To Find Your Location: Secure and Practical Point Location Scheme

Jingjing Guo (jennifer.jing.sun@gmail.com)
Xidian University

Jiacong Sun
Peking University Shenzhen Graduate School

Research

Keywords: Location-based Service, Point Location, Data Privacy, Query Privacy

Posted Date: October 29th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-93394/v1

RESEARCH

To Find Your Location: Secure and Practical Point Location Scheme

Jingjing Guo[1*] and Jiacong Sun[2]

*Correspondence:
jennifer.jing.sun@gmail.com
[1]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
Full list of author information is available at the end of the article

Abstract

With the ubiquitous mobile devices and the advanced wireless communication, location-based service (LBS) helps people to enjoy a convenient lifestyle and has attracted numerous research interests. As a basic query process in LBS system, point location requires to find a region containing the query point. Since location belongs to sensitive information and also leads other private information leaked, it is urgent to design a secure and efficient point location scheme. In this paper, we propose a point location scheme named SecPL to protect sensitive information while supporting high efficient location query. Specifically, we introduce a LineTest scheme from asymmetric scalar-product-preserving encryption (ASPE) to facilitate the checking of whether a point lies above or below a line. Furthermore, the SecPL scheme is designed by using LineTest and order-preserving encryption (OPE) scheme. Through detailed security analysis, we demonstrate that SecPL scheme achieves data privacy and query privacy at the same time. Finally, the performance evaluation demonstrates the high efficiency of the proposed SecPL scheme.

Keywords: Location-based Service; Point Location; Data Privacy; Query Privacy

1 Introduction

According to the report on smartphone users worldwide from website Statista, the number of users today have surpassed three billion and will further grow by several hundred million in the next few years. With the ubiquity of mobile devices and wireless communication, location-based service (LBS) enables people to enjoy a convenient life by providing basic services, such as transportation navigation, nearby interest point inquiry, online dating and advertising push [1, 2, 3, 4]. Users only submits a LBS query and the LBS provider can list all the relevant information according to the query, most important of which is point location.

Point location requires to find a region containing the query point and is very useful in many settings. For instance, when a sailor sails on an unfamiliar sea with rocks and sand banks, he wants to know the region where he is. Furthermore, he retrieves the information (e.g., the weather, the geographical setting) about the region through a LBS system. If everyone uses this sort of navigation, LBS provider will face heavy computation tasks. To achieve a low computation and communication burden, LBS provider chooses to outsource his storage and computation tasks to a cloud server [5, 6]. However, location information belongs to sensitive information and further will be revealed by the cloud server.

As the simplest way, masking identity has been introduced to preserve the sensitive information in access authentication [7]. But, the real identity is easily revealed in

LBS system through frequent queries [8]. To enhance the security, Zhang et al. [9] propose a privacy-preserving LBS scheme from k-anonymity, which ensures the probability of identity an user is at most $1/k$. Scheme [9] meanwhile needs a trusted third party (TTP) to store original LBS data and further hides the result with other $k-1$ garble locations. Cloaking technique [10, 11] as an alternative way has been illustrated recently. Unfortunately, above schemes based on k-anonymity approach and cloaking approach bring heavy communication cost for mobile devices.

Moving a step forward, Paulet et al. [12] propose a privacy-preserving location-based service scheme by leveraging private information retrieval technique (PIR). In scheme [12], LBS users can obtain results from a remote database without revealing which record they are interested. Moreover, all information are performed in plaintexts by the cloud server contradicting with the privacy requirements in cloud computing. Shao et al. [1] introduce a fine-grained privacy-preserving location-based service scheme called FINE. The FINE provides accurate results and acceptable computation cost for the mobile services, but it also brings unacceptable computation cost for the cloud server. Zhu et al. [2] design a polygon range by using an improved homomorphic encryption scheme [13]. However, scheme [2] returns all related information in a specified distance e.g., 1 km. In this condition, it brings heavy communication and decryption cost for mobile devices.

All above schemes are performed after knowing location. Unfortunately, location involving sensitive information should be preserved. To the best of our knowledge, there is no privacy-preserving point location helping LBS users find a region containing their location. Therefore, we aim to design a secure and efficient point location scheme to enable LBS users to know their locations, namely finding a region Δ containing the query point q.

1.1 Our Contributions

To achieve the target of privacy-preserving point location query in LBS system, we first introduce a basic tool and combine order preserving encryption technique to construct a secure and practical point location scheme, named SecPL. The main contributions are introduced as follows.

- We propose a basic tool called LineTest to check whether a query point lies below or above a line. The LineTest scheme is secure under knwon-plaintext attacks in the presence of an adversary $\mathcal{A}$.
- We design a secure and efficient point location scheme over encrypted database in LBS system, and formally introduce and prove its security with indistinguishability under order-ciphertext-only attacks (IND-OCOA).
- To illustrate the efficiency of SecPL scheme, we leverage Python language to implement our SecPL. The results show the high efficiency on processing point location query in LBS system.

1.2 Related Work

Location-based service enables people to enjoy a convenient life that has attracted numerous interests in recent years. With the huge storage and computation burden, a LBS provider chooses to outsource database under encryption. In this section, we illustrate some works that are closely related to privacy-preserving location-based service.

Masking identity as the simplest approach has been proposed to preserve the identity in access authentication [7]. However, real identity is easily leaked by the location information from frequent queries. To enhance the security, some works [9, 14, 15, 16, 17] from k-anonymity have been introduced. k-anonymity requires an adversary can recognize a real identity with an advantage at most $1/k$. Scheme [14] constructs a location k-anonymity by leveraging $k-1$ other mobile devices. It also requires a trusted platform to store locations and obtains the subject that an advantage for identifying a user is very low. Moving a step forward, Khoshgozaran et al. [18] introduce a k-anonymity approach to construct a location-based service scheme without a trusted third party. Different from the above schemes based on k-anonymity, cloaking technique is introduced to design privacy assurance LBS schemes [11, 19, 20]. The cloaking technique blurs the location into a cloaked region, e.g., polygon[11] and rectangular [19]. Unfortunately, above schemes from cloaking technique return a number of candidate results instead of accurate and then bring a heavy burden for communication. Paulet et al. [12] demonstrate a LBS scheme from private information retrieval technique to protect sensitive information. However, all the information are performed over plaintexts in scheme [12], which do not apply to the outsource scenario.

Shao et al. [1] illustrate a fine-grained privacy assurance LBS scheme named FINE, which provides rectangle LBS queries. Although it brings accurate results, FINE brings unacceptable computation for the cloud server. Zhu et al. design privacy-preserving circle range LBS query [21] and polygon range LBS query [2] based on enhanced homomorphic encryption. However, both two schemes bring heavy computation and communication burden for the mobile devices. To enhance the efficiency, Zeng et al. [22] design a circle range LBS query from asymmetric scalar-product preserving encryption scheme and the circle range query scheme achieves a high efficiency. However, all above schemes limit to special scenario, circle range query [21, 22] and polygon range query [2].

Wang et al. [23] illustrate a general solution to design geometric range query. In scheme [23], Wang et al. propose a symmetric-key searchable encryption scheme from Bloom filter [24] and symmetric-key probabilistic functional encryption (SSW scheme) techniques [25]. As an improvement, Wang et al. further design an efficient geometric range query scheme, which is referred as FastGeo [26]. FastGeo is interpreted as two level search, where the first level depends on equality checking solved by pseudo-random function and the second level relies on the checking of whether inner products are zeros. The second level can be handled from SSW technique. Unfortunately, both schemes require to enumerate all possible points in the geometric range query, which makes it not quite suitable in LBS system.

1.3 Organization

The rest of this paper is illustrated as follows. We present some preliminaries in section 2, and illustrate the system model and the formal definition of SecPL in section 3. Then, in section 4, we design a secure LineTest scheme to check whether a point lies above or below a line and prove the security of LineTest. The proposed SecPL scheme is illustrated in section 5 and the security analysis is presented in section 6. In section 7, we demonstrate a performance evaluation of the proposed LineTest and our SecPL scheme. Finally, conclusions are described in section 8.

2 Preliminaries

In this section, we introduce some preliminaries for order-preserving encryption approach and randomized incremental algorithm of point location.

2.1 Order-Preserving Encryption

Order-preserving encryption [27, 28] is a special type of encryption scheme where the order of ciphertexts retains that of plaintexts, namely, if $m_1 < m_2$, then $[m_1] < [m_2]$. It facilitates order comparison on encrypted database in cloud computing without revealing plaintexts. The ideal security requires ciphertexts reveals nothing besides order information and has been achieved by [29, 30].

An OPE scheme consists of three polynomial-time algorithms KeyGen, Enc, and Dec. The algorithm is defined as follows.

- KeyGen(λ)$\rightarrow$ sk: On input a security parameter λ, it outputs a secret key sk;
- Enc(sk, m)$\rightarrow$ $[m]$: On input a secret key sk and a plaintext m, it outputs a ciphertext $[m]$;
- Dec($sk, [m]$)$\rightarrow$ m: On input a secret key sk and a ciphertext $[m]$, it outputs the corresponding plaintext m.

2.2 Randomized Incremental Algorithm for Point Location

A randomized incremental algorithm presents a data structure $\mathcal{T}(L)$ which is used to carry out a point location query. The randomized incremental algorithm has a $O(n \log n)$ expected construction time, a $O(n)$ expected storage, and $O(\log n)$ expected time for point location, where n is the number of line segments in L. Since one-time construction of data structure, the construction time is acceptable.

The data structure $\mathcal{T}(L)$ has two types of inner nodes and one type of leaf node for every trapezoid of $\mathcal{T}(L)$. The two types of inner nodes are $x - node$ and $y - node$. The $x - node$ stores an endpoint of line segment in L which indicates a query point q lies to the left or to the right of the vertical line through this endpoint. The $y - node$ stores a line segment which indicates a query point q lies above or below the segment. We introduce details of the randomized incremental algorithm for solving location queries as follows.

Tree Construction. According to a set of planar subdivision segments $L = \{l_1, l_2, \cdots, l_n\}$, a LBS provider constructs a data structure $\mathcal{T}(L)$ for it. First, the LBS provider computes a bounding rectangle R which includes the set of line segments and he initializes an incremental tree structure $\mathcal{T}$ for it. For $i = 1$ to n, the LBS provider searches data structure $\mathcal{T}$ to find leaf nodes Δ_0, Δ_1, $\cdots$, Δ_k intersected with l_i. Finally, the LBS provider removes Δ_0, Δ_1, $\cdots$, Δ_k and replaces them with new trapezoids which appears because of new segments l_i.

An instance of tree construction is introduced in Figure 1. First, it takes segment l_1 with endpoints p_1, q_1 to construct data structure as described in Figure 1(a). Subsequently, when inserting segment l_2, line segment l_2 intersects with regions A, C, D in Figure 1(a). Then, remove original regions A, C, D and replace them with new regions A, E, C, F, G, D. The data structure after inserting line segment l_2 is illustrated in Figure 1(b).

Search algorithm: For a query point q, the search algorithm traverses the data structure $\mathcal{T}(S)$ to find the leaf node which contains q. From root to leaf node, if

non-leaf node Δ is a x-node, the LBS provider checks whether query point q lies to the left or to the right of the vertical line through endpoint. That is, check the order relation of horizontal axis for x-node and query point q. If non-leaf node Δ is a y-node, the LBS provider checks whether query point q lies above or below the line segment l. After checking, the LBS provider updates the new child as Δ until leaf nodes. An instance for search algorithm is shown in Fig. 1(b).

Figure 1 An architecture of randomized incremental algorithm.(a)The insertion of line l_1. (b) The insertion of line l_2.

3 Overview

In this section, we illustrate the system model and the formal definition of our scheme in brief.

3.1 System Model

The system model of the proposed scheme consists of a LBS provider, LBS users and a cloud server. The LBS provider is a resource-constraint entity (e.g., a company) who wants to outsource the heavy storage and computation tasks. The LBS users are an entities (e.g., a user of the company) who submit a search request and obtain results from the cloud server. The cloud server is an entity who provides powerful storage and search resources. In this paper, we focus on the problem of point location for the plana subdivision. It means that the LBS provider outsources a tree structure for the plana subdivision to the cloud server. The purpose of point location is to search a polygon containing a location $q = (x, y)$.

We assume that the cloud server is honest-but-curious. It means that the cloud server follows protocol and returns results faithfully. But, the cloud server intends to learn sensitive information about the outsourced database and the search request. To protect the private data, the LBS provider only outsources an encrypted database and LBS users perform an encrypted search request. The architecture of the system model is illustrated in Fig. 2.

Figure 2 The architecture of system model.

3.2 Formal Definition

A secure point location query scheme is a tuple of polynomial-time algorithms Π = {**KeyGen, TreeBuild, Encrypt, Search** } defined as follows:

- **KeyGen** $(\lambda) \rightarrow sk$: On input a security parameter λ, this algorithm outputs a secret key sk.
- **TreeBuild** $(L) \rightarrow \mathcal{T}(L)$: On input a graph data $L = \{l_1, l_2, \cdots, l_n\}$, this algorithm constructs an incremental tree $\mathcal{T}(L) = \{p_1, p_2, \cdots, p_{2n}, l_1, l_2, \cdots, l_n, \Delta_1, \cdots, \Delta_k, P\}$, where p_i is the endpoint of line segment, Δ_i is the region of plana subdivision and P is a set of pointers to cover the parent-child relations.

- **Encrypt** $(sk, \mathcal{T}(L)) \rightarrow \mathcal{T}^*(L)$: On input the secret key sk and incremental tree $\mathcal{T}(L)$, it encrypts the incremental tree as $\mathcal{T}^*(L) = \{p_1^*, p_2^*, \cdots, p_{2n}^*, l_1^*, l_2^*, \cdots, l_n^*, \Delta_1^*, \cdots, \Delta_k^*, P\}$.
- **Search** $(sk, \mathcal{T}^*(L), q) \rightarrow \Delta_i$: On input the security key sk, encrypted incremental tree $\mathcal{T}^*(L)$ and a query point q, this algorithm outputs the region Δ_i containing q.

In the following, we illustrate the security requirements for the proposed SecPL scheme in cloud computing.

- **Data privacy.** The data tuple in the outsourced database should keep secret from an adversary $\mathcal{A}$. General speaking, the adversary $\mathcal{A}$ could not obtain any information besides the order about the outsourced database.
- **Query Privacy.** The point location query should keep secret from the adversary $\mathcal{A}$. That is, a location query may contain some sensitive information about the data user, thus the actual information should keep secret even though all queries have been recorded.

4 Basic Tools

Before illustrating the proposed SecPL scheme, we present a basic tool, which is referred as LineTest. In the following, we describe the proposed LineTest scheme and then illustrate the security analysis.

4.1 The LineTest Scheme

The tool enables us to test a query point $q = (x, y)$ lies above or below a line $l = (a, b, c)$ in privacy. Specifically, it gives us the power to check whether $ax + by + c > 0$. In the following, we demonstrates the tool in five polynomial-time algorithms, KeyGen, Enc, TokenGen, Test, Dec.

- KeyGen$(\lambda) \rightarrow sk$: Takes as input a security parameter λ, it outputs a $m - bit$ string S, two $m \times m$ invertible matrix M_1, M_2 and $k = m - d$ random numbers $w_1, w_2, \cdots, w_k$. The secret key is

$$sk = (S, M_1, M_2, w_1, w_2, \cdots, w_k).$$

- Enc$(sk, l) \rightarrow C$: Takes as input a secret key sk and a line $l = (a, b, c)^T$, it encrypts the line into a ciphertext in the following processes.
  1 Adding dimensions. Extend a 3-dimensional data tuple $l = (a, b, c)^T$ to a $m-$dimensional data l' as

$$l' = (a, b, c, w_1, w_2, \cdots, w_k)^T.$$

  2 Random splitting. Split the $m-$dimensional data l' into two parts l'_1 and l'_2. For $i = 1$ to m, if $S_i = 1$, then $l'_1[i]$ and $l'_2[i]$ are random numbers such that $l'_1[i] + l'_2[i] = l'[i]$; otherwise, $l'_1[i] = l'_2[i] = l'[i]$.
  3 Data encryption. Encrypt the data as $\bar{l}_1 = M_1^T l'_1$ and $\bar{l}_2 = M_2^T l'_2$. The ciphertext is

$$C = (\bar{l}_1, \bar{l}_2) = (M_1^T l'_1, M_2^T l'_2).$$

- TokenGen(sk, q)$\rightarrow TK$: Takes as input a secret key sk and a query point q, it computes the search token for the query point. The details are presented as follows.
  1 Adding dimensions. Extend a 2-dimensional query data $q = (x, y)^T$ into a m-dimensional data q' as

$$q' = (rx, ry, r, t_1, t_2, \cdots, t_k)^T,$$

  where r, t_1, t_2, $\cdots$, t_{k-1} are random numbers and $t_k = \frac{-\sum_{i=1}^{k-1} w_i t_i}{w_k}$.
  2 Random splitting. Split q' into two random parts q'_1 and q'_2. For $i = 1$ to m, if $S_i = 0$, $q'_1[i]$ and $q'_2[i]$ are chosen randomly satisfying $q'_1[i] + q'_2[i] = q'[i]$; otherwise, $q'_1[i] = q'_2[i] = q'[i]$.
  3 Data Encryption. Compute the search token in ciphertext as $tk = (\bar{q}_1, \bar{q}_2)$, where $\bar{q}_1 = M_1^{-1} q'_1$, and $\bar{q}_2 = M_2^{-1} q'_2$.
- Test(C, TK)$\rightarrow$ ord: Takes as input a ciphertext $C = (\bar{l}_1, \bar{l}_2)$ of line l and search token $tk = (\bar{q}_1, \bar{q}_2)$ for query point q, it tests whether

$$\bar{l}_1 \cdot \bar{q}_1 + \bar{l}_2 \cdot \bar{q}_2 > 0.$$

  If it holds, $ord = 1$ which means the query point lies above the line l; otherwise, $ord = 0$ which means the query point lies on or below the line l.
- Dec(C, sk)$\rightarrow$ l: Takes as input a ciphertext $C = (\bar{l}_1, \bar{l}_2)$ and a secret key sk, it performs the following processes to decrypt.
  1 Compute

$$l_1 = \pi_3 M_1^{T^{-1}} \bar{l}_1, \quad l_2 = \pi_3 M_2^{T^{-1}} \bar{l}_2,$$

  where $\pi_3 = (I_3, 0)$ and I_3 is a 3×3 identity matrix.
  2 For $i = 1$ to 3, if $S_i = 1$, $l[i] = l_1[i] + l_2[i]$; otherwise, $l[i] = l_1[i] = l_2[i]$.

4.2 Security Analysis

Theorem 4.1 *The proposed LineTest scheme is correct.*

Proof We prove the correctness of LineTest as follows.

$$\begin{aligned}
&\bar{l}_1 \cdot \bar{q}_1 + \bar{l}_2 \cdot \bar{q}_2 \\
&= (M_1^T l'_1) \cdot (M_1^{-1} q'_1) + (M_2^T l'_2) \cdot (M_2^{-1} q'_2) \\
&= (M_1^T l'_1)^T \times (M_1^{-1} q'_1) + (M_2^T l'_2)^T \times (M_2^{-1} q'_2) \\
&= (l'_1)^T M_1 \times M_1^{-1} q'_1 + (l'_2)^T M_2 \times M_2^{-1} q'_2 \\
&= (l'_1)^T \times q'_1 + (l'_2)^T \times q'_2 \\
&= l' \cdot q' \\
&= ax + by + c
\end{aligned}$$

From the equation $\bar{l}_1 \cdot \bar{q}_1 + \bar{l}_2 \cdot \bar{q}_2 > 0$, we have the conclusion $ax + by + c > 0$, which means that the query point $q = (x, y)$ lies above the line $l = (a, b, c)$. Otherwise, the query point $q = (x, y)$ lies on or below the line $l = (a, b, c)$. □

Theorem 4.2 *The proposed LineTest scheme is secure under the known-plaintext attack (KPA) in the presence of an adversary.*

Proof The security proof contains two aspects, data privacy and query privacy. First, we introduce the data privacy under known-plaintext attack as follows.

In known-plaintext attack (KPA), suppose that an adversary $\mathcal{A}$ owns the knowledge $\{E(DB), P, C = E(P)\}$, namely $\mathcal{A}$ knows the encrypted database $E(DB)$, a set of plaintexts P and its corresponding ciphertexts $C = E(P)$. The KPA security requires that the adversary $\mathcal{A}$ could not recover a set plaintexts $DB_{\mathcal{A}}$ from the known information.

For a line segment $l_i \in P$, the security definition requires an adversary $\mathcal{A}$ knows its corresponding ciphertext $(\bar{l}_{1i}, \bar{l}_{2i})$. In the following, we proof the LineTest scheme is KPA secure if the adversary $\mathcal{A}$ connot derive the splitting secret key S. We further suppose that $m = 4$, without considering adding dimensions. Note that adding dimensions will increase the security of LineTest scheme. If an adversary $\mathcal{A}$ doesnot know the secret key S, for a segment $l_i \in P$, he has to model l'_{1i} and l'_{2i} as random numbers. Next, he tries to recover two 4×4 invertible matrices M_1 and M_2 from equation $M_1^T l'_{1i} = \bar{l}_{1i}$ and $M_2^T l'_{2i} = \bar{l}_{2i}$. There are $8|P|$ unknowns in l'_{1i} and l'_{2i} and 32 unknowns in M_1, M_2 and only $8|P|$ equations. Therefore, the adversary $\mathcal{A}$ could not solve matrices M_1 and M_2.

Furthermore, the query privacy under known-plaintext attack can be proved in a similar way as above. In conclusion, the LineTest is secure under known-plaintext attack if a adversary $\mathcal{A}$ does not know the splitting secret key S. □

5 The Proposed SecPL Scheme

In this section, we first illustrate the methodology of the proposed SecPL scheme and then demonstrate the detailed description of point location in privacy.

5.1 High Description

The problem of point location is referred as an resource-constraint LBS provider outsources a data structure $\mathcal{T}(L)$ for a planar subdivision and later LBS users perform location queries to find a region containing their location q. Because of privacy requirements, the outsourced structure $\mathcal{T}(L)$ and the query request should be encrypted in ciphertexts while keeping search functionality. The data structure contains two types of inner nodes and one type of leaf node. The inner node consists of $x-$node and $y-$node two aspects. The leaf node is referred as a label for the region.

In this paper, we aim to adapt order preserving encryption approach to protect the privacy of $x - node$ and LineTest algorithm to assurance privacy of $y - node$. That is, for a $x - node$ $p_i = (x_i, y_i)$, we compute $[p_i] = [x_i]$, where $[x_i] = \text{OPE.Enc}(sk, x_i)$. For a $y - node$ $l = (a, b, c)$, we perform the LineTest algorithm $C = \text{LineTest.Enc}(sk, l)$. For a query point $q = (x, y)$, compute its ciphertexts from OPE and LineTest algorithm, respectively. Compute the ciphertext of query point $q = (x, y)$ as $TK = (TK_1, TK_2)$, where $TK_1 = \text{OPE.Enc}(sk, x)$ and $TK_2 = \text{LineTest.TokenGen}(sk, q)$. During search process, if inner node is a $x - node$, the cloud server performs the order comparison directly over ciphertexts

from OPE algorithm. Otherwise, the cloud server performs comparison operation from LineTest.Test. Therefore, the cloud server can search the leaf node which contains query point q.

5.2 The Main Construction

For sake of clarity, suppose that OPE=(KeyGen, Enc, Dec) is an order-preserving encryption scheme with IND-OCPA security, and LineTest = (KeyGen, Enc, TokenGen, Test, Dec) is a new asymmetric scalar-preserving encryption scheme with IND-KPA security. We describe the proposed SecPL scheme in four algorithms as follows.

- **KeyGen** $(\lambda) \rightarrow sk$: Takes as input a security parameter λ and then computes $sk_1 \leftarrow \text{OPE.KeyGen}(\lambda)$ and $(S, M_1, M_2, w_1, w_2, \cdots, w_k) \leftarrow \text{LineTest.KeyGen}(\lambda)$. Thus, we have the secret key

$$sk = (sk_1, S, M_1, M_2, w_1, w_2, \cdots, w_k).$$

- **TreeBuild** $(L) \rightarrow \mathcal{T}(L)$: Takes as input a graph database $L = \{l_1, l_2, \cdots, l_n\}$, this algorithm proceeds the randomized incremental algorithm and constructs an incremental tree $\mathcal{T}(L) = \{p_1, p_2, \cdots, p_{2n},\ l_1, l_2,\ \cdots, l_n, \Delta_1, \cdots, \Delta_k,\ P\}$, where p_i is an endpoint of a line segment, Δ_i is a region of plana subdivision and P is a set of pointers to cover the parent-child relations.
- **Encrypt** $(sk, \mathcal{T}(L)) \rightarrow \mathcal{T}^*(L)$: Takes as input a secret key sk and incremental tree $\mathcal{T}(S)$, it encrypts the incremental tree as $\mathcal{T}^*(L) = \{p_1^*, p_2^*, \cdots, p_{2n}^*, l_1^*, l_2^*, \cdots, l_n^*, \Delta_1^*, \cdots, \Delta_k^*, P\}$.
  - $x - node$ encryption. For a $x - node\ p_i = \{x_i, y_i\}$ in the incremental tree $\mathcal{T}(L)$, LBS provider encrypts it from the OPE scheme and computes its ciphertext as

$$[p_i] = [x_i] = \text{OPE.Enc}(sk_1, x_i), i = 1, 2, \cdots, 2n.$$

  - $y - node$ encryption. For a $y - node\ l_i$ in the incremental tree $\mathcal{T}(L)$, it can be represented as 3-dimensional tuple (a, b, c). Subsequently, the LBS provider computes a ciphertext as

$$C_i = \text{LineTest.Enc}(sk, l_i).$$

- **Search** $(sk, \mathcal{T}^*(S), q) \rightarrow \Delta_i$: Takes as input the secret key sk, encrypted tree $\mathcal{T}^*(S)$ and query point q, it outputs the polygon Δ_i containing q. The following processes should be performed.
  - Token generation. For a query point $q = (x, y)$, its ciphertext including two parts. Compute its OPE ciphertext as

$$TK_1 = [x] = \text{OPE.Enc}(sk_1, x).$$

    It performs the order comparison with $x - node$ in the tree structure. Furthermore, it computes the ciphertext C_2 to proceed the comparison

with $y - node$. For a query point $q = (x, y)$, LBS provider performs the LineTest encryption as

$$TK_2 = \text{LineTest.TokenGen}(sk, q).$$

Thus, the search token is

$$TK = (TK_1, TK_2) = ([x], (\bar{q}_1, \bar{q}_2)).$$

– Tree search. LBS users submit the search token tk to the cloud server. Afterwards, the cloud server proceeds the search algorithm in **Algorithm 1** to find the polygon Δ_i containing q. From root to leaf node, if Δ is a $x - node$, the cloud server check whether query point q lies to the left or to the right of the vertical line through endpoint p_i by checking the order of $[x] = \text{OPE.Enc}(sk_1, x)$ with $[x_i] = \text{OPE.Enc}(sk_1, x_i)$. If Δ is a $y - node$, the cloud server computes $\text{LineTest.Test}(C_i, TK)$ and updates the corresponding child node as a new node. In this condition, the cloud server finally finds the leaf node Δ containing query point q.

Algorithm 1 Search algorithm of SecPL scheme

Input:

- an encrypted query point q
- an encrypted incremental tree: $\mathcal{T}^*(L)$

Output:

- leaf node Δ containing query point q

1: Δ = root;
2: **while** Δ is not a leaf node **do**
3: **if** Δ is a $x - node$ p_i **then**
4: Perform order comparison between $[x] = \text{OPE.Enc}(sk_1, x)$ and $[x_i] = \text{OPE.Enc}(sk_1, x_i)$. Afterwards, the LBS provider obtains the result for whether query point q lies to the left or to the right of the vertical line through endpoint p_i;
5: **if** q lies to the left **then**
6: Update Δ to the left child of p_i;
7: **else**
8: Update Δ to the right child of p_i;
9: **end if**
10: **else** {Δ is a $y - node$ l_i}
11: Check whether q lies above of below the segment l_i through performing $\text{LineTest.Test}(C_i, TK)$;
12: **if** q lies above the segment l_i **then**
13: Update Δ to the left child of l_i;
14: **else**
15: Update Δ to the right child of l_i;
16: **end if**
17: **end if**
18: **end while**

6 Security Analysis

In this section, we demonstrate formal security definition of IND-OCOA and illustrate the security proof of our SecPL scheme. In the security proof, we prove that the proposed SecPL scheme achieves data privacy and query privacy.

6.1 Formal Security Definition

Definition 1 *(IND-OCOA Data Privacy.) We say that a scheme* Π *is secure against order ciphertext-only attack (OCOA) on data privacy, if for any polynomial-time adversaries* $\mathcal{A}$*, it has most negligible advantage in the following game,*

$$Adv_{\Pi,\mathcal{A}} = |Pr[b' = b] - \frac{1}{2}| < negl(\lambda).$$

Security game. The security game between a LBS provider and an adversary $\mathcal{A}$ proceeds as follows:

1 The LBS provider runs the KeyGen(λ) algorithm to generate a secret key sk and chooses a bit $b \leftarrow \{0, 1\}$;
2 The adversary $\mathcal{A}$ sends $x - node$ sequences $\{p_i^0\}_i$ and $\{p_i^1\}_i$, and $y - node$ sequences $\{l_j^0\}_j$ and $\{l_j^1\}_j$ to the LBS provider, where $\{p_i^0\}_i$ and $\{p_i^1\}_i$ have the same order relations;
3 The LBS provider computes the OPE ciphertext as $[p_i^b] = [x_i] = \text{OPE.Enc}(sk, x_i^b)$ and $C_j^b = \text{LineTest.Enc}(sk, l_j^b)$. Thus, the LBS provider has the ciphertexts $\{[p_i^b]\}_i$ and $\{C_j^b\}_j$, and returns back to the adversary $\mathcal{A}$;
4 Afterwards, adversary $\mathcal{A}$ outputs b', its guess for b.

Definition 2 *(IND-OCOA Data Privacy.) We say that a scheme* Π *is secure against order ciphertext-only attack (OCOA) on query privacy, if for any polynomial-time adversaries* $\mathcal{A}$*, it has most negligible advantage in the following game,*

$$Adv_{\Pi,\mathcal{A}} = |Pr[b' = b] - \frac{1}{2}| < negl(\lambda).$$

Security game. The security game between a LBS provider and an adversary $\mathcal{A}$ proceeds as follows:

1 The LBS provider runs the KeyGen(λ) algorithm to generate a secret key sk and chooses a bit $b \leftarrow \{0, 1\}$;
2 The adversary $\mathcal{A}$ sends sequences $\{q_i^0\}_i = \{x_{q_i}^0, y_{q_i}^0\}$ and $\{q_i^1\}_i = \{x_{q_i}^1, y_{q_i}^1\}$, where $\{x_{q_i}^0\}_i$ and $\{x_{q_i}^1\}_i$ have the same order relations;
3 The LBS provider computes the OPE ciphertext as $[q_i^b] = [x_{q_i}^b] = \text{OPE.Enc}(sk, x_{q_i}^b)$ and $TK_i^b = \text{LineTest.TokenGen}(sk, q_i^b)$. Thus, the LBS provider has the ciphertexts $\{[x_{q_i}^b]\}_i$ and $\{C_i^b\}_i$, and returns back to the adversary $\mathcal{A}$;
4 Afterwards, the adversary $\mathcal{A}$ outputs b', its guess for b.

6.2 Security Proof

In the following, we analyze the security of the proposed SecPL scheme from two aspects, data privacy and query privacy.

Theorem 6.1 *The proposed SecPL scheme achieves IND-OCOA secure on data privacy in the presence of an adversary* $\mathcal{A}$*.*

Proof For the convenience of description, we illustrate the concept of computationally indistinguishability. Suppose that $X = \{X_i\}_i$ and $Y = \{Y_i\}_i$ are two ensembles of distributions. X and Y are said to be computationally indistinguishability (written $X \overset{C}{\approx} Y$), if for all probabilistic polynomial time adversary $\mathcal{A}$, there is a negligible function *negl* such that

$$|Pr[\mathcal{A}(X_k) = 1] - Pr[\mathcal{A}(Y_k) = 1]| < negl(\lambda).$$

We will proof the theorem through hybrid methods.

In the security game on data privacy, the adversary $\mathcal{A}$ has the ciphertexts $\{[p_i^b]\}_i$ and $\{C_j^b\}_j$. Because the order preserving encryption OPE=(KeyGen, Enc, Dec) is ideal security (i.e., the ciphertexts reveals nothing about plaintexts besides order information), we conclude that $\{[p_i^0]\}_i$ and $\{[p_i^1]\}_i$ are computationally indistinguishability in above game, namely

$$\{[p_i^0]\}_i \overset{C}{\approx} \{[p_i^1]\}_i.$$

Thus, it is easy to draw the conclusion that

$$\{\{[p_i^0]\}_i, \{C_j^b\}_j\} \overset{C}{\approx} \{\{[p_i^1]\}_i, \{C_j^b\}_j\}.$$

Moving a step forward, since LineTest = {KeyGen, Enc, TokenGen, Test, Dec} is secure under known-plaintext attack proved in theorem 4.2, it also achieves IND-OCOA security (a weaker security definition). Thus, ciphertexts $\{C_j^0\}_j$ and $\{C_j^1\}_j$ are computationally indistinguishability, i.e.,

$$\{C_j^0\}_j \overset{C}{\approx} \{C_j^1\}_j.$$

In this condition, we further have

$$\{\{[p_i^0]\}_i, \{C_j^0\}_j\} \overset{C}{\approx} \{\{[p_i^0]\}_i, \{C_j^1\}_j\}$$
$$\{\{[p_i^0]\}_i, \{C_j^1\}_j\} \overset{C}{\approx} \{\{[p_i^1]\}_i, \{C_j^1\}_j\}.$$

Owing to the transitivity principle of computationally indistinguishability, we conclude that

$$\{\{[p_i^0]\}_i, \{C_j^0\}_j\} \overset{C}{\approx} \{\{[p_i^1]\}_i, \{C_j^1\}_j\}.$$

As a result, an adversary $\mathcal{A}$ con not distinguish ciphertexts $\{\{[p_i^0]\}_i, \{C_j^0\}_j\}$ and $\{\{[p_i^1]\}_i, \{C_j^1\}_j\}$ in IND-OCOA security game. We complete the proof of data privacy. □

In the following, we prove our SecPL scheme achieves query privacy under order-ciphertext-only attack.

Theorem 6.2 *The proposed SecPL scheme achieves IND-OCOA secure on query privacy in the presence of an adversary $\mathcal{A}$.*

Proof The proof of query privacy under IND-OCOA is extremely similar to that of data privacy as above. Thus, we skip the proof of query privacy owing to the limit space. □

7 Performance Evaluation

In this section, we illustrate an experiment simulation between LineTest scheme and the proposed SecPL scheme to test the practical utility. The proposed SecPL scheme consists of a LBS provider, LBS users and a cloud server. In the following experiment, we simulate the three entities on the same machine. More specifically, the code is run with Python3 language on a machine with Intel Xeon(R) i7-8565U processor running at 16GHz and 1 T memory.

7.1 Performance of LineTest

The aim of LineTest is to check whether a query point lies above or below a line segment. In experiment simulation, we perform experiments on a synthetic database with 1 million data items and precalculate the invertible matrices of secret keys M_1 and M_2 to save the time cost in query token generation and decryption. We vary the size of synthetic database from 10,000 to 100,000 and vary expanded dimension m from 10 to 100. The length $m = 80$ is comparable to the security level of a 1024-bit RSA. The performance of LineTest is simulated in the following aspects, line test, query token encryption, distance comparison and decryption.

Figure 3 The performance of LineTest scheme. (a) Set expanded dimension as $m = 80$. (b) Set data item as $n = 50000$.

Figure 3 demonstrates the performance of LineTest tool. Figure 3(a) shows the time cost for various n with a fixed $m = 80$, while Figure 3(b) shows the time cost for various m with a fixed $n = 50,000$. We note that token generation needs a slightly expensive time cost than three other processes and it is very efficient, about 195s for 100,000 queries with $m = 80$. The test and decryption processes are very efficient and nearly stay flat, 11s for 100,000 tests and 30s for 100,000 decryptions ($m = 80$). Figure 3(b) shows that the expanded dimension has a small affect on the time cost. Token generation is on the top and it is only affected with $m = 100$. Therefore, Figure 3(a) and 3(b) demonstrate that the time cost of LineTest scheme is very small and acceptable.

7.2 Performance of SecPL scheme

In this section, we use a real database to evaluate the overhead. We run the code on the Gowalla database containing 6,442,890 check-in locations. We randomly choose 1,000,000 locations to evaluate the proposed SecPL scheme. The OPE is simulated as scheme [29] and the symmetric encryption is illustrated as AES-ECB-128 provided by the PyCrypto library. The matrix operations is run in numpy library. We evaluate the performance of SecPL scheme from following aspects, key generation, tree construction, and location search.

Key generation. For the proposed SecPL scheme, the secret key consists of a AES-ECB-128 secret key, a $m-$bit string S, two $m \times m$ invertible matrices M_1, M_2

and $(m-3)$ random numbers. In our experiments, it takes less than 5ms to generate secret keys for m from 10 to 100.

Tree construction. In tree construction, we simulate the tree build and tree encryption processes together. We vary the size of database from $n = 10,000$ to $100,000$ in the tree construction process and evaluate the time cost and storage cost for various n with $m = 80$.

Figure 4 The cost of tree construction in SecPL scheme. Set expanded dimension as $m = 80$. (a)Average database encryption time vs. n. (b)Average encrypted database storage vs. n.

Figure 4 shows the performance of tree construction under various n with $m = 80$. From Figure 4(a) and Figure 4(b), we note that the time cost and storage cost is linearly increased with the growth of database. Figure 4(a) demonstrates that the database encryption time are relatively small, 11s for 10,000 data items and 157s for 100,000 data items. In Figure 4(b), we evaluate the storage cost with the growth of database. It shows that the storage cost researches 101M for 10,000 data items and 1,013M for 100,000 data items.

Figure 5 Average search time for 1,000 queries of SecPL scheme. (a) Set expanded dimension as $m = 80$. (b) Set data item as $n = 50,000$.

Location search. In this process, we measure the execution time for token generation and tree search. The token generation algorithm is performed by LBS users and tree search algorithm is executed by cloud server. We evaluate the time cost for 1,000 queries under various database's size n and expanded dimension m.

Fig. 5 shows the time cost for token generation and tree search under various n with $m = 80$ and various m with $n = 50,000$. We note that time cost for token generation and tree search is not affected with the growth n and m. The token generation time even stays flat. Furthermore, tree search time is far bigger than token generation time, about 2s for tree search and 0.2s for token generation.

8 Conclusions

In this paper, we study a point location approach to securely find a region containing a query point over an encrypted database in LBS system. More specifically, a LineTest scheme has been proposed as a basic tool to check whether a point lie above or below a line. Furthermore, the introduced tool LineTest approach has considerable potentials, such as polygon range LBS queries and circle range LBS queries. Combined LineTest and order preserving encryption techniques, SecPL scheme is introduced to help people in LBS system. The proposed scheme has a faster-than-linear search complexity and achieves IND-OCOA security. That is, the capacity of an adversary is limited by ciphertext-only attack. It is possible for the adversary $\mathcal{A}$ to obtain other related back-ground knowleddge, e.g., knows a set of plains and the corresponding ciphertexts. How to design a more secure point location scheme is a subject for future work.

Acknowledgements
The authors are grateful to the anonymous referees for their invaluable suggestions.

Funding
Not applicable.

Abbreviations
LBS: location based services; ASPE: asymmetric scalar-product-preserving encryption; OPE: order-preserving encryption; TTP: trusted third party; PIR: private information retrieval technique; IND-OCOA: indistinguishability under order-ciphertext-only attacks.

Availability of data and materials
The data used to support the findings of this study are available from the corresponding author upon request.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
JG conceived and designed this study in consultation with JS. JG performed a pre-study, and positioned the work in the current state of research. JS helped carrying out the main experiments. Both authors have contributed in writing the manuscript and have approved the submitted version.

Author details
[1]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China. [2]School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, China.

References

1. J. Shao, R. Lu, and X. Lin, FINE: A fine-grained privacy-preserving location-based service framework for mobile devices, in Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Toronto, Canada, April 27 - May 2, 2014, pp. 244-252.
2. H. Zhu, F. Liu, and H. Li, Efficient and privacy-preserving polygons spatial query framework for location-based services, IEEE Internet Things J., vol. 4, no. 2, pp. 536-545, 2017.
3. F. Farouk, Y. Alkady, and R. Rizk, Efficient privacy-preserving scheme for location based services in VANET system, IEEE Access, vol. 8, pp. 60101-60116, 2020.
4. J. Zhou, Z. Cao, Z. Qin, X. Dong, and K. Ren, LPPA: lightweight privacy-preserving authentication from efficient multi-key secure outsourced computation for location-based services in vanets, IEEE Trans. Information Forensics and Security, vol. 15, pp. 420-434, 2020.
5. M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, Secure outsourcing of scientific computations, Adv. Comput., vol. 54, pp. 215-272, 2001.
6. H. Hacigümüs, S. Mehrotra, and B. R. Iyer, Providing database as a service, in Proceedings of the 18th International Conference on Data Engineering (ICDE), San Jose, CA, USA, February 26- March 1, 2002, pp. 29-38.
7. G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan, Private queries in location based services: anonymizers are not necessary, in Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), Vancouver, BC, Canada, June 10-12, 2008, pp. 121-132.
8. H. Hu, Q. Chen, and J. Xu, VERDICT: privacy-preserving authentication of range queries in location-based services, in Proceedings of the 29th International Conference on Data Engineering (ICDE), Brisbane, Australia, April 8-12, 2013, pp. 1312-1315.
9. Y. Zhang, W. Tong, and S. Zhong, On designing satisfaction ratio-aware truthful incentive mechanisms for k-anonymity location privacy, IEEE Trans. Information Forensics and Security, vol. 11, no. 11, pp. 2528-2541, 2016.
10. H. Hu and J. Xu, 2pass: Bandwidth-optimized location cloaking for anonymous location-based services, IEEE Trans. Parallel Distrib. Syst., vol. 21, no. 10, pp. 1458-1472, 2010.
11. Y. Liu, X. Chen, Z. Li, Z. Li, and R. C. Wong, An efficient method for privacy preserving location queries, Frontiers Comput. Sci., vol. 6, no. 4, pp. 409-420, 2012.
12. R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, Privacy-preserving and content-protecting location based queries, IEEE Trans. Knowl. Data Eng., vol. 26, no. 5, pp. 1200-1210, 2014.
13. D. Boneh, E. Goh, and K. Nissim, Evaluating 2-dnf formulas on ciphertexts, in Proceeding of IACR Theory of Cryptography Conference (TCC), Cambridge, MA, USA, February 10-12, 2005, pp. 325-341.
14. B. Gedik and L. Liu, Protecting location privacy with personalized k-anonymity: Architecture and algorithms, IEEE Trans. Mob. Comput., vol. 7, no. 1, pp. 1-18, 2008.
15. G. Zhong and U. Hengartner, Toward a distributed k-anonymity protocol for location privacy, in Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES), Alexandria, VA, USA, October 27, 2008, pp. 33-38.
16. K. Vu, R. Zheng, and J. Gao, Efficient algorithms for k-anonymous location privacy in participatory sensing, in Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Orlando, FL, USA, March 25-30, 2012, pp. 2399-2407.
17. T. Dargahi, M. Ambrosin, M. Conti, and N. Asokan, ABAKA: A novel attribute-based k-anonymous collaborative solution for lbss, Comput. Commun., vol. 85, pp. 1-13, 2016.
18. A. Khoshgozaran and C. Shahabi, A taxonomy of approaches to preserve location privacy in location-based services, IJCSE, vol. 5, no. 2, pp. 86-96, 2010.
19. M. F. Mokbel, C. Chow, and W. G. Aref, The new casper: Query processing for location services without compromising privacy, in Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, September 12-15, 2006, pp. 763-774.

20. X. Wu, J. Liu, X. Hong, and E. Bertino, Anonymous geo-forwarding in manets through location cloaking, IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 10, pp. 1297-1309, 2008.
21. H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li, An efficient privacy-preserving location-based services query scheme in outsourced cloud, IEEE Trans. Veh. Technol., vol. 65, no. 9, pp. 7729-7739, 2016.
22. M. Zeng, K. Zhang, J. Chen, and H. Qian, P3GQ: A practical privacy-preserving generic location-based services query scheme, Pervasive Mob. Comput., vol. 51, pp. 56-72, 2018.
23. B. Wang, M. Li, and H. Wang, Geometric range search on encrypted spatial data, IEEE Trans. Information Forensics and Security, vol. 11, no. 4, pp. 704-719, 2016.
24. B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Commun. ACM, vol. 13, no. 7, pp. 422-426, 1970.
25. E. Shen, E. Shi, and B. Waters, Predicate privacy in encryption systems, in Proceeding of IACR Theory of Cryptography Conference (TCC), San Francisco, CA, USA, March 15-17, 2009, pp. 457-473.
26. B. Wang, M. Li, and L. Xiong, Fastgeo: Efficient geometric range queries on encrypted spatial data, IEEE Trans. Dependable Secur. Comput., vol. 16, no. 2, pp. 245-258, 2019.
27. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, Order-preserving encryption for numeric data, in Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France, June 13-18, 2004, pp. 563-574.
28. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, Order-preserving symmetric encryption, in Proceeding of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Cologne, Germany, April 26-30, 2009, pp. 224-241.
29. R. A. Popa, F. H. Li, and N. Zeldovich, An ideal-security protocol for order-preserving encoding, in Proceeding of the IEEE Symposium on Security and Privacy (SP), Berkeley, USA, May 19-22, 2013, pp. 463-477.
30. D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, POPE: partial order preserving encoding, in Proceedings of the ACM Conference on Computer and Communications Security (CCS), Vienna, Austria, October 24-28, 2016, pp. 1131-1142.

Figures

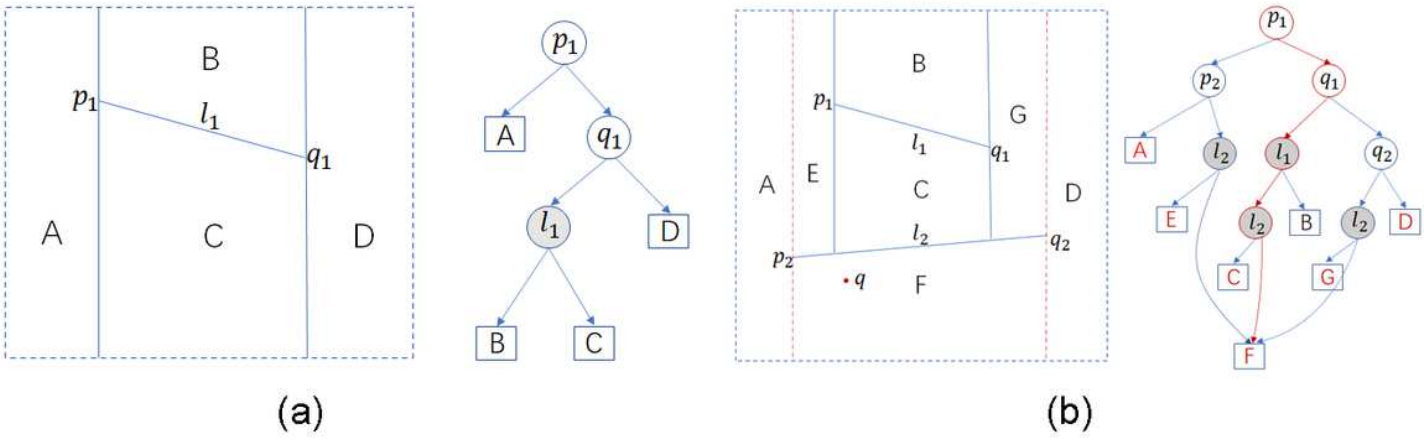

Figure 1

An architecture of randomized incremental algorithm.(a)The insertion of line l1. (b) The insertion of line l2.

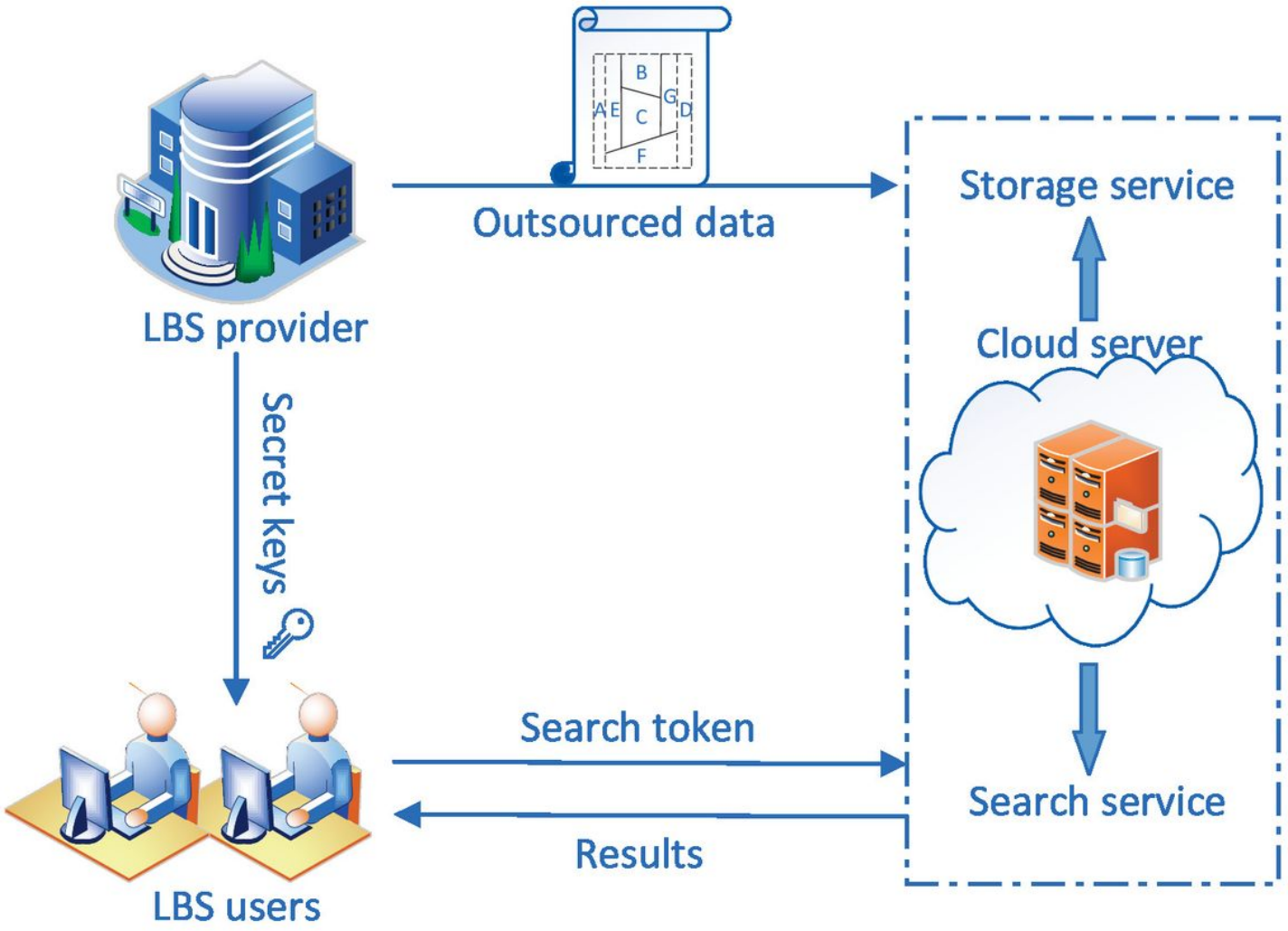

Figure 2

The architecture of system model.

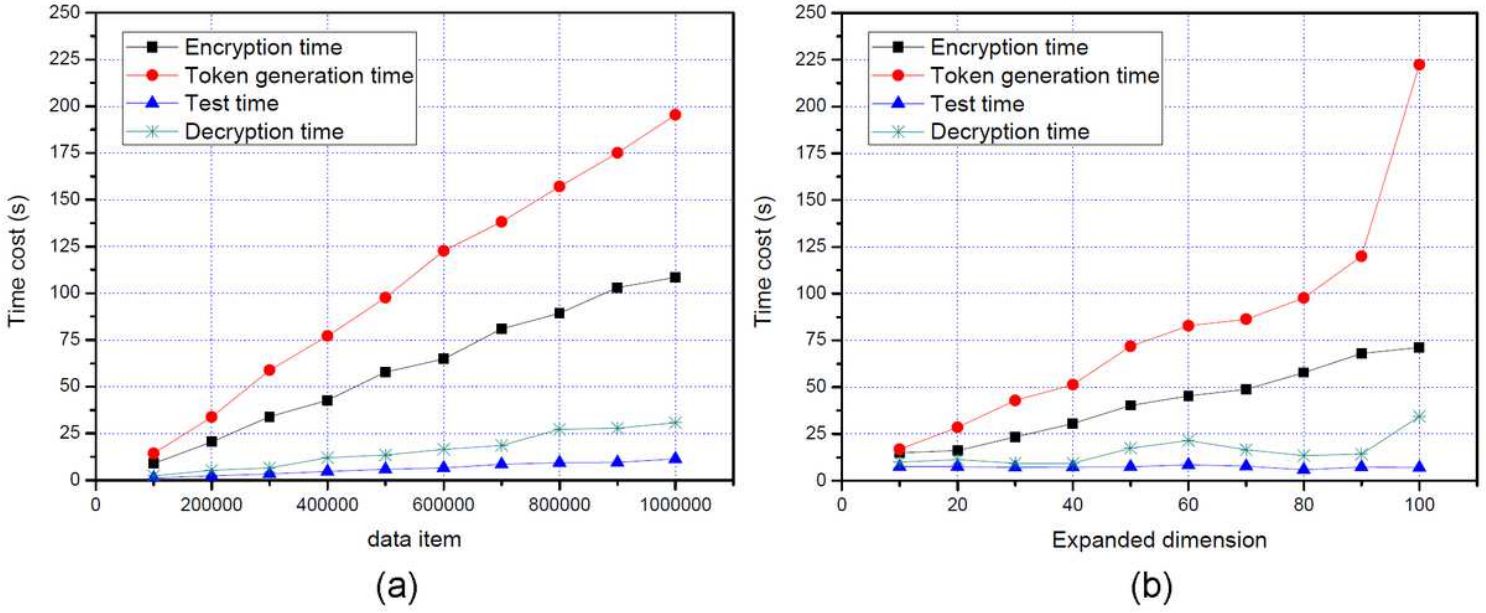

Figure 3

The performance of LineTest scheme. (a) Set expanded dimension as m = 80. (b) Set data item as n = 50000.

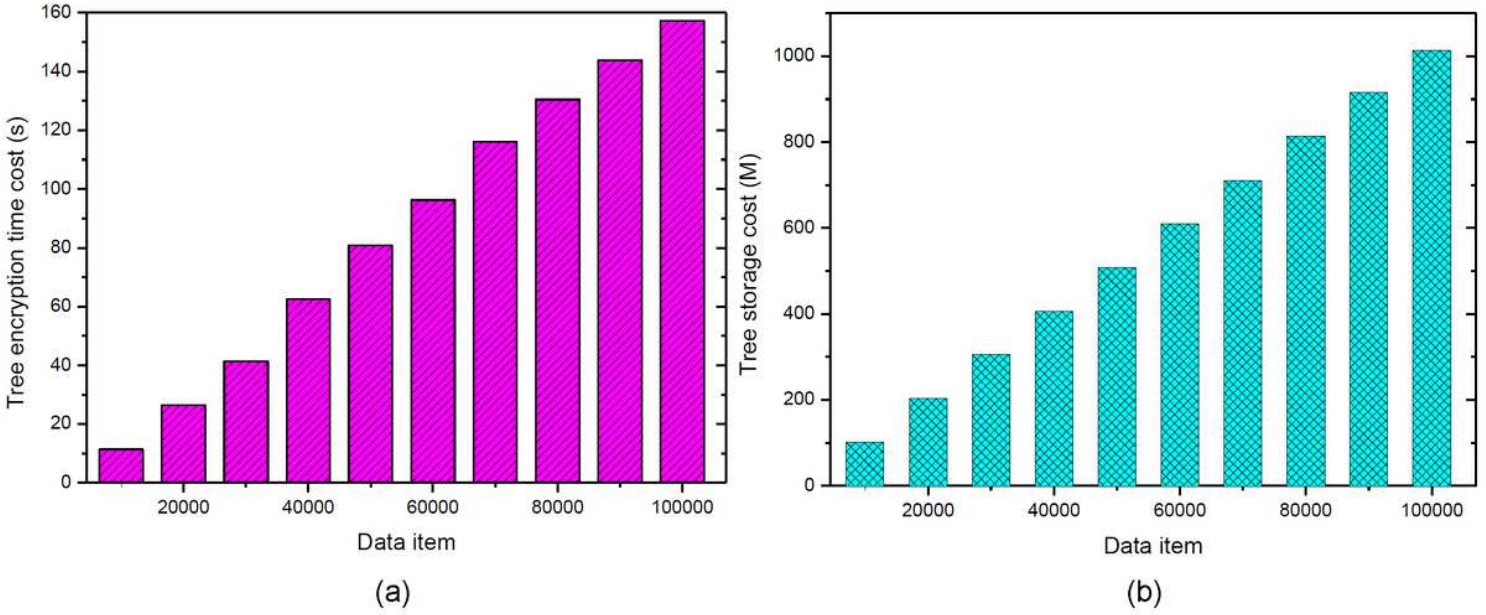

Figure 4

The cost of tree construction in SecPL scheme. Set expanded dimension as m = 80. (a)Average database encryption time vs. n. (b)Average encrypted database storage vs. n.

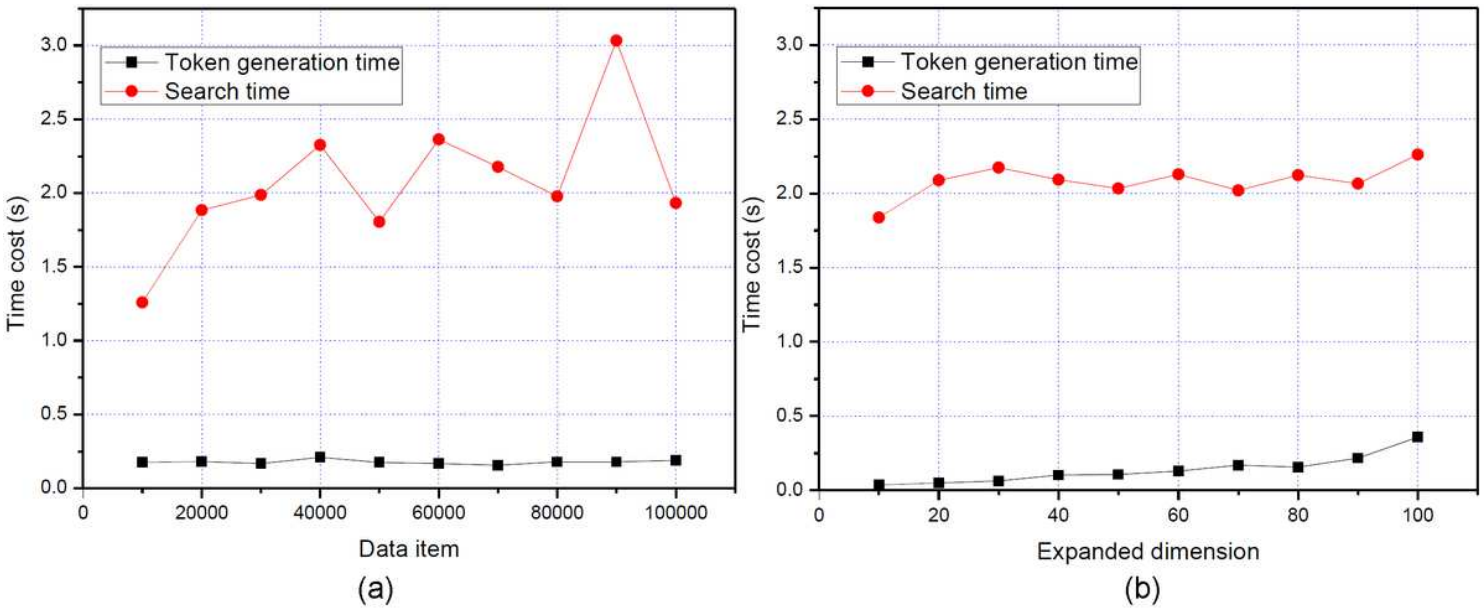

Figure 5

Average search time for 1,000 queries of SecPL scheme. (a) Set expanded dimension as m = 80. (b) Set data item as n = 50; 000.