Efficient Privacy-preserving Contact Query Processing over Trajectory Data in Cloud

Siqi Cai
1024041101
Nanjing University of Posts and Telecommunications
Nanjing,China

Abstract. With the expansion of cloud computing, its effectiveness in handling large datasets, particularly trajectory, has become evident. However, due to the sensitivity of trajectory data, sharing it in plaintext could lead to privacy risks. It is a challenge to implement secure and efficient contact query based on trajectory data in the cloud. In this paper, we propose a privacy-preserving contact query processing over trajectory data. The trajectory vectorization is designed to encrypt trajectories that support secure distance comparison through vector inner product. Based on the trajectory vectors, the baseline privacy-preserving contact query processing scheme (EPCQ) is introduced. To improve the efficiency of contact query, the quick-filtering checkpoint optimization strategy is presented. By adopting it, the enhanced privacy-preserving contact query processing scheme (EPCQ+) is introduced. Experimental results show that the proposed schemes perform well in query accuracy and efficiency.

Keywords: cloud computing, privacy-preserving, moving objects, contact query

1 Introduction

Cloud computing, as an important and popular "X-as-Service" computing model, is now transforming the way we live and work. By centralizing resources, cloud computing provides great convenience for massive data processing. In this era of data explosion, trajectory data is rapidly growing as the spatio-temporal sampling devices widely deployed. Due to its massive scale, most companies choose to leverage cloud computing to improve performance and reduce cost [1]. In practice, trajectory data contains a significant amount of sensitive information, which can be used for meaningful operations such as contact query [19, 17], convoy discovery [22], etc. However, outsourcing trajectory data to the cloud could cause privacy leakage issues. Therefore, the privacy issues over the outsourced trajectory data in the cloud need to be confronted [23, 24].

Among the various trajectory-based pattern discovery, we focus on the contact query in this paper. A contact refers to the proximity of two moving objects to each other for a certain time period. For example, as shown in Fig.1, an object o is in continuous contact with a contact source s, then o fulfills the criteria for a contact. One important application is to query contacts during infectious disease epidemics [21, 8, 20, 26]. And due to the ease of access and computational simplicity of trajectory data, it is more suitable for performing contact query. Li et al. [16] accomplish the contact query by clustering trajectories. Chao et al. [6] propose a Trajectory Close Search (TCS) algorithm. But they all use the plaintext data and do not consider the privacy protection. Some people accomplish privacy-preserving contact query by using security hardware, but this are limited by circumstances. Therefore, it is necessary to investigate the trajectory-based privacy-preserving contact query processing schemes in the cloud.

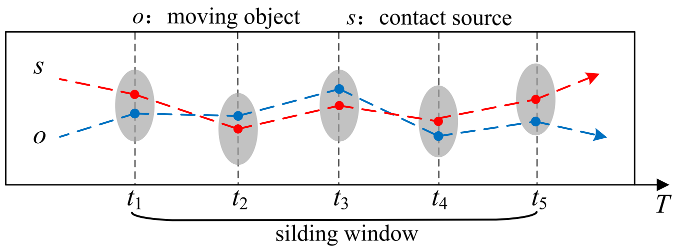


Fig. 1: Example of contact

In this paper, we propose a privacy-preserving contact query processing scheme, which can implement contact query over trajectory data in the cloud. In this novel scheme, the trajectory vectorization is designed to encrypt the plaintext trajectory data into vectors. The trajectory vectors product allows a secure comparison of the distance between two moving objects with the contact distance threshold, which is the key to implementing privacy-preserving contact query processing. Based on this, we propose the baseline privacy-preserving contact query processing (EPCQ). To improve the efficiency of query, we further propose an optimization strategy in the enhanced privacy-preserving contact query processing (EPCQ+). The optimization strategy is to apply quick-filtering checkpoint in query processing, which can accelerate the contact event determination, and thus improved the query efficiency. Experimental results show that the proposed schemes have good performance in query accuracy and efficiency.

2 Related Work

2.1 Plaintext Contact Query Processing

Recently, the widespread use of information technology has greatly promoted contact queries, primarily through Bluetooth and smartphone applications. [10, 27, 5, 7, 14, 18, 3, 4, 11]. For example, TraceTogether [14], launched by the Singapore government, can track the proximity of two app users via Bluetooth. Hisada et al. [11] performed early contact exclusion by using the search query logs of smart devices. However, the above approach is limited by the hardware environment. To avoid the above limitations, some others use spatio-temporal data-based approach for contact query [16, 6, 2, 29]. For example, Li et al. [16]

perform contact querying by clustering trajectory and combining it with a KNN query. Chao et al. [6] propose a generalized Trajectory Close Search (TCS), which models the contact tracking problem and introduces an iterative algorithm to gradually identify the existence of contacts.

2.2 Privacy-preserving Contact Query Processing

All the above schemes overlook the protection of users' private information. Whether based on device or spatio-temporal data, these schemes involve the collection of users' personal information, highlighting the crucial need for privacy preservation. In this regard [12, 9, 13, 28, 15], Kato et al. [12] proposed PCT, which achieves the function of privacy-preserving by using secure hardware to build a trusted execution environment (TEE). However, this approach still receives the requirement limitation of secure hardware. Fitzsimons et al. [9] used a cryptographic method based on secure two-party computation (2PC) and Li et al. [15] combined MPC and Geo-I to query contact, but the overhead of their query is very high. To address the above problems, we propose a privacy-preserving contact query processing scheme and further propose an optimization scheme to improve the query efficiency.

3 Preliminaries

3.1 Bloom Filter

The Bloom filter is a data structure that can be used to detect the presence of an element in a set. Its structure is a m-bit array, i.e., $BF = [B_1, B_2, \cdots, B_m]$. It also needs to be equipped with k different hash functions H_i, where $i \in [1, k]$. When an element is inserted into the Bloom filter, the element is first computed using k different hash functions, producing k hash values and finding the corresponding k different positions in the bit array to set 1 and generate a m-bit vector. For example, given a number 7, a 4-bit bloom filter with one hash function $H(\cdot)$, it will first compute the value $H(7)\ mod\ 4 = 2$, and then set the $B_2 = 1$, eventually generate the vector $BF =$ [0,1,0,0].

3.2 Secure Inner Product

The secure inner product is a computational algorithm designed to securely compute inner products, which can compute the inner product of two vectors in the ciphertext without knowing the plaintext. Assuming that V_p and V_q are two m-dimensional vectors and M is a random $m \times m$ dimensional invertible matrice which is used as the secret key, V_p and V_q are encrypted into $\widetilde{V_p} = M^T p$ and $\widetilde{V_q} = M^{-1} q$, respectively. In this case, the inner product of encrypted vectors $\widetilde{V_p}$ and $\widetilde{V_q}$ is computed as shown in Eq.1.

$$\widetilde{V_p} \cdot \widetilde{V_q} = (M^T V_p)^T (M^{-1} V_q) = V_p^T M M^{-1} V_q = V_p \cdot V_q \tag{1}$$

Eq.1 shows that the inner product result of the encrypted vectors $\widetilde{V_p}$ and $\widetilde{V_q}$ equals that of the plaintext vectors V_p and V_q.

4 Models And Problem Statement

4.1 System Model and Threat Model

System Model. The system model is composed of three components: data owner (DO), cloud server (CS) and data user (DU). DO is in charge of encrypting the trajectory data and uploading them to CS, and it also transmits the key to DU; DU generates the trapdoor based on the contact source trajectory and sends it to CS; CS executes the contact query processing on the encrypted data according to the trapdoor and then returns the query results to the DU.

Threat model. The threat model used in this paper is the "curious but honest" model. In this model, the cloud server offers dependable data and query services, executes queries accurately, and returns results to the DU without altering or deleting data. However, it maintains a "curious" stance towards the stored data and queries, attempting to extract plaintext information through statistics or other means.

4.2 Problem Formulation

Definition 1. Trajectory. The trajectory of a moving object $o_i \in O$ can be represented as a polyline consisting of a finite sequence of positions at a sequence of sampling time points. The trajectory of o_i is denoted as $tra_i = \{(p_1^i, t_1), (p_2^i, t_2), \cdots, (p_n^i, t_n)\}$, where $p_j^i = (x_j^i, y_j^i)$ represents two-dimensional space coordinates of o_i at time t_j. The set of sampling time points for the trajectory is $T = \{t_1, t_2, \cdots, t_n\}$.

Definition 2. Sliding Window. Given a window width threshold τ, the sliding window W_i is composed of τ consecutive sampling time points starting from the time point t_i, i.e., $W_i = \{t_i, t_{i+1}, \cdots, t_{i+\tau-1}\}$, where $W_i \subset T$. We use $W_i[j]$ to denote the j-th $(1 \le j \le \tau)$ time point in W_i, thus the start and end times of W_i are $W_i[1] = t_i$ and $W_i[\tau] = t_{i+\tau-1}$, respectively.

Definition 3. Contact Event. Given a trajectory tra_i of moving object o_i, a contact distance threshold d and a sliding window width τ, for a trajectory tra_j of the contact source object o_j, if there exists an earliest sliding window W_p satisfying the distance between o_i and o_j at each time point in W_p is not larger than d, then a contact event between o_i and o_j occurs in W_p and we denoted it as a triple $c =< o_i, o_j, W_p >$.

Definition 3 indicates that $< o_i, o_j, W_p >$ is a contact event if and only if the sliding window $W_p \subset T$ exists and satisfies the following two conditions:

- $|W_p| = \tau$ and $\forall t_s \in W_p(dist(tra_i, tra_j, t_s) \le d))$ hold;
- If any sliding window $W_q \subset T$ exists, which satisfies that $|W_q| = \tau$, $W_q \ne W_p$ and $\forall t_s \in W_q(dist(tra_i, tra_j, t_s) \le d)$, then $W_p[1] < W_q[1]$ holds.

where $dist(tra_i, tra_j, t_s)$ is the Euclidean distance between the sampling points in tra_i and tra_j at time point t_s.

Definition 4. Contact Query. Given a trajectory set Tra of a moving object set O, a contact distance threshold d and a sliding window width τ, for source object s with the trajectory tra_s, a contact query is to get a result set R which contains all the contact events caused by s in a time period $[t_b, t_e]$, i.e.,

$$R = \{< s, o_i, W_p > \mid o_i \in O \wedge W_p \subset [t_b, t_e]\}, \tag{2}$$

where $< s, o_i, W_p >$ is a contact event and W_p is a sliding window with width τ.

The goal of the proposed scheme is described as follows: **1) Query Efficiency.** The proposed scheme is designed to achieve efficient contact query by adopting the trajectory vectorization and an efficient query algorithm. **2) Privacy Preservation.** The proposed scheme is designed to prevent curious CS from snooping on private information. Specifically, the trajectory privacy and query command privacy should be preserved.

5 The Baseline Privacy-preserving Contact Query

In this section, we will present the trajectory vectorization and the details of the baseline privacy-preserving contact query processing scheme (EPCQ).

5.1 Trajectory Vectorization

Definition 5. Binary Encoding on Spatial Grids. Assuming that the spatial space is divided into $\theta \times \theta$ equal grids of side length kd, where d is the contact source threshold and k is a factor ($k \geq 1$), each grid corresponds to a unique integer id, $id \in (1, \theta^2)$. The (i, j)-grid will be encoded as the binary code of its integer id, i.e., $g_{i,j} = bit(id)$, where (i, j)-grid refers to the grid located in i row and j column, and bit is the binary code of the integer id.

It is noticeable that when actually dividing spatial space, the contact distance threshold d needs to be divided by $\sqrt{2}$ to ensure that the diagonal of the generated grid is d. This ensures that the distance between positions within the same grid must be less than the contact distance threshold d.

Definition 6. Bloom Filter-based Grid Vector. Given a binary code of the grid $g_{i,j}$ with n bits and a l-bit bloom filter BF with h hash functions, for the k-th bit in $g_{i,j}$ will be mapped into a BF_k according to the hash functions, the binary code after the mapping is a $(l \times n)$-bit vector as VB and generated follows Eq.3,

$$VB = BF_1 || BF_2 || \cdots || BF_n, k \in (1, n) \tag{3}$$

where $||$ is the vector concatenate operator and BF_k is the bloom filter corresponding to the k-th bit in $g_{i,j}$.

Definition 7. Position Vector. Given a l-bit bloom filter BF with h hash functions, a position p which is assumed in the (i, j)-grid and the binary code of the grid is $g_{i,j}$ with n bits, the binary code after the vectorization is a $(l \times n + 1)$-bit vector as V_p and generated follows Eq.4,

$$V_p[k] = \begin{cases} VB_p[k] & k \in \{1, 2, \cdots, l \times n\} \\ n \times h & k = l \times n + 1 \end{cases} \tag{4}$$

where VB_{p_u} is a $(l \times n)$-bit vector generated according to Definition 6.

Definition 8. Contact source Position Vector. Given a l-bit bloom filter BF with h hash functions, a contact source position s which is assumed in the (i, j)-grid and the binary code of the grid is $g_{i,j}$ with n bits, the binary code after the vectorization is a $(l \times n + 1)$-bit vector as V_s and generated follows Eq.5,

$$V_s[k] = \begin{cases} VB_s[k] & k \in \{1, 2, \cdots, l \times n\} \\ -1 & k = l \times n + 1 \end{cases} \tag{5}$$

where VB_{p_s} is a $(l \times n)$-bit vector generated according to Definition 6.

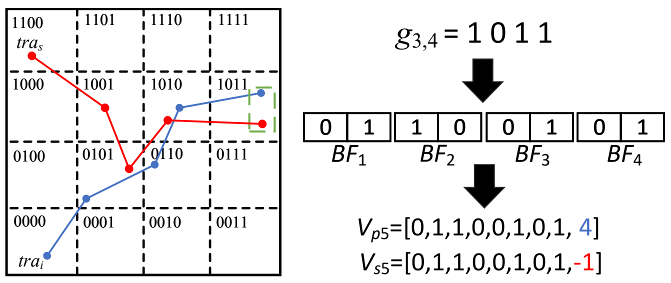


Fig. 2: Example of position vector and contact source position vector

To clearly explain Definition 7 and Definition 8, we give an example as shown in Fig.2. There is a trajectory tra_i, a contact source trajectory tra_s and a 4×4-grid space, they both have a position in $g_{3,4} = 1011$ at time 5. Given a 2-bit bloom filter with one hash function, each bit in binary code of $p_5 = g_{3,4} = 1011(p_5 \in tra_i)$ will be mapped into a bloom filter BF_i and then generate the position vector V_{p_5} according to Eq.3 and Eq.4. Similarly, the contact source position vector V_{s_5} can be generated according to Eq.3 and Eq.5.

Definition 9. Trajectory Vectors. Given a trajectory $tra_i = \{(p^i_1, t_1), (p^i_2, t_2), \cdots, (p^i_n, t_n)\}$, the trajectory vectors of tra_i, denoted as VT_i, is a sequence of positoin vector of points V_{p_u} in tra_i, i.e.,

$$VT_i = \{(V_{p_1}, t_1), (V_{p_2}, t_2), \cdots, (V_{p_n}, t_n)\}, \tag{6}$$

where V_{p_i} is the position vector of p_u generated according to Definition 7.

Definition 10. Contact Source Trajectory Vectors. Given a contact source trajectory $tra_s = \{(p^s_1, t_1), (p^s_2, t_2), \cdots, (p^s_n, t_n)\}$, the contact source trajectory vectors of tra_s is denoted as VS, which is a sequence of positoin vector of points V_{s_i} in tra_s, i.e.,

$$VS = \{(V_{s_1}, t_1), (V_{s_2}, t_2), \cdots, (V_{s_n}, t_n)\}, \tag{7}$$

where V_{s_i} is the position vector of p_i generated according to Definition 8.

The details of the trajectory vectors and contact source trajectory vectors generation are presented in the following Algorithm 1.

Algorithm 1: $GenTraVec(tra_i, d)$

```
Input: the trajectory $tra_i$, the contact distance threshold $d$
Output: the trajectory vector $VT$ or the contact trajectory vector $VS$
1  $VT, VS = \varnothing$;
2  $d = d/\sqrt{2}$;
3  Divide the spatial space into grids according to $d$ and generate all binary
     codes on spatial grids;
4  for each $(p_j, t_j) \in tra_i$ do
5      if $tra_i$ is not a contact source trajectory then
6          Create the $V_{p_j}$ according to Definition 7;
7          $VT = VT + \{(V_{p_j}, t_j)\}$;
8      else
9          Create the $V_{s_j}$ according to Definition 8;
10         $VS = VS + \{(V_{s_j}, t_j)\}$;
11 return $VT, VS$;
```

Lemma 1. Given a position p_u and a contact source position s_u at time u, the position vector and contact source position vector of which are V_{p_u} and V_{s_u}, respectively, we have

$$V_{p_u} \cdot V_{s_u} = 0 \Rightarrow dist(p_u, s_u) \le d \tag{8}$$

Proof. Assuming that V_{p_u} is the position vector of p_u which in (i, j)-grid, and V_{s_u} is the position vector of s_u which in (i', j')-grid, according to Definition 7 and Definition 8, the

$$\begin{aligned} V_{p_u} \cdot V_{s_u} &= V_{p_u}[1, 2, \cdots, ln] \cdot V_{s_u}[1, 2, \cdots, ln] + V_{p_u}[ln+1] \cdot V_{s_u}[ln+1] \\ &= \sum_{k=1}^{n} V_{p_u}[(k-1)l+1, kl] \cdot V_{s_u}[(k-1)l+1, kl] - nh \end{aligned} \tag{9}$$

where the $V_{p_u}[(k-1)l+1, kl]$ is a BF correspond to one bit in $g_{i,j}$ and the $V_{s_u}[(k-1)l+1, kl]$ is a BF correspond to one bit in $g_{i',j'}$.

If $V_{p_u} \cdot V_{s_u} = 0$, according to the Eq.9, then $\sum_{k=1}^{n} V_{p_u}[(k-1)l+1, kl] \cdot V_{s_u}[(k-1)l+1, kl] = nh$ holds. This shows that the BF corresponding to each bit of $g_{i,j}$ and $g_{i',j'}$ is the same, i.e., $g_{i,j} = g_{i',j'}$. Since $g_{i,j}$ is unique, these two positions must fall within the same grid. According to Definition 1, the maximum distance of points within the same grid is d, i.e., $dist(p_u, s_u) \le d$ holds. ■

Theorem 1. Given the trajectory vectors V_{t_i} and contact source trajectory vectors V_s corresponding to moving object o_i and contact source s, if there is a sliding window $W_l \subset T$ with the width τ satisfying the following two conditions, then the contact event $< o_i, o_j, W_l >$ is determined.

– **Condition 1:** $\forall t_k \in W_l (V_{p_k} \cdot V_{s_k} = 0)$ holds, where V_{p_k} and V_{s_k} are position vector in V_{t_i} and V_s at the time point t_s, respectively.

– **Condition 2:** If any sliding window $W_q \subset T$ exists, which satisfies that $|W_q| = \tau$, $W_q \neq W_l$ and $\forall t_m \in W_q(V_{p_m} \cdot V_{s_m} = 0)$, then $W_l[1] < W_q[1]$ holds.

Proof. According to Lemma 1, we have that $V_{p_u} \cdot V_{s_u} = 0 \Rightarrow dist(p_u, s_u) \leq d$ holds. Since p_u and s_u are the positions of trajectory tra_i and contact source tra_s at t_u, the $dist(p_u, s_u) \leq d$ is equivalent to $dist(tra_i, tra_s, t_u) \leq d$, thus $V_{p_u} \cdot V_{s_u} = 0 \Rightarrow dist(tra_i, tra_s, t_u) \leq d$. At this time, the above condition 1 and condition 2 satisfy the contact determination condition in Definition 3, so a contact event $< o_i, o_j, W_l >$ can be determined. ■

Theorem 1 indicates that the trajectory vectors can be used to determine the contact event, this is because the plaintext comparison between the distance of two moving objects and the contact distance threshold is converted into a problem of vector inner product, which is the key to achieving the privacy-preserving contact query processing.

5.2 Design of EPCQ

In this section, we give the details of the efficient privacy-preserving contact query processing scheme (EPCQ) from the setup and query processing modules.

• ***Algorithms in the Setup Module***

$\widetilde{\boldsymbol{SK}} \leftarrow$ ***GenKey***(1^λ). The *GenKey* algorithm is to generate a set of secret keys $SK = \{s, M_1, M_2, k\}$, where s is an m-dimensional bit vector, M_1 and M_1 are two random $m \times m$-dimensional invertible matrices, k is a key used to encrypt the trajectory dataset.

Algorithm 2: $EncVec(V, M_1, M_2, b, flag)$

```
  Input: the vector $V$, two invertible matrices $M_1$ and $M_2$ and the bit vector $b$
  Output: the encrypted vector $\widetilde{V}$
1 if $flag = true$ then
2     Split $V$ into $\{V', V''\}$, according to the rules:
      $\begin{cases} V'[i] = V''[i] = V[i], & b[i] = 0 \\ V'[i] + V''[i] = V[i], & b[i] = 1 \end{cases}$;
3     $\widetilde{V} = \{M_1^T V', M_2^T V''\}$;
4 else
5     Split $V$ into $\{V', V''\}$, according to the rules:
      $\begin{cases} V'[i] + V''[i] = V[i], & b[i] = 0 \\ V'[i] = V''[i] = V[i], & b[i] = 1 \end{cases}$;
6     $\widetilde{V} = \{V' M_1^{-1}, V'' M_2^{-1}\}$;
7 Return $\widetilde{V}$;
```

$\widetilde{\boldsymbol{Tra}} \leftarrow$ ***Setup(Tra, SK, kd)***. This algorithm is performed by DO to conduct the trajectory vectorization to all trajectories in the trajectory set Tra. The details of the algorithm are as follows: for each $tra_i \in Tra$, assuming that

$tra_i = \{(p_1^i, t_1), (p_2^i, t_2), \cdots, (p_n^i, t_n)\}$, DO takes tra_i as input and generates the trajectory vectors VT_i according to algorithm *GenTraVec* (Algorithm 1). Then, DO uses the algorithm *EncVec* (Algorithm 2) to encrypted VT_i to $\widetilde{VT}_i$. At last, the encrypted trajectory set $\widetilde{Tra} = \{\widetilde{VT}_1, \widetilde{VT}_2, \cdots, \widetilde{VT}_n\}$ is generated.

- ***Algorithms in the Query Processing Module***

$\widetilde{\boldsymbol{Q}} \leftarrow$ ***GenTrapdoor(Q,SK)***. Once DU starts a contact query, i.e., $Q = (tra_s, [t_u, t_v])$, where tra_s is the contact source trajectory and $[t_u, t_v]$ is the queried time period, it takes tra_s as the input and uses the algorithm *GenTraVec* and *EncVec* to generate the encrypted trajectory vector $\widetilde{VS}$. Then, the trapdoor $\widetilde{Q} = (\widetilde{VS}, [t_u, t_v])$ is submitted to CS as the query command for privacy-preserving contact query processing.

$\boldsymbol{R} \leftarrow$ ***ContactQuery(***$\widetilde{\boldsymbol{Q}}$, $\widetilde{\boldsymbol{Tra}}$***).*** Once CS receives the trapdoor $\widetilde{Q}$, it conducts Algorithm 3 to perform the privacy-preserving contact query processing and get the query result R. At last, CS returns the R to DU. The details of the contact query algorithm are shown in Algorithm 3.

Algorithm 3: $ContactQuery(\widetilde{Q}, \widetilde{Tra})$

```
Input: the trapdoor $\widetilde{Q} = (\widetilde{VS}, [t_u, t_v])$, the encrypted trajectory vectors $\widetilde{Tra}$
Output: the contact query result $R$ storing contact events caused by the
        contact source object $s$ in the time period $[t_u, t_v]$
1  $R = \varnothing$ ;
2  for each $\widetilde{VT}_i \in \widetilde{Tra}$ do
3      for each $(V_{p_j}, t_j) \in \widetilde{VT}_i, t_j \in [t_u, t_v]$ do
4          Create a sliding window $W_j = \{t_j, t_{j+1}, \cdots, t_{j+\tau-1}\}$ where
            $t_{j+\tau-1} \le t_v$;
5          $count = 0$;
6          for each $t_q \in W_j$ do
7              Get $V_{p_q}$ from $\widetilde{VT}_i$, which is the position vector of $VT_i$ at $t_q$;
8              Get $V_{s_q}$ from $\widetilde{VS}$, which is the position vector of $VS$ at $t_q$;
9              if $V_{p_q} \cdot V_{s_q} = 0$ then
10                 $count = count + 1$;
11         if $count = \tau$ then
12             $R = R + \{< s, o_i, W_p >\}$;
13 return $R$;
```

The idea of Algorithm 3 is based on Theorem 1, where the vector inner product is used to check whether two moving objects are engaged in a contact event, without knowing the plaintext trajectory locations. Thus protecting the privacy information of the trajectory data.

6 The Optimized Privacy-preserving Contact Query

In this section, we propose the optimized privacy-preserving contact query processing EPCQ+. We first discuss the idea of optimization and then introduce the quick-filtering checkpoint. Finally, we give the design details of EPCQ+.

6.1 Optimization idea and strategy

Idea of Optimization. In EPCQ, finding all possible contact events requires performing the vector inner product at all time points, which needs a significant cost in query processing. If we can filter some time points that do not meet the condition of the contact event in advance, the time cost of the vector inner product will be saved and thus improve the query efficiency.

Definition 11. Quick-filtering Checkpoint. Given a sliding window with width τ and a time list $T = \{t_1, t_2, \cdots, t_n\}$, the quick-filtering checkpoint h is a special time point in T. The set of the quick-filtering checkpoint is denoted as H, which is calculated according to Eq.10.

$$H = \{h_k | (h_k = t_i, i = (k-1) \cdot \lfloor \frac{\tau}{2} \rfloor + 1)\} \tag{10}$$

To further illustrate it, we give an example as shown in Fig.3. Assuming that the sliding window $\tau = 4$, the points marked by triangles represent the quick-filtering checkpoints, e.g. h_1 and h_2, and the brackets represent the sliding window, e.g. W_1 and W_2. From the sliding windows, it is evident that there must be two consecutive quick-filtering checkpoints in any sliding window. Building upon this, we present Lemma 2.

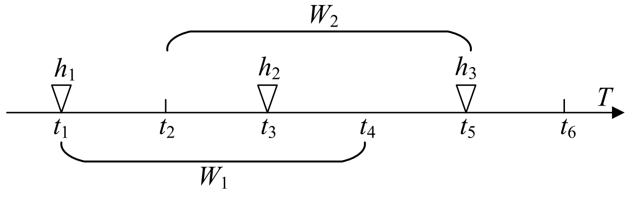


Fig. 3: Example of quick-filtering checkpoint

Lemma 2. Given a trajectory tra_i of moving object o_i, a contact source tra_s of contact source moving object s, the contact distance threshold d and a quick-filtering checkpoint set $H = \{h_1, h_2, \cdots, h_n\}$, for any two consecutive quick-filtering checkpoints h_k and h_{k+1} in the set, we have

$$dist(p_k^i, p_k^s) > d \vee dist(p_{k+1}^i, p_{k+1}^s) > d \Rightarrow \nexists e = < s, o_i, W_l > (h_k \vee h_{k+1} \in W_l), \tag{11}$$

where $dist(\cdot)$ is the distance between two position point and e represents a contact event occurring between s and o_i in sliding window W_l.

Proof. According to Definition 11, we find that each sliding window must contain two consecutive points. At this point, if $dist(p_k^i, p_k^s) > d \vee dist(p_{k+1}^i, p_{k+1}^s) > d$ holds, it means that the distance between o_i and s at these two points is greater than d in the sliding window containing them. As a result, Condition 1 for judging contact event in Definition 1 does not hold, i.e., there will be no contact event occurring in the sliding window containing these two points. ■

6.2 Design of EPCQ+

We adopt the above optimization to propose an enhanced version of EPCQ which is named EPCQ+. Specifically, we only need to improve the algorithm *ContactQuery* of EPCQ. The enhanced algorithm is presented as follows.

Algorithm 4: $ContactQuery(\widetilde{Q}, \widetilde{Tra})$

```
Input: the trapdoor $\widetilde{Q} = (\widetilde{VS}, [t_u, t_v])$, the encrypted trajectory vectors $\widetilde{Tra}$
Output: the contact query result $R$ storing contact events caused by the
        contact source object $s$ in the time period $[t_u, t_v]$
1  $R = \varnothing$ ;
2  Select the quick-filtering checkpoint point set $H_s = \{h_{s_1}, h_{s_2}, \cdots, h_{s_n}\}$ to $\widetilde{VS}$,
     where $h_{s_k} = t_k, t_k \in [t_u, t_v]$;
3  for each $\widetilde{VT}_i \in \widetilde{Tra}$ do
4      Select the quick-filtering checkpoint point set $H_i = \{h_{i_1}, h_{i_2}, \cdots, h_{i_n}\}$,
         where $h_{i_k} = t_k, t_k \in [t_u, t_v]$;
5      for each $h_{i_k}, h_{i_{k+1}} \in H_i$ do
6          if $h_{i_k} \cdot h_{s_k} = 0$ and $h_{i_{k+1}} \cdot h_{s_{k+1}} = 0$ then
7              Get the sliding window set $WS = \{W_l | h_{i_k}, h_{i_{k+1}} \in W_l\}$;
8              for each $(V_{p_j}, t_j) \in \widetilde{VT}_i, t_j \in W_l$ do
9                  $count = 0$;
10                 Get $V_{p_j}$ from $\widetilde{VT}_i$, which is the position vector of $VT_i$ at $t_j$;
11                 Get $V_{s_j}$ from $\widetilde{VS}$, which is the position vector of $VS$ at $t_j$;
12                 if $V_{p_q} \cdot V_{s_q} = 0$ then
13                     $count = count + 1$;
14                 if $count = \tau$ then
15                     $R = R + \{< s, o_i, W_p >\}$;
16 return $R$;
```

R $\leftarrow$ ***ContactQuery***$(\widetilde{Q}, \widetilde{Tra})$**.** On the basis of the original *ContactQuery*, when CS receives the trapdoor $\widetilde{Q} = (\widetilde{VS}, [t_u, t_v])$, it will select the quick-filtering checkpoint set H from the query period $[t_u, t_v]$. Then CS checks quick-filtering checkpoints in H according to Lemma 2 to identify the sliding windows that need further inspection for finding contact events. At last, CS returns the query result R to DU. The details of the improved *ContactQuery* are shown in Algorithm 4.

By using quick-filtering checkpoint in contact query processing, we can pre-filter time points where no contact events have occurred, reducing the number of sliding windows that require checking and thereby improving query efficiency.

7 Performance Evaluation

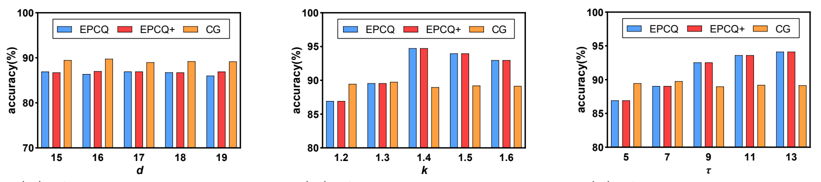
In this section, we perform a comprehensive evaluation for EPCQ and EPCQ+ and compare them with CG scheme [15]. Since the definition of contact in CG differs from ours, we modify it to align with our definition. All the evaluations are realized by using Python in Win11 with Intel Core i7-12700H CPU. The evaluation is based on two metrics, query accuracy and query time cost. The dataset used in the evaluation is Taxi [25]. The default setting of the contact distance threshold d, the factor k, and the sliding window width τ are $d = 19$, $k = 1.2$, and τ, respectively.

7.1 Query Accuracy Evaluation

In the experiments, the following formula is used to calculate the query accuracy:

$$Accuracy = 1 - \frac{|R_{\text{mis}}|}{|R_{\text{real}}|} \times 100\% \tag{12}$$

where R_{mis} is the missing results set, and R_{real} is the real result set. The evaluation results are given in Fig.x. It is noticeable that in CG, since vectorization is not involved, accuracy and time cost are not influenced by different parameters, so the following analysis will focus on EPCQ and EPCQ+.



(a) Accuracy versus d (b) Accuracy versus k (c) Accuracy versus τ

Fig. 4: Query accuracy under different parameters

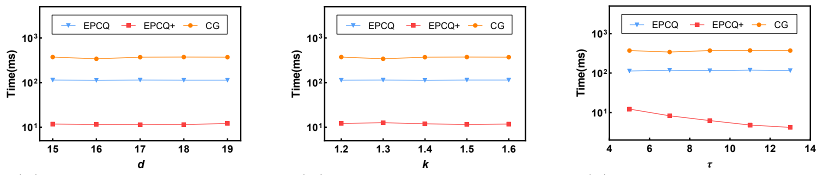
Fig.4a shows that as the increase of contact distance threshold d, the accuracy of EPCQ and EPCQ+ remains stable. The reason for this is that the increase of d only affects the spatial grid, resulting in different position vectors and contact source position vectors, but it does not affect the results of the query using the vector inner product. Therefore, the accuracy of the queries is not affected.

Fig.4b shows that EPCQ and EPCQ+ tend to increase and then decrease with the increase of k. The reason for this trend is that k is used to adjust the grid size. When k is small, it can mitigate the issue of missing contacts caused by judging distance by grid range. However, as k increases, the grid range becomes too large, resulting in misjudgments and a decrease in overall accuracy after an initial increase.

Fig.4c shows that EPCQ and EPCQ+ tend to increase with the increase of τ. The reason for this is that larger τ impose stricter criteria for judging contact. This results in fewer contact events being queried, reducing the likelihood of missing and causing the accuracy to appear to trend upwards.

7.2 Query Time Cost Evaluation

In this subsection, the query time is evaluated under different parameters. The results are shown in Fig.5.



(a) Time cost versus d (b) Time cost versus k (c) Time cost versus τ

Fig. 5: Query time cost under different parameters

Fig.5a and Fig.5b show that the time cost of EPCQ and EPCQ+ remains stable under different d and k. The reason for this is that the number of positions in the trajectory does not change under different d and k, i.e., the number of vector inner products required does not change, and hence the time cost remains stable.

Fig. 5c shows that the time cost of EPCQ remains stable under different τ, but the time cost of EPCQ+ tends to decrease with the increase of τ. The reason for this is that in EPCQ+, as τ increases, the distance between quick-filtering checkpoints becomes further apart. Consequently, fewer quick-filtering checkpoints need to be checked, which allows for the quick location of sliding windows that may contain contact events, thereby improving the time cost of the query.

The experimental results in Figs. 5(a), (b), and (c) all show that the time cost of EPCQ+ is much less than the other two. This is because in EPCQ and CG, both have to traverse all the positions to perform the vector inner product, whereas in EPCQ+, it is only necessary to check the quick-filtering checkpoints and then perform the vector inner product for a portion of the positions. This greatly reduces the computational overhead and improves the time cost of the query.

8 Conclusion

In this paper, we design a trajectory vectorization method that supports secure distance comparison in contact determination. Based on this, the baseline privacy-preserving contact query processing scheme (EPCQ) is proposed. To improve the efficiency of contact query, the quick-filtering checkpoint optimization strategy is presented. By adopting it, the enhanced privacy-preserving contact query processing scheme (EPCQ+) is proposed. Experimental results show that our schemes perform well in query accuracy and efficiency.

References

1. Alam, T.: Cloud computing and its role in the information technology. IAIC Transactions on Sustainable Digital Innovation (ITSDI) **1**(2), 108–115 (2020)
2. Alarabi, L., Basalamah, S., Hendawi, A., Abdalla, M.: Traceall: A real-time processing for contact tracing using indoor trajectories. Information **12**(5), 202 (2021)
3. Banasiak, J., et al.: Application protego stop covid-could it have been an opportunity to prevent covid-19 spreading? Political Preferences (29), 47–59 (2021)
4. Bell, J., Butler, D., Hicks, C., Crowcroft, J.: Tracesecure: Towards privacy preserving contact tracing. arXiv preprint arXiv:2004.04059 (2020)
5. Carmela, T.: Decentralized privacy-preserving proximity tracing. IEEE Data (Base) Engineering Bulletin **43** (2020)
6. Chao, P., He, D., Li, L., Zhang, M., Zhou, X.: Efficient trajectory contact query processing. In: Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I 26. pp. 658–666. Springer (2021)
7. Cho, H., Ippolito, D., Yu, Y.W.: Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. arXiv preprint arXiv:2003.11511 (2020)
8. Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., Fraser, C.: Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. science **368**(6491), eabb6936 (2020)
9. Fitzsimons, J.K., Mantri, A., Pisarczyk, R., Rainforth, T., Zhao, Z.: A note on blind contact tracing at scale with applications to the covid-19 pandemic. In: Proceedings of the 15th International Conference on Availability, Reliability and Security. pp. 1–6 (2020)
10. Gvili, Y.: Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc. Cryptology ePrint Archive (2020)
11. Hisada, S., Murayama, T., Tsubouchi, K., Fujita, S., Yada, S., Wakamiya, S., Aramaki, E.: Surveillance of early stage covid-19 clusters using search query logs and mobile device-based location information. Scientific Reports **10**(1), 18680 (2020)
12. Kato, F., Cao, Y., Yoshikawa, M.: Pct-tee: trajectory-based private contact tracing system with trusted execution environment. ACM Transactions on Spatial Algorithms and Systems (TSAS) **8**(2), 1–35 (2021)
13. Kermanshahi, S.K., Sun, S.F., Liu, J.K., Steinfeld, R., Nepal, S., Lau, W.F., Au, M.H.A.: Geometric range search on encrypted data with forward/backward security. IEEE Transactions on Dependable and Secure Computing **19**(1), 698–716 (2020)
14. Lai, S.H.S., Tang, C.Q.Y., Kurup, A., Thevendran, G.: The experience of contact tracing in singapore in the control of covid-19: highlighting the use of digital technology. International orthopaedics **45**, 65–69 (2021)
15. Li, M., Zeng, Y., Zheng, L., Chen, L., Li, Q.: Accurate and efficient trajectory-based contact tracing with secure computation and geo-indistinguishability. In: International Conference on Database Systems for Advanced Applications. pp. 300–316. Springer (2023)

16. Li, Z., Zuo, J., Song, M., Wei, Z., Zhang, Y., Xie, Y.: Query and clustering of spatio-temporal trajectory big data under the background of covid-19. In: Proceedings of the 2021 1st International Conference on Control and Intelligent Robotics. pp. 676–680 (2021)
17. Liu, Y., Dai, H., Li, B., Li, J., Yang, G., Wang, J.: Ecma: An efficient convoy mining algorithm for moving objects. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 1089–1098 (2021)
18. Michael, K., Abbas, R.: Behind covid-19 contact trace apps: The google–apple partnership. IEEE Consumer electronics magazine **9**(5), 71–76 (2020)
19. Orakzai, F.M., Calders, T., Pedersen, T.B.: k/2-hop: fast mining of convoy patterns with effective pruning. Proceedings of the VLDB Endowment **12**(9), 948–960 (2019)
20. Reichert, L., Brack, S., Scheuermann, B.: Privacy-preserving contact tracing of covid-19 patients. Cryptology ePrint Archive (2020)
21. Samuel, B.: Decentralized contact tracing using a dht and blind signatures. IACR Cryptol. Eprint Arch. **2020**, 398 (2020)
22. Sanches, D.E., Alvares, L.O., Bogorny, V., Vieira, M.R., Kaster, D.S.: A top-down algorithm with free distance parameter for mining top-k flock patterns. In: Geospatial Technologies for All: Selected Papers of the 21st AGILE Conference on Geographic Information Science 21. pp. 233–249. Springer (2018)
23. Sun, P.: Security and privacy protection in cloud computing: Discussions and challenges. Journal of Network and Computer Applications **160**, 102642 (2020)
24. Tabrizchi, H., Kuchaki Rafsanjani, M.: A survey on security challenges in cloud computing: issues, threats, and solutions. The journal of supercomputing **76**(12), 9493–9532 (2020)
25. Tang, L.A., Zheng, Y., Yuan, J., Han, J., Leung, A., Peng, W.C., Porta, T.L.: A framework of traveling companion discovery on trajectory data streams. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(1), 1–34 (2014)
26. Tang, Q.: Privacy-preserving contact tracing: current solutions and open questions. arXiv preprint arXiv:2004.06818 (2020)
27. Trieu, N., Shehata, K., Saxena, P., Shokri, R., Song, D.: Epione: Lightweight contact tracing with strong privacy. arXiv preprint arXiv:2004.13293 (2020)
28. Xiong, L., Shahabi, C., Da, Y., Ahuja, R., Hertzberg, V., Waller, L., Jiang, X., Franklin, A.: React: Real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking. SIGSPATIAL Special **12**(2), 3–14 (2020)
29. Xu, J., Lu, H., Bao, Z.: Imo: A toolbox for simulating and querying" infected" moving objects. Proceedings of the VLDB Endowment **13**(12), 2825–2828 (2020)