have $s_u = Dist(\tau_i^A, T^B)$ or $s_u = Dist(\tau_j^B, T^A)$. Finally, the DSED can be defined as follows.

$$\boldsymbol{DSED(T^A, T^B)} = (\frac{\Delta t_{1,2} \cdot s_1 + \Delta t_{h-1,h} \cdot s_h}{2} + \sum_{u=2}^{h-1} \frac{\Delta t_{u-1,u+1}}{2} \cdot s_u) \cdot \frac{1}{\Delta t_{1,h}}, \tag{2}$$

where $h = l_a + l_b - 2$ and $\Delta t_{i,j} = t_i - t_j$.

To further illustrate the computation of DSED, an example is given as shown in Fig. 1. Suppose T^A and T^B are two trajectories ($l_a = 3, l_b = 4$). In the order of time, we have 5 synchronous pairs at timestamp $\{4, 5, 7, 9, 15\}$, and the distance between points within each pair is computed as s_u, $u \in [1, 5]$. Then, according to Eq.(2), the DSED value between T^A and T^B is equal to $(s_1 + 3s_2 + 4s_3 + 8s_4 + 6s_5)/22$.

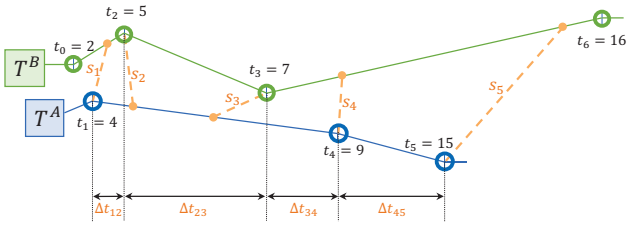


Fig. 1: Illustration of trajectory distance computation.

B. Hilbert Curve

Hilbert curve [15] is a space-filling curve that can traverse through all cells in a multi-dimensional space. It has been widely applied in spatial indexing of large-scale databases for its superior locality-preserving property [16]. A Hilbert curve in 2-D space can be determined by four parameters, i.e., $\phi = \{N, \theta, P_0, \gamma\}$, where N is the curve order, $\theta \in \{\sqcap, \sqcup, \sqsubset, \sqsupset\}$ denotes the curve's orientations, $P_0 = (x_0, y_0)$ is the starting point, and γ is the scale factor. Since the curve passes through each cell exactly once, a unique integer in $[0, 2^{2N} - 1]$ is assigned to each cell, which represents its encoding *H-value*. Then, each coordinate point $p = (x, y)$ in the space can be indexed with ϕ as a space decoding/encoding key [8], and

the original coordinates cannot be revealed from the encoded values without knowing ϕ. Accordingly, we can represent the encoding algorithm **HIL.Enc** as $v^{\phi}: p \rightarrow H$, where $v^{\phi}()$ is a bijective function from $\mathbb{R}^2$ to $\mathbb{R}^1$, and its inverse function is also known as the decoding algorithm **HIL.Dec**.

C. SHE Cryptosystem

SHE [17] is an efficient symmetric homomorphic encryption algorithm that supports homomorphic addition and multiplication, and it contains three algorithms:

- $\mathbf{SHE.KeyGen}(k_0, k_1, k_2)$: Given a set of security parameters $\{k_0, k_1, k_2\}$ satisfying $k_1 \ll k_2 < k_0$, the key generation algorithm outputs the secret key $sk = (p, q, \mathcal{L})$, where p, q are two large prime numbers in $\{0,1\}^{k_0}$ and $\mathcal{L} \in \{0,1\}^{k_2}$ is a random number. Next, it sets $\mathcal{N} = pq$, the public parameter $pp = (k_0, k_1, k_2, \mathcal{N})$ and the basic message space $\mathcal{M} = [-2^{k_1-1}, 2^{k_1-1})$.
- $\mathbf{SHE.Enc}(sk, m)$: Given a secret key sk and a plaintext m, the encryption algorithm outputs a ciphertext $E(m) = (r\mathcal{L} + m)(1 + r'p) \bmod \mathcal{N}$, where $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$ are random numbers.
- $\mathbf{SHE.Dec}(sk, E(m))$: Given sk and $E(m)$, the decryption algorithm recovers the plaintext m' by computing $m' = (E(m) \bmod p) \bmod \mathcal{L} = (r\mathcal{L} + m) \bmod \mathcal{L}$. If $m' < \mathcal{L}/2, m = m'$, otherwise $m = m' - \mathcal{L}$.

For the SHE under public key setting, by setting $pk = \{E(0)_1, E(0)_2, pp\}$, a message $m \in \mathcal{M}$ can also be encrypted by $E(m) = m + r_1 \cdot E(0)_1 + r_2 \cdot E(0)_2 \bmod \mathcal{N}$, where $E(0)_i$ is a ciphertext of 0, and r_i is a k_2-bit random number, $i = 1, 2$.

D. Proxy Re-encryption

Proxy re-encryption (PRE) [18] is a special type of public key encryption that allows a proxy to convert a ciphertext encrypted under the delegator i's key into an encryption of the same message under the delegatee j's key, while the proxy has no knowledge of the original message. A PRE often consists of the following algorithms:

- $\mathbf{PRE.GenKey}(k_i)$: Given a security parameter k_i as an input, the PRE key generation algorithm outputs a pair of secret and public keys $\{sk_i, pk_i\}$ for user i.
- $\mathbf{PRE.ReKey}(\{sk_i, pk_i\}, \{sk_j, pk_j\})$: Given two pairs of keys, where sk_j is optional, the PRE re-encryption key generation algorithm outputs a re-encryption key rk_{ij}.
- $\mathbf{PRE.Enc}(pk_i, m)$: Given a public key pk_i and a message m, the PRE encryption algorithm outputs a ciphertext c_i of user i.
- $\mathbf{PRE.ReEnc}(rk_{ij}, c_i)$: Given a re-encryption key rk_{ij} and user i's ciphertext c_i, the PRE re-encryption algorithm outputs a re-encryption ciphertext c_j for user j.
- $\mathbf{PRE.Dec}(sk_j, c_j)$: Given a secret key sk_j and a ciphertext c_j, the PRE decryption algorithm outputs a plaintext.

III. Problem, Models and Design Goals

In this section, we formalize the top-k trajectory similarity query problem. Then, we introduce the system model, security model and the design goals of PTTSQ.

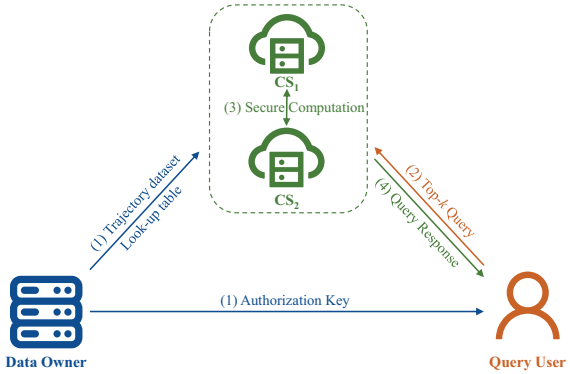


Fig. 2: System Model.

A. Problem Definition

Given a trajectory dataset $\mathcal{T}$ containing n trajectories with identity information, a query trajectory Q and an integer number k, the Privacy-preserving Top-k Trajectory Similarity Query outputs the k most similar trajectories to Q along with their identities and distance values in a privacy-preserving manner, i.e., to find $\mathcal{R} \subseteq \mathcal{T}$, such that $|\mathcal{R}| = k$, for $\forall\ R \in \mathcal{R}$ and $\forall\ T \in \mathcal{T} - \mathcal{R}$, we have $DESD(R, Q) \leq DESD(T, Q)$. In this paper, the identity information and DSED values in the results $\mathcal{R}$ are also denoted as $\mathcal{R}.ID$ and $\mathcal{R}.D$, respectively.

B. System Model

The system model of PTTSQ is shown in Fig. 2, which mainly consists of three types of entities: a data owner DO, dual cloud servers (CS_1 and CS_2), and a query user (QU).

- **Data Owner** DO**:** DO has a large dataset with n trajectories, i.e., $\mathcal{T} = \{\langle ID_i, T^i \rangle\}_{i=1}^{n}$, where T^i, ID_i denotes the i-th trajectory and its identity, respectively. For instance, an item of ID_i may include sensitive personal information such as an individual's name, ID number and mobile phone number, etc. To save computational and storage costs, DO outsources the trajectory data and similarity search services to cloud servers. For privacy concerns, DO encrypts the dataset $\mathcal{T}$ before data outsourcing. Meanwhile, it constructs a Hilbert-based look-up table Γ for the privacy-preserving search on the cloud servers.
- **Query User** QU**:** QU is a query user authorized by DO. QU can initiate a top-k trajectory similarity query request $\{Q, k\}$ to CS_1, and obtains the query results including identities, trajectories and DSED values without revealing privacy, i.e., $\mathcal{R} = \{\langle ID_j, R^j, D_j \rangle\}_{j=1}^{k}$, where $D_j = DSED(R^j, Q)$.
- **Cloud Servers** (CS_1**,** CS_2**):** Generally, privacy preserving computing on a single cloud server is often difficult and inefficient, so dual-cloud model is adopted in the system. CS_1 stores the Hilbert-based look-up table, the encoded trajectory data and the encrypted identity information, while CS_2 keeps the secret key of SHE. When receiving a query request from QU, CS_1 initially filters the massive trajectory data to obtain K candidate similar trajectories. Then through the secure protocol, the two cloud servers further refine the actual k most similar trajectories and return them as the query results to QU.

TABLE I: Notations

Notation	Description
Q	The queried trajectory submitted by QU
$\mathcal{T}$	A dataset of n trajectories owned by DO
$\mathcal{F}$	The candidate result set of K trajectories after filtration
$\mathcal{R}$	The final result set of k trajectories after refinement
n, K, k	The number of trajectories in dataset $\mathcal{T}$, $\mathcal{F}$, $\mathcal{R}$
ID_i, T^i	The identity information and trajectory in $\mathcal{T}$
(t_j^i, p_j^i)	The j-th timestamp and geo-location of T^i
l_i	The trajectory length of T^i
id	The index of the trajectory in $\mathcal{T}$
ϕ	The parameters of Hilbert curve, i.e., $\{N, \theta, P_0, \gamma\}$
N, θ, P_0, γ	The order, orientation, starting point and scale factor
Γ	The Hilbert-based look-up table
sk, pk	The secret key and public key of SHE encryption
pk_d, pk_q, pk_c	The public key of DO, QU, CS_1
rk_{dq}	The re-encryption key created by DO for QU

C. Security Model

In PTTSQ, all above three entities are assumed to be *honest-but-curious*, i.e., they perform the protocol honestly but may attempt to infer privacy information of other parties. Besides, CS_1 and CS_2 are supposed to be non-colluding entities, which is realistic since the commercial competition and interest conflicts between different cloud service providers [19]. Furthermore, DO and QU are considered not to collude with CS_1 or CS_2 for the sake of their own trajectory privacy. As semi-honest participants, cloud servers are supposed to launch cloud inference attacks to get the plaintext of the trajectory data, the identity information, the query requests and results. In addition, QU and DO may want to reveal the privacy of each other's trajectory data.

D. Design Goals

Based on above system and security models, we aim to propose a PTTSQ scheme with following goals:

1) Query Precision of Top-k: The filtration method is adopted to reduce similarity computations on the large-scale trajectory data, which may slightly weaken the query precision of top-k. Therefore, the precision of returned query results $\mathcal{R}$ should be ensured.

2) Privacy Preservation: PTTSQ should protect the data privacy of DO, the query request and result privacy of QU. That is, the dataset $\mathcal{T}$ owned by DO cannot be revealed by CS_1 and CS_2, and that QU can only get the query result $\mathcal{R}$ of k trajectories without knowing other knowledge of $\mathcal{T} - \mathcal{R}$. Besides, the privacy of query trajectory Q, the result trajectory set $\mathcal{R}$ and the similarity values $\{DSED(R^j, Q)\}_{j=1}^k$ should be preserved against DO, CS_1 and CS_2.

3) Efficiency: Similarity search on encrypted trajectory data will greatly increase the computational cost and communication overhead of the system, so PTTSQ should be efficient for similarity search.

IV. Our Proposed Scheme: PTTSQ

In this section, we describe the proposed privacy-preserving trajectory top-k query scheme (PTTSQ) in detail, before which we first introduce a modified Hilbert distance used in PTTSQ. The used notations are summarized in Table I.

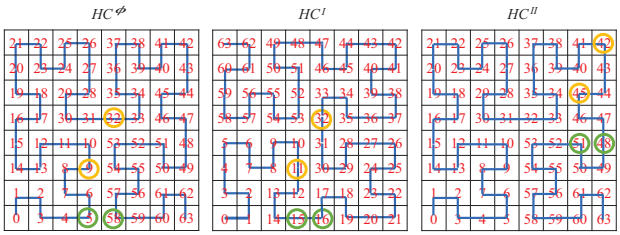


Fig. 3: An example of modified Hilbert distance.

A. Modified Hilbert Distance

In order to search the trajectories efficiently while preserving privacy, PTTSQ filters out the dissimilar trajectories preliminarily by combining a cluster of Hilbert curves. In other words, for the approximate but quick computation of trajectory similarity, we introduce a modified Hilbert distance based on the locality-preserving property of Hilbert curve [16].

Firstly, given a Hilbert curve HC^{ϕ} with parameter ϕ, the distance between every two points can simply be approximated by calculating the difference of their H-values, i.e.,

$$d_H(p_i, p_j) = |v^{\phi}(p_i) - v^{\phi}(p_j)| = |H_i^{\phi} - H_j^{\phi}|, \quad (3)$$

where p_i, p_j are two coordinate points in high-dimensional space and H_i^{ϕ}, H_j^{ϕ} denote their H-values on HC^{ϕ}.

However, errors of distance estimation may occur in some particular circumstances where the points with separated H-values are actually very close in the 2-D space. To ensure the precision of filtration, we modify the Hilbert-based approximate distance in Eq.(3) by combining a cluster of Hilbert curves. Specifically, they are generated from HC^{ϕ} by doing rotations and translations, i.e., varying parameters θ and P_0 in ϕ. Then, given a cluster of Hilbert curves $\mathcal{HC} = \{HC^{\phi}, HC^{I}, HC^{II}, \ldots, HC^{r}\}$ with same scale γ, whose encoding rules are $\mathcal{V} = \{v^{\phi}; v^{I}, v^{II}, \ldots, v^{r}\}$, the **modified Hilbert distance** is defined as the minimum value of the Hilbert-based distances on all above Hilbert curves, i.e.,

$$d_{\mathcal{H}}(p_i, p_j) = \min_{\eta \in \varphi} |v^{\eta}(p_i) - v^{\eta}(p_j)| = \min_{\eta \in \varphi} |H_i^{\eta} - H_j^{\eta}|, \quad (4)$$

where $\varphi = \{\phi, I, II, \ldots, r\}$.

To better understand the modified Hilbert distance, Fig. 3 shows a simple example of three Hilbert curves $\mathcal{HC} = \{HC^{\phi}, HC^{I}, HC^{II}\}$, in which HC^{I} and HC^{II} are generated by rotating and translating HC^{ϕ} respectively, i.e., $\{\theta^{\phi}, \theta^{I}, \theta^{II}\} = \{\sqcap, \sqsupset, \sqcap\}$, $\{P_0^{\phi}, P_0^{I}, P_0^{II}\} = \{(0,0), (0,0), (-3,-3)\}$. Given points $p_1 = (3,0)$ and $p_2 = (4,0)$, their H-values encoded by the three Hilbert curves are $H_1^{\phi} = 5$, $H_1^{I} = 15$, $H_1^{II} = 51$, $H_2^{\phi} = 58$, $H_2^{I} = 16$, $H_2^{II} = 48$. Then the modified Hilbert distance $d_{\mathcal{H}}(p_1, p_2) = \min_{\eta \in \{\phi, I, II\}} |H_1^{\eta} - H_2^{\eta}| = \min\{53, 1, 3\} = 1$. Similarly, for points $p_3 = (3,2)$ and $p_4 = (4,4)$, we have $d_{\mathcal{H}}(p_3, p_4) = \min_{\eta \in \{\phi, I, II\}} |H_3^{\eta} - H_4^{\eta}| = \min\{23, 21, 3\} = 3$.

B. Description of PTTSQ

Our PTTSQ scheme contains four parts, i.e., initialization, querying, preliminary filtration and refining, and the work flow is shown in Fig. 4. In such a filter-and-refine framework, most

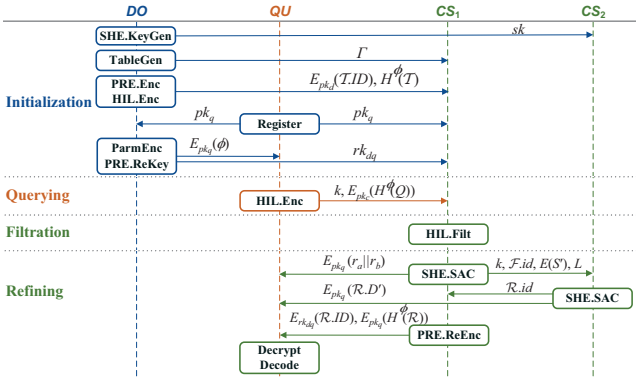


Fig. 4: System Flow Diagram.

dissimilar trajectories are firstly ruled out by the Hilbert curve-based filtration and only K candidates ($k \leq K \ll n$) are retained for the final refining to get the final top-k results.

1) Initialization: To initialize the system, DO generates a cluster of Hilbert curves $\mathcal{HC}$ and a public/secret key pair $\{pk, sk\}$ of SHE, and then constructs a lookup-table. Next, it outsources the encrypted data to CS_1. When a QU registers in the system, DO returns the encrypted curve parameters and generates a re-encryption key rk_{dq}. The details are as follow:
Step-1: Data Pre-processing.

DO first utilizes the Douglas-Peucker(DP) algorithm [20] for data thinning process and extracts representative point sequences from the trajectory. We assume that the points on the trajectory are located in cells of a $2^N * 2^N$ resolution square grid and the coordinates are integers. The pre-processed dataset of n trajectories are denoted as $\mathcal{T} = \{\langle ID_i, T^i\rangle\}_{i=1}^{n}$, where ID_i denotes the moving object's identity, and $T^i = \{(t_1^i, p_1^i), (t_2^i, p_2^i), \cdots, (t_{l_i}^i, p_{l_i}^i)\}$ is the i-th trajectory in $\mathcal{T}$.
Step-2: Hilbert-based Lookup-table Construction.

DO runs **TableGen** to obtain a Hilbert-based lookup-table Γ by following steps. First, DO picks a specific Hilbert curve parameter ϕ, then by varying orientation θ and starting point P_0, a cluster of Hilbert curves are generated as $\mathcal{HC} = \{HC^{\phi}; HC^{I}, HC^{II}, \cdots, HC^{r}\}$, where r is the number of transformed curves. Next, for each coordinate point (x, y) in the targeted space, DO runs $\mathbf{HIL.Enc}((x, y), \phi^{\eta})$ to generate the corresponding H-values of every HC^{η}, and obtains H_{xy}^{η}, where $x, y \in [0, 2^N - 1]$ and $\eta \in \{\phi, I, II, \ldots, r\}$. Meanwhile, DO runs $\mathbf{SHE.KeyGen}$ to generate a public/secret key pair $\{pk, sk\}$, and encrypts the above 2-D coordinates as $(E(x), E(y))$, correspondingly. At last, DO constructs the lookup-table Γ by matching the encrypted 2-D coordinates with the corresponding H-values of different Hilbert curves, i.e., $\Gamma = \langle H_{xy}^{\phi} : [H_{xy}^{I}, H_{xy}^{II}, \cdots, H_{xy}^{r}, (E(x), E(y))]\rangle$, where H_{xy}^{ϕ} is set as the key of the lookup-table for convenience of the later data retrieval. For instance, a look-up table of Hilbert curves with order 3 is illustrated in Table II.
Step-3: Trajectory Encoding & Encryption.

DO encodes the trajectories by running $\mathbf{HIL.Enc}(p_j^i, \phi)$ on each coordinate point, i.e., $H^{\phi}(\mathcal{T}) = \{H^{\phi}(T^i) = [(t_1^i, H^{\phi}(p_1^i)), (t_2^i, H^{\phi}(p_2^i)), \cdots, (t_{l_i}^i, H^{\phi}(p_{l_i}^i))] | i = 1, 2, \cdots, n\}$. Meanwhile, DO runs $\mathbf{PRE.Enc}(pk_d, ID)$ with public key for proxy re-encryption to obtain the encrypted trajectory identities $E_{pk_d}(\mathcal{T}.ID) = \{E_{pk_d}(ID_i) | i = 1, 2, \ldots, n\}$.

Finally, DO outsources the lookup-table Γ, the encoded trajectory data $H^{\phi}(\mathcal{T})$ and the encrypted identities $E_{pk_d}(\mathcal{T}.ID)$ to CS_1. Meanwhile, DO shares the sk of SHE to CS_2 secretly.
Step-4: Query User Registration.

To register to the system for the top-k similar trajectory query on DO's data, QU submits the public key pk_q to DO and CS_1. Then, DO encrypts the selected Hilbert curve parameter ϕ using pk_q and runs $\mathbf{PRE.ReKey}(sk_d, pk_d, pk_q)$ to generate a conversion key rk_{dq} for QU. Finally, DO returns $E_{pk_q}(\phi)$ and rk_{dq} to QU.

2) Querying: For the query trajectory $Q = [(t_1, q_1), (t_2, q_2), \cdots, (t_{l_0}, q_{l_0})]$, QU runs $\mathbf{HIL.Enc}(q_j, \phi)$ to encode each point of the query trajectory, i.e., $H^{\phi}(Q) = [(t_1, H^{\phi}(q_1), (t_2, H^{\phi}(q_2), \cdots, (t_{l_0}, H^{\phi}(q_{l_0}))]$. Then, QU encrypts $H^{\phi}(Q)$ with CS_1's public key and submits the query request $\{E_{pk_c}(H^{\phi}(Q)), k\}$ to CS_1, where k is an integer.

3) Preliminary Filtration: In this phase, CS_1 blindly processes similarity queries with the assistance of the Hilbert-based lookup-table Γ, and obtains K candidate trajectories as the preliminary filtration results $\mathcal{F}$.
Step-1: Hilbert-based Similarity Calculation.

After receiving $H^{\phi}(Q)$, CS_1 calculates the Hilbert-based similarity $DSED_{\mathcal{H}}$ between the query trajectory Q and each trajectory $T^i \in \mathcal{T}$ to approximate the value of DSED, where the modified Hilbert distance is adopted for computing s_u in Eq.(2). Thus, according to the look-up table Γ, CS_1 obtains $D^{\mathcal{H}} = \{D_i^{\mathcal{H}} = DSED_{\mathcal{H}}(Q, T^i) | (i = 1, 2, \cdots, n)\}$.
Step-2: Trajectory Filtration.

CS_1 filters out dissimilar trajectories by sorting the trajectories based on the similarity values $D^{\mathcal{H}}$, and retains a top-K candidate set as the preliminary filtration results $\mathcal{F}$ from the entire n trajectories , where $|\mathcal{F}| = K \ll n$. To facilitate further refining, CS_1 keeps the K filtered indexes as $\mathcal{F}.id$.

4) Refining: CS_1 and CS_2 further determine the final query results by operating a series of secure protocols on encrypted trajectory data. Finally, QU can recover the top-k query results, the corresponding similarities and identities by decoding and decryption algorithms.
Step-1: Encrypted Trajectory Retrieval.

To refine the exact top-k similar trajectories from $\mathcal{F}$, CS_1 looks up the SHE ciphertexts of the coordinates in the

TABLE II: An illustration of the look-up table

H_{xy}^{ϕ}	H_{xy}^{I}	H_{xy}^{II}	...	H_{xy}^{r}	$(E(x), E(y))$
000000	010101	101010	...	111111	$(E(0), E(0))$
000001	010110	101011	...	111100	$(E(0), E(1))$
000010	010111	101000	...	111101	$(E(1), E(1))$
⋮	⋮	⋮	⋮	⋮	⋮
111111	000000	010101	...	101010	$(E(7), E(7))$

query trajectory by matching their H^{ϕ}-values in table Γ, and obtains $E(Q) = [(t_1, E(q_1)), (t_2, E(q_2)), \cdots, (t_{l_0}, E(q_{l_0}))]$. Similarly, a set of K encrypted trajectories can be obtained by looking up the SHE ciphertexts of the coordinates in candidate trajectories, i.e., $E(\mathcal{F}) = \{E(T^j) = [(t_1^j, E(p_1^j)), (t_2^j, E(p_2^j)), \cdots, (t_{l_j}^j, E(p_{l_j}^j))] | j \in \mathcal{F}.id\}$.

Step-2: Homomorphic Calculation of Exact Distances.

After retrieving $E(\mathcal{F})$ and $E(Q)$, CS_1 computes the encrypted distance s_u in Eq.(2) between Q and each $T^j \in \mathcal{F}$ by homomorphic operations on SHE ciphertexts, denoted as $E(s_u^{(j)})$, where $j \in [1, K]$, $u \in [1, l_0 + l_j]$. Then, for each $T^j \in \mathcal{F}$ with trajectory length l_j, it obtains the sum of synchronous distances at all the timestamps by calculating:

$$E(S_j) = \Delta t_{1,2} E(s_1^{(j)}) + \Delta t_{h_j-1,h_j} E(s_{h_j}^{(j)}) + \sum_{u=2}^{h_j-1} \Delta t_{u-1,u+1} E(s_u^{(j)}),$$

where $h_j = l_0 + l_j$. Suppose that $L_j = \Delta t_{1,h_j}$ denotes the total time interval, then $DSED(Q, T^j) = S_j/(2L_j)$. However, SHE do not support the division and comparison operations of the ciphertext, thus the following $\mathbf{SHE.SAC}$ protocol is called to refine $\mathcal{F}$ over the ciphertext.

Step-3: Secure and Exact Top-k Search.

To find the exact top-k similar trajectories $\mathcal{R}$ among the candidate trajectories $\mathcal{F}$, the dual clouds run the Secure Average Comparison protocol $\mathbf{SHE.SAC}$ without collusion. Specifically, CS_1 first chooses two random numbers $r_a \in (0, 2^{k_1}), r_b \in (-2^{k_1}, 2^{k_1})$ and sends $E_{pk_q}(r_a \| r_b)$ to QU. Then it computes $\{E(S_j') = r_a \cdot E(S_j) + r_b \cdot L_j | j = 1, 2, \cdots, K\}$ and sends them to CS_2 with the corresponding L_j and id_j. On receiving $\{\langle E(S_j'), L_j, id_j \rangle\}_{j=1}^{K}$, CS_2 recovers S_j' by using sk and computes $\bar{D}_j' = S_j'/L_j$ $(j = 1, 2, \cdots, K)$. Next, since the DSED values of the K results are basically arranged in order after filtration, CS_2 can simply utilize heapsort method on D_j' to obtain the refined top-k results $\mathcal{R}.D' = \{\bar{D}_v' | v = 1, \ldots, k\}$ and the corresponding indexes, recorded as $\mathcal{R}.id$. Then, CS_2 returns $E_{pk_q}(\mathcal{R}.D')$ to QU, and sends $\mathcal{R}.id$ to CS_1.

Step-4: Proxy Re-encryption of Query Result.

On receiving the indexes $\mathcal{R}.id$, CS_1 retrieves the corresponding k encoded trajectories $H^{\phi}(\mathcal{R})$ and their encrypted identity information $E_{pk_d}(\mathcal{R}.ID)$ from $H^{\phi}(\mathcal{T})$ and $E_{pk_d}(\mathcal{T}.ID)\rangle$, respectively. Then, CS_1 re-encrypts $\mathcal{R}.ID$ by running $\mathbf{PRE.ReEnc}(rk_{dq}, \mathcal{R}.ID)$ with the proxy re-encryption key rk_{dq}. Finally, the ciphertexts $\{E_{rk_{dq}}(\mathcal{R}.ID), E_{pk_q}(H^{\phi}(\mathcal{R}))\}$ are sent to QU.

Step-5: Query Result Recovery.

At last, on receiving $\{E_{rk_{dq}}(\mathcal{R}.ID), E_{pk_q}(H^{\phi}(\mathcal{R})), E_{pk_q}(r_a \| r_b)\}$ from CS_1 and $E_{pk_q}(\mathcal{R}.D')$ from CS_2, QU gets the query results of identities, trajectories and DSED similarities in the following way, denoted as $\mathcal{R} = \{\langle ID_j, R^j, D_j \rangle\}_{j=1}^{k}$. To obtain the top-$k$ trajectories, the user runs $\mathbf{HIL.Dec}(\phi, H^{\phi}(\mathcal{R}))$ to recover the trajectory locations. Besides, the identity information of the trajectories can be recovered by running $\mathbf{PRE.Dec}(sk_q, E_{rk_{dq}}(\mathcal{R}.ID))$. Moreover, QU obtains the exact top-k distances $\mathcal{R}.D$ by calculating $\bar{D}_v = \frac{\bar{D}_v' - r_b}{r_a}$ for $v = 1, 2, \cdots, k$.

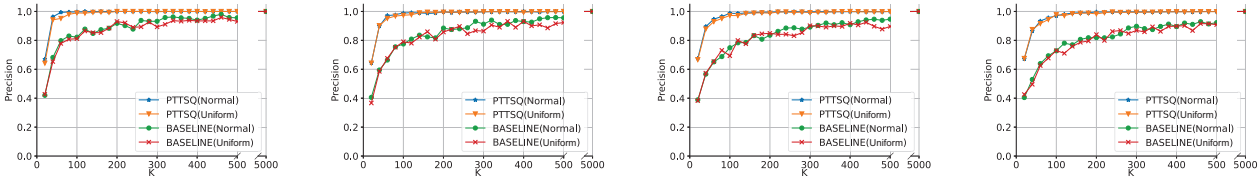
V. Security Analysis

In this section, we analyze the security of PTTSQ and show that PTTSQ achieves the aforementioned security goals.

1) Data privacy of the DO is preserved: During the system initialization stage, DO's trajectory data are encoded by a Hilbert curve HC^{ϕ} as $H^{\phi}(\mathcal{T})$ before being outsourced to CS_1. To recover these encoded trajectories, the parameter ϕ is necessary as a transformation key, without which CS_1 and CS_2 cannot infer DO's trajectories due to the blind evaluation based on Hilbert curve encoding [8]. Besides, although the encrypted trajectories $E(\mathcal{F})$ can be obtained by CS_1, the trajectories in $\mathcal{T}$ remain unknown to CS_1 for the lack of the secret key sk. Based on the non-collusive assumption of the dual cloud servers, even though CS_2 has sk, it just can recover D' from the ciphertexts of the disturbed DSED similarity values $E(D')$ by using the sk, which is only the intermediate result with no information about the original trajectories in $\mathcal{T}$. Therefore, the privacy of the trajectories owned by DO cannot be inferred by the dual could servers. Meanwhile, in PTTSQ, the identity information $\mathcal{T}.ID$ is encrypted by the DO with pk_d before uploading, and CS_1 cannot infer them without knowing sk_d. Even though CS_1 can conduct re-encryption on $E_{pk_d}(\mathcal{T}.ID)$ by using rk_{dq}, the obtained ciphertexts $E_{rk_{dq}}(\mathcal{T}.ID)$ can only be decrypted by using sk_q [18]. The dual cloud servers have neither sk_d nor sk_q, so they are prevented from inferring the identity information of any trajectory. For a curious QU who does not collude with the dual cloud servers, only the top-k trajectories $H^{\phi}(\mathcal{R})$ and their identities $E_{rk_{dq}}(\mathcal{R}.ID)$ are returned from the dual cloud servers. Then, although QU can recover the queried trajectories in $\mathcal{R}$ and their identities $\mathcal{R}.ID$ by using ϕ and sk_q respectively, it cannot infer the rest of trajectory data $\mathcal{T} - \mathcal{R}$. In a word, the trajectory data of DO is privacy-protected to CS_1, CS_2 and QU.

2) Privacy of QU's query request is preserved: In PTTSQ, QU's query request $E_{pk_c}(H^{\phi}(Q))$ is obtained by first encoding the query trajectory Q with the Hilbert curve of DO, and then encrypting it with the public key of CS_1. Even though $H^{\phi}(Q)$ can be decrypted by CS_1 using the secret key sk_c, the query trajectory Q still cannot be revealed by CS_1 without knowing the curve parameters used by DO. Similarly, both DO and CS_2 are not collude with CS_1, so they cannot obtain $H^{\phi}(Q)$ without knowing CS_1's secret key sk_c, and also cannot get QU's query request Q. Therefore, the query request of QU is privacy-protected to DO, CS_1 and CS_2.

3) Privacy of QU's query results is preserved: Based on the query responses $\{E_{rk_{dq}}(\mathcal{R}.ID), E_{pk_q}(H^{\phi}(\mathcal{R})), E_{pk_q}(r_a \| r_b)\}$ received from CS_1 and $\{E_{pk_q}(\mathcal{R}.D')\}$ received from CS_2, QU can recover the query results $\mathcal{R}$ by using the secret key sk_q. For the dual cloud servers, as above analysed in V.*1)*, they cannot reveal the identity information $\mathcal{R}.ID$ from $E_{rk_{dq}}(\mathcal{R}.ID)$ without knowing sk_q, and are also prevented from inferring the trajectories based on $H^{\phi}(\mathcal{R})$ because they have no knowledge of the curve parameter ϕ. Besides, the queried DSED similarities $\mathcal{R}.D$ are kept unknown to the dual



(a) The filtration precision ($k = 5$) (b) The filtration precision ($k = 10$) (c) The filtration precision ($k = 15$) (d) The filtration precision ($k = 20$)

Fig. 5: The filtration precision with K for $k = 5, 10, 15, 20$.

cloud servers via the **SHE.SAC** protocol, where CS_1 just conducts homomorphism operations based on SHE without knowing sk, while CS_2 only obtains the disturbed similarities D' from CS_1 without knowing r_a or r_b. Since the two pieces of ciphertexts $E_{pk_q}(D')$ and $E_{pk_q}(r_a \| r_b)$ are encrypted by the non-collusive dual cloud servers separately before being returned to QU, the actual queried similarity values $\mathcal{R}.D$ could be protected against CS_1, CS_2 and DO. Even though DO owns the whole trajectory dataset $\mathcal{T}$ and knows the Hilbert curve transformation key ϕ, it cannot infer the queried top-k trajectories in $\mathcal{R}$ from $E_{pk_q}(H^{\phi}(\mathcal{R}))$ or $E_{rk_{dq}}(\mathcal{R}.ID)$ due to the lack of sk_q. Thus, the query results including the queried identity information, top-k trajectories and the corresponding similarities are all privacy-protected to CS_1, CS_2 and DO.

VI. Performance Evaluation

In this section, we state our experimental settings and then evaluate our PTTSQ system from the aspects of the filtration precision, communication overhead and computational cost.

A. Experimental Settings

To validate the effectiveness of our system, we conducted experiments on a *Dell Precision 7920 Tower Server* with two *Intel Xeon Gold 6248R (96) @4GHz CPUs* and *128GB RAM running Ubuntu 20.04* using *Python 3.8*. Four trajectory datasets are applied for the evaluation, including real life datasets *T-drive*, *Geolife* [21, 22], and two synthetic datasets generated by random walk algorithm with a *uniform*-distributed and a *normal*-distributed pace accordingly. In the experiment, the numbers of trajectories n contained in *T-drive*, *Geolife*, *Uniform* and *Normal* are $2000, 2000, 5000, 5000$, and the corresponding lengths l are $100, 200, 100, 200$. In the evaluation, the querying map is set to a grid of $2^N \times 2^N$ for $N \in \{8, 9, 10, 11\}$, with the querying resolution ranging from $65,536$ up to $4,194,304$. For PTTSQ, a cluster of six Hilbert curves $\mathcal{HC} = \{HC^{\phi}; HC^{I}, HC^{II}, \cdots, HC^{VI}\}$ is used for the modified Hilbert similarity calculation, where $\mathcal{HC}$ is generated based on the original Hilbert curve parameter $\phi = \{11, \sqcap, (0,0), 5\text{m}\}$ by doing rotations for $\theta \in \{\sqcup, \sqsubset, \sqsupset\}$ and shifting the starting point for $P_0 \in \{(1,1), (2,2), (4,4)\}$. As a comparison, we set a baseline scheme which calculates the similarity based on a single curve HC^{ϕ}. For the SHE encryption system, the security parameters k_0, k_1 and k_2 are set to $2048, 24$ and 160, respectively. Besides, the query number k of top-k is set to $5, 10, 15, 20$ for the evaluation.

B. Evaluations and Experimental Results

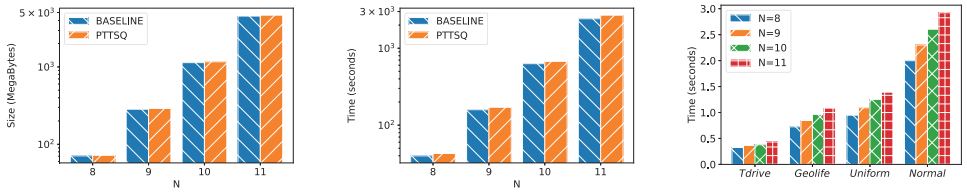
In this subsection, we first evaluate the filtration precision of top-k query and then analyze the communication overhead of PTTSQ. Finally, we compare the time cost of system initialization and query processing phases with latest works [12, 13] by conducting experiments on both real life and synthetic datasets.

1) The filtration precision: In a single Hilbert curve, two adjacent points may have low similarities. To alleviate this problem, modified Hilbert distance is designed in PTTSQ. To demonstrate the precision improvement of our Hilbert based filtration scheme, we carry out comparison experiments on filtering with/without modification, whose effectiveness is evaluated by the following defined precision: $pr = \frac{|R \cap C|}{|C|}$, where R denotes the preliminary result set of top-K similar candidates and C denotes the exact top-k result set.

We conduct top-5, top-10, top-15 and top-20 similarity query evaluation experiments on the *Uniform* and *Normal* with 5,000 trajectories, changing the filtration scope K from k to n, and repeat 100 times to evaluate the average precision in each case. The comparison results of filtration precision are shown in Fig. 5, from which we can see that for both *Uniform* and *Normal*, our modified Hilbert distance based PTTSQ significantly improves the filtration precision than the baseline scheme. More exactly, for top-5, top-10, top-15 and top-20 similarity query, our filtration method achieves more than 99% precision when setting the filtration scope $K = 125, 190, 215, 220$. It means that the preliminary filtering mechanism in PTTSQ is effective, which can reduce the computational cost while ensuring the search precision.

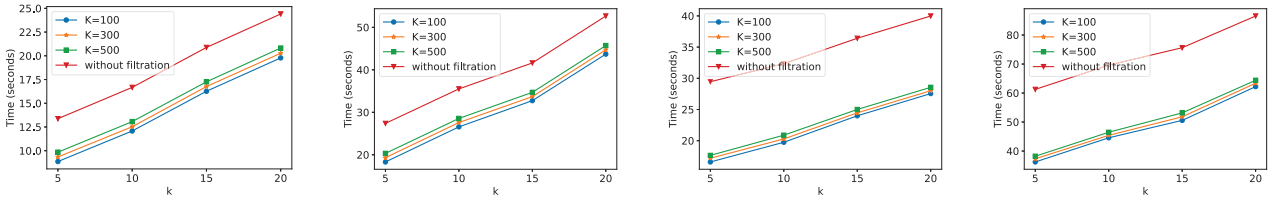
2) Communication overhead: For convenience, we denote Ω as the total number of grids in the target map, i.e., 2^{2N}. Let $\{E_S, E_H, E_P, E_{R_E}, E_{R_R}\}$ represent the operations of SHE encryption, Hilbert curve encoding, public key encryption, PRE encryption and PRE re-encryption algorithms, respectively, and the corresponding ciphertext lengths are expressed as $\{|E_S|, |E_H|, |E_P|, |E_{R_E}|, |E_{R_R}|\}$. Similarly, $\{D_S, D_H, D_P, D_{R_R}\}$ stand for the corresponding decryption or decoding operations.

In the initialization phase of PTTSQ, the communication overhead mainly comes from the delivery of the Hilbert-based look-up table Γ, which is $\Omega \cdot [(1+r) \cdot |E_H| + 2 \cdot |E_S|]$ bits, and the size of the look-up table Γ with different N is shown in Fig. 6(a). Besides, the communication overhead of uploading trajectory data and identities is $2 \sum_{i=1}^{n} l_i \cdot |E_H| + n \cdot |E_{R_E}| +$



(a) The size of Γ (b) The time of constructing Γ (c) The time of Hilbert encoding

Fig. 6: The initialization performance in Γ construction and trajectory encoding.



(a) *Tdrive* (b) *Geolife* (c) *Uniform* (d) *Normal*

Fig. 7: The query processing time of PTTSQ over different datasets.

$(1 + l_0) \cdot |E_P|$ bits. During the preliminary filtration phase, CS_1 takes $K \cdot |E_S|$ bits communication to send preliminary filtration results to CS_2. In the refining phase, CS_1 sends the encrypted random numbers and the encrypted top-k results to QU, and the communication overheads are $|E_P|$ bits and $k \cdot |E_{R_R}| + \sum_{j=1}^{k} l_j \cdot |E_P|$ bits. Meanwhile, CS_2 sends the encrypted similarity values to QU, which costs $k \cdot |E_P|$ bits communication.

3) Computational cost: In the system initialization phase, DO executes **TableGen** to generate Ω pairs of key-values in the look-up table that leads to 2Ω times E_S and $(1+r)\Omega$ times E_H. Secondly, for the local data outsourcing, the encryption of trajectory data consumes $2(l_q + \sum_{i=1}^{n} l_i)$ times E_H, n times E_{R_E}, one E_P and one **PRE.ReKey**. As for the computational cost during the query processing, CS_1 preliminarily filters the trajectories and then refines the results with the assistance of CS_2, which requires one **Hil.Filt** and one **SHE.SAC**, including $\mathcal{O}(nl)$ times homomorphic operations and K times D_S. Besides, the generation of encrypted query results needs $(1 + k + \sum_{j=1}^{k} l_j)$ times E_P and k times E_{R_R}. Finally, it takes QU k times D_{R_R}, and $\sum_{j=1}^{k} l_j$ times D_P, D_H to obtain the query results.

To exhibit the computational efficiency more clearly, we simulate the schemes and show the results as follows.

The Time of the Table Construction. To show the time of the table construction, we construct the look-up tables based on a cluster of six Hilbert curves $\mathcal{HC}$ (generated as in VI-A) for each curve order $N \in \{8, 9, 10, 11\}$, and compare the constructing time cost with the baseline scheme. As displayed in Fig. 6(b), the time cost of constructing Γ is mainly related to the order N of the selected Hilbert curve. That is because an increasing N means the more information since the total number of points in the space is 2^{2N}. Besides, there is no significant difference when appending transformations and increasing the cluster size of the curves, so it is convenient to setup a look-up table with several transformations to modify the approximation of trajectory similarities.

The Time of Trajectory Encoding. Intuitively, the computational cost of trajectory encoding increases as the size of trajectory data, i.e., the number of trajectories n and the average length l. We evaluate the time of Hilbert curve encoding on the above given four trajectory datasets, i.e., *T-drive*, *Geolife*, *Uniform* and *Normal*. As can be seen in Fig. 6(c), the trajectory encoding by Hilbert curve is highly efficient, and the time cost is linearly correlated to the size of trajectory dataset, and also increases as the curve order rises.

The Time of Filtration & Refining. We carry out experiments on the basis of an 11-order Hilbert-based look-up table as constructed above. To show the efficiency of our scheme, we simulate top-5, top-10, top-15 and top-20 similarity queries over four datasets. Hence, when varying $K = 100, 300, 500$, we can obtain the top-k query results in a different precision demands. A larger K may achieve higher precision but bring more calculations, and note that setting $K = n$ implies that refining all n trajectories without filtration. We displays the query processing time for our filter-and-refine procedure after initialization, which consists of the preliminary filtration time (t_F), refined searching time (t_R) and time of recovering query results. As can be seen in Fig. 7, the querying time increases with K. Since t_F is the filtration time over n trajectories at high speed, while t_R is the refined searching time over only K ($K \ll n$) trajectories, the filtration step over large-scale trajectory data is of great efficiency.

Total Time Cost. The total query efficiency of our PTTSQ is compared with related schemes STFSM [12] and PDRQ [13] by simulating the similarity queries on four datasets. Since the query efficiency of two comparison schemes is very low, we just randomly choose 50 trajectories of length 50 from four datasets in the comparative experiments. In our top-k similarity query scheme and their similarity range queries STFSM [12] and PDRQ [13], they all output 10 most similar trajectories. The comparison results of total query time are shown in Table III, from which we can see that for all four

datasets and different order N, both STFSM [12] and PDRQ [13] cost thousands of seconds or tens of thousands of seconds to perform a query. In contrast, the construction of the look-up table Γ in our PTTSQ takes tens of seconds or thousands of seconds, while the query user can complete a query within one second. What we need special attention here is that the query table of our PTTSQ can be constructed in advance, and it only needs to be constructed once for different queries.

STFSM [12] and PDRQ [13] are time-consuming for a dataset of only 50 trajectories, and they are not suitable for large-scale trajectory datasets in terms of computational efficiency. On the contrary, as previously demonstrated in Fig. 7, PTTSQ just needs no more than 70 seconds for a top-20 query among 2000 or 5000 trajectories, which is much lower than the query time of STFSM [12] and PDRQ [13] in 50 trajectories. Meanwhile, as illustrated in Fig. 5, PTTSQ can also achieve high-precision query. Therefore, PTTSQ is computationally efficient for large-scale trajectory datasets.

VII. Conclusion

In this paper, we propose a privacy-preserving top-k similarity query system, i.e., PTTSQ, for trajectory data based on a spatio-temporal similarity measure. To improve the query efficiency, we develop a filtration method by utilizing the location-preserving property of Hilbert curve, which maps the high-dimensional geo-location points into their H-values and screens out the dissimilar trajectories while preserving data privacy. Then, the preliminary filtered K trajectories are refined to get the exact query results. Security analysis shows that PTTSQ guarantees the privacy of the query request, query results, and DO's trajectory data. Finally, the experimental results show that PTTSQ has over 99% precision rate and higher computational efficiency compared to existing studies.

Acknowledgment

This work is supported in part by National Key R&D Program of China under Grants 2022YFB3103404, National Natural Science Foundation of China under Grants 62172080, and Natural Science Foundation of Sichuan Province under Grants 2023NSFSC0478.

References

[1] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *2007 IEEE 23rd International Conference on Data Engineering*, 2007, pp. 816–825.

[2] P. Zhao, W. Rao, C. Zhang, G. Su, and Q. Zhang, "SST: Synchronized spatial-temporal trajectory similarity search," *Geoinformatica*, vol. 24, no. 4, p. 777–800, Oct 2020.

[3] H. He, R. Li, S. Ruan, T. He, J. Bao, T. Li, and Y. Zheng, "Trass: Efficient trajectory similarity search based on key-value data stores," in *2022 IEEE 38th International Conference on Data Engineering*, 2022, pp. 2306–2318.

[4] R. Cover, "Platform for privacy preferences (p3p) project," 2000.

[5] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 620–629.

[6] S. Shaham, M. Ding, B. Liu, S. Dang, Z. Lin, and J. Li, "Privacy preservation in location-based services: A novel metric and attack model," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 3006–3019, Oct 2021.

[7] S. Yuan, D. Pi, X. Zhao, and M. Xu, "Differential privacy trajectory data protection scheme based on r-tree," *Expert Systems with Applications*, vol. 182, no. C, Nov 2021.

[8] A. Khoshgozaran, H. Shirani-Mehr, and C. Shahabi, "Blind evaluation of location based queries using space transformation to preserve location privacy," *GeoInformatica*, vol. 17, no. 4, pp. 599–634, 2013.

[9] M. Ashouri-Talouki, A. Baraani-Dastjerdi, and A. A. Selquk, "Glp: A cryptographic approach for group location privacy," *Computer Communications*, vol. 35, no. 12, pp. 1527–1533, 2012.

[10] Q. Liu, Z. Hao, Y. Peng, H. Jiang, J. Wu, T. Peng, G. Wang, and haobo Zhang, "Secvkq: Secure and verifiable knn queries in sensor–cloud systems," *Journal of Systems Architecture*, vol. 120, p. 102300, 2021.

[11] A. Liu, K. Zhengy, L. Liz, G. Liu, L. Zhao, and X. Zhou, "Efficient secure similarity computation on encrypted trajectory data," in *2015 IEEE 31st International Conference on Data Engineering*, 2015, pp. 66–77.

[12] Y. Teng, Z. Shi, F. Zhao, G. Ding, L. Xu, and C. Fan, "Signature-based secure trajectory similarity search," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2021, pp. 196–206.

[13] Y. Guan, R. Lu, Y. Zheng, S. Zhang, J. Shao, and G. Wei, "Achieving privacy-preserving discrete frechet distance range queries," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2022.

[14] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, no. 1, p. 3–32, Oct 2019.

[15] A. R. Butz, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, no. 4, pp. 314–330, 1968.

[16] Z. He and A. B. Owen, "Extensible Grids: Uniform Sampling on a Space Filling Curve," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 78, no. 4, pp. 917–931, 10 2015.

[17] S. Zhang, S. Ray, R. Lu, Y. Zheng, Y. Guan, and J. Shao, "PPAQ: Privacy-preserving aggregate queries for optimal location selection in road networks," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[18] Z. Qin, H. Xiong, S. Wu, and J. Batamuliza, "A survey of proxy re-encryption for secure data sharing in cloud computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2016.

[19] 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, p. 1–13, May 2019.

[20] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, May 2015.

[21] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, p. 99–108.

[22] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife2.0: A location-based social networking service," in *Proceedings of the 10th International Conference on Mobile Data Management (MDM 2009)*, May 2009, p. 357–358.

TABLE III: Total time cost on four datasets with 50 trajectories and $N = 8, 9, 10, 11$

	System Initialization (sec)	**Query Processing** (sec)	**Ref. [12]** (sec)	**Ref. [13]** (sec)
	Order 8 / 9 / 10 / 11	Order 8 / 9 / 10 / 11	Order 8 / 9 / 10 / 11	Order 8 / 9 / 10 / 11
T-drive	43.0 / 173.6 / 698.6 / 2753.1	0.418 / 0.416 / 0.416 / 0.415	3863.5 / 4759.4 / 8167.7 / 10456.1	6201.3 / 6206.1 / 6674.2 / 23806.5
Geolife	43.0 / 173.6 / 698.6 / 2753.1	0.417 / 0.414 / 0.415 / 0.416	1303.8 / 1888.2 / 3122.5 / 4236.4	5959.8 / 6029.1 / 5969.6 / 7427.5
Uniform	43.0 / 173.6 / 698.6 / 2753.1	0.418 / 0.416 / 0.415 / 0.416	2130.1 / 4532.5 / 9506.9 / 13795.9	6176.8 / 6242.4 / 9340.1 / 21780.6
Normal	43.0 / 173.6 / 698.6 / 2753.1	0.420 / 0.417 / 0.418 / 0.417	4661.1 / 7363.5 / 12513.6 / 16459.8	5966.5 / 5969.5 / 11331.3 / 20849.4