This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

Achieving Privacy-Preserving Discrete Fréchet Distance Range Queries

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

Abstract—The advances in Internet of Things, big data, and machine learning technologies have greatly transformed our daily lives into much more intelligent ones by offering various promising services. Among those services, the discrete Fréchet distance (DFD) range query, which aims to obtain a set of trajectories whose distances to a given query trajectory do not exceed a given threshold, has been widely applied to support applications such as vehicle trajectory clustering and other data processing tasks. Meanwhile, due to the huge data volume issue in the big data era, there is a trend towards outsourcing various query services to the cloud for achieving a better performance. However, since the cloud is not fully trustable, designing privacy-preserving query services becomes a research focus. Over the past years, many schemes focusing on privacy-preserving trajectory analysis have been proposed, but none of them can well support privacy-preserving DFD range queries. Aiming at addressing this challenge, this paper proposes a novel privacy-preserving DFD range query scheme, in which queries are conducted in a filtration-and-verification manner and the privacy of the dataset and queries can be preserved. Specifically, by indexing the dataset with two R-trees, a query can be conducted by i) querying the two R-trees to obtain a candidate set and ii) verifying each trajectory in the set, which involve two basic operations, namely, rectangle intersection detection and proximity detection. To preserve the privacy of the dataset and queries, we build the two basic operations upon a novel Inner-Product Preserving Encryption (IPPE) scheme, which is proved to be selectively secure with trivial leakages. Besides, extensive experiments are conducted, and the results demonstrate that our proposed scheme can significantly reduce the computational cost by effectively reducing the candidate set's size.

Index Terms—Trajectory similarity, discrete Fréchet distance, range query, privacy-preserving, outsourced encrypted data.

1 Introduction

With the proliferation of Internet of Things (IoT), big data and machine learning technologies, smart cities have been attracting a lot of attention from both industry and academia [1]. It is reported that the global smart cities market size is USD 410.8 billion in 2020, and it is estimated to be 820.7 billion by 2025 with a compound annual growth rate of 14.8% [2]. One of the most important goals of a smart city is to improve the efficiency and security of urban transportation, which naturally involves collecting and analyzing trajectory data. Meanwhile, as the trajectory dataset grows in size, there is a trend towards outsourcing the data as well as the trajectory analysis services to the cloud for more flexible computational resources and guaranteed service quality [3]–[5]. In this paper, we will focus on outsourced trajectory similarity range query services, where the trajectory similarity metric is selected to be discrete Fréchet distance (DFD). The DFD between two trajectories can be defined intuitively (for a formal definition, see Section 3.1) as the shortest leash connecting two objects, each of which goes forward with non-negative speed along one of the two trajectories, and enabling them to move from the beginnings of the corresponding trajectories to the ends. As demonstrated in DFD's definition, it is well suited for scenarios where there is an upper-bound for the distance between two moving objects [6]–[8]. An example of a DFD range query in a smart city is efficient carpooling services [9]. That is, a passenger may have a plan to go to multiple locations for some quick tasks (e.g., picking up items), while hailing a car at each task location will inevitably result in longer waiting time for the passenger and unnecessary operating cost for the carpooling service. In this case, it might be a better option for he/she to launch a DFD range query to find one vehicle from the carpooling service. Thereby, he/she can complete the tasks along the vehicle's trajectory, while the distance between each stop point in the vehicle's trajectory and the corresponding task location(s) does not exceed a threshold.

While being benefited by cloud computing, the outsourced services also suffer from privacy issues. The cloud server, which is not fully trusted, can obtain the trajectory data and queries containing sensitive information. A natural solution for this problem is to encrypt the data before outsourcing. Nevertheless, it is also commonly acknowledged that these techniques will obstruct implementing the range query service. Although many privacy-preserving range query schemes have been proposed, none of the existing works focuses on privacy-preserving DFD range queries to the best of our knowledge. The most related one to our work is Zhu et al.'s proposal [10]. However, in order to achieve privacy-preserving DFD range query with Zhu et al.'s work or other existing privacy-preserving range query works on two-dimensional points, the cloud server needs to linearly scan the dataset to determine whether the DFD between each trajectory in the dataset and the query trajectory exceeds the given threshold or not. As a result, these schemes

- *Y. Guan, R. Lu, Y. Zheng, and S. Zhang are with the Faculty of Computer Science, University of New Brunswick, Fredericton, Canada E3B5A3. E-mail: yguan4@unb.ca, rlu1@unb.ca, yzheng8@unb.ca, szhang17@unb.ca.*
- *J. Shao and G. Wei are with Zhejiang Gongshang University, Hangzhou, China 310018. E-mail: chn.junshao@gmail.com, weigy@zjgsu.edu.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

do not apply to our scenario due to high computational costs, especially when dealing with large datasets.

To address the above challenges, in this paper, we propose a privacy-preserving DFD range query (PDRQ) scheme, which processes DFD range queries in a filtration-and-verification manner. Specifically, our scheme first indexes the given trajectory dataset with two R-trees and uploads the encrypted dataset and R-trees to the cloud. Then, upon receiving an encrypted query request from a user, the cloud server first obtains a set of candidates by querying the two encrypted R-trees. After that, it refines the candidate set by verifying the DFD between each trajectory in the candidate set and the query trajectory over ciphertexts. To this end, we design an Inner-Product Preserving Encryption scheme to enable the cloud server to conduct DFD range queries over the encrypted dataset and index while preserving the data privacy. Specifically, the contributions of this work are four-fold.

1) First, based on the bilinear pairing and Bloom filter techniques, we design an Inner-Product Preserving Encryption (IPPE) scheme, which can reveal whether the inner-product of two vectors is negative or non-negative while protecting the plaintext values of two vectors or their inner-product.
2) Second, we design two IPPE-based approaches to respectively support privacy-preserving proximity detection and rectangle intersection detection. In specific, given two points and a threshold, the former detection approach can securely verify whether the distance between the points exceeds the threshold or not. Meanwhile, given two rectangles, the latter one can securely check whether they intersect or not.
3) Third, we propose our privacy-preserving discrete Fréchet distance range query scheme. In specific, the scheme first employs an index containing two R-trees for efficient filtration, in which the IPPE-based rectangle intersection detection approach is deployed for privacy-preserving R-tree traverse. Then, it employs the IPPE-based proximity detection approach to securely verify each trajectory in the filtration result.
4) Finally, we analyze the security of the proposed scheme and conduct extensive experiments to demonstrate its efficiency. The result of security analysis shows that our proposed scheme can preserve the privacy of the trajectory dataset and user queries. Besides, the result of performance analysis shows that our scheme can significantly reduce the computational cost by effectively reducing the candidate set's size.

The remainder of this paper is organized as follows. In Section 2, we formalize the system model and security model, and identify our design goal. Then, we recall some preliminaries in Section 3. In Section 4, we present our DFD range query scheme, followed by the security and performance analysis in Section 5 and Section 6, respectively. We also review some related works in Section 7. Finally, we conclude this work in Section 8.

2 Models and Design Goal

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

2.1 System Model

In this work, we consider a privacy-preserving DFD range query scenario, which mainly consists of three types of entities, namely, a service provider SP, a cloud server CS, and a set of query users $\mathcal{U}$, as shown in Fig. 1.

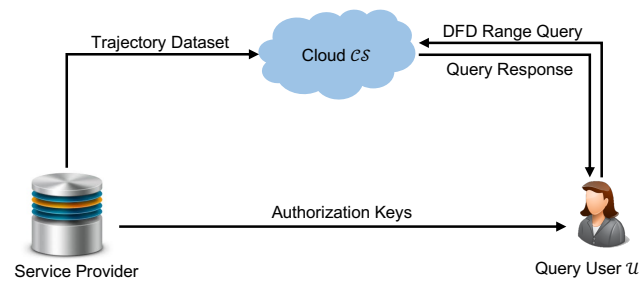

Fig. 1. The system model under consideration.

• *Service Provider SP:* In our system model, the service provider SP has a set of trajectory records $\mathcal{T} = \{t_i = (ID_i, \langle P_{i,1}, P_{i,2}, \cdots, P_{i,l_i}\rangle) \mid i = 1, 2, \cdots, N\}$, in which ID_i represents the identity of trajectory t_i, $\langle P_{i,1}, P_{i,2}, \cdots, P_{i,l_i}\rangle$ denotes an array of length l_i, and each element in the array is a k-dimensional point $(x_1, x_2, \cdots, x_k)$. Since SP could be weak in computing and storage, he/she is willing to outsource his/her data to the cloud for providing data services, e.g., the DFD range query service considered in this paper. However, as these trajectory records may contain some sensitive information and the cloud is not fully trusted, they need to be encrypted before being outsourced to the cloud.

• *Cloud Server CS:* In our system model, we consider CS is equipped with powerful computational resources and abundant storage space. Therefore, it is employed to store the encrypted dataset from SP and offer the DFD range query service to query users $U_i \in \mathcal{U}$.

• *Query Users* $\mathcal{U} = \{U_1, U_2, \cdots\}$: As for a query user $U_i \in \mathcal{U}$, he/she is authorized by SP and can launch DFD range queries. In specific, by submitting a query with a query trajectory t_q and a distance threshold ε, the user U_i can obtain a set of trajectories' identities $Results \subset \{ID_i \mid i = 1, 2, \cdots, N\}$, such that for each $ID_i \in Results$, the DFD between the corresponding trajectory t_i and the query trajectory t_q is not larger than the threshold ε, i.e., $\delta_d(t_q, t_i) \leq \varepsilon$. Note that, the distances between points in the trajectories are given by the Euclidean distance.

2.2 Security Model

In our security model, we consider the service provider SP is trusted as he/she is the data owner and has no motivation to deviate from the protocol. That is, the service provider will outsource a correct dataset and honestly distribute secret keys. However, the query users $U_i \in \mathcal{U}$ are considered to be honest-but-curious, i.e., they will honestly launch queries, but they may be curious about others' query requests and query results. As for the cloud server CS, it is considered to be honest-but-curious. In specific, although it will honestly store the trajectory data from SP and correctly respond to DFD range queries from U_i, it is curious about the plaintext of the trajectory data from SP, the plaintext of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

query requests, and the corresponding query results. Note that, the external attackers may launch other active attacks, i.e., Denial of Service (DoS) attacks, to the network. Since this work focuses on privacy preservation, those attacks are beyond the scope of this paper, and will be discussed in our future work.

2.3 Design Goal

The design goal of this work is to present a privacy-preserving DFD range query scheme to meet the requirements defined in the system model and security model. In specific, the proposed scheme should have the following two properties.

• *The proposed scheme should be privacy-preserving.* The server CS should not be able to obtain the private information in the system, including the plaintext of the trajectory dataset from SP and user queries.

• *The proposed scheme should be efficient:* To achieve privacy-preserving DFD range queries, some cryptographic techniques need to be employed for preserving the data privacy. However, as the proposed scheme focuses on the outsourced computation scenario, its efficiency should be taken into consideration. That is, to make the scheme practical, its overhead in terms of computation should be minimized.

3 Preliminaries

In this section, we first review the discrete Fréchet distance. Then, we respectively recall three techniques which will be used in our proposed scheme, namely, R-tree [11], bilinear pairing, and Bloom filter.

3.1 Discrete Fréchet Distance

The Fréchet distance is a measure of similarity between curves, in which the location and the ordering of points along the curves are considered. The intuitive idea of the Fréchet distance is that, considering two curves are respectively the trajectories of a person and a dog that is walked by him/her, the Fréchet distance between the curves is the minimum length of leash that is sufficient for the person and the dog to traverse their separate paths from beginning to end. Note that, during the walk, the speeds of the person and the dog are two non-negative numbers changing according to time, i.e., both of them cannot move backward. In this work, we consider discrete Fréchet distance (DFD), i.e., the curves are comprised of limited numbers of points, and we only consider the length of the leash when both endpoints are located at vertices of the curves. In specific, the definition of DFD is as follows.

Definition 1. (Discrete Fréchet Distance). Given two polygonal curves $t_i = \langle P_{i,1}, P_{i,2}, \cdots, P_{i,l_i} \rangle$ and $t_j = \langle P_{j,1}, P_{j,2}, \cdots, P_{j,l_j} \rangle$ of lengths l_i and l_j, and a distance function $d(\cdot,\cdot)$ for any two points in the curves, the discrete Fréchet distance between t_i and t_j is

$$\delta_d(t_i, t_j) = \min_{\alpha,\beta} \max_{\mathtt{t}\in[0,1]} d(\alpha(\mathtt{t}), \beta(\mathtt{t})),$$

where $\alpha(\cdot)$ (resp., $\beta(\cdot)$) maps time $\mathtt{t}$ to a point in t_i (resp., t_j), and both $\alpha(\cdot)$ and $\beta(\cdot)$ are monotone. Moreover, when $\mathtt{t} = 0$ and $\mathtt{t} = 1$, the two functions respectively returns the head and tail nodes in the two curves, e.g., $\alpha(0) = P_{i,1}$, $\alpha(1) = P_{i,l_i}$, $\beta(0) = P_{j,1}$, and $\beta(1) = P_{j,l_j}$.

3.2 R-Tree

An R-tree [11] is a data structure used for organizing multi-dimensional data records $r_i = \langle (x_{i,1}, x_{i,2}, \cdots), y \rangle$, and it can answer range queries and nearest neighbor queries over the data points. Its main idea is to recursively group nearby data points and represent each group as its minimum bounding rectangle (MBR) R_j in the corresponding parent node. Then, during conducting a range query, a node can be pruned if its MBR has no intersection with the query range. As shown in Fig. 2, an R-tree consists of two types of nodes, namely, inner nodes and leaf nodes. Specifically, an inner node has several children, and it stores the MBR of each child node; while a leaf node contains an array of data points that are adjacent.

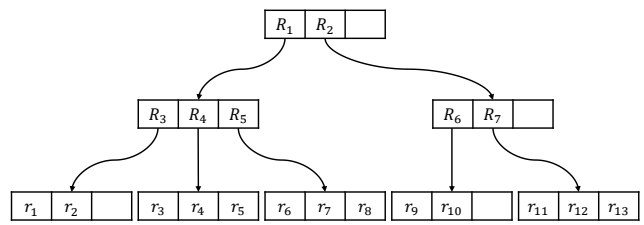

Fig. 2. An example of an R-tree.

Given a set of data points, an R-tree can be constructed in top-down or bottom-up approaches [12]. Then, to obtain data points in query range R_q, the query algorithm traverses the R-tree from its root. Specifically, the query is conducted in the following three steps.

• *Step 1:* Initialize an empty queue and an empty result set, and put the R-tree's root node into the queue.

• *Step 2:* Take a node from the queue, test the intersection between each MBR R_j in the node and the query range R_q, and ignore a child node if the corresponding R_j and R_q do not intersect. Otherwise, if the child node is a leaf node, then add the data records within $R_q \cap R_j$ into the result set; if the child node is an inner node, append it to the queue.

• *Step 3:* Repeat *Step 2* until the queue is empty and the result set contains all the data records that are covered by the query range R.

3.3 Bilinear Pairing

A bilinear pairing can be generated based on a security parameter λ, and it can be denoted as a 5-tuple $(\mathbb{G}_1, \mathbb{G}_2, p, g, e)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of the same big prime order p, g is an arbitrary generator of $\mathbb{G}_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ be a bilinear map, which has the following three properties:

• *Bilinearity:* $e(P^x, Q^y) = e(P, Q)^{xy}$, for any $x, y \in \mathbb{Z}_p^*$ and any $P, Q \in \mathbb{G}_1$.

• *Non-degeneracy:* $e(g, g) \neq 1$.

• *Computability:* $e(\cdot,\cdot)$ can be computed efficiently.

3.4 Bloom Filter

A Bloom filter (BF) is a space-efficient data structure for answering membership queries, i.e., identifying whether an item is a member of a set or not with a bounded false-positive rate. Specifically, an empty Bloom filter is an array

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

of Δ bits, and a set of f hash functions $\{\mathtt{h}_i\}_{i=1}^{f}$, where all bits in the array are set to 0 and each hash function $\mathtt{h}_i : \{0,1\}^* \mapsto \{1, 2, \cdots, \Delta\}$ maps an element to one of the Δ positions in the array. Then, the Bloom filter is initialized for a set $\mathcal{S}$ by setting locations $\{h_i(\mathtt{s}_j) \mid i = 1, 2, \cdots, f, \text{ and } \mathtt{s}_j \in \mathcal{S}\}$ to 1, and we denote the resulting Bloom filter as $BF(\mathcal{S})$. After that, $BF(\mathcal{S})$ can determine whether an element $\mathtt{s}$ is a member of $\mathcal{S}$ by testing whether the corresponding bits of locations $\{\mathtt{h}_i(\mathtt{s}) \mid i = 1, 2, \cdots, f\}$ are 1. If any of these bits is 0, then the element $\mathtt{s}$ is definitely not a member of $\mathcal{S}$, i.e., $\mathtt{s} \notin \mathcal{S}$; otherwise, $\mathtt{s} \in \mathcal{S}$ with a false-positive rate less than $(1 - \mathtt{e}^{-f \cdot |\mathcal{S}|/\Delta})^f$, where $\mathtt{e}$ is the Euler's number.

4 Our Proposed Scheme

In this section, we propose our PDRQ scheme. As mentioned in the Introduction, the scheme is built upon two basic operations, namely, rectangle-intersection detection and proximity detection, and the privacy of them is preserved by a novel Inner-Product Preserving Encryption (IPPE) scheme. Therefore, for the simplicity of introduction, we first introduce the IPPE scheme and the IPPE-based constructions for proximity detection and rectangle-intersection detection. Then, based on these building blocks, we present the construction of our PDRQ scheme. We summarize the some notations used in this section in Table. 1.

TABLE 1
Notations

Notation	Description
$\mathcal{T} = \{t_i \mid i = 1, \cdots, N\}$	The dataset of trajectories
$t_i = \{ID_i, \langle P_{i,1}, \cdots, P_{i,l_i}\rangle\}$	A trajectory consisting of l_i k-dimensional points
$Q = \{t_q, \varepsilon\}$	A DFD range query request with a query trajectory t_q and a distance threshold ε
$\delta(t_i, t_j)$	The DFD between two trajectories t_i and t_j
R_{t_i} and $\widehat{R}_{t_i}$	The minimum boundary rectangle (MBR) and the extended MBR of t_i
$[x_{i,m,1}, x_{i,m,2}]$	The range covered by R_{t_i} on the m-th dimension
$\{T_\ell\}_{\ell=1}^{2}$	The two R-tree built from $\{x_{i,m,\ell}\}$ of all trajectories' MBRs
$\mathbf{u}_{P_i}$, $[\![\mathbf{u}_{P_i}]\!]$, $\mathbf{u}_{m,i}$, $[\![\mathbf{u}_{i,m}]\!]$	The vector and the corresponding ciphertext built from a point P_i, and those for the m-th dimension of a rectangle R_i
$\mathbf{v}_{P_j,\varepsilon}$, $[\![\mathbf{v}_{P_j,\varepsilon}]\!]$, $\mathbf{v}_{j,m}$, $[\![\mathbf{v}_{j,m}]\!]$	The vector and the corresponding token built from a point P_j and ε, and those for the m-th dimension of a rectangle R_j

4.1 Inner-Product Preserving Encryption Scheme

In this subsection, we introduce our Inner-Product Preserving Encryption (IPPE) scheme, which can determine whether the inner product of two vectors is negative or non-negative without revealing the plaintexts of the vectors and their inner product. Specifically, assume there are two d-dimensional vectors $\mathbf{u} \in \mathbb{U}$ and $\mathbf{v} \in \mathbb{V}$, where $\mathbb{U}$ and $\mathbb{V}$ are the spaces of $\mathbf{u}$ and $\mathbf{v}$, respectively. After encrypting $\mathbf{u}$ and $\mathbf{v}$ with different algorithms, i.e., $[\![\mathbf{u}]\!]_1$ and $[\![\mathbf{v}]\!]_2$, the IPPE scheme can determine whether $\langle \mathbf{u}, \mathbf{v}\rangle < 0$ or not, and the values of $\mathbf{u}$, $\mathbf{v}$, and $\langle \mathbf{u}, \mathbf{v}\rangle$ are privacy-preserving. In the following, we respectively refer to $\mathbf{u}$ and $\mathbf{v}$ as a data vector and a token vector. Then, the IPPE scheme consists of the following four algorithms, namely, Key Generation, Encryption, Token Generation, and Check.

4.1.1 Key Generation

Given a security parameter λ, the dimension of vectors d, and the maximum possible absolute value of inner products T, the key generation algorithm generates keys for the IPPE scheme as follows.

- *Step 1:* The algorithm generates a bilinear pairing $(\mathbb{G}_1, \mathbb{G}_2, p, g, e)$ with the security parameter λ, where $p > 2(T^2 + T)$.
- *Step 2:* The algorithm generates two random numbers a and $b \in \mathbb{Z}_p^*$, and it computes $A = g^a$ and $B = g^b$.
- *Step 3:* It constructs a set $\mathcal{H} = \{H(e(g,g)^{abi})\}_{i=1}^{T^2+T}$, where $H(\cdot)$ is a secure hash function. Then, it builds a Bloom filter $BF(\mathcal{H})$ from the set $\mathcal{H}$ with an acceptable false-positive rate, regarding the expected quality of services, by adjusting the length of the array and the number of hash functions, i.e., Δ and f.
- *Step 4:* It randomly generates an invertible matrix M over $\mathbb{Z}_p$ of order $(d+2) \times (d+2)$.

Finally, the algorithm outputs a public parameter $PP = ((\mathbb{G}_1, \mathbb{G}_2, p, g, e), BF(\mathcal{H}))$, an encryption key $EK = (\mathrm{M}^{-1}, A)$ and a token generation key $TK = (\mathrm{M}, B)$.

4.1.2 Encryption

Given the public parameter PP and the encryption key EK, the encryption algorithm encrypts a data vector $\mathbf{u} = (u_1, u_2, \cdots, u_d)$ as follows.

- *Step 1:* The algorithm selects two random numbers γ_1 and γ_2, such that $|\gamma_1| < \gamma_2 \leq \lfloor\sqrt{\frac{T^2}{T_{\mathbf{u}}}}\rfloor$, where T is the maximum possible value of inner products, $T_{\mathbf{u}} = \max_{\mathbf{v}} \langle \mathbf{u}, \mathbf{v}\rangle$ is the maximum inner products of $\mathbf{u}$ with different $\mathbf{v} \in \mathbb{V}$.
- *Step 2:* The algorithm constructs a vector $\mathbf{u}' = (u'_1, u'_2, \cdots, u'_d, u'_{d+1}, u'_{d+2})$ of length $d+2$, where $u'_i = \gamma_2 \cdot u_i$, for $i = 1, 2, \cdots, d$, $u'_{d+1} = \gamma_1$ and $u'_{d+2} = 1$.
- *Step 3:* The algorithm computes a vector $\mathbf{u}''^T = \mathrm{M}^{-1} u'^T \bmod p = (u''_1, u''_2, \cdots, u''_{d+2})$ and outputs the ciphertext of the data vector $\mathbf{u}$, i.e., $[\![\mathbf{u}]\!]_1 = (\{A^{u''_i}\}_{i=1}^{d+2})$.

4.1.3 Token Generation

Given the public parameter PP and the token generation key TK, the token generation algorithm encrypts a token vector $\mathbf{v} = (v_1, v_2, \cdots, v_d)$ in the following steps.

- *Step 1:* The algorithm selects two random numbers γ_3 and γ_4, such that $|\gamma_3| < \gamma_4 \leq \lfloor\sqrt{\frac{T^2}{T_{\mathbf{v}}}}\rfloor$, where $T_{\mathbf{v}} = \max_{\mathbf{u}} \langle \mathbf{u}, \mathbf{v}\rangle$ is the maximum $\langle \mathbf{u}, \mathbf{v}\rangle$ with different $\mathbf{u} \in \mathbb{U}$.
- *Step 2:* The algorithm constructs a vector $\mathbf{v}' = (v'_1, v'_2, \cdots, v'_d, v'_{d+1}, v'_{d+2})$, where $v'_i = \gamma_4 \cdot v_i$, for $i = 1, 2, \cdots, d$, $v'_{d+1} = 1$ and $v'_{d+2} = \gamma_3$.
- *Step 3:* The algorithm computes a vector $\mathbf{v}'' = \mathbf{v}\mathrm{M} \bmod p = (v''_1, v''_2, \cdots, v''_{d+2})$ and outputs the ciphertext of the token vector $\mathbf{v}$, i.e., $[\![\mathbf{v}]\!]_2 = (\{B^{v''_i}\}_{i=1}^{d+2})$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

4.1.4 Check

Given the encrypted data vector $[\![\mathbf{u}]\!]_1$ and token vector $[\![\mathbf{v}]\!]_2$, the check algorithm determines whether $\langle \mathbf{u}, \mathbf{v} \rangle$ is negative or non-negative by computing $\Psi = \prod_{i=1}^{d+2} e(A^{u''_i}, B^{v''_i})$. If $H(\Psi) \in BF(\mathcal{H})$, then $\langle \mathbf{u}, \mathbf{v} \rangle \geq 0$; and $\langle \mathbf{u}, \mathbf{v} \rangle < 0$, otherwise. For simplicity, we denote the check algorithm as $Check(\cdot)$, e.g., $Check([\![\mathbf{u}]\!]_1, [\![\mathbf{v}]\!]_2)$, which will return either 0 or 1 to indicate $\langle \mathbf{u}, \mathbf{v} \rangle < 0$ or ≥ 0.

4.1.5 Correctness

The correctness of the scheme can be analyzed as follows. Based on the properties of bilinear pairing, we have

$$\Psi = \prod_{i=1}^{d+2} e(A^{u''_i}, B^{v''_i}) = e(g^a, g^b)^{\sum_{i=1}^{d+2} u''_i \cdot v''_i}$$
$$= e(g, g)^{ab \cdot (\gamma_2 \cdot \gamma_4 \cdot \langle \mathbf{u}, \mathbf{v} \rangle + \gamma_1 + \gamma_3)},$$

as $\sum_{i=1}^{d+2} v''_i \cdot u''_i = \mathbf{v}'' \cdot \mathbf{u}''^T = \mathbf{v}' \mathrm{M} \cdot \mathrm{M}^{-1} \mathbf{u}'^T = \gamma_2 \cdot \gamma_4 \cdot \langle \mathbf{u}, \mathbf{v} \rangle + \gamma_1 \cdot 1 + 1 \cdot \gamma_3$. Therefore, if $\langle \mathbf{u}, \mathbf{v} \rangle \in [0, T]$, then we have $\gamma_2 \cdot \gamma_4 \cdot \langle \mathbf{u}, \mathbf{v} \rangle + \gamma_1 + \gamma_3 \in (0, T^2 + T)$, i.e., $\Psi \in \mathcal{H}$; otherwise, we have $\gamma_2 \cdot \gamma_4 \cdot \langle \mathbf{u}, \mathbf{v} \rangle + \gamma_1 + \gamma_3 \in (-T^2 - T, 0)$, i.e., $\Psi \notin \mathcal{H}$. Based on the property of the Bloom filter, $BF(\mathcal{H})$ can determine whether $\Psi \in \mathcal{H}$ or not with an acceptable false-positive rate. Thus, the correctness of the scheme holds.

4.2 IPPE-based Proximity Detection

In this subsection, based on the IPPE scheme, we introduce an approach to detect whether the distance between two given points exceeds a threshold or not. That is, given two k-dimensional points $\{P_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,k}), P_j = (x_{j,1}, x_{j,2}, \cdots, x_{j,k})\}$ and a threshold ε, the approach detects whether the Euclidean distance between them $d(P_i, P_j) = \sqrt{\sum_{m=1}^{k} (x_{i,m} - x_{j,m})^2} \leq \varepsilon$ or not. The idea of the detection approach is to covert the inequality evaluation into testing the sign of two vectors' inner-product, in which, the data vector represents a point (e.g., P_i), and the token vector represents the other point (e.g., P_j) and the threshold ε. Specifically, given a group of keys $\{PP, EK, TK\}$ generated for the IPPE scheme with the dimension parameter $d = k+2$, the approach consists of the following three parts, namely, Encryption, Token Generation, and Check.

- *Encryption:* To encrypt the point P_i with the IPPE scheme, the encryption algorithm first converts it into a data vector $\mathbf{u}_{P_i} = \left(-\sum_{m=1}^{k} x_{i,m}^2, 1, \{2x_{i,m}\}_{m=1}^{k}\right)$ of length $k+2$. Then, given PP and EK, the algorithm encrypts $\mathbf{u}_{P_i}$ into $[\![\mathbf{u}_{P_i}]\!]_1$ with the IPPE scheme.
- *Token Generation:* Similarly, the token generation algorithm first converts the point P_j into a token vector $\mathbf{v}_{P_j,\varepsilon} = \left(1, \varepsilon^2 - \sum_{m=1}^{k} x_{j,m}^2, \{x_{j,m}\}_{m=1}^{k}\right)$ of length $k+2$. Then, given PP and TK, the algorithm encrypts the vector $\mathbf{v}_{P_j,\varepsilon}$ into $[\![\mathbf{v}_{P_j,\varepsilon}]\!]_2$ with the IPPE scheme.
- *Check:* Given the public parameter PP and the two encrypted vectors $\{[\![\mathbf{u}_{P_i}]\!]_1, [\![\mathbf{v}_{P_j,\varepsilon}]\!]_2\}$, the check algorithm can determine whether the distance $d(P_i, P_j) \leq \varepsilon$ or not by checking whether $Check([\![\mathbf{u}_{P_i}]\!]_1, [\![\mathbf{v}_{P_j,\varepsilon}]\!]_2) \stackrel{?}{=} 1$. Specifically, if $Check([\![\mathbf{u}_{P_i}]\!]_1, [\![\mathbf{v}_{P_j,\varepsilon}]\!]_2) = 1$, then $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon} \rangle \geq 0$, i.e., $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon} \rangle \geq 0$, then we have $d(P_i, P_j) \leq \varepsilon$. Otherwise, we have $d(P_i, P_j) > \varepsilon$.

Correctness. Based on the correctness of the IPPE scheme, we can obtain the correct sign of $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon} \rangle$. Therefore, we show the correctness of this approach by analyzing the sign of $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon} \rangle = 1 \times \left(-\sum_{m=1}^{k} x_{i,m}^2\right) + \left(\varepsilon^2 - \sum_{m=1}^{k} x_{j,m}^2\right) \times 1 + \sum_{m=1}^{k} 2x_{i,m} \times x_{j,m} = \varepsilon^2 - d(P_i, P_j)^2$. Since $\varepsilon \geq 0$ and $d(P_i, P_j) \geq 0$, we have $\varepsilon \geq d(P_i, P_j)$ when $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon} \rangle \geq 0$; and $\varepsilon < d(P_i, P_j)$, otherwise. Thus, the correctness of the approach holds.

4.3 IPPE-based Rectangle-Intersection Detection

In this subsection, we introduce an approach to detect whether two rectangles intersect or not. That is, given two rectangles R_i and R_j, the approach detects whether $R_i \cap R_j = \emptyset$ or not. To this end, we respectively denote the two k-dimensional rectangle as $\prod_{m=1}^{k} [x_{i,m,1}, x_{i,m,2}]$ and $\prod_{m=1}^{k} [x_{j,m,1}, x_{j,m,2}]$, where $[x_{i,m,1}, x_{i,m,2}]$ and $[x_{j,m,1}, x_{j,m,2}]$ are the ranges respectively covered by R_i and R_j on the m-th dimension. Then, the idea of the approach is to detect overlap on each dimension, i.e., for each dimension $m = 1, 2, \cdots, k$, the approach checks whether $[x_{i,m,1}, x_{i,m,2}] \cap [x_{j,m,1}, x_{j,m,2}] = \emptyset$ or not. If there exists an $m \in [1, \cdots, k]$ such that $[x_{i,m,1}, x_{i,m,2}] \cap [x_{j,m,1}, x_{j,m,2}] = \emptyset$, the two rectangles R_i and R_j do not intersect. Moreover, for each dimension m, we convert ranges $[x_{i,m,1}, x_{i,m,2}]$ and $[x_{j,m,1}, x_{j,m,2}]$ into two vectors and employ the IPPE to securely check the sign of their inner-product. Specifically, given a group of keys $\{PP, EK, TK\}$ and a random number $\widehat{\gamma}_m \in \mathbb{Z}_p$, the approach detects overlap on the m-th dimension in three steps, namely, Encryption, Token Generation, and Check.

- *Encryption:* In this step, the algorithm first converts the range of R_i on the m-th dimension into a data vector $\mathbf{u}_{i,m} = (\widehat{\gamma}_m - x_{i,m,1}x_{i,m,2}, 1, x_{i,m,1}, x_{i,m,2})$. Then, given the public parameter PP and the encryption key EK, it encrypts the vector $\mathbf{u}_{i,m}$ into $[\![\mathbf{u}_{i,m}]\!]_1$.
- *Token Generation:* In the Token Generation step, the algorithm first converts the range of R_j on the m-th dimension into a token vector $\mathbf{v}_{j,m} = (1, -\widehat{\gamma}_m - x_{j,m,1}x_{j,m,2}, x_{j,m,1}, x_{j,m,2})$. Then, given the public parameter PP and the token generation key TK, it encrypts the token vector $\mathbf{v}_{j,m}$ into $[\![\mathbf{v}_{j,m}]\!]_2$.
- *Check:* This algorithm checks whether two rectangles overlap on the m-th dimension. The input of this algorithm includes the public parameter PP and two encrypted vectors $\{[\![\mathbf{u}_{i,m}]\!]_1, [\![\mathbf{v}_{j,m}]\!]_2\}$ representing the ranges covered by R_i and R_j on the m-th dimension. Then, if $Check([\![\mathbf{u}_{i,m}]\!]_1, [\![\mathbf{v}_{j,m}]\!]_2) = 1$, i.e., $\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m} \rangle \geq 0$, then we know the two rectangles R_i and R_j overlap on the m-th dimension; otherwise, $Check([\![\mathbf{u}_{i,m}]\!]_1, [\![\mathbf{v}_{j,m}]\!]_2) = 0$.

Correctness. Based on the correctness of the IPPE scheme, we can obtain the correct sign of $\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m} \rangle$. Therefore, we show the correctness of this approach by analyzing the sign of $\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m} \rangle = \widehat{\gamma}_m - x_{i,m,1}x_{i,m,2} - \widehat{\gamma}_m - x_{j,m,1}x_{j,m,2} + x_{i,m,1}x_{j,m,1} + x_{i,m,2}x_{j,m,2} = (x_{i,m,2} - x_{j,m,1})(x_{j,m,2} - x_{i,m,1})$. Since $x_{i,m,1} \leq x_{i,m,2}$ and $x_{j,m,1} \leq x_{j,m,2}$, when R_1 and R_2 overlap on the m-th dimension, we have $x_{i,m,2} \geq x_{j,m,1}$ and $x_{j,m,2} \geq x_{i,m,1}$, and $\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m} \rangle \geq 0$; and $\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m} \rangle < 0$, otherwise. Thus, the correctness of this approach holds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

4.4 PDRQ Scheme

Based on the three building blocks, we propose our PDRQ scheme in this subsection. Before delving into our scheme, we first explain the intuitive idea of our scheme on plaintext. After that, we respectively detail the three parts of the scheme, namely, System Initialization, Token Generation, and Query.

4.4.1 Scheme Overview

Given the trajectory dataset, the scheme conducts privacy-preserving DFD range queries in two steps, i.e., filtration and verification. That is, to conduct a query, the scheme first filters the original dataset to obtain a set of possible query results, and then it verifies the set and removes elements from the set that does not satisfy the query. Then, we respectively explain the ideas of these two steps as follows.

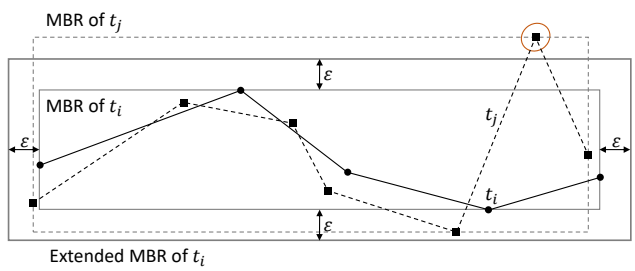

Fig. 3. An example of two trajectories t_i and t_j comprised of 2 dimensional points. In the figure, we draw t_i's MBR and its extension, as well as t_j's MBR. The MBR of t_j is not covered by the extended MBR of t_i, as the distance between the MBR of t_i and the circled point on t_j is greater than the threshold ε. Therefore, $\delta_d(t_i, t_j) > \varepsilon$.

• *Filtration:* The filtration step of the scheme is based on an observation as shown in Fig. 3. That is, given two trajectories t_i and t_j, we can draw their minimum boundary rectangles (MBR), i.e., R_{t_i} and R_{t_j}. Then, we compute the extended MBR $\widehat{R}_{t_i}$ by expending each dimension of R_{t_i} by ε. Similarly, we compute $\widehat{R}_{t_j}$ from R_{t_j}. As shown in Fig. 3, if the MBR of one trajectory (e.g., R_{t_j}) is not fully covered by the extended MBR of the other trajectory (e.g., $\widehat{R}_{t_i}$), then there must exist at least one point $P_{j,m}$ in t_j that satisfies the inequality $d(P_{i,m}, P_{j,m'}) > \varepsilon$ for all $P_{i,m'}$ in t_i, i.e., its distances to all points in t_i exceed the threshold. Therefore, $\delta_d(t_i, t_j) > \varepsilon$.

Based on this observation, we can index the trajectories in the dataset based on their MBRs, and during conducting a query, the scheme can ignore the trajectories whose MBRs are not covered by the extended MBR of the query trajectory or whose extended MBR cannot cover the MBR of the query trajectory. In specific, we first initialize two empty R-trees T_1 and T_2. Then, for each trajectory $t_i \in \mathcal{T}$, we compute its MBR $R_{t_i} = \prod_{m=1}^{k}[x_{i,m,1}, x_{i,m,2}]$ and put points $(x_{i,1,1}, x_{i,2,1}, \cdots, x_{i,k,1})$ and $(x_{i,1,2}, x_{i,2,2}, \cdots, x_{i,k,2})$ respectively into T_1 and T_2, as shown in Fig. 4. Upon receiving a query trajectory t_q, our scheme obtains a candidate set containing possible query result in the following three steps: i) It computes the MBR of t_q, i.e., $R_{t_q} = \prod_{m=1}^{k}[x_{q,m,1}, x_{q,m,2}]$. ii) It respectively queries T_1 and T_2 to obtain two sets

$$\mathcal{C}_1 = \{v_i \mid x_{i,m,1} \in [x_{q,m,1} - \varepsilon, x_{q,m,1} + \varepsilon], m = 1, \cdots, k\},$$

$$\mathcal{C}_2 = \{v_i \mid x_{i,m,2} \in [x_{q,m,2} - \varepsilon, x_{q,m,2} + \varepsilon], m = 1, \cdots, k\}.$$

iii) It outputs the candidate set $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$, which will be further refined in the verification step.

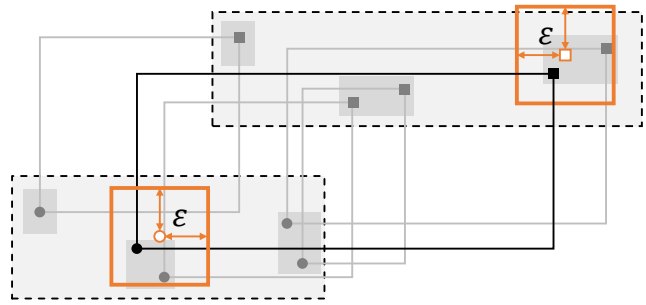

Fig. 4. An example of indexing a 2-dimensional dataset. In the index, two R-trees are respectively constructed for the lower-left (•) and upper-right (■) vertices of t_i's MBR, for $t_i \in \mathcal{T}$. To conduct a query, the algorithm first obtains the lower-left (○) and upper-right (□) vertices of the query trajectories, and queries each R-tree with a square query range centered on the corresponding vertex with a side length of 2ε. Then, one trajectory is selected by intersecting the query results from both R-trees.

• *Verification:* In the verification step, trajectories $t_i \in \mathcal{C}$ that do not satisfy the query, i.e., $\delta_d(t_i, t_q) > \varepsilon$ will be removed. However, instead of directly computing the DFD between the query trajectory t_q and each trajectory $t_i \in \mathcal{C}$, our scheme verifies whether each distance $\delta_d(t_q, t_i)$ is greater than ε or not. If $\delta_d(t_q, t_i) > \varepsilon$, then t_i does not satisfy the query and will be removed from $\mathcal{C}$. Specifically, as demonstrated in Alg. 1, a trajectory $t_i \in \mathcal{C}$ in verified in the following steps.

- *Step 1:* The algorithm computes the Euclidean distance $d(P_{q,1}, P_{i,1})$ between the first two points of the two trajectories t_q and t_i. If the $d(P_{q,1}, P_{i,1}) > \varepsilon$, the algorithm outputs *false* indicating that the DFD $\delta_d(t_q, t_i) > \varepsilon$. Otherwise, the algorithm proceeds to *Step 2*.
- *Step 2:* It applies the verification algorithm to sub-trajectory pairs $(t_q[2..], t_i)$, $(t_q, t_i[2..])$, and $(t_q[2..], t_i[2..])$, where $t_q[2..]$ and $t_i[2..]$ respectively denote the resulting trajectories after removing the first nodes from t_q and t_i. If any of the three invokes of the algorithm returns *true*, the algorithm outputs *true* indicating that the DFD $\delta_d(t_q, t_i) \leq \varepsilon$. Otherwise, the algorithm proceeds to *Step 3*.
- *Step 3:* The algorithm outputs whether both trajectories respectively only contain one points. If true, then the algorithm has completely checked the two trajectories and $\delta_d(t_q, t_i) \leq \varepsilon$; otherwise, the rests of the two trajectories cannot satisfy the query, and t_i should be removed from the candidate set $\mathcal{C}$.

After the verification step, the candidate set will contain only trajectories that satisfy the query, and the set can be returned to the query user as the query result.

4.4.2 System Initialization

In the system initialization part, the service provider SP initializes the whole system in the following two phases, namely, Key Generation and Data Outsourcing.

• *Key Generation:* In the Key Generation phase, the service provider SP generates keys for all entities in the system through three steps.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

Algorithm 1 Verification process for a single trajectory.

Input: The query trajectory $t_q = \{P_{q,1}, P_{q,2}, \cdots\}$,
a trajectory in the dataset, $t_i = \{P_{i,1}, P_{i,2}, \cdots\}$.
Output: Whether $\delta_d(t_q, t_i) \leq \varepsilon$ or not.

1: $\mathtt{j} \leftarrow 1, \mathtt{k} \leftarrow 1$
2: **return** VERIFY(t_q, t_i, $\mathtt{j}$, $\mathtt{k}$)
3:
4: **function** VERIFY(t_q, t_i, $\mathtt{j}$, $\mathtt{k}$)
5: ▷ Verify the distance between the nodes pointed by both pointers
6: **if** $d(P_{q,\mathtt{j},P_{i,\mathtt{k}}}) > \varepsilon$ **then**
7: **return** false
8:
9: ▷ Try moving pointers forward and recursively checking the rest parts of the two trajectories
10: **if** $\mathtt{j}+1 \leq |t_q|$ **and** VERIFY(t_q, t_i, $\mathtt{j}+1$, $\mathtt{k}$) **then**
11: **return** true
12: **if** $\mathtt{k}+1 \leq |t_i|$ **and** VERIFY(t_q, t_i, $\mathtt{j}$, $\mathtt{k}+1$) **then**
13: **return** true
14: **if** $\mathtt{j}+1 \leq |t_q|$ **and** $\mathtt{k}+1 \leq |t_i|$
15: **and** VERIFY(t_q, t_i, $\mathtt{j}+1$, $\mathtt{k}+1$) **then**
16: **return** true
17:
18: ▷ Accept if $d(P_{q,\mathtt{j},P_{i,\mathtt{k}}}) \leq \varepsilon$ and both pointers at are the ends
19: **return** $\mathtt{j} == |t_q|$ **and** $\mathtt{k} == |t_i|$

- *Step 1: SP* runs the Key Generation algorithm of the IPPE scheme and obtain the keys PP, EK, TK. Note that, SP needs to randomly generate two invertible matrices M_1 of order 6 and M_2 of order $k+2$.
- *Step 2:* It chooses $2k$ random numbers $\widehat{\gamma}_{1,m}$ and $\widehat{\gamma}_{2,m}$ that will be used encrypting query ranges for the two R-trees $\{T_1, T_2\}$.
- *Step 3:* It generates a secret key sk for a Symmetric Key Encryption (SKE) scheme.

Finally, he/she publishes the public parameter $PP = ((\mathbb{G}_1, \mathbb{G}_2, g, p, e), BF(\mathcal{H}))$, securely distributes $TK = (\mathrm{M}_1, \mathrm{M}_2, \{\widehat{\gamma}_{1,m}\}_{m=1}^k, \{\widehat{\gamma}_{2,m}\}_{m=1}^k, B)$ and the SKE secret key sk to each query user $U_i \in \mathcal{U}$, and keeps the encryption key $EK = (\mathrm{M}_1, \mathrm{M}_2, \{\widehat{\gamma}_{1,m}\}_{m=1}^k, \{\widehat{\gamma}_{2,m}\}_{m=1}^k, A)$ as well as sk for the Data Outsourcing phase.

• *Data Outsourcing:* In the Data Outsourcing phase, SP constructs an encrypted index for the dataset with the encryption key EK and uploads it to CS. Specifically, it mainly consists of the following three steps.

- *Step 1: SP* encrypts trajectories in the dataset. Specifically, for each $t_i \in \mathcal{T}$, SP encrypts each point $P_{i,j} \in t_i$ to $[\![\mathbf{u}_{P_{i,j}}]\!]_1$ with the encryption key (M_2^{-1}, A) and the public parameter PP. Furthermore, the identity ID_i of t_i will be encrypted as $SKE(sk, ID_i)$. Then, we denote the ciphertext of trajectory t_i as

$$E(t_i) = \left(SKE\left(sk, ID_i\right), \left\langle [\![\mathbf{u}_{P_{i,1}}]\!]_1, \cdots, [\![\mathbf{u}_{P_{i,l_i}}]\!]_1 \right\rangle\right).$$

- *Step 2:* In this step, SP builds an index for the dataset. Firstly, it constructs two R-trees T_1 and T_2 from the MBR of each trajectory $t_i \in \mathcal{T}$ as detailed in Section 4.4.1. Then, with the encryption key $(\mathrm{M}_1, A, \{\widehat{\gamma}_{\ell,m}\}_{m=1}^k)$, to encrypt each R-tree T_ℓ, for $\ell = 1, 2$, SP respectively encrypts its inner nodes and leaf nodes as follows. i) For each inner node in T_ℓ, SP encrypts each MBR $R_{\ell,j}$ of the node's children to be k encrypted vectors $\{[\![\mathbf{u}_{R_{\ell,j},m}]\!]_1\}_{m=1}^k$. ii) For each leaf node in T_ℓ, SP converts each record $r_{\ell,j}$ in it to be a rectangle $\prod_{m=1}^k [x_{\ell,m}, x_{\ell,m}]$, i.e., the MBR of $r_{\ell,j}$, which will be further encrypted into $\{[\![\mathbf{u}_{r_{\ell,j},m}]\!]_1\}_{m=1}^k$.
- *Step 3:* The service provider SP outsources the encrypted index $\{E(T_1), E(T_2)\}$ and the encrypted dataset $E(\mathcal{T}) = \{E(t_i) \mid t_i \in \mathcal{T}\}$ to CS.

4.4.3 Token Generation

To launch a DFD range query, a query user U_i needs to generate a query token, which mainly consists of three parts, namely, two query tokens for the R-trees $\{T_1, T_2\}$, and the encrypted trajectory for the verification step. Specifically, given a query trajectory t_q and a distance threshold ε, the query token is built in the following two steps.

• *Step 1:* With the token generation key (M_2, B) and the public parameter PP, the query user U_i runs the token generation algorithm in Section 4.2 to convert each point $P_{q,j} \in t_q$ to $[\![\mathbf{v}_{P_{q,j},\varepsilon}]\!]_2$.

• *Step 2:* The query user U computes the query trajectory t_q's MBR $R_{t_q} = \prod_{m=1}^k [x_{q,m,1}, x_{q,m,2}]$ and respectively generates tokens for the two rectangles $R_{q,\ell} = \prod_{m=1}^k [x_{q,m,\ell} - \varepsilon, x_{q,m,\ell} + \varepsilon]$, for $\ell = 1, 2$. Then, to generate query tokens respectively for T_ℓ, for $\ell = 1, 2$, $R_{q,\ell}$ can be further converted into $\{[\![\mathbf{v}_{\ell,m}]\!]_2\}_{m=1}^k$ with the token generation key $(\mathrm{M}_1, \{\widehat{\gamma}_{\ell,m}\}_{m=1}^k, B)$ and the public parameter PP.

Finally, U obtains the query token $Token = \{\{[\![\mathbf{v}_{1,m}]\!]_2\}_{m=1}^k, \{[\![\mathbf{v}_{2,m}]\!]_2\}_{m=1}^k, \{[\![\mathbf{v}_{P_{q,j},\varepsilon}]\!]_2 \mid P_{q,j} \in t_q\}\}$.

4.4.4 Query

In this phase, U submits the query token $Token$ to the cloud server CS, and the latter will conduct the query in the following steps.

• *Step 1:* CS queries the two encrypted R-trees $E(T_\ell)$, for $\ell = 1, 2$, with $\{[\![\mathbf{v}_{\ell,m}]\!]_2\}_{m=1}^k$, i.e., the encrypted query range on T_ℓ. The process of querying an encrypted R-tree is generally consistent with the idea given in Section 3.2, except that computing intersections between MBRs in the R-tree's nodes and the query trajectory's MBR. In specific, given an encrypted MBR $\{[\![\mathbf{u}_{R_{\ell,j},m}]\!]_1\}_{m=1}^k$ in T_ℓ and the query trajectory t_q's encrypted MBR $\{[\![\mathbf{v}_{\ell,m}]\!]_2\}_{m=1}^k$, the intersection is detected through the IPPE-based rectangle-intersection detection approach detailed in Section 4.3. Similarly, given a record $r_{\ell,j}$'s encrypted MBR, the intersection with t_q's encrypted MBR is also computed through the IPPE-based approach. After querying T_1 and T_2, the cloud server CS will respectively obtain two candidate sets $\mathcal{C}_1$ and $\mathcal{C}_2$.

• *Step 2:* In this step, CS verifies each candidate trajectory $E(t_i) \in \mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$. Specifically, it verifies whether $\delta_d(t_q, t_i)$ through Alg. 1. However, as CS only have $E(t_i) = ([\![\mathbf{u}_{P_{i,1}}]\!]_1, [\![\mathbf{u}_{P_{i,2}}]\!]_1, \cdots)$ and $([\![\mathbf{v}_{P_{q,1},\varepsilon}]\!]_2, [\![\mathbf{v}_{P_{i,2},\varepsilon}]\!]_2, \cdots)$, it cannot directly compute the distance between two points in t_i and t_q as shown on Line 6 of Alg. 1. Instead, CS runs the IPPE-based proximity detection approach detailed in Section 4.2. That is, to determine whether the Euclidean distance between $P_{i,j_1} \in t_i$ and $P_{q,j_2} \in t_q$ exceeded the distance threshold ε, CS computes whether $\left\langle \mathbf{u}_{P_{i,j_1}}, \mathbf{v}_{P_{q,j_2},\varepsilon} \right\rangle > 0$ or not through the IPPE scheme. Then, CS will remove t_i from $\mathcal{C}$ if Alg. 1 returns *false*. After this step, CS can obtain the query result $E(Results) = \{SKE(sk, ID_i) \mid E(t_i) \in \mathcal{C}, \delta_d(t_q, t_i) \leq \varepsilon\}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

• *Step 3: CS* sends $E(Results)$ to the query user U_i through a secured channel, and U_i can obtain the plaintext query result $Results = \{ID_i\}$ by decrypting each element in $E(Results)$ with the SKE secret key sk.

5 Security Analysis

In this section, we analyze the security of the proposed privacy-preserving discrete Fréchet distance range query scheme. Specifically, we first analyze the security of the three building blocks. Then, following the security requirements discussed earlier, we will show how the proposed scheme can achieve the data privacy.

5.1 Security of Building Blocks

As detailed in Section 4, there are three building blocks in our proposed scheme, namely, the IPPE scheme, and the proximity detection approach and rectangle-intersection detection approach based on the IPPE scheme. In the following, we will respectively analyze their security.

5.1.1 Security of The IPPE scheme

In this subsection, we prove that our IPPE scheme is *selectively secure* in the real/ideal world security model, following the formal definition of this model in [13], [14]. This security model subsumes the Known Plaintext Attack (KPA) security definition in the distinguishability setting and is widely used to prove the security of functional schemes. In the real/ideal world model, the real world is our IPPE scheme, while in the ideal world, the scheme is replaced with an ideal function that only leaks some information specified by a leakage function. Then, we prove that the IPPE scheme will only leak the information specified by the leakage function, i.e., our scheme is selectively secure, by showing that the two worlds are indistinguishable. Before delving into the two worlds, we first define the trivial leakage of our IPPE scheme. Let $[\![\mathbf{u}]\!]_1$ and $[\![\mathbf{v}]\!]_2$ are the ciphertexts of $\mathbf{u}$ and $\mathbf{v}$, the leakage of IPPE is $\mathsf{sign}(\langle \mathbf{u}, \mathbf{v}\rangle)$, i.e., $\mathcal{L} = \{\mathsf{sign}(\langle \mathbf{u}, \mathbf{v}\rangle)\}$. Based on $\mathcal{L}$, we define the real and ideal worlds.

Real world. The real world involves two participants, i.e., a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a challenger, and they interact with each other as follows.

• **Setup phase.** $\mathcal{A}$ randomly chooses p_1 d-dimensional data vectors $\{\mathbf{u}_i\}_{i=1}^{p_1}$ and sends them to the challenger. On receiving them, the challenger runs the key generation algorithm of the IPPE scheme to obtain (PP, EK, TK), and uses EK to encrypt them into ciphertexts, i.e., $\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$. Then, the challenger sends the public parameter PP to $\mathcal{A}$.

• **Query phase I.** In the query phase I, $\mathcal{A}$ randomly chooses p_2 d-dimensional token vectors $\{\mathbf{v}_i\}_{i=1}^{p_2}$. Then, $\mathcal{A}$ sends $\{\mathbf{v}_i\}_{i=1}^{p_2}$ to the challenger. On receiving these token vectors, the challenger responds with the ciphertexts of the token vectors encrypted by TK, i.e., $\{[\![\mathbf{v}_i]\!]_2\}_{i=1}^{p_2}$.

• **Challenge phase.** In the challenge phase, the challenger sends $\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$ (i.e., the ciphertexts of the data vectors $\{\mathbf{u}_i\}_{i=1}^{p_1}$) to $\mathcal{A}$.

• **Query phase II.** In the query phase II, $\mathcal{A}$ randomly chooses $(p_2' - p_2)$ d-dimensional token vectors $\{\mathbf{v}_i\}_{i=p_2+1}^{p_2'}$, where p_2' is a polynomial number. Then, same as **Query Phase I**, the challenger responds with $\{[\![\mathbf{v}_i]\!]_2\}_{i=p_2+1}^{p_2'}$.

Ideal world. The ideal world involves two participants, i.e., a PPT adversary $\mathcal{A}$ and a simulator, and they interact with each other as follows.

• **Setup phase.** In the setup phase, $\mathcal{A}$ randomly chooses p_1 d-dimensional data vectors $\{\mathbf{u}_i\}_{i=1}^{p_1}$, where p_1 is a polynomial number. Then, it sends $\{\mathbf{u}_i\}_{i=1}^{p_1}$ to the simulator. On receiving them, the simulator runs the following steps.

- ∗ It first randomly generates a bilinear pairing $(\mathbb{G}_1, \mathbb{G}_2, p, g, e)$, a and $b \in \mathbb{Z}_p^*$, and a matrix M over $\mathbb{Z}_p$ of order $(d+2) \times (d+2)$.
- ∗ It computes a Bloom filter $BF(\mathcal{H})$, where $\mathcal{H} = \{H(e(g,g)^{abi}) \mid i = 1, 2, \cdots, T^2 + T\}$, and sends $PP = \{(\mathbb{G}_1, \mathbb{G}_2, p, g, e), BF(\mathcal{H})\}$ to $\mathcal{A}$.
- ∗ For each data vector $\mathbf{u}_i$, it randomly generates a $(d+2)$-dimensional vector $\mathbf{u}_i'$ over $\mathbb{Z}_p$ and computes $\mathbf{u}_i$'s ciphertext $[\![\mathbf{u}_i]\!]_1' = \{g^{a u_{i,k}''}\}_{k=1}^{d+2}$, where $\mathbf{u}_i'' = \{u_{i,k}''\}_{k=1}^{d+2} = \mathrm{M}^{-1}\mathbf{u}_i'^T \bmod p$.

• **Query phase I.** In the query phase I, $\mathcal{A}$ randomly chooses p_2 d-dimensional token vectors $\{\mathbf{v}_j\}_{j=1}^{p_2}$, where p_2 is a polynomial number, and sends them to the simulator. On receiving them, the simulator uses the leakage function $\mathcal{L}$ and $\{[\![\mathbf{u}_i]\!]_1'\}_{i=1}^{p_1}$ (i.e., the ciphertexts of $\{\mathbf{u}_i\}_{i=1}^{p_1}$) to generate the ciphertexts of $\{[\![\mathbf{v}_j]\!]_2'\}_{j=1}^{p_2}$. For each token vector $\mathbf{v}_j$, the simulator generates a ciphertext $[\![\mathbf{v}_j]\!]_2'$ as follows.

- ∗ First, the simulator generates a p_1-dimensional vector $S_j = \{s_i\}_{i=1}^{p_1}$, in which each element s_i is a random integer and satisfies that $|s_i| \in (0, T^2 + T]$ and $\mathrm{sign}(s_i) = \mathrm{sign}(\langle \mathbf{u}_i, \mathbf{v}_j\rangle)$ for $i = 1, 2, \cdots, p_1$.
- ∗ Second, the simulator randomly chooses a vector $\mathbf{v}_j'$ over $\mathbb{Z}_p$ such that $\langle \mathbf{u}_i, \mathbf{v}_j\rangle = s_i$, for $i = 1, 2, \cdots, p_1$.
- ∗ The simulator further computes $[\![\mathbf{v}_j]\!]_2' = \{g^{b v_{j,k}''}\}_{k=1}^{d+2}$, where $\mathbf{v}_j'' = \{v_{j,k}''\}_{k=1}^{d+2} = \mathbf{v}_j \mathrm{M} \bmod p$.

After that, the simulator responds $\mathcal{A}$ with $\{[\![\mathbf{v}_j]\!]_2'\}_{j=1}^{p_2}$.

• **Challenge phase.** In the challenge phase, the simulator sends $\{\mathbf{u}_i\}_{i=1}^{p_1}$ (i.e., the ciphertexts of $\{\mathbf{u}_i\}_{i=1}^{p_1}$) to $\mathcal{A}$.

• **Query phase II.** In the query phase II, $\mathcal{A}$ randomly chooses $(p_2' - p_2)$ d-dimensional token vectors $\{\mathbf{v}_j\}_{j=p_2+1}^{p_2'}$, where p_2' is a polynomial number. Then, $\mathcal{A}$ sends $\{\mathbf{v}_j\}_{j=p_2+1}^{p_2'}$ to the simulator. On receiving the token vectors, the simulator responds with the ciphertexts $\{[\![\mathbf{v}_j]\!]_2'\}_{j=p_2+1}^{p_2'}$ same as the query phase I.

In the real world, the view of $\mathcal{A}$ is $\mathsf{View}_{\mathcal{A},\mathrm{Real}} = \left\{\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}, \{[\![\mathbf{v}_j]\!]_2\}_{j=1}^{p_2'}\right\}$, while that in the ideal world is $\mathsf{View}_{\mathcal{A},\mathrm{Ideal}} = \left\{\{[\![\mathbf{u}_i]\!]_1'\}_{i=1}^{p_1}, \{[\![\mathbf{v}_j]\!]_2'\}_{j=1}^{p_2'}\right\}$. Then, based on the views of $\mathcal{A}$ in the real and ideal worlds, we define the security of the IPPE scheme.

Definition 2. (Security of the IPPE scheme). The IPPE scheme is selectively secure with the leakage $\mathcal{L}$ *iff* for any $\mathcal{A}$ issuing a polynomial number of data vector encryption and token vector encryption there exists a simulator such that the advantage that $\mathcal{A}$ can distinguish the views of real and ideal worlds is negligible, i.e. $|\Pr[\mathsf{View}_{\mathcal{A},\mathrm{Real}} = 1] - \Pr[\mathsf{View}_{\mathcal{A},\mathrm{Ideal}} = 1]|$ is negligible.

In the following, we prove that the IPPE scheme is selectively secure with the leakage $\mathcal{L}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

Theorem 1. *The IPPE scheme is selectively secure with $\mathcal{L}$.*

Proof. We prove the security of our IPPE scheme by proving that $\mathcal{A}$ cannot distinguish the views of real and ideal worlds. Specifically, the views in the real world and the ideal world are $\mathsf{View}_{\mathcal{A},\mathrm{Real}} = \left\{\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}, \{[\![\mathbf{v}_j]\!]_2\}_{j=1}^{p_2'}\right\}$ and $\mathsf{View}_{\mathcal{A},\mathrm{Ideal}} = \left\{\{[\![\mathbf{u}_i]\!]_1'\}_{i=1}^{p_1}, \{[\![\mathbf{v}_j]\!]_2'\}_{j=1}^{p_2'}\right\}$, respectively. Since all ciphertexts in the ideal world are randomly generated by the simulator, distinguishing the two views is equivalent to distinguish $\mathsf{View}_{\mathcal{A},\mathrm{Real}}$ from random ciphertexts. To prove the indistinguishability, we consider the following two cases.

Case 1: The ciphertexts $\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$ and $\{[\![\mathbf{v}_j]\!]_2\}_{i=1}^{p_2'}$ are indistinguishable from random ciphertexts;

Case 2: The intermediate results $\{\psi_{i,j} = \gamma_{i,2}\gamma_{j,4}\langle \mathbf{u}_i, \mathbf{v}_j\rangle + \gamma_{i,1} + \gamma_{j,3} \mid 1 \le i \le p_1, 1 \le j \le p_2'\}$ are indistinguishable from random integers in range $(0, T^2 + T)$, where $\gamma_{i,1}$ and $\gamma_{i,2}$ (resp. $\gamma_{j,2}$ and $\gamma_{j,4}$) represent the random numbers γ_1 and γ_2 (resp. γ_3 and γ_4) in the ciphertext $[\![\mathbf{u}_i]\!]_1$ (resp. $[\![\mathbf{v}_j]\!]_2$).

Specifically, in addition to the ciphertexts considered in **Case 1**, we need to consider the indistinguishability between intermediate results that can be obtained by $\mathcal{A}$, and $\{\psi_{i,j} \mid 1 \le i \le p_1, 1 \le j \le p_2'\}$ are the intermediate results with the least number of unknown variables. This is mainly due to that, $\mathcal{A}$ can remove the randomness introduced by the random matrix M by computing

$$\Psi_{i,j} = \prod_{k=1}^{d+2} e(g^{au_i''}, g^{bv_j''}) = e(g,g)^{ab(\gamma_{i,2}\gamma_{j,4}\langle \mathbf{u}_i, \mathbf{v}_j\rangle + \gamma_{i,1} + \gamma_{j,3})}.$$

Then, $\mathcal{A}$ can easily guess the corresponding $\psi_{i,j} = \gamma_{i,2}\gamma_{j,4}\langle \mathbf{u}_i, \mathbf{v}_j\rangle + \gamma_{i,1} + \gamma_{j,3}$ of each $\Psi_{i,j}$ with $BF(\mathcal{H})$ and multiple $\Psi_{i,j}$, as $|\psi_{i,j}| \in (0, T^2 + T)$ is not sufficiently large. Thus, if $\mathcal{A}$ cannot distinguish $\{\psi_{i,j} \mid 1 \le i \le p_1, 1 \le j \le p_2'\}$ from random integers in range $(0, T^2+T)$, all other intermediate results are indistinguishable from random ciphertexts. Therefore, we consider **Case 1** and **Case 2** in the indistinguishability proof as follows.

- **$\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$ and $\{[\![\mathbf{v}_j]\!]_2\}_{i=1}^{p_2'}$ are indistinguishable from random ciphertexts.** For $[\![\mathbf{u}_i]\!]_1$, the k-th element in it is $(g^a)^{u_{i,k}''}$ and $u_{i,k}'' = \langle \mathrm{M}_k^{-1}, \mathbf{u}_i'\rangle \in \mathbb{Z}_p^*$, where M_k^{-1} is the k-th row of matrix M^{-1}. Without knowing the value of M, $\mathcal{A}$ can neither recover $u_{i,k}''$ based on the DLP problem nor distinguish $(g^a)^{u_{i,k}''}$ from a random element in $\mathbb{Z}_p$. Thus, $\mathcal{A}$ cannot distinguish $\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$ from random ciphertexts. Similarly, $\{[\![\mathbf{v}_j]\!]_2\}_{i=1}^{p_2'}$ is indistinguishable from random ciphertexts. Therefore, $\{[\![\mathbf{u}_i]\!]_1\}_{i=1}^{p_1}$ and $\{[\![\mathbf{v}_j]\!]_2\}_{i=1}^{p_2'}$ are indistinguishable from random ciphertexts.
- **$\{\psi_{i,j} \mid 1 \le i \le p_1, 1 \le j \le p_2'\}$ are indistinguishable from random integers in range $(0, T^2 + T)$.** Since $\psi_{i,j} = \gamma_{i,2}\gamma_{j,4}\langle \mathbf{u}_i, \mathbf{v}_j\rangle + \gamma_{i,1} + \gamma_{j,3}$ and it contains many random numbers including $\{\gamma_{i,1}, \gamma_{i,3}, \gamma_{j,2}, \gamma_{j,4}\}$. These random numbers can make $\psi_{i,j}$ indistinguishable from a random integer in range $(0, T^2 + T)$. Thus, $\{\psi_{i,j} \mid 1 \le i \le p_1, 1 \le j \le p_2'\}$ are indistinguishable from random integers in range $(0, T^2 + T)$.

Thus, we can deduce that $\mathsf{View}_{\mathcal{A},\mathrm{Real}}$ is indistinguishable from random ciphertexts. Hence, $\mathcal{A}$ cannot distinguish $\mathsf{View}_{\mathcal{A},\mathrm{Real}}$ and $\mathsf{View}_{\mathcal{A},\mathrm{Ideal}}$, and thereby, it cannot distinguish the two worlds. Therefore, the IPPE scheme is selectively secure with the leakage $\mathcal{L}$. □

5.1.2 Security of The IPPE-Based Proximity Detection and Rectangle Intersection Detection Approaches

Based on the security of the IPPE scheme, we respectively analyze the security of the IPPE-based proximity detection approach and the IPPE-based rectangle-intersection detection approach as follows.

- *The IPPE-based proximity detection approach is privacy-preserving:* Given an encrypted data vector $[\![\mathbf{u}_{P_i}]\!]_1$ and an encrypted token vector $[\![\mathbf{v}_{P_j,\varepsilon}]\!]_2$ representing the coordinates of two points $\{P_i, P_j\}$ and a distance threshold ε, the server can run the IPPE-based proximity detection approach to determine whether the distance between the two points exceeds the distance threshold, i.e., $d(P_i, P_j) > \varepsilon$, or not. In the meantime, the server may be curious about the coordinates of $\{P_i, P_j\}$, the distance $d(P_i, P_j)$, and ε. However, since P_i and $\{P_j, \varepsilon\}$ are respectively encrypted in $[\![\mathbf{u}_{P_i}]\!]_1$ and $[\![\mathbf{v}_{P_j,\varepsilon}]\!]_2$, based on the security of the IPPE scheme, the server cannot obtain P_i, P_j and ε. Meanwhile, the inner-product of the two vectors $\langle \mathbf{u}_{P_i}, \mathbf{v}_{P_j,\varepsilon}\rangle$ represents the difference between the distance and the distance threshold, i.e., $d(P_i, P_j) - \varepsilon$. Based on the property of the IPPE scheme, the server can only obtain $sign(d(P_i, P_j) - \varepsilon)$, but it cannot recover the distance $d(P_i, P_j)$. Thus, the IPPE-based proximity detection is privacy-preserving.
- *The IPPE-based rectangle-intersection detection approach is privacy-preserving:* Given the ciphertexts of two rectangles $\{R_i, R_j\}$, i.e., $\{[\![\mathbf{u}_{i,m}]\!]_1\}_{m=1}^{k}$ and $\{[\![\mathbf{v}_{j,m}]\!]_2\}_{m=1}^{k}$, the server can run the IPPE-based rectangle-intersection detection approach to determine whether R_i intersects with R_j or not. In the meantime, the server may be curious about the locations and the sizes of R_1 and R_2. However, based on the random number for each dimension $\widehat{\gamma}_m$, the inner-product between data vectors and token vectors from different dimensions will not reveal any useful information. Therefore, we mainly focus on the privacy-preservation for each dimension. That is, given an encrypted data vector $[\![\mathbf{u}_{i,m}]\!]_1$ and an encrypted token vector $[\![\mathbf{v}_{j,m}]\!]_2$, the server can determine whether R_i and R_j overlap on the m-th dimension or not. Meanwhile, it may be curious about the ranges covered by R_i and R_j on this dimension, i.e., $[x_{i,m,1}, x_{i,m,2}]$ and $[x_{j,m,1}, x_{j,m,2}]$, and the relative locations of the two ranges, i.e., $x_{i,m,2} - x_{j,m,1}$ and $x_{j,m,2} - x_{i,m,1}$. Nevertheless, since the ranges $R_{i,m}$ and $R_{j,m}$ are respectively encrypted into $[\![\mathbf{u}_{i,m}]\!]_1$ and $[\![\mathbf{v}_{j,m}]\!]_2$, based on the privacy of the IPPE scheme, the server cannot obtain the two ranges. Moreover, based on the property of the IPPE scheme, the server can only know the sign of the inner-product, i.e., $sign(\langle \mathbf{u}_{i,m}, \mathbf{v}_{j,m}\rangle)$. Hence, the server only obtains which dimensions do R_1 intersect with R_2, but no more useful information related to the two rectangles are revealed. That is, the IPPE-based rectangle-intersection detection approach is privacy-preserving.

5.2 Security of the Proposed Scheme

In this subsection, we prove that our proposed PDRQ scheme is *selectively secure* against honest-but-curious adversary with a leakage function $\mathcal{L}_{\mathsf{Scheme}}$, following the formal definition of *selective security* model in [13], [14]. Given the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

dataset $\mathcal{T}$ and a set of queries $\mathcal{Q}$, the leakage function $\mathcal{L}_{\mathsf{Scheme}} = \{\mathsf{IS}(\mathcal{T}), \mathsf{AP}(\mathcal{T}, \mathcal{Q}), \mathsf{SP}(\mathcal{T}, \mathcal{Q})\}$ consists of three parts:

• Index Structure $\mathsf{IS}(\mathcal{T})$: the fan-out of each inner node in the two R-trees $\{T_\ell\}_{\ell=1}^2$, the identical relationships between leaf nodes in the two R-trees, and the number of points in each trajectory $t_i \in \mathcal{T}$.

• Access Pattern $\mathsf{AP}(\mathcal{T}, \mathcal{Q})$: the collection of all records in $\mathcal{T}$ that match each query $Q \in \mathcal{Q}$, i.e., $\mathsf{AP}(\mathcal{T}, \mathcal{Q}) = \{E(Results_Q) \mid \forall Q \in \mathcal{Q}\}$.

• Search Pattern $\mathsf{SP}(\mathcal{T}, \mathcal{Q})$: for each query $Q = (t_q, \varepsilon) \in \mathcal{Q}$, the search pattern consists of two parts:

i) $\{M_{\ell,Q}\}_{\ell=1}^2$: two $k \times |\mathcal{T}_\ell|$ binary matrices indicating whether the MBR of Q intersects each node in T_ℓ on each dimension, where $|\mathcal{T}_\ell|$ represents the number of nodes in T_ℓ;

ii) $\{M_{t_i,Q}\}_{t_i \in \mathcal{T}}$: $|\mathcal{T}|$ binary matrices of dimension $|t_i| \times |t_q|$, where each $M_{t_i,Q}$ indicates whether the distance between each pair of points in t_i and t_q exceeds ε.

That is, the search pattern can be defined as $\mathsf{SP}(\mathcal{T}, \mathcal{Q}) = \{M_{\ell,Q} \mid Q \in \mathcal{Q}, \ell = 1, 2\} \cup \{M_{t_i,Q} \mid Q \in \mathcal{Q}, t_i \in \mathcal{T}\}$.

Theorem 2 (Security of PDRQ)**.** *PDRQ is selectively secure against honest-but-curious adversary with $\mathcal{L}_{\mathsf{Scheme}}$ if the IPPE scheme is selectively secure.*

Proof. We proof the security of PDRQ by constructing a simulator $\mathcal{S}$, which interacts with $\mathcal{A}$ in the following phases.

• **Setup phase.** $\mathcal{A}$ randomly generates a plaintext dataset $\mathcal{T}$ and submits it to $\mathcal{S}$. Then, $\mathcal{S}$ generates PP by running the key generation algorithm of IPPE and sends PP to $\mathcal{A}$. Furthermore, $\mathcal{S}$ generates ciphertexts for $\mathcal{T}$ in two steps.

Step 1. For each trajectory $t_i \in \mathcal{T}$, $\mathcal{S}$ randomly generates $|t_i|$ vectors $\{\widetilde{\mathbf{u}}_{P_{i,j}} \in \mathbf{Z}_p^{(k+2)}\}_{k=1}^{|t_i|}$ and encrypts them through IPPE as t_i's ciphertext.

Step 2. $\mathcal{S}$ generates two empty R-trees $\{\widetilde{T}_\ell\}_{\ell=1}^2$ based on $\mathsf{IS}(\mathcal{T})$. Then, $\mathcal{S}$ generates the ciphertext for each k-dimensional MBR $\prod_{m=1}^k [x_{\ell,m,1}, x_{\ell,m,2}]$ in $\widetilde{T}_\ell$ by i) randomly generates k random vectors $\widetilde{\mathbf{u}}_{\ell,m} \in \mathbb{Z}_p^4$ and ii) encrypts them through IPPE.

• **Query phase I.** $\mathcal{A}$ randomly generates p_1 queries $\{Q_i = \{t_{q_i}, \tau_i\}\}_{i=1}^{p_1}$, where p_1 is a polynomial number, and sends them to $\mathcal{S}$. On receiving them, $\mathcal{S}$ generates the token $\widetilde{Token}_{Q_i}$ for each Q_i in two steps.

Step 1. $\mathcal{S}$ generates tokens for points in t_{q_i} by computing $|t_{q_i}|$ vectors $\{\widetilde{\mathbf{v}}_{q_i,j} \in \mathbb{Z}_p^{(k+2)}\}_{j=1}^{|t_{q_i}|}$ such that $\{M_{t_i,Q_i}\}_{t_i \in \mathcal{T}}$ in the leakage $\mathcal{L}_{\mathsf{Scheme}}$ is satisfied.

Step 2. $\mathcal{S}$ computes $2 \times k$ vectors $\{\widetilde{\mathbf{v}}_{\ell,m} \in \mathbb{Z}_p^4 \mid m = 1, \cdots, k, \text{ and } \ell = 1, 2\}$ such that $\{\widetilde{M_{\ell,Q_i}}\}_{\ell=1}^2$ is satisfied.

Finally, $\mathcal{S}$ sends the tokens $\{\widetilde{Token}_{Q_i}\}_{i=1}^{p_1}$ to $\mathcal{A}$.

• **Challenge phase.** In the challenge phase, $\mathcal{S}$ sends the two encrypted R-tree $\{\widetilde{T}_\ell\}_{\ell=1}^2$ and $E(\widetilde{\mathcal{T}})$ to $\mathcal{A}$.

• **Query phase II.** Similar to **Query phase I**, $\mathcal{A}$ submits p_2 queries $\{Q_i\}_{i=p_1+1}^{p_1+p_2}$ to $\mathcal{S}$, where p_2 is a polynomial number, and $\mathcal{S}$ responses with their tokens $\{\widetilde{Token}_{Q_i}\}_{i=p_1}^{p_1+p_2}$.

We can obtain the simulated view $\mathsf{View}_{\mathcal{A},\mathrm{Ideal}}$ of $\mathcal{A} = \{\{\widetilde{T}_\ell\}_{\ell=1}^2, E(\widetilde{\mathcal{T}}), \{\widetilde{Token}_{Q_i}\}\}$, which is conspicuously indistinguishable to $\mathsf{View}_{\mathcal{A},\mathrm{Real}}$, i.e., $\mathcal{A}$'s view during running our PDRQ scheme, when our IPPE scheme is selectively secure. Hence, our PDRQ scheme is selectively secure. □

6 Performance Analysis

In this section, we evaluate the performance of our proposed scheme with respect to the computational cost of data outsourcing, token generation, and query phases. As detailed in Section 7, Zhu et al.'s work [10] is most related to our work, but adapting it to our scenario will introduce heavy computational costs and communication overhead to the query user. Therefore, it is not applicable to the IoT scenario where U_i is resource-constrained, and we will not compare it with our work in this section. To demonstrate the efficiency of our encrypted index, we build a baseline scheme by solely employ our IPPE-based proximity detection technique and compare the time consumption for the cloud server to conduct a query in our PDRQ scheme and in the baseline scheme.

6.1 Experimental Setting

We implemented our scheme[1] with Java and Java Pairing-Based Cryptography Library (jPBC) [15], and we conducted experiments on a machine with an Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz, 8GB RAM and Debian 11 operating system, and an Android 10.0 platform with a Qualcomm (R) Snapdragon (TM) 712 2.30GHz CPU and 8GB RAM. In our experiment, we select the security parameter $\kappa = 512$ and run the key generation phase of the IPPE scheme to obtain the keys PP, EK, and TK, and we employ the Bloom filter implementation provided by Google Guava [16] with parameters `expectedInsertions` $= 30,000$ and `fpp` $= 0.03$ (equivalent to $\Delta = 6.57 \times 10^9$ and $f = 5$). We evaluate the performance of PDRQ on a commonly used synthetic dataset [17], which contains 20,200 GPS trajectories of varied lengths. Since the values in the dataset are real numbers, by respectively selecting offsets for latitude and longitude, we first map each of them to a positive number by adding the corresponding offset. Then, we further round them to integers after dividing them by 100. In addition, for each experiment, we conduct 50 times and the average result is reported.

6.2 Experimental Results

In this subsection, we respectively show the computational cost of data outsourcing, query token generation, and discrete Fréchet distance range query phases.

6.2.1 Data Outsourcing

To protect the privacy of the trajectory data, the service provider SP encrypts his/her it in the data outsourcing phase. Therefore, the computational cost is related to two parameters, namely, the number of trajectories in the dataset N and the length of each trajectory l. To evaluate the relationship between the computational cost and the two parameters, we construct sets of trajectory from the original dataset with numbers of trajectories ranging from 200 to 1000, while the lengths of trajectories in these sets range from 10 to 50. As shown in Fig. 5, the average time consumption of the data outsourcing phase increases with the number of trajectories N and the trajectory length l of the dataset.

1. The code is available at https://github.com/guanyg/frechet .

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

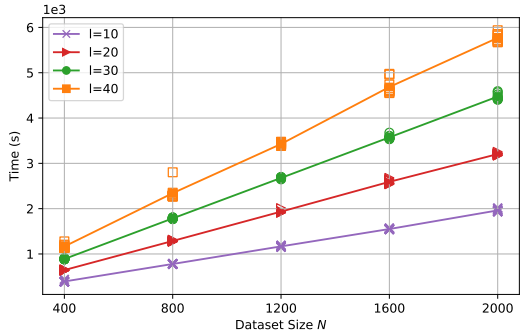

Fig. 5. Average time consumption of data outsourcing with different number of trajectories and different trajectory lengths.

6.2.2 Query Token Generation

To launch a query, the query user U encrypts his/her query trajectory and its MBR, in which the computational cost for encrypting the trajectory is linear to its length, and the computational cost for encrypting the MBR is constant. Therefore, the overall computational cost of this phase is related to the length of the query trajectory. As shown in Fig. 6, the time consumption for the PC and the Android platforms increases with the trajectory length l. When $l = 50$, the average time consumption for the PC platform to generate a query token is around 2.4s, while that for the Android platform is around 4.3s.

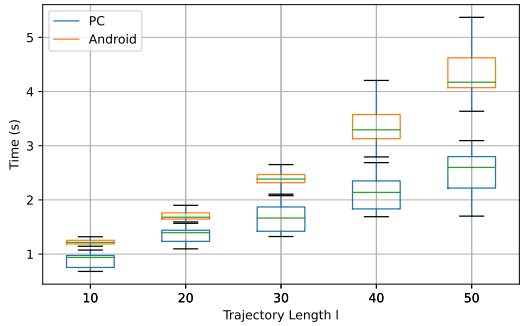

Fig. 6. Average time consumption of query token generation with different trajectory lengths.

6.2.3 DFD Range Query Processing

As detailed in Section 4.4, given a DFD range query token $Token$, the cloud server CS first obtains a set of candidates by querying the two R-trees, and then it refines the candidate set by verifying each trajectory in it. Specifically, the computational cost of the first step is related to the number of trajectories in the dataset and the threshold ε, while that of the second step is related to the length of trajectories and the query threshold ε. We respectively analyze the computational cost of these two steps as follows.

• *The computational cost of the filtration step:* As shown in Fig. 7, the average time consumption for the filtration step increases with the number of trajectories in the dataset and the threshold ε. This is mainly because that, as the number of trajectories increases, the two R-trees will become deeper, which results in more rectangle intersection operations. In addition, as a larger threshold ε results in a larger query range for each R-tree, the computation cost also increases with ε.

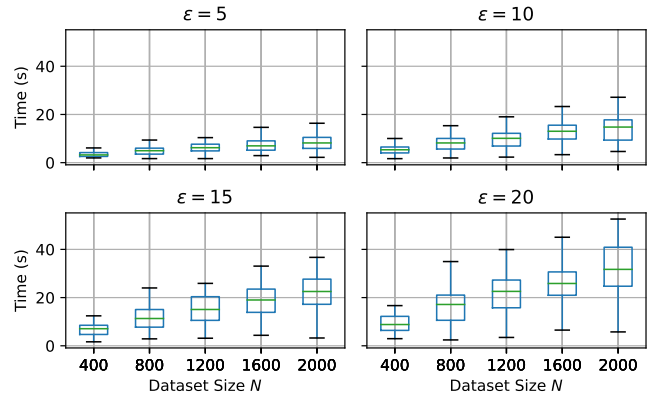

Fig. 7. Average time consumption for the server to obtain candidate sets for queries of different threshold ε on datasets containing different numbers of trajectories.

As shown in Fig. 8, the number of candidates obtained in the filtration step increases with the number of trajectories in the dataset and the query threshold. However, the number of candidates is considerably smaller than the original dataset. For instance, while conducting a query with $\varepsilon = 20$ over a dataset containing 1000 trajectories, the candidate set averagely contains around 20 trajectories. That is, our filtration step can effectively reduce the computational cost for conducting queries.

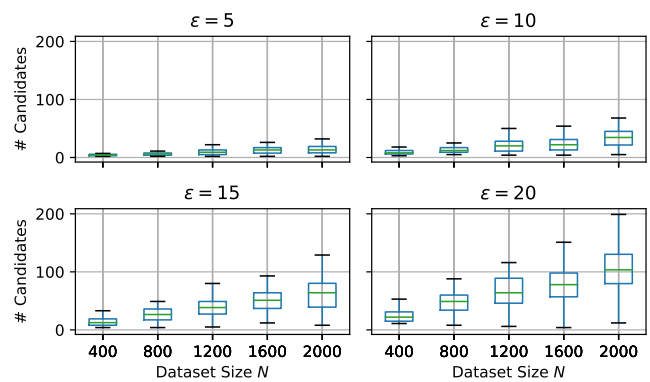

Fig. 8. Number of candidates obtained in the filtration step of different numbers of trajectories in the dataset and different query thresholds ε.

• *The computational cost of the verification step:* As shown in Fig. 9, the upper bound of the time required for the verification step increases with the length of trajectories, since the dynamic programming algorithm for verifying the discrete Fréchet distance may take more steps for verifying longer trajectories. However, as demonstrated in the figure, the average time consumption only slightly increases with the trajectory length. This is mainly because that, for most

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3171980, IEEE Transactions on Dependable and Secure Computing

TABLE 2
Comparison on Existing Works

	Scenario	Technique	Efficiency	Drawbacks
[18]–[24]	Trajectory publish / query	Anonymization-based	Efficient	Cannot obtain accurate query results
[10], [25], [26]	Distance metric evaluation / query	Work-specific protocols	Inefficient	Cannot directly support DFD range queries
[27], [28]	Proximity testing	PET or PSI	Inefficient	Cannot efficiently handle DFD range queries

cases, the algorithm do not need to go through the two trajectories to verify whether their DFD is larger than ε.

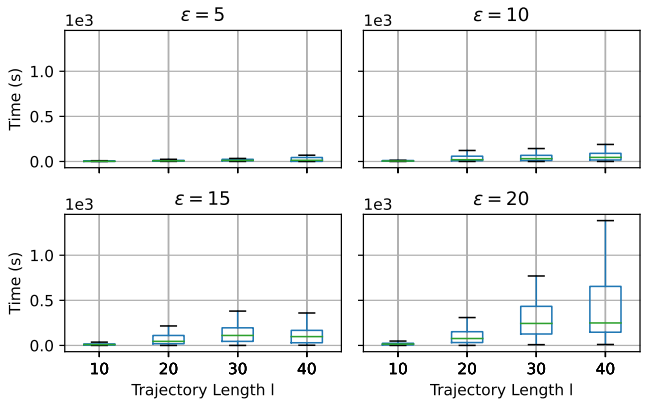

Fig. 9. Time required for the server to verify each trajectory of different lengths and different thresholds ε. In this figure, the orange line represents the average time consumption of the verification step for different length of trajectories.

In Fig. 10, we plot the overall computational cost for the cloud to handle DFD range queries in our PDRQ scheme and the baseline scheme. As shown in the figure, the time consumption for the cloud server both schemes increases with the dataset size N and the distance threshold ε. However, the time consumption for the cloud to handle a DFD range query in the baseline scheme is much higher than that in our PDRQ scheme, and it increases linearly to the dataset size N.

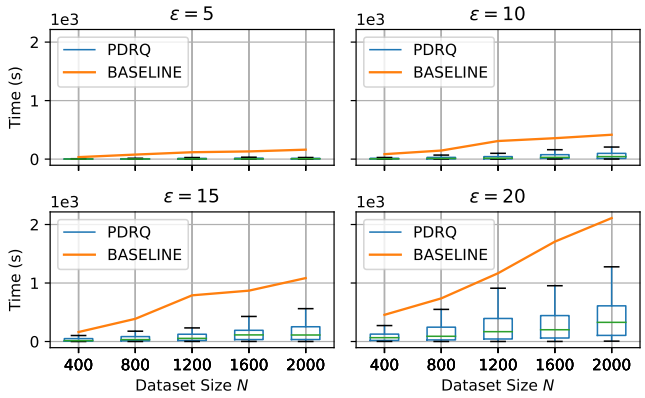

Fig. 10. Time consumption for the cloud to handle a DFD range query in our PDRQ scheme and the baseline scheme with different distance thresholds ε and different dataset sizes N. In the figure, the orange lines indicate the average time consumption for the baseline scheme to handle a DFD range query, and the box plots demonstrate the query time consumption for our PBRQ scheme.

7 Related Work

Many studies have been proposed to handle range queries over records other than trajectories (e.g., two-dimensional points [29]). However, as their distance metrics cannot be directly adopted to handle trajectories, in this section, we mainly focus on privacy-preserving trajectory analysis. Based on the techniques employed in these schemes, they can be divided into two categories, namely, k-anonymity based schemes [18]–[24] and encryption based schemes [10], [25], [26]. We compare these related works and conclude their drawbacks when deploying them to handle DFD range queries in Table. 2.

• *k-anonymity based Schemes:* Chow and Mokbel's work [18] achieves trajectory k-anonymity in LBSs by dynamically expending cloaked areas such that each area will always include k trajectories. Abul et al. [19] proposed the Never Walk Alone (NWA) scheme to achieve (k, δ)-anonymity for mobile object databases based on clustering and space translation. Then, Xu and Cai [20] improved the NWA scheme by anonymizing the stay points on users' trajectories through grid-based and clustering-based approaches. Gao et al. [21] proposed a trajectory privacy-preserving framework, which involves a mix-zone model enhanced by considering the time factor from the perspective of graph theory. Focusing on continuously LBS queries, Peng et al. [22] proposed a collaborative trajectory privacy preserving scheme, in which users protect their actual trajectories by launching fake queries. Tian et al. [23] proposed a scheme for mining social ties from users' trajectories, which cloaks each location in the trajectories as semantic regions. Zhang et al. [24] proposed a trajectory privacy-preserving scheme, in which multiple anonymizers and the dynamic pseudonym mechanism are employed to preserve users' trajectory privacy. The above-mentioned works cannot apply to our scenario, since their privacy-preserving feature is obtained by reducing the accuracy of locations, which may significantly affect the correctness of DFD range queries.

• *Encryption based Schemes:* Zhu et al. [10] proposed a privacy-preserving time series distance evaluation scheme in which Dynamic Time Warping (DTW) and discrete Fréchet distance are considered, which can be also applied on trajectory data. However, to apply it to our scenario, the server needs to compute the encrypted DFDs between the query trajectory and every trajectories in the dataset and further compare them with the encrypted threshold under the user's help, which has unacceptable costs on both computation and communication. Based on the Paillier cryptosystem and garbled circuits, Liu et al.'s work [25] can securely evaluate DTW, Longest Common Subsequences (LCSS) and Edit Distance on Real sequence (EDR) distances, but it cannot be applied to conduct DFD range queries due to the same reason as Zhu et al.'s work. Based on Paillier

and Sharmir's secret sharing, Hallgren et al. [26] proposed a scheme to test whether two encrypted trajectories satisfy the criteria for ride sharing. However, it is built upon a secure protocol specifically tailored for the ride sharing scenario and cannot be easily adapted to support DFD range queries. Hence, all the above-mentioned works cannot apply to our scenario, as they employ specifically tailored protocols or cannot efficiently support range queries.

Therefore, although many privacy-preserving trajectory analysis schemes have been proposed, some of them cannot directly apply to our scenario, while some others suffer from sever accuracy or efficiency issues. On the other hand, according to the definition of DFD, existing works on privacy-preserving proximity testing seem to be a sound solution for DFD range queries. Nevertheless, many of them [27], [28] are built upon private equality testing (PET) or private set intersection (PSI) protocols and cannot efficiently support dynamic distance thresholds with one party offline (to simulate the encrypted static points in the outsourced trajectories). More importantly, as demonstrated by our experimental results, applying these schemes to handle DFD range queries requires linearly scanning the whole dataset to verify whether each trajectory matches the query requests, resulting in an unacceptable performance on large datasets. In this paper, we propose a PDRQ scheme to efficiently achieve accurate DFD range query while protecting the privacy of the dataset and users' queries.

8 Conclusion

In this paper, we have proposed a novel privacy-preserving discrete Fréchet distance range query scheme, which can return accurate query results while preserving the privacy of the dataset and user queries. Specifically, to build our privacy-preserving discrete Fréchet distance range query scheme, we employed bilinear pairing to design an Inner-Product Preserving Encryption (IPPE) scheme. Then, based on the IPPE scheme, we respectively developed two approaches for proximity detection and rectangle intersection detection. Furthermore, we built our privacy-preserving discrete Fréchet distance range query scheme based on the IPPE-based proximity detection and rectangle intersection detection approaches, in which user queries are conducted in a filtration-and-verification manner. Detailed security analysis shows that our proposed scheme is indeed privacy-preserving, and the performance analysis also indicates that our filtration process can significantly reduce the computational cost. In our future work, we will further evaluate our proposed scheme in real platform and exploit how to improve the efficiency of the verification step.

Acknowledgments

This research was supported in part by NSERC Discovery Grants (04009).

References

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

[2] "Global smart cities market by smart transportation, smart buildings, smart utilities, smart citizen services and region - forecast to 2025," https://www.researchandmarkets.com/reports/5146372/\\global-smart-cities-market-by-smart, 2020.

[3] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data," *J. Medical Syst.*, vol. 43, no. 5, pp. 123:1–123:13, 2019.

[4] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Achieving efficient and privacy-preserving exact set similarity search over encrypted data," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[5] Y. Zheng, R. Lu, J. Shao, F. Yin, and H. Zhu, "Achieving practical symmetric searchable encryption with search pattern privacy over cloud," *IEEE Transactions on Services Computing*, 2020.

[6] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3306–3317, 2016.

[7] Z. Lin, F. Wen, Y. Ding, Y. Xue, S. Liu, Y. Zhao, and S. Yi, "Wams-based coherency detection for situational awareness in power systems with renewables," *IEEE Trans. on Power Syst.*, vol. 33, no. 5, pp. 5410–5426, 2018.

[8] P. Shui, X. Xia, and Y. Zhang, "Sea-land segmentation in maritime surveillance radars via k-nearest neighbor classifier," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 5, pp. 3854–3867, 2020.

[9] H. Wei, R. Fellegara, Y. Wang, L. D. Floriani, and H. Samet, "Multi-level filtering to retrieve similar trajectories under the fréchet distance," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. ACM, 2018, pp. 600–603.

[10] H. Zhu, X. Meng, and G. Kollios, "Privacy preserving similarity evaluation of time series data," in *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*. OpenProceedings.org, 2014, pp. 499–510.

[11] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*. ACM Press, 1984, pp. 47–57.

[12] Y. J. García, M. A. López, and S. T. Leutenegger, "A greedy algorithm for bulk loading r-trees," in *ACM-GIS '98, Proceedings of the 6th international symposium on Advances in Geographic Information Systems, November 6-7, 1998, Washington, DC, USA*. ACM, 1998, pp. 163–164.

[13] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan, "From selective to adaptive security in functional encryption," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 9216. Springer, 2015, pp. 657–677.

[14] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. ACM, 2018, pp. 745–762.

[15] A. D. Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28 - July 1, 2011*. IEEE Computer Society, 2011, pp. 850–855.

[16] Google, "Guava," https://github.com/google/guava.

[17] M. Werner and D. Oliver, "ACM SIGSPATIAL GIS cup 2017: range queries under fréchet distance," *ACM SIGSPATIAL Special*, vol. 10, no. 1, pp. 24–27, 2018.

[18] C. Chow and M. F. Mokbel, "Enabling private continuous queries for revealed user locations," in *Advances in Spatial and Temporal Databases, 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16-18, 2007, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4605. Springer, 2007, pp. 258–275.

[19] O. Abul, F. Bonchi, and M. Nanni, "Never walk alone: Uncertainty for anonymity in moving objects databases," in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*. IEEE Computer Society, 2008, pp. 376–385.

[20] T. Xu and Y. Cai, "Exploring historical location data for anonymity preservation in location-based services," in *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*. IEEE, 2008, pp. 547–555.

[21] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "Trpf: A trajectory privacy-preserving framework for participatory sensing," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 6, pp. 874–887, 2013.

[22] T. Peng, Q. Liu, D. Meng, and G. Wang, "Collaborative trajectory privacy preserving scheme in location-based services," *Inf. Sci.*, vol. 387, pp. 165–179, 2017.

[23] Y. Tian, W. Wang, J. Wu, Q. Kou, Z. Song, and E. C. H. Ngai, "Privacy-preserving social tie discovery based on cloaked human trajectories," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1619–1630, 2017.

[24] S. Zhang, X. Mao, K. R. Choo, T. Peng, and G. Wang, "A trajectory privacy-preserving scheme based on a dual-k mechanism for continuous location-based services," *Inf. Sci.*, vol. 527, pp. 406–419, 2020.

[25] A. Liu, K. Zheng, L. Li, G. Liu, L. Zhao, and X. Zhou, "Efficient secure similarity computation on encrypted trajectory data," in *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. IEEE Computer Society, 2015, pp. 66–77.

[26] P. A. Hallgren, C. Orlandi, and A. Sabelfeld, "Privatepool: Privacy-preserving ridesharing," in *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*. IEEE Computer Society, 2017, pp. 276–291.

[27] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.

[28] P. Kotzanikolaou, C. Patsakis, E. Magkos, and M. Korakakis, "Lightweight private proximity testing for geospatial social networks," *Comput. Commun.*, vol. 73, pp. 263–270, 2016.

[29] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 704–719, 2016.

Yunguo Guan is a PhD student of the Faculty of Computer Science, University of New Brunswick, Canada. His research interests include applied cryptography and game theory.

Rongxing Lu (Fellow, IEEE) is Mastercard IoT Research Chair, a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. 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 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Yandong Zheng received her M.S. degree from the Department of Computer Science, Beihang University, China, in 2017 and she is currently pursuing her Ph.D. 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.

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

Jun Shao (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science 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, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Guiyi Wei is a professor of the School of Computer and Information Engineering at Zhejiang Gongshang University. He obtained his Ph.D. in Dec 2006 from Zhejiang University, where he was advised by Cheung Kong chair professor Yao Zheng. His research interests include wireless networks, mobile computing, cloud computing, social networks and network security.