Regular Paper

LPCQP: Lightweight Private Circular Query Protocol with Divided POI-table and Somewhat Homomorphic Encryption for Privacy-Preserving *k*-NN Search

Yasuhito Utsunomiya[1,a) Kentaroh Toyoda[1,b) Iwao Sasase[1,c)

Received: March 10, 2015, Accepted: September 2, 2015

Abstract: With the recent growth of mobile communication, the location-based *k*-nearest neighbor (*k*-NN) search is getting much attention. While the *k*-NN search provides beneficial information about points of interest (POIs) near users, users' locations could be revealed to the server. Lien et al. have recently proposed a highly-accurate privacy-preserving *k*-NN search protocol with the additive homomorphism. However, it requires a heavy computation load due to the unnecessary multiplication on the server in the encryption domain. In this paper, we propose a lightweight private circular query protocol (LPCQP) with divided POI-table and the somewhat homomorphic encryption for privacy-preserving *k*-NN search. Our proposed scheme removes unnecessary POI information for the request user by dividing and aggregating a POI-table, and this reduces both the computational and the communication costs. In addition, we use both additive and multiplicative homomorphisms to perform the above process in the encryption domain. We evaluate the performance of our proposed scheme and show that our scheme reduces both the computational and the communication costs while maintaining high security and high accuracy.

Keywords: location privacy, k-nearest neighbor query, location-based service, somewhat homomorphic encryption

1. Introduction

Location-based services (LBSs) are a major service among recent mobile communication trends to exploit mobile users. The key technique of an LBS is to find points of interest (POIs), e.g., cafes or drugstores, in the vicinity of a user. In particular, finding the nearest *k* POIs around the user is called *k*-nearest neighbor (*k*-NN) search. An LBS basically consists of two entities, a user who wants to find *k*-nearest POIs and an LBS server which returns them. An LBS server manages POI information such as names or addresses. For instance, when a user at a station wants to find nearby cafes, the user informs his/her current location, e.g., latitude and longitude, to the server. Then, since the server knows the user's location and it can provide the information about cafes near the station. However, the user's current location is sensitive and informing it might harm the privacy of the user.

A private circular query protocol (PCQP) proposed by Lien et al. [1] is a state-of-the-art protocol for privacy-preserving *k*-NN search. This protocol adopts the Moore curve [2] and the Paillier cryptosystem [3] to securely rotate the POI-table to conceal a user's location. No adversary (even an LBS server) knows the user's location in PCQP because the user's query and the search results are encrypted. However, it takes a high computational cost since it involves a large number of multiplications on a matrix in the encryption domain on the server side and this must be reduced to achieve a real-time LBS.

In this paper, we propose a lightweight private circular query protocol (LPCQP) with the divided POI-table and the somewhat homomorphic encryption for privacy-preserving *k*-NN search. We notice that PCQP multiplies an encrypted matrix by the entire POI-table for circularly shifting although most POIs are not required for search results. Accordingly, we propose to divide a POI-table into some sub-tables and aggregate them into a table before shifting a POI-table in order to obtain partial entries of a POI-table. This reduces the computational cost in the matrix multiplication on the server. In addition, we use a somewhat homomorphic cryptosystem with the properties of additive and multiplicative homomorphisms to perform the above process in the encryption domain. As a result, our scheme achieves low complexity and high accuracy of *k*-NN search without losing security. Moreover, the communication cost is also reduced in our scheme because the size of a matrix becomes small.

We evaluate the performance of our proposed scheme with both real and synthetic POI datasets and show that our scheme reduces the computational costs on the server and the user by up to 99% and 98% respectively and the communication cost by at least 27% by allowing a 5% reduction in search accuracy without losing security.

The rest of this paper is constructed as follows. Section 2 describes the background techniques. The related work is introduced in Sections 3 and 4. System model and comprehensive description regarding the proposed scheme are described in Section 5. Simulation result and evaluation are discussed in Section 6. Finally we conclude our discussion in Section 7.

1 Keio University, Yokohama, Kanagawa 223–8522, Japan
a) utsunomiya@sasase.ics.keio.ac.jp
b) toyoda@sasase.ics.keio.ac.jp
c) sasase@ics.keio.ac.jp

2. Background

2.1 *k*-Nearest Neighbor Search

Finding the nearest k POIs from the querying point is called k-NN search. In particular, $k = 1$ means that the user obtains the information about the nearest POI. An LBS basically consists of two entities, a user who wants to find k-nearest POIs and an LBS server which manages POI information such as names or addresses. In k-NN search, the user first sends his/her current location to the server as a query. The server retrieves POIs on the database with the user's location and returns the k POIs which are near the user. As a result, the user obtains the information about the nearby POIs. However, the current location is highly sensitive for the user and should not be revealed to untrusted entities unnecessarily. The query may be leaked not only over the network by eavesdropping but also from the LBS server by the attack. In addition, an adversary may masquerade as an LBS server. Therefore, privacy-preserving k-NN search is required for the LBS.

2.2 Homomorphic Encryption

For preserving the user's privacy, especially in private information retrieval (PIR), homomorphic encryption is adopted. In order to describe the homomorphism, we denote by $\mathcal{E}_{pk}(m; r)$ and $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m; r))$ the encryption of a message m under an encryption key pk with a pseudo-random number r and the decryption of that under a decryption key sk, respectively. For given plaintexts m_1 and m_2, pseudo-random numbers r_1 and r_2, a public-key pk, and a private-key sk, Eqs. (1) and (2) are satisfied in the homomorphic cryptography.

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m_1; r_1) +_c \mathcal{E}_{pk}(m_2; r_2)) = m_1 + m_2. \tag{1}$$

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m_1; r_1) \times_c \mathcal{E}_{pk}(m_2; r_2)) = m_1 \times m_2. \tag{2}$$

Here, $+_c$ and $\times_c$ denote the operator of additive homomorphism and that of multiplicative homomorphism, respectively. In other words, the server can execute addition and/or multiplication of plaintexts without decryption. For example, the NTRU cryptosystem [4] is a cryptosystem with both additive and multiplicative homomorphisms [5]. In particular, if an encryption scheme fully supports additive and multiplicative homomorphisms, it is called a fully homomorphic encryption (FHE) scheme. In contrast, a somewhat homomorphic encryption (SHE) scheme supports these properties with a limited number of operations. Although an FHE scheme can reduce the noise of the encrypted data by bootstrapping [6], the cost is significantly heavy compared with that of an SHE. We, therefore, adopt an SHE scheme for our scheme. The effects on cryptography from noise in our scheme are discussed in Section 5.8.

3. Related Work

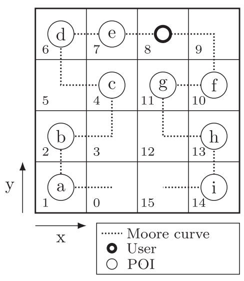
Many schemes have been proposed to allow mobile users to search nearby POI without revealing their own location. They are categorized into three types; cloaking-based [7], [8], [9], [10], [11], transformation-based [12], [13], [14], and PIR-based methods [1], [15], [16], [17]. The cloaking-based methods generate a cloaking region that an LBS server cannot distinguish the user from other $K - 1$ users to obscure the location information. One of most active challenges in cloaking-based methods is to reduce useless spaces in a cloaking region. For example, the scheme in Ref. [7] utilizes adjacent cell information where it does not pass through an original Hilbert curve and Tan et al. propose a multiple anonymizing spatial regions algorithm [8]. In another study, Kalnis et al. propose a circular range k-NN search which reduces the number of redundant results [9]. Chow et al. identify three new privacy-aware query types, private queries over public data, public queries over private data, and private queries over private data, and propose a framework, Casper*, which supports all the three query types [10]. Meanwhile, the scheme in Ref. [11] uses proximity information among users instead of their coordinates for location cloaking while the schemes [7], [8], [9], [10] use users' coordinates. The transformation-based methods transform the queried location into other location via a trusted third party (TTP) to conceal a user's location. The key point of transformation-based methods is how to conceal the user's location from a malicious user/server. Khoshgozaran et al. propose a dual curve query resolution which achieves highly-accurate k-NN search and guarantees stringent privacy via a trusted third party [12]. On the other hand, the schemes in Refs. [13] and [14] do not rely on any trusted third party. In LocX [13], the transformation key is shared among the user's friends, so that only the friends know the user's actual location. Yiu et al. propose SpaceTwist [14] which obscures the user's location by incrementally retrieving POIs from a fake location (i.e., the user's actual location is transformed) until the user's privacy and quality requirements are satisfied. The PIR-based methods enable a user to retrieve a POI from a database on the server without revealing which POI is retrieved, e.g., by using cryptographic techniques. However, PIR techniques require high computational and communication costs. Therefore, an efficient search technique is required in order to reduce these costs. Papadopoulos et al. propose an aggregate Hilbert grid (AHG) scheme which achieves low computational and communication costs for arbitrary k-NN search by eliminating the empty space in the database blocks [15]. Lien et al. propose a private circular query protocol with homomorphic encryption [1]. They also propose a cross-like search algorithm and achieve the high accuracy utilizing the algorithm. While the scheme in Ref. [1] supports only k-NN search, Khoshgozaran et al. propose an approach which supports range and k-NN searches [16], [17]. The cloaking-based and the transformation-based methods achieve high accuracy of k-NN search, but they require a TTP to hide a user's location [7], [8], [9], [10], [11], [12]. However, the TTP possesses sensitive information for many LBSs and could be the target of attacks, e.g., the denial of service (DoS) attack. Moreover, the TTP becomes the bottleneck of the service. In contrast, the PIR-based methods achieve high-level security without a TTP. Although this type of k-NN search requires a high computational cost for searching POIs, it is robust to the correlation attack or the background knowledge attack, whereas the cloaking-based and the transformation-based methods are vulnerable to these attacks [1].

4. Private Circular Query Protocol

For high quality k-NN search with strong security, we pay attention to PCQP which is a state-of-the-art PIR-based scheme proposed by Lien et al. [1]. PCQP offers highly-accurate privacy-preserving k-NN search without a TTP. Moreover, even if an adversary masquerades as an LBS server, the user's location cannot be revealed to that in PCQP due to the PIR technique. A user's location is hidden by circularly shifting the entries of a POI-table through k-NN search. The shift amount is randomly chosen by the user whenever he/she queries to the LBS. Then, the user requires k-nearest POIs of the shifted location. Since the POI-table is also shifted, the user can obtain nearby k POIs by informing the shifted location. In order to securely shift the POI-table, Lien et al. adopt the Paillier cryptosystem as a building tool of additive homomorphism [1]. Shifting is securely performed in the encryption domain, and thus no user's location information is revealed to the server. In addition, k-NN search is performed in a one-dimensional space in PCQP using a space-filling curve [2]. Since the distance between a user and a POI is calculated in a one-dimensional one, the calculation cost is lower than that of k-NN search performed in a two-dimensional one. The architecture of PCQP is described below.

4.1 Initialization Process

In the initialization process, an LBS server constructs the Moore curve. The Moore curve is a space-filling curve with end-point-connected property. **Figure 1** (a) indicates the second order Moore curve. As shown in Fig. 1 (a), the dotted line crosses every cell. As the characteristic of the Moore curve, the start point (point 0 in Fig. 1 (a)) and the end point (point 15 in Fig. 1 (a)) are adjacent. The number on the corner of each cell represents an index in the curve, which is called *H-value*. Thus, the two-dimensional coordinates of a POI can be converted into the one-dimensional one, i.e., *H-value*. Utilizing the *H-value* makes the k-NN search more efficient because (i) calculating the distance between *H-value*s is low cost and (ii) it partially retains the adjacency relation of the two-dimensional one. The server generates two tables, lookup-table and POI-table, from POIs stored on the server. The lookup-table contains *H-value* and *H-index* of each POI, where *H-index* denotes the evenly distributed value numbered in the ascending order of *H-value* with common difference d (i.e., *H-index* of i-th POI in the ascending order of *H-value* is calculated as $d \times i$). The POI-table contains POI information, e.g., the names or the latitude and longitude of POIs, and *H-index* of each POI. Only the lookup-table is publicly announced to users for retrieving *H-index* of their current location. Figure 1 (b) and (c) indicate a POI-table and a lookup-table of the POIs on the map in Fig. 1 (a), respectively. Here, $d = 2$ in Fig. 1. The user knows which cell the user belongs to from the Moore curve and the *H-index* of the cell from the lookup-table. Meanwhile, each user generates their public-key pk and private-key sk of the Paillier cryptosystem and sends their pk to the server. Note that each user retains their sk.



(a) Moore curve.

H-index	POI-info
2	a
4	b
6	c
8	d
10	e
12	f
14	g
16	h
18	i

(b) POI-table.

H-value	H-index
1	2
2	4
4	6
6	8
7	10
10	12
11	14
13	16
14	18

(c) lookup-table.

Fig. 1 Example of location information on an LBS server.

4.2 Query Process

In the query process, a user chooses an arbitrary amount of shifts t and generates an $n_p \times n_p$ t-offset circular shift permutation matrix $\mathbf{P}^t$, which is defined as

$$\mathbf{P}^t = \left[P_{i,j}\right]_{0 \leq i,j \leq n_p - 1}, \quad P_{i,j} = \begin{cases} 1 & (j = (i + n_p - t) \bmod n_p) \\ 0 & (otherwise) \end{cases}, \tag{3}$$

where n_p denotes the number of all entries in a POI-table. In order to hide the amount of shifts, the user encrypts $\mathbf{P}^t$ before sending it. However, it requires high communication cost for sending the encrypted whole $\mathbf{P}^t$. Here, $(i + 1)$-th row of $\mathbf{P}^t$ can be obtained by circularly shifting the element of i-th row by one. Therefore, the user encrypts only the first row of $\mathbf{P}^t$ and sends it. For a given set of pseudo-random numbers $\boldsymbol{r} = \{r_1, \cdots, r_{n_p}\}$ and the user's public-key pk, the encrypted first row of $\mathbf{P}^t$ is represented as

$$\mathcal{E}_{pk}(\boldsymbol{P}^t_{0,j}; \boldsymbol{r}) = \left[\mathcal{E}_{pk}(P_{0,j}; r_{j+1})\right]_{0 \leq j \leq n_p - 1}, \tag{4}$$

where $\boldsymbol{P}^t_{0,j}$ denotes the first row of $\mathbf{P}^t$. After encrypting $\boldsymbol{P}^t_{0,j}$, the user sends $\mathcal{E}_{pk}(\boldsymbol{P}^t_{0,j}; \boldsymbol{r})$ and the shifted location, which is calculated as $H\text{-}index_u + t \times d$, to the server, where $H\text{-}index_u$ denotes the *H-index* of the user retrieved from the lookup-table by using his current location as a key. The server receives $\mathcal{E}_{pk}(\boldsymbol{P}^t_{0,j}; \boldsymbol{r})$ and constructs the encrypted $\mathbf{P}^t$, i.e., $\mathcal{E}_{pk}(\mathbf{P}^t; \boldsymbol{r})$, by circularly shifting the element of $\mathcal{E}_{pk}(\boldsymbol{P}^t_{0,j}; \boldsymbol{r})$ to the right by one consecutively. Then, the server multiplies $\mathcal{E}_{pk}(\mathbf{P}^t; \boldsymbol{r})$ by a POI-table. By the homomorphic properties of the Paillier cryptosystem, the POI-table is shifted by t without decryption. That is, for given entries of POI $\boldsymbol{I} = \{I_1, \cdots, I_{n_p}\}$, the i-th entry of the shifted POI-table I^t_i is calculated as:

$$I^t_i = \prod_{j=0}^{n_p - 1} \mathcal{E}_{pk}(P_{i-1,j}; r_{j+1})^{I_{j+1}}. \tag{5}$$

Here, I^t_i is still encrypted. In addition, because of the homomorphic properties of the Paillier cryptosystem, I^t_i satisfies

$$\mathcal{D}_{sk}(I^t_i) = I_{((i + n_p - t) \bmod n_p)}. \tag{6}$$

Therefore, only the user who possesses the correspondent private key can decrypt the shifted POIs and obtain the decrypted them. Finally, the server returns k-nearest rows in the shifted POI-table from the shifted location specified by the user. Note that the server cannot obtain any information regarding the user's location since the amount of shifts t varies every time the user queries and every shifted POI (i.e., I^t_i) is encrypted with the user's public-key as shown in Eq. (5). After receiving encrypted k results from the server, the user decrypts them with its private-key sk.

4.3 Cross-Like k-NN Search

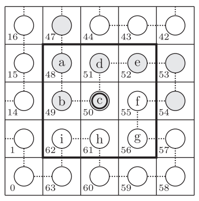
Under the condition that each cell has one POI on an average, most of exact k-nearest POIs are covered by a $(2D+1)\times(2D+1)$ area centered at the querying point, where D is the minimum positive integer satisfying $(2D + 1)^2 \geq k$. For example, when a user issues a 9-NN search query at the cell of 50 in **Fig. 2** (a), exact 9-nearest POIs (i.e., a, b, c, d, e, f, g, h, and i) are covered by a 3×3 area. Here, as shown in Fig. 2 (a), some POIs in the area (i.e., f, g, h, and i) cannot be searched by 9-NN search because these are far from the querying point on the curve. For the full coverage of the area, a user issues additional queries from the central cells of the area's borders in PCQP. These are selected in the descending order by the differences between *H-value*s to reduce duplicated POIs. In Fig. 2 (a), POIs b, d, f, and h are candidates for that query. By issuing additional queries from these cells, more POIs in the area can be searched as shown in Fig. 2 (b), Fig. 2 (c), Fig. 2 (d), and Fig. 2 (e). Though this technique improves the accuracy of k-NN search, the computational cost and the communication cost increase in proportion to the number of additional queries.

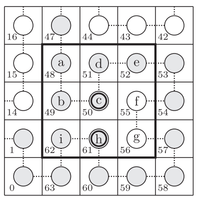
4.4 PCQP with Somewhat Homomorphic Encryption

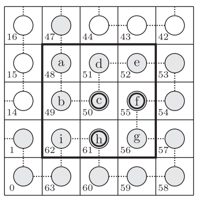
As shown in Eq. (4), the communication cost for sending $\mathcal{E}(\boldsymbol{P}^t_{0,j};\boldsymbol{r})$ increases in proportion to n_p. Moreover, data size of a ciphertext should be extremely large to ensure high security, and this causes burden on a mobile user. To address this problem, Lien et al. also proposed to decompose $\boldsymbol{P}^t_{0,j}$ into two vectors. For instance, $\boldsymbol{P}^3_{0,j}$, *i.e.*, [0 1 0 0], can be obtained from the outer products of $\boldsymbol{u} = [1, 0]$ and $\boldsymbol{v} = [0, 1]$ calculated as

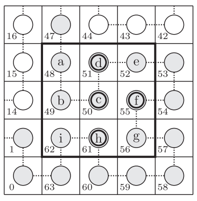
$$\boldsymbol{u}^T\boldsymbol{v} = \begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}0 & 1\\0 & 0\end{bmatrix}, \tag{7}$$

by reading in raster-scan ordering. Thereby, the communication cost becomes $O(\sqrt{n_p})$ from $O(n_p)$ by sending $\mathcal{E}(\boldsymbol{u};\boldsymbol{r^u})$ and $\mathcal{E}(\boldsymbol{v};\boldsymbol{r^v})$ instead of $\mathcal{E}(\boldsymbol{P}^t_{0,j};\boldsymbol{r})$, where $\boldsymbol{r^u}$ and $\boldsymbol{r^v}$ are sets of $\sqrt{n_p}$ random numbers (i.e., $\boldsymbol{r^u} = \{r_1,\cdots,r_{\sqrt{n_p}}\}$ and $\boldsymbol{r^v} = \{r'_1,\cdots,r'_{\sqrt{n_p}}\}$). Note that a server must compute the product of two encrypted vectors in this technique. Therefore, this technique requires a somewhat/fully homomorphic encryption that is a cryptosystem with additive and multiplicative homomorphic properties instead of the Paillier cryptosystem.


(a) 9-NN search from a query point c.


(b) 9-NN search from query points c and h.


(c) 9-NN search from query points c, h, and f.


(d) 9-NN search from query points c, h, f, and d.

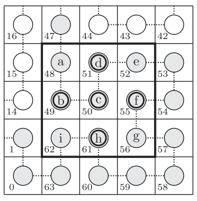

(e) 9-NN search from query points c, h, f, d, and b.

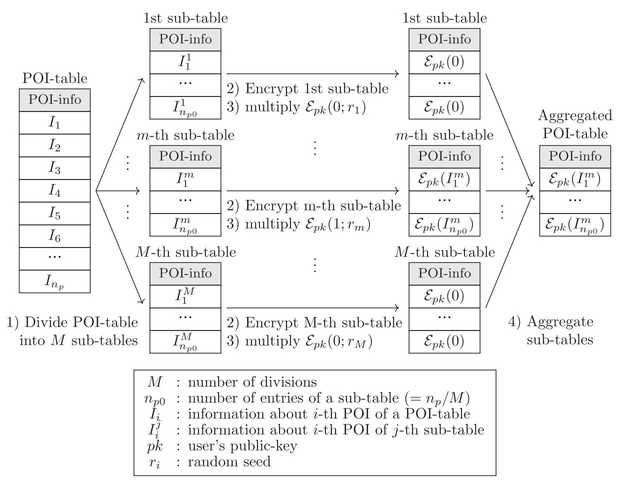
Fig. 2 Example of cross-like k-NN search when $k = 9$. Shaded circles are reachable POIs by 9-NN search.

4.5 Drawback of PCQP

Although PCQP achieves high-level security and high accuracy, it requires a high computational load in the matrix multiplication on the LBS server. As mentioned in Section 4.2, for a given k-NN query, the server requires $k(n_p - 1)$ additions and kn_p multiplications in the plaintext domain. Traditional homomorphic encryption schemes incur a huge number of calculations even for a simple calculation of plaintexts. Although fast homomorphic encryption schemes have been proposed to improve the calculation speed [18], [19], especially for real-time mobile services, the high-cost computation is still a serious problem and may keep users waiting for a long time in each query. In PCQP, obviously, multiplication applied across the entire POI-table is too wasteful to find at most k entries of POI. Focusing on this point, we propose a lightweight k-NN search scheme to mitigate the drawback without impairing advantages of PCQP.

5. Proposed Scheme

Here, we propose a lightweight private circular query protocol (LPCQP) for low-complexity privacy-preserving k-NN search by dividing and aggregating a POI-table. Our idea is to shift the partial POI-table during the k-NN search whereas PCQP shifts the entire one. We notice that PCQP multiplies an encrypted matrix by the entire POI-table for circularly shifting although most POIs are not required for search results. **Figure 3** indicates our technique for partial shifting. In our scheme, the server divides a POI-table into M sub-tables in advance. When a user issues a query, he/she chooses the sub-table that involves POIs the user wants, i.e., the m-th sub-table in Fig. 3. Then the user sends M encrypted elements with his public-key, where only the m-th element is the encrypted '1' and the others are the encrypted '0's. After receiving them, the server multiplies each of them by the corresponding sub-table. Here, the server can reduce the number of entries to be calculated to n_p/M by aggregating them into a table. Note that the server cannot see which sub-table the user



M	:	number of divisions
n_{p0}	:	number of entries of a sub-table ($= n_p/M$)
I_i	:	information about i-th POI of a POI-table
I^j_i	:	information about i-th POI of j-th sub-table
pk	:	user's public-key
r_i	:	random seed

Fig. 3 Operations on an LBS server in LPCQP.

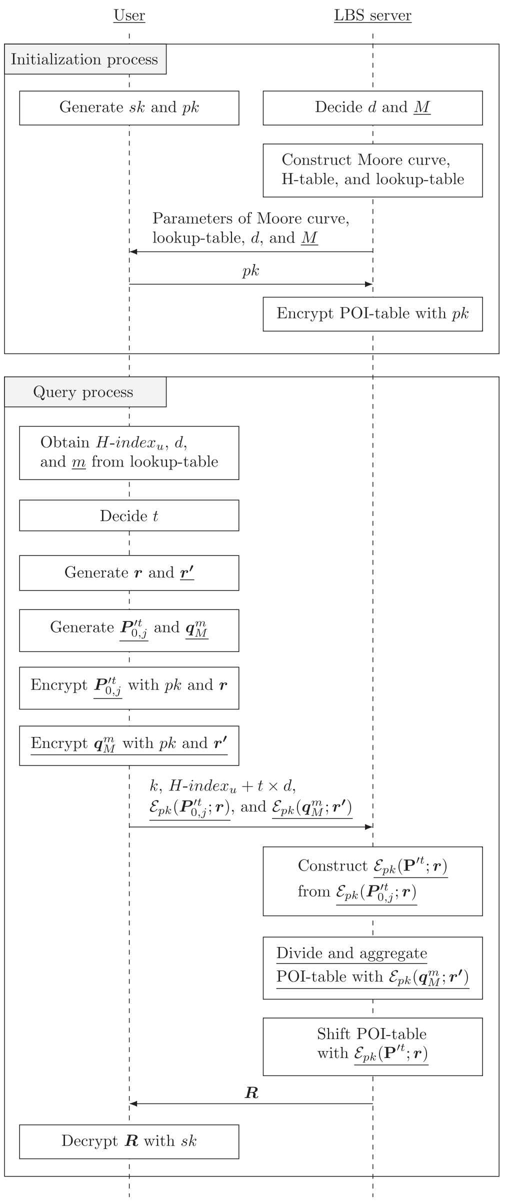

User
LBS server
Initialization process
Generate sk and pk
Decide d and $\underline{M}$
Construct Moore curve, H-table, and lookup-table
Parameters of Moore curve, lookup-table, d, and $\underline{M}$
pk
Encrypt POI-table with pk
Query process
Obtain $H\text{-}index_u$, d, and $\underline{m}$ from lookup-table
Decide t
Generate $\boldsymbol{r}$ and $\underline{\boldsymbol{r}'}$
Generate $\underline{\boldsymbol{P}'^{t}_{0,j}}$ and $\underline{\boldsymbol{q}^{m}_{M}}$
Encrypt $\underline{\boldsymbol{P}'^{t}_{0,j}}$ with pk and $\boldsymbol{r}$
$\underline{\text{Encrypt } \boldsymbol{q}^{m}_{M} \text{ with } pk \text{ and } \boldsymbol{r}'}$
k, $H\text{-}index_u + t \times d$, $\mathcal{E}_{pk}(\underline{\boldsymbol{P}'^{t}_{0,j}}; \boldsymbol{r})$, and $\underline{\mathcal{E}_{pk}(\boldsymbol{q}^{m}_{M}; \boldsymbol{r}')}$
Construct $\underline{\mathcal{E}_{pk}(\mathbf{P}'^{t}; \boldsymbol{r})}$ from $\underline{\mathcal{E}_{pk}(\boldsymbol{P}'^{t}_{0,j}; \boldsymbol{r})}$
$\underline{\text{Divide and aggregate}}$ $\underline{\text{POI-table with } \mathcal{E}_{pk}(\boldsymbol{q}^{m}_{M}; \boldsymbol{r}')}$
Shift POI-table with $\underline{\mathcal{E}_{pk}(\mathbf{P}'^{t}; \boldsymbol{r})}$
$\boldsymbol{R}$
Decrypt $\boldsymbol{R}$ with sk

Fig. 4 Sequence chart of LPCQP. The additional and modified processes from the conventional protocol [1] are underlined.

wants since these processes are performed in the encryption domain with additive and multiplicative homomorphisms. Thus, our scheme reduces the computational cost on the server side without sacrificing the security. We describe how to divide and aggregate a POI-table in Sections 5.2 and 5.3 in detail.

Advantages of our scheme are as follows:

- The computational cost on the server is significantly reduced while keeping the high search accuracy and the high security of PCQP.
- The communication cost and the computational cost on the user side are reduced to the same level as that of PCQP with the decomposition technique with a somewhat homomorphic encryption.

Figure 4 indicates a sequence chart of our proposed protocol. In the following sections, we first explain the system model and the addressed problem in this study. Subsequently, we describe the protocol sequence of LPCQP.

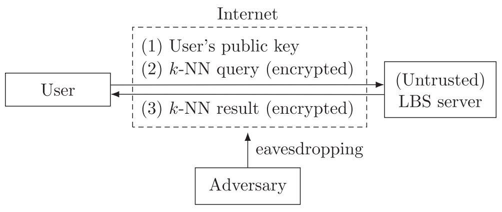


Fig. 5 System model of LPCQP.

5.1 System Model

In this section, we explain the system model and the addressed problem. **Figure 5** indicates our system model that includes the LBS and adversary models. In Fig. 5, there are three entities, which are a user, an LBS server, and an adversary. We assume that the LBS server is untrusted and the adversary has the capability to eavesdrop on the network. The user possesses both his/her private key and public key for encrypting/decrypting the k-NN query/result. The user's public key is sent to the LBS server over the Internet for privacy-preserving k-NN search, so that the LBS server has only the user's public key. Hence, the adversary can also obtain the user's public key. When the user wants to obtain nearby POIs, he/she issues a k-NN query to the LBS server over the Internet. After receiving the query, the LBS server returns the result to the user. Following the assumption, the adversary can obtain both query and result. The goal of an untrusted LBS server and an adversary is to identify the user's location.

In the above settings, we then explain the addressed problems in this paper. There are three important issues to be addressed in privacy-preserving k-NN search: (1) user's privacy, (2) search accuracy, and (3) computational and communication costs. These requirements are not commonly satisfied simultaneously. Cloaking-based and transformation-based methods provide weak privacy protection or require third trusted party while achieving low computational and communication costs and precise search results. By contrast, PIR-based methods require high computation and communication costs while achieving strong privacy protection. The challenge in LPCQP is to fulfill the above three properties at the same time. In other words, LPCQP offers highly-accurate k-NN search with a strict privacy requirement while the computational and communication costs are low.

5.2 Initialization Process

In the initialization process, the server first divides a POI-table into M sub-tables. The j-th sub-table $\boldsymbol{t}^{j}_{\mathsf{POI}}$ is defined as

$$\begin{aligned} \boldsymbol{t}^{j}_{\mathsf{POI}} &= \left[\ I^{j}_{i}\ \right]_{1 \le i \le n_{p0}}, \\ I^{j}_{i} &= I_{(n_{p0}(j-1)+i)}, \end{aligned} \tag{8}$$

where n_{p0}, I^{j}_{i}, and I_i denote the number of all entries of a sub-table calculated as $\lceil n_p/M \rceil$, the i-th entry of the j-th sub-table, and the i-th entry of the original POI-table, respectively. M is an arbitrary integer value and pre-defined from two to n_p by the server depending on the requirements for search accuracy or cost. The complexity becomes lower as M is larger, but the accuracy rate also becomes lower then. The reasons are described in Sections 5.5 and 5.7. For ease of understanding, we give an example on how to divide the POI-table when $n_p = 4$ and M=2. We define a POI-table I, which has four POIs, a, b, c, and d, as

$$\boldsymbol{I} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}. \tag{9}$$

Then the POI-table I is divided into two sub-tables, $\boldsymbol{t}^1_{\mathsf{POI}}$ and $\boldsymbol{t}^2_{\mathsf{POI}}$, using Eq. (8) as follows:

$$\boldsymbol{t}^1_{\mathsf{POI}} = \begin{bmatrix} I^1_1 \\ I^1_2 \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}, \tag{10}$$

$$\boldsymbol{t}^2_{\mathsf{POI}} = \begin{bmatrix} I^2_1 \\ I^2_2 \end{bmatrix} = \begin{bmatrix} I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}. \tag{11}$$

As a result, the first half of POIs, a and b, are assigned to the first sub-table $\boldsymbol{t}^1_{\mathsf{POI}}$ and the others, c and d, are assigned to the second sub-table $\boldsymbol{t}^2_{\mathsf{POI}}$.

Now, let ID_{table} denote an index of a sub-table in which a POI is contained. ID_{table} is added to the lookup-table so that a user knows which sub-table the user should search. The lookup-table of our scheme, therefore, contains ID_{table}, *H-index*, and *H-value* of each POI. Although the processes mentioned above should be performed only once in advance, they have to be performed whenever any POI information is updated, inserted, or deleted as with PCQP.

5.3 Query Process

In the query process, a user first generates a pseudo-random number t and an $n_{p0} \times n_{p0}$ t-offset circular shift permutation matrix $\mathbf{P}'^t$, which is defined as follows:

$$\begin{aligned} \mathbf{P}'^t &= \left[P'_{i,j}\right]_{0 \le i,j \le n_{p0}-1}, \\ P'_{i,j} &= \begin{cases} 1 & (j = (i + n_{p0} - t) \bmod n_{p0}) \\ 0 & (otherwise) \end{cases} . \end{aligned} \tag{12}$$

Before sending $\mathbf{P}'^t$ to the server, the user encrypts it with his/her public-key and pseudo-random numbers $\boldsymbol{r} = \{r_1, \cdots, r_{n_{p0}}\}$ as $\mathcal{E}_{pk}(\mathbf{P}'^t; \boldsymbol{r})$ in order to hide the amount of shifts. Next, the user obtains the index of the sub-table, i.e., m, by retrieving from the lookup-table. The user, then, generates a vector $\boldsymbol{q}^m_M$ defined as

$$\begin{aligned} \boldsymbol{q}^m_M &= [q_i]_{1 \le i \le M}, \\ q_i &= \begin{cases} 1 & (i = m) \\ 0 & (otherwise) \end{cases} . \end{aligned} \tag{13}$$

$\boldsymbol{q}^m_M$ is used for specifying which sub-table a user searches. It is also encrypted with the user's public-key and pseudo-random numbers $\boldsymbol{r}' = \{r'_1, \cdots, r'_M\}$ as $\mathcal{E}_{pk}(\boldsymbol{q}^m_M; \boldsymbol{r}')$ and sent to the server. After receiving them, the server multiplies each element of $\mathcal{E}_{pk}(\boldsymbol{q}^m_M; \boldsymbol{r}')$ by each entry of the corresponding sub-table. That is, the j-th sub-table $\boldsymbol{t}^j_{\mathsf{POI}}$ becomes $\boldsymbol{t}'^j_{\mathsf{POI}}$ defined as

$$\begin{aligned} \boldsymbol{t}'^j_{\mathsf{POI}} &= \mathcal{E}_{pk}(q_j; r'_j) \times_c \mathcal{E}_{pk}(\boldsymbol{t}^j_{\mathsf{POI}}) \\ &= \left[\ \mathcal{E}_{pk}(q_j; r'_j) \times_c \mathcal{E}_{pk}(I^j_i)\ \right]_{1 \le i \le n_{p0}}. \end{aligned} \tag{14}$$

As a result, all entries of sub-tables except the m-th sub-table become zero in the plaintext domain. Then, the server aggregates all $\boldsymbol{t}'^j_{\mathsf{POI}}$ into a table $\boldsymbol{t}'_{\mathsf{POI}}$ as follows:

$$\begin{aligned} \boldsymbol{t}'_{\mathsf{POI}} &= \boldsymbol{t}'^1_{\mathsf{POI}} +_c \boldsymbol{t}'^2_{\mathsf{POI}} +_c \cdots +_c \boldsymbol{t}'^M_{\mathsf{POI}} \\ &= \left[\ I'_i\ \right]_{1 \le i \le n_{p0}}, \\ I'_i &= \mathcal{E}_{pk}(q_1; r'_1) \times_c \mathcal{E}_{pk}(I^1_i) \\ &\quad +_c\ \mathcal{E}_{pk}(q_2; r'_2) \times_c \mathcal{E}_{pk}(I^2_i) \\ &\quad +_c\ \cdots +_c \mathcal{E}_{pk}(q_M; r'_M) \times_c \mathcal{E}_{pk}(I^M_i). \end{aligned} \tag{15}$$

Here, I'_i is still encrypted and satisfies

$$\mathcal{D}_{sk}(I'_i) = I^m_i. \tag{16}$$

Therefore, each entry in an aggregated table becomes corresponding entry of the specified sub-table in the plaintext domain. Note that the server cannot obtain any information regarding the user's location and the sub-table since $\boldsymbol{q}^m_M$ and $\boldsymbol{P}'^t$ are encrypted with the user's public-key and distinct random numbers against each element. In our scheme, the server multiplies $\mathcal{E}_{pk}(\mathbf{P}'^t; \boldsymbol{r})$ instead of $\mathcal{E}_{pk}(\mathbf{P}^t; \boldsymbol{r})$ by $\boldsymbol{t}'_{\mathsf{POI}}$ for circularly shifting. Therefore, as compared with PCQP, our scheme reduces the computational cost on the shift process without sacrificing security. Finally, the server returns the result set $\boldsymbol{R}$, and the user obtains the result by decrypting it with sk.

5.4 Cross-Like *k*-NN Search

In PCQP, additional query points are selected in the decreasing order by the differences between *H-value*s. However, it may contain duplicated entries of POI in the search result set, especially when a user is near the boundary between groups. This is because that it may be skewed towards a specific group. We, therefore, propose a group-based query point selection method for LPCQP.

At first, the user obtains the querying point Q_0 and candidates of additional query points, $Q_1, \cdots, Q_4$ from the lookup-table as with PCQP. In Fig. 2, these are the 50th cell, the 49th cell, the 51st cell, the 55th cell, and the 61st cell, respectively. Each cell has *H-value* and ID_{table}. Next, query points that belong to the unselected groups are chosen with priority from the candidates set $\boldsymbol{Q}$. If the number of the chosen points is more than one, the point whose *H-value* is farthest is chosen. The *H-value* of the chosen point is added to the result set $\boldsymbol{H}$ and removed from the candidates set $\boldsymbol{Q}$. This process is repeated until the cardinality of the result equals to the number of additional queries n_q. Finally, the user obtains a set of *H-value*s, $\boldsymbol{H}$. This algorithm efficiently selects query points without depending on a specific group, and thus the search accuracy increases. Algorithm 1 indicates a pseudo code of the group-based query point selection method.

5.5 Computational Cost

In this section, we discuss the computational cost on the server in LPCQP. In the following, we argue the computational cost in the plaintext domain since it depends on the encryption scheme the system uses. For example, the Paillier cryptosystem [3] calculates the addition of plaintexts by multiplying the encrypted them, whereas the NTRU cryptosystem [4] can compute them by adding the ciphertexts. For this reason, the terms 'addition' and

Algorithm 1 Group-based query point selection

```
Input: $Q_0, \boldsymbol{Q} = \{Q_1, \cdots, Q_4\}, n_q$
Output: $\boldsymbol{H} = \{H\text{-}value_1, \cdots, H\text{-}value_{n_q}\}$
 1: $\boldsymbol{H}, \boldsymbol{G_0}, \boldsymbol{G_1}, \boldsymbol{G_2}, \boldsymbol{G_3}, \boldsymbol{G_4} \leftarrow \emptyset$
 2: for $i \leftarrow 1$ to 4 do
 3:     $\boldsymbol{G_0} \leftarrow \boldsymbol{G_0} \cup Q_i.\mathsf{ID}_{table}$
 4: end for
 5: if $\boldsymbol{G_0} \cap Q_0.\mathsf{ID}_{table} \neq \emptyset$ then
 6:     $\boldsymbol{G_1} \leftarrow \boldsymbol{G_1} \cup Q_0.\mathsf{ID}_{table}$
 7:     $\boldsymbol{G_0} \leftarrow \boldsymbol{G_0} \setminus Q_0.\mathsf{ID}_{table}$
 8: end if
 9: repeat
10:     $j \leftarrow 0$
11:     while $\boldsymbol{G_j} = \emptyset$ do
12:         $j \leftarrow j + 1$
13:     end while
14:     Initialize $H'$
15:     $k \leftarrow 1$
16:     for $i \leftarrow 1$ to $|\boldsymbol{Q}|$ do
17:         if $\boldsymbol{G_j} \cap Q_i.\mathsf{ID}_{table} \neq \emptyset$ then
18:             $H'[k].H\text{-}value \leftarrow Q_i.H\text{-}value$
19:             $H'[k].distance \leftarrow |Q_0.H\text{-}value - Q_i.H\text{-}value|$
20:             $H'[k].\mathsf{ID}_{table} \leftarrow Q_i.\mathsf{ID}_{table}$
21:             $H'[k].index \leftarrow i$
22:             $k \leftarrow k + 1$
23:         end if
24:     end for
25:     Sort $H'$ in descending order based on $distance$
26:     $\boldsymbol{H} \leftarrow \boldsymbol{H} \cup H'[1].H\text{-}value$
27:     $\boldsymbol{G_{j+1}} \leftarrow \boldsymbol{G_{j+1}} \cup H'[1].\mathsf{ID}_{table}$
28:     $\boldsymbol{G_j} \leftarrow \boldsymbol{G_j} \setminus H'[1].\mathsf{ID}_{table}$
29:     $\boldsymbol{Q} \leftarrow \boldsymbol{Q} \setminus Q_{H'[1].index}$
30: until $|\boldsymbol{H}| = n_q$
```

'multiplication' mean $+_c$ and $\times_c$, respectively, in this section.

In the initialization process of LPCQP, the server divides a POI-table into sub-tables in addition to the initialization process of PCQP. This extra process is negligible since its cost is significantly low as compared with encryption/decryption or calculation of ciphertexts with homomorphism. In the query process, the server first multiplies the encrypted $\boldsymbol{q}_M^m$ by the corresponding sub-table and thus it requires n_p multiplications. Next, the server aggregates sub-tables and thus it requires $n_{p0}(M-1)$ additions. The server, finally, shifts the aggregated table by multiplying $\mathbf{P}'^t$ and thus it requires $k(n_{p0}-1)$ additions and kn_{p0} multiplications.

Here, if a single-bit SHE/FHE is adopted in LPCQP, the encrypted j-th sub-table $\boldsymbol{t}'^j_{\mathsf{POI}}$ in Eq. (14) can be constructed from C_0^j and C_1^j defined as:

$$C_0^j = \mathcal{E}_{pk}(q_j; r'_j) \times_c \mathcal{E}_{pk}(0). \tag{17}$$

$$C_1^j = \mathcal{E}_{pk}(q_j; r'_j) \times_c \mathcal{E}_{pk}(1). \tag{18}$$

By constructing $\boldsymbol{t}'^j_{\mathsf{POI}}$ from C_0^j and C_1^j instead of using Eq. (14), the number of multiplications required in the pre-process becomes $2M$. This technique omits the redundant multiplications in the pre-process and thus it more reduces the computational cost when $M < n_p/2$. Note that this technique can be adopted when the protocol is implemented with a single-bit homomorphic encryption.

Table 1 summarizes the computational costs of PCQP and LPCQP. From Table 1, we can find that the computational cost

Table 1 Comparison of the computational cost on the server between PCQP and LPCQP.

	Pre-process		Shifting process	
Scheme	Add. $(+_c)$	Mult. $(\times_c)$	Add. $(+_c)$	Mult. $(\times_c)$
PCQP	-	-	$k(n_p-1)$	kn_p
PCQP with decomposition	-	n_p	$k(n_p-1)$	kn_p
LPCQP	$n_{p0}(M-1)$	n_p	$k(n_{p0}-1)$	kn_{p0}
LPCQP(optimized)	$n_{p0}(M-1)$	$2M$	$k(n_{p0}-1)$	kn_{p0}

Table 2 Comparison of the communication cost between PCQP and LPCQP.

	Communication cost [bit]	
Scheme	Uplink	Downlink
PCQP	$l_c n_p$	$l_c k$
PCQP with decomposition	$2l_c\sqrt{n_p}$	$l_c k$
LPCQP	$l_c(n_{p0}+M)$	$l_c k$

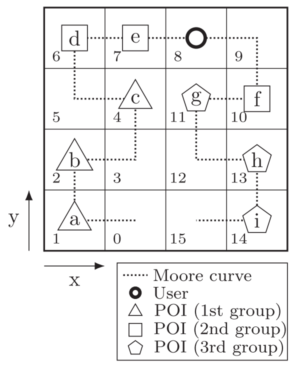
on the shifting process in LPCQP becomes $1/M$ compared with that in PCQP. However, LPCQP requires the additional cost for aggregating sub-tables. Since the pre-process is performed only once, the additional cost for pre-process can be negligible when k is large.

5.6 Communication Cost

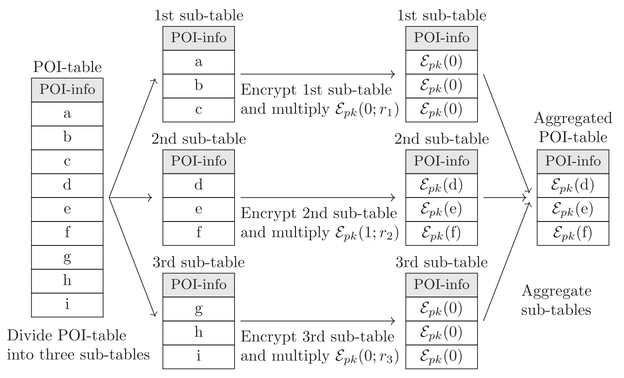
As described in Section 5.3, the size of $\boldsymbol{P}'^t_{0,j}$ is $1/M$ of that of $\boldsymbol{P}^t_{0,j}$. In other words, our scheme can also reduce the communication cost for sending a query. Let l_c denote the bit-length of an encrypted entry of POI information. In PCQP, a user sends $\boldsymbol{P}^t_{0,j}$ as $l_c n_p$ bits to the server, and its size becomes $2l_c\sqrt{n_p}$ bits by adopting the decomposition technique described in Section 4.4. On the other hand, in LPCQP, a user sends $\boldsymbol{P}'^t_{0,j}$ and $\boldsymbol{q}_M^m$, i.e., $l_c(n_{p0}+M)$ bits. Obviously, our scheme can reduce the communication cost from that of the basic PCQP by choosing a certain number M since n_p is generally large. In particular, the communication cost of our scheme becomes $O(\sqrt{n_p})$ as well the technique in Section 4.4 when $M = \sqrt{n_p}$. Incidentally, the user receives a fixed-length (i.e., $l_c k$ bits) result set from the server regardless of the scheme. **Table 2** summarizes the communication costs of PCQP and LPCQP.

5.7 Search Accuracy

In our scheme, n_p POIs are aggregated into a table with n_{p0} POIs. This causes losing $(n_p - n_{p0})$ entries of POI. The more tables are divided, the less accurate the search results get. We demonstrate that in the case of $n_p = 9$ and $M = 3$ by using **Fig. 6**. In Fig. 6 (a), nine POIs (a, b, c, d, e, f, g, h, and i) and a querying user are located on the map. The map is encoded into a one-dimensional space, which is indicated as the dotted line. Here, these POIs are sorted in a POI-table in the ascending order of H-$index$ on the Moore curve and divided into three sub-tables as shown. When the user queries k-NN search with $k = 3$, the user first obtains the index of a sub-table containing the nearest POI. In this example, e is the nearest H-$index$ to the user; hence the user obtains the 2nd sub-table in Fig. 6 (b). Therefore, sub-tables on the server are aggregated into a table with entries of the 2nd sub-table as shown in Fig. 6 (b). Here, the server returns the three encrypted POIs (d, e, and f). Note that c and g are nearer than d to the user on the map as shown in Fig. 6 (a). However,



(a) Nine POIs and a user are located.


POI-table
POI-info
a
b
c
d
e
f
g
h
i
Divide POI-table into three sub-tables
1st sub-table
2nd sub-table
3rd sub-table
Encrypt 1st sub-table and multiply $\mathcal{E}_{pk}(0; r_1)$
Encrypt 2nd sub-table and multiply $\mathcal{E}_{pk}(1; r_2)$
Encrypt 3rd sub-table and multiply $\mathcal{E}_{pk}(0; r_3)$
$\mathcal{E}_{pk}(0)$
$\mathcal{E}_{pk}(\mathrm{d})$
$\mathcal{E}_{pk}(\mathrm{e})$
$\mathcal{E}_{pk}(\mathrm{f})$
Aggregated POI-table
Aggregate sub-tables

(b) POI-table are divided and sub-tables are aggregated into a table with entries of 2nd sub-table.

Fig. 6 Example of losing some POI information by aggregating sub-tables.

it is impossible to obtain them by k-NN search even if k is large since the encrypted entries of c and g become zero in the plaintext domain. For this reason, some entries are out of search results and the number of such entries becomes large when M is large. Therefore, we choose a certain number M for balance between accuracy and complexity. We evaluate the relationship between search accuracy and the number of sub-tables M in Section 6.

5.8 Security Analysis

We discuss the security of the proposed scheme in this section. Lien et al. study privacy issues in k-NN search and show that the correlation attack and the background knowledge attack may violate the user's privacy in k-NN search [1]. The correlation attack utilizes some queries/results obtained through eavesdropping over the network and the background knowledge attack adopts prior knowledge about the user such as age or occupation to infer the user's location. Following the assumption in Ref. [1], we consider the correlation attack and the background knowledge attack in the paper. In addition to the above attacks, we consider the off-line keyword guessing attack [20] and the inference attack [21] as attacks which may also threaten the user's privacy on search over the encrypted data. Both of these attacks utilize a trapdoor which is generated from a search word with the user's private key for search over the encrypted data and it does not reveal any information about the search word. The off-line keyword guessing attack calculates trapdoors for commonly used words to guess the content of encrypted documents. The inference attack identifies a trapdoor for an arbitrary word by combining access patterns with background knowledge about the content of documents. Our objective is to introduce an efficient scheme for privacy-preserving k-NN search; therefore, the encryption strength (e.g., required key length against attacks) is outside a consideration in this study.

The correlation attack: In our scheme, a user sends $\mathcal{E}_{pk}(\boldsymbol{P}'^{t}_{0,j}; \boldsymbol{r})$ and $\mathcal{E}_{pk}(\boldsymbol{q}^{m}_{M}; \boldsymbol{r}')$ instead of $\mathcal{E}_{pk}(\boldsymbol{P}^{t}_{0,j}; \boldsymbol{r})$. They are encrypted with distinct random numbers and the random numbers are scrambled by the user every time the user queries. Therefore, the server cannot correlate queries issued by the user.

The background knowledge attack: Our scheme does not transfer any plaintext data except the shifted location calculated by the user for k-NN search. Obviously, the shifted location cannot be the sensitive information since the shift amount is randomly chosen by the user and it varies every time the user queries. Therefore, no sensitive information about the user is leaked to the server, and thus our scheme is secure against this attack.

The off-line keyword guessing attack: In our scheme, a user never sends a trapdoor of a certain keyword (i.e., user's location) to a server. In addition, location data stored on a server is not sensitive information about the user. Therefore, any adversary or untrusted server cannot guess the user's location by off-line keyword guessing attack.

The inference attack: The inference attack exploits the user's access pattern, e.g., a document which contains the queried keyword. However, no information about the user's access pattern will be leaked to a server and an adversary since $\boldsymbol{P}'^{t}_{0,j}$ and $\boldsymbol{q}^{m}_{M}$ are encrypted with distinct random numbers and the random numbers are scrambled by the user every time the user queries. Therefore, our scheme is robust to this attack.

Next, we discuss the requirements for a homomorphic encryption in LPCQP. As shown in Table 1, our scheme requires some operations in the encryption domain. Contrary to this, one ciphertext is added to an another one $M + n_{p0} - 1$ times and it is multiplied by an another one twice in total. Hence, in LPCQP, a homomorphic encryption scheme which guarantees $M + n_{p0} - 1$ additions and two multiplications is required.

6. Results

We evaluate the k-NN search of our scheme in terms of the computational cost, the communication cost, and the search accuracy. Our scheme is implemented in C language and Java language with Scarab library[*1] and performed on a laptop computer, which has Intel Core i5 1.8 GHz processor and 4 GB memory. Scarab library is based on the work by Gentry [6], and Smart and Vercauteren [22]. The library supports a single-bit SHE/FHE using large integers. Here, our scheme requires only two multiplications for one encrypted data as mentioned in Section 5.8. Since the noise of an SHE is mainly caused by multiplication, it is acceptable in LPCQP. We, therefore, do not use the bootstrapping scheme for the fast computation. In our experiments, the key lengths of a public-key and a private-key are 288 bytes and 32 bytes, respectively. In addition, one bit is encrypted as 16 bytes data.

We use three POI datasets: a uniform dataset, a Gaussian dataset, and a real-world dataset. POIs are uniformly distributed

[*1] http://hcrypt.com/scarab-library/

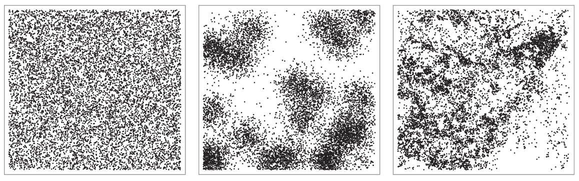


Fig. 7 Distribution of POIs in three different datasets.

in the uniform dataset in a square map and normally distributed from randomly chosen twenty points in a square map with $\sigma = 0.06$ in the Gaussian dataset, respectively. On the other hand, POIs in the real-world dataset are obtained from OpenStreetMap[*2] in a 10 km × 10 km area surrounding Tokyo, Japan. OpenStreetMap provides POI data in the world as extensible markup language (XML). The XML data contains IDs, names, latitudes and longitudes, and types of POIs. We extract the latitudes and longitudes as the positions of POIs from the data for the real-world dataset. Note that we do not consider the other attributes in this experiment. Each dataset contains 10,000 POIs. **Figure 7** indicates the distribution of POIs of each dataset. The information about one POI is represented as 64 bytes data.

We experimentally evaluate the performance of our k-NN search scheme by comparing PCQP and the dual curve query resolution (DCQR) [12] for different values of M and k and the three datasets. DCQR is a transformation-based privacy-preserving k-NN search with a TTP. The reason why we chose DCQR for comparison is because this method is a well-studied k-NN search scheme with a space-filling curve. M and k vary from 10 to 500 and from 1 to 50, respectively. Khoshgozaran et al. suggest that the average number of POIs which are assigned to the same *H-value* should be two or less for practical use [12]. Thus, we set the curve order of the Moore curve to seven that satisfies the above condition against our datasets. In each experiment, queries are issued at randomly chosen 1,000 locations on the map and the results are averaged.

6.1 Computational Cost

Table 3 indicates the computational time of each scheme and **Fig. 8** indicates the total computational time for a query versus k. Our scheme without redundant multiplications described in Section 5.5 is represented as optimized LPCQP in Table 3 and Fig. 8. Note that our scheme returns at most n_{p0} results even if a user requires more than k POIs; therefore, the computational time of our scheme is not indicated in Fig. 8 when $M = 500$ and $k > 20$. From Table 3, we can see that our scheme requires heavy computation on the pre-process, i.e., aggregating sub-tables in the encryption domain, whereas the shifting process is significantly reduced. For this reason, in particular when k is small, the total computational time does not exactly become $1/M$ although the computational time on the shifting process becomes $1/M$. However, this cost can be reduced by 96% or more by omitting redundant multiplications as shown in Table 3. That is, omitting redundant multiplications makes LPCQP more lightweight, and thus the total computational time can be reduced even if k is small, e.g., $k = 1$. From Fig. 8, we

Table 3 Computational time for k-NN search on the server.

Scheme	M	Computational time [ms]	
		Pre-process	Shifting process (per k)
PCQP	-	-	3.2×10^4
PCQP with decomposition	-	6.7×10	3.2×10^4
LPCQP	10	3.0×10^4	3.2×10^3
LPCQP	50	3.1×10^4	5.9×10^2
LPCQP	100	3.1×10^4	3.0×10^2
LPCQP	200	3.1×10^4	1.5×10^2
LPCQP	500	3.1×10^4	6.0×10
LPCQP (optimized)	10	1.1×10^3	3.2×10^3
LPCQP (optimized)	50	1.0×10^3	5.9×10^2
LPCQP (optimized)	100	9.7×10^2	3.0×10^2
LPCQP (optimized)	200	1.0×10^3	1.5×10^2
LPCQP (optimized)	500	9.5×10^2	6.0×10

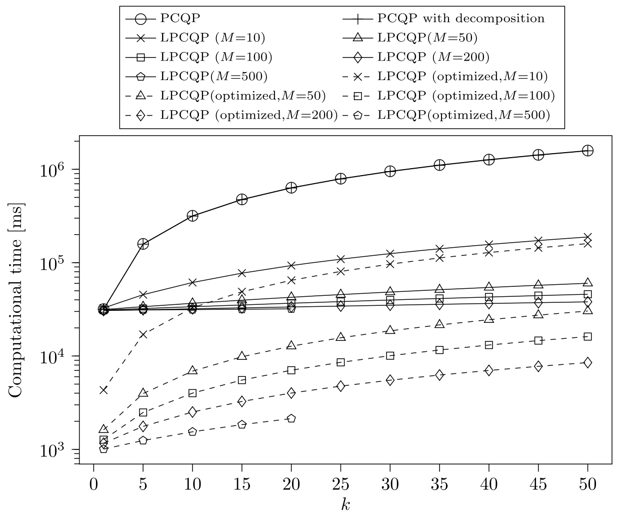


Fig. 8 Total computational time on the server versus k.

can see that our scheme in any M significantly reduces the total computational time. In particular, optimized LPCQP can reduce 99% of that required in the conventional scheme when $M \geq 100$ and k is large. Even if $M = 10$, our scheme can reduce 90% of that. These results indicate that the reduction rate follows the theoretical one. However, the total computational time of LPCQP without applying the optimization technique is larger than that of PCQP when $k = 1$. As mentioned in Section 5.5, the optimization technique can be applied to LPCQP with a single-bit SHE/FHE. Hence, our scheme with a multi-bit SHE/FHE cannot reduce the computational cost when $k = 1$ because of the heavy cost in the pre-process.

6.2 Communication Cost

Table 4 indicates the communication cost for a query with two additional queries ($n_q = 2$). From Table 4, we can see that our scheme in any M reduces the communication cost. In particular, the communication cost for sending a query can be reduced by 98% as with the conventional scheme with the decomposition technique when $M = 100$ (i.e., $M = \sqrt{n_p}$).

We denote by the total communication cost the sum of the communication cost for sending a query with two additional queries and that for receiving its results. **Figure 9** indicates the decreasing rate of the total communication cost of LPCQP to that of

[*2] http://planet.openstreetmap.org/

Table 4 Communication cost for a k-NN query.

Scheme	M	Communication cost [bit]	
		Uplink	Downlink (per k)
PCQP	-	1.6×10^5	8.2×10^3
PCQP with decomposition	-	3.2×10^3	8.2×10^3
LPCQP	10	1.6×10^4	8.2×10^3
LPCQP	50	4.0×10^3	8.2×10^3
LPCQP	100	3.2×10^3	8.2×10^3
LPCQP	200	4.0×10^3	8.2×10^3
LPCQP	500	8.3×10^3	8.2×10^3

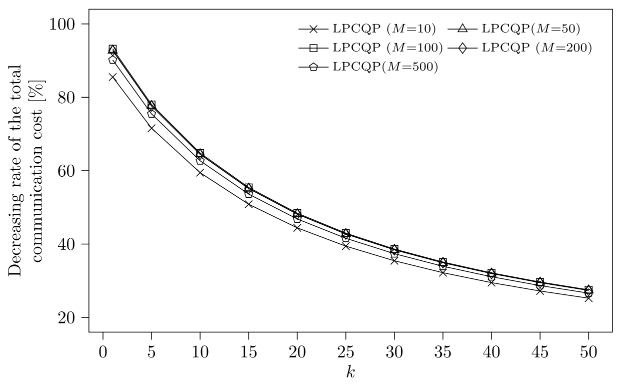


Fig. 9 The decreasing ratio of the total communication cost of LPCQP to that of PCQP.

Table 5 Computational cost for generating a k-NN query.

Scheme	M	Computational time [ms]
PCQP	-	9.2×10^3
PCQP with decomposition	-	1.9×10^2
LPCQP	10	9.3×10^2
LPCQP	50	2.4×10^2
LPCQP	100	1.9×10^2
LPCQP	200	2.3×10^2
LPCQP	500	4.5×10^2

PCQP. As shown in Fig. 9, the decreasing rate of the total communication cost decreases when k gets larger. There exist two reasons for this result. The first reason is that the communication cost of results increases proportionally to k. The second one is that the data size of POI information is larger than that of an element of a shift matrix, i.e., '0' or '1'. Thus, the decreasing rate further increases when k or the size of POI information gets smaller. In addition, large n_p makes the decreasing rate high because the size of a shift matrix becomes large.

Here, reducing the calculation cost for encrypting a query is required especially for a mobile application. **Table 5** indicates the required time for encrypting a shift matrix and a group vector. As shown in Table 5, the computational time on the user is also reduced because the size of a shift matrix becomes $1/M$. In this experiment, the computational cost for generating a query is reduced by 98% when $M = 100$. From this result, LPCQP reduces not only the communication cost but also the computational cost on the user side.

6.3 Search Accuracy

We evaluate two metrics, accuracy rate and displacement, to evaluate the quality of k-NN search results referring to [12]. The accuracy rate and the displacement are defined as how exactly nearby POIs are retrieved and how close the ground-truth result set and the query result set, respectively. Let $\boldsymbol{G} = \{o_1^G, \cdots, o_k^G\}$ and $\boldsymbol{R} = \{o_1^R, \cdots, o_{k'}^R\}$ denote the ground-truth result set and the query result set obtained by k-NN search, respectively. The accuracy rate of k-NN search, r_{acc}, is defined as

$$r_{acc} = \frac{|\boldsymbol{R} \cap \boldsymbol{G}|}{|\boldsymbol{G}|}, \tag{19}$$

where $|\boldsymbol{R}|$ denotes the cardinality of the set $\boldsymbol{R}$. Let $\boldsymbol{R}' = \{o_1^{R'}, \cdots, o_k^{R'}\}$ denote the set of k-nearest POIs of $\boldsymbol{R}$ from the queried point. Then, the displacement of k-NN search, l_{dis}, is defined as

$$l_{dis} = \frac{1}{k}\left(\sum_{i=1}^{k} \left\|q - o_i^{R'}\right\| - \sum_{i=1}^{k} \left\|q - o_i^{G}\right\|\right), \tag{20}$$

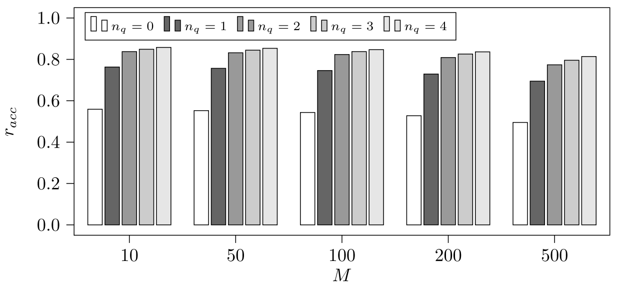
where $\|q - o_i^{R'}\|$ denotes the Euclidean distance between the fixed query point q and the location of a POI $o_i^{R'}$. For instance, $r_{acc} = 0.5$ means that the result set contains $k/2$ near POIs and $k/2$ far POIs from the user whereas $l_{dis} = 10$ [m] means that each POI in the result set is farther from the user than each of k-nearest POIs by 10 m on an average. Here, the conventional scheme returns at least k POIs in total for a query, and thus $k \le k'$. In contrast, the condition above is not always satisfied in LPCQP because our scheme may return less than k POIs when M is large. For this reason, the displacement cannot be calculated in LPCQP when $M = 500$ and $k > 20$. Our objective is to keep high accuracy rate and low displacement of PCQP while reducing the computational cost.

We also evaluate two query point selection methods in cross-like search. The one is the *H-value*-based selection technique used in PCQP (mentioned in Section 4.3) and the other is the group-based selection technique proposed in Section 5.4. The former one is represented as LPCQP and the latter one is represented as LPCQP+ in the following. Note that the computational cost on the server and the communication cost of these schemes are same.

6.3.1 Parameter Tuning

We first tune the two parameters n_q and M to find optimal values for high-accurate and low-complexity k-NN search. In this setting, we use the real-world dataset.

Figure 10 (a) and (b) represent the accuracy rate and the displacement of LPCQP with varying the number of additional queries n_q when $k = 20$, respectively. From Fig. 10, we can see that the more additional queries are issued in any cases, the more accurate the search results get. Obviously, the quality of k-NN search is further enhanced by issuing more additional queries; however, the computational cost and the communication cost are proportionally increased as mentioned in Section 4.3. We notice that there is little difference between the quality of search with two additional queries and that with four additional queries, and hence we set the number of additional queries n_q to two in the following experiments.


(a) Accuracy rate.

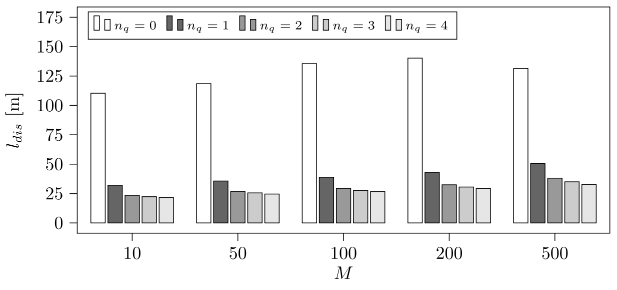
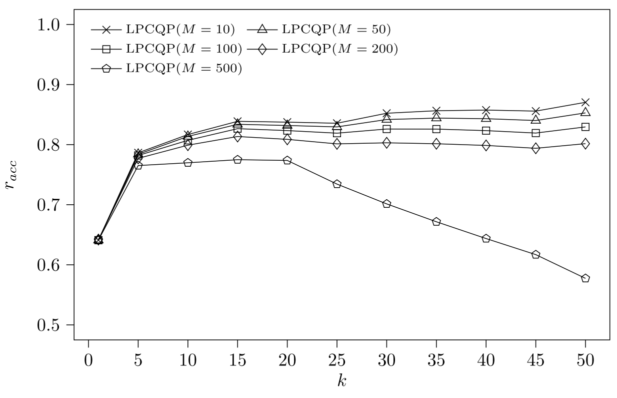

(b) Displacement.

Fig. 10 Accuracy rate and displacement versus n_q.


(a) Accuracy rate.

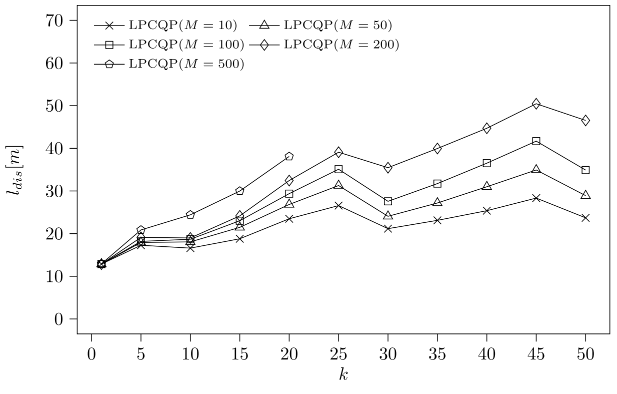

(b) Displacement.

Fig. 11 Accuracy rate and displacement versus M.

Figure 11 (a) and (b) represent the accuracy rate and the displacement of LPCQP versus k, respectively. As shown in Fig. 11, the quality of k-NN search gets worse as M gets larger. The reason is that there is too much loss from some POI entries near the user from aggregating sub-tables as mentioned in Section 5.7. In particular, the accuracy rate is no longer exact when $M = 500$ and k is large because LPCQP can return up to n_{p0} POIs to the user. That is, there is a trade-off between the quality of results

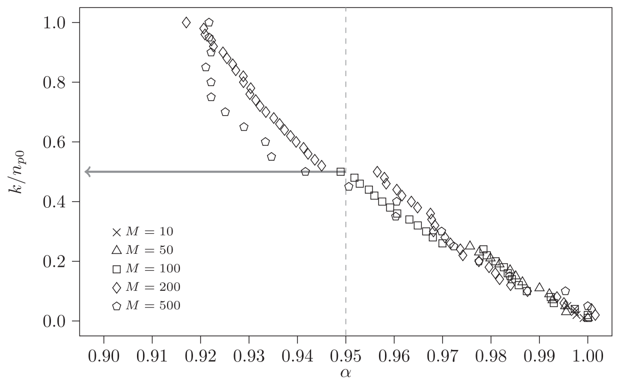


Fig. 12 Ratio k/n_{p0} versus α.

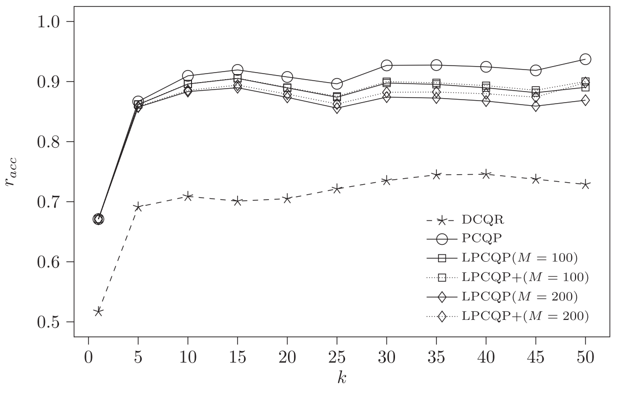
and the computational cost on the server. Additionally, we can see that there are variations in both metrics when the window size is changed. This is because that the querying points of additional queries are changed according to k in cross-like search. Although it is difficult to choose the optimal M, we find the M that most reduces the computational cost with required quality of k-NN search. For the given accuracy rate of our proposed scheme, r_{acc}^{LPCQP}, and that of PCQP, r_{acc}^{PCQP}, we define the ratio α as:

$$\alpha = \frac{r_{acc}^{\mathsf{LPCQP}}}{r_{acc}^{\mathsf{PCQP}}}. \tag{21}$$

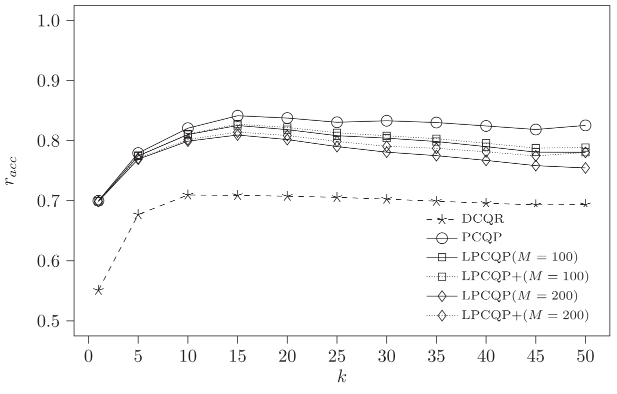
As the definition above, $\alpha = 0$ means that the proposed scheme is not accurate at all while $\alpha = 1$ means that ours achieves the same accuracy as that of the conventional one achieves. In order to provide high-accurate and low-complexity k-NN search, we find the maximum M that achieves $\alpha \geq 0.95$. Note that the optimal M varies depending on n_p and k. Here, we use the ratio k/n_{p0} for deciding the optimal M since the search accuracy relies on k against n_p. The server can adjust this ratio by choosing a certain M under the condition that k is specified by the user and n_p is pre-defined for the service. **Figure 12** represents the ratio k/n_{p0} versus α. As shown in Fig. 12, there is a correlation between k/n_{p0} and α, and thus the server can control α against pre-defined n_p and k. From this result, k/n_{p0} should be 0.5 or less at least to achieve $\alpha \geq 0.95$. In other words, the decreasing accuracy rate of LPCQP is suppressed to 5% or below when $n_{p0} \geq 2k$. Therefore, we use $M = 100$ that is the maximum M satisfies the above condition when $n_p = 10{,}000$ and $k \leq 50$ in the following simulation.

6.3.2 Accuracy Rate

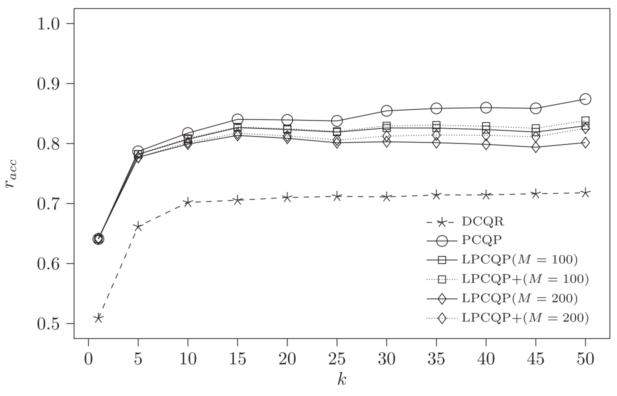
In this section, we compare our scheme where $M = 100, 200$ with PCQP and DCQR in terms of the accuracy rate. **Figure 13** indicates the accuracy rate versus k for the three datasets. From Fig. 13, we can find that the ratio α is kept to 0.95 or above regardless of the dataset even if k is large when $M = 100$. Note that the accuracy rate of LPCQP is still higher than that of DCQR although that is lower than that of PCQP. From this result, we can say that LPCQP keeps high accuracy of PCQP while reducing the computational cost of that. Furthermore, LPCQP+ achieves higher accuracy rate than LPCQP for all datasets. Especially for the uniform and Gaussian datasets, LPCQP+ achieves $\alpha \geq 0.95$ when $M = 200$ whereas α of LPCQP is 0.91-0.93 then. From this result, we can say that the group-based query point selection method improves the quality of k-NN search in LPCQP.



(a) Uniform.

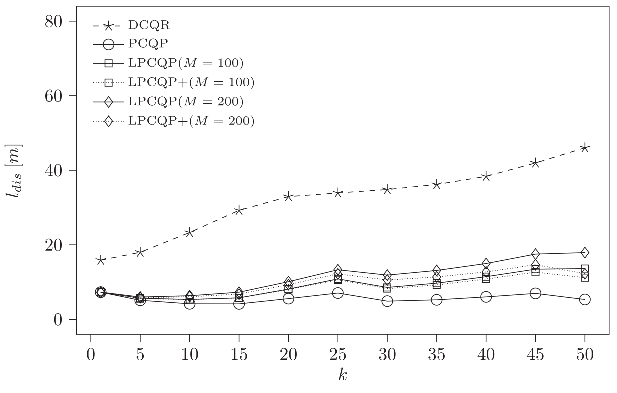


(b) Gaussian.

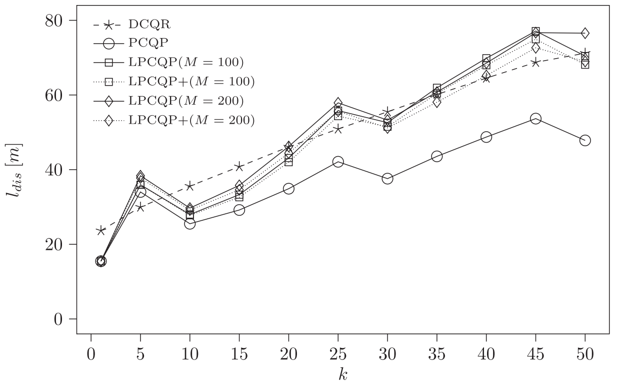


(c) Real-world.

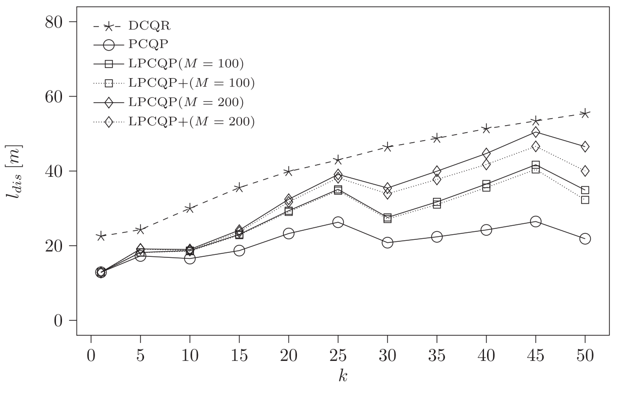
Fig. 13 Search accuracy r_{acc} versus k.



(a) Uniform.



(b) Gaussian.



(c) Real-world.

Fig. 14 Displacement l_{dis} versus k.

6.3.3 Displacement

We also compare our scheme with the related works in terms of the displacement. **Figure 14** indicates the displacements versus k for the three datasets. From Fig. 14, we can find that the displacement of our scheme increases when M gets larger. That is, the displacement is also gets worse in LPCQP. Especially for the Gaussian dataset, the displacement of LPCQP is larger than that of DCQR whereas those of LPCQP are still smaller than that of DCQR regardless of k for the uniform and real-world datasets. This is because that the cross-like k-NN search mentioned in Section 4.3 obtains POIs far from the user to reduce duplicated POIs. For this reason, LPCQP returns the information about the distant POIs as well as near POIs for the skewed POI map. Algorithm 1 also decreases the displacement of LPCQP. Especially for the uniform dataset, when $k = 50$, the displacements are reduced by 18% and 32% when $M = 100, 200$, respectively.

From these results, our proposed scheme can reduce both the computational and the communication costs while keeping the high quality of k-NN search in PCQP. In our experiments, allowing a 5% reduction in search accuracy, the computational costs on the server and the user are reduced by up to 99% and 98% respectively and the communication cost is reduced by at least 27%. As a result, LPCQP provides highly-accurate and low-complexity privacy preserving k-NN search for an LBS.

7. Conclusion

In this paper, we have proposed a low-complexity privacy-preserving k-NN search scheme by dividing and aggregating a POI-table with additive and multiplicative homomorphisms. Our scheme is as secure as the conventional scheme because all processes on the server are performed in the encryption domain. We evaluate the computational time, the communication cost, and the search accuracy of our scheme with three POI datasets; a uniform dataset, a Gaussian dataset, and a real-world dataset. The results show that our scheme reduces the computational cost and the communication cost of k-NN search while maintaining high search accuracy without sacrificing security whereas the conventional scheme with additive and multiplicative homomorphisms can reduce only the communication cost. Furthermore, the computational cost is reduced more by omitting the redundant multiplications in the pre-process. In our environment, the computational cost on the pre-process is reduced by 96% or more regardless of M by applying the optimization technique, and then the total computational costs on the server and the user are reduced by up to 99% and 98% respectively and the total communication cost is reduced by at least 27% by allowing a 5% reduction in search accuracy when $M = 100$.

Acknowledgments This work is partly supported by the Grant in Aid for Scientific Research (No.26420369) from Ministry of Education, Sport, Science and Technology, Japan.

References

[1] Lien, I.-T., Lin, Y.-H., Shieh, J.-R. and Wu, J.-L.: A Novel Privacy Preserving Location-Based Service Protocol With Secret Circular Shift for k-NN Search, *IEEE Trans. Information Forensics and Security*, Vol.8, No.6, pp.863–873 (2013).

[2] Moore, E.H.: On Certain Crinkly Curves, *Transactions of the American Mathematical Society*, Vol.1, No.1, pp.72–90 (1900).

[3] Pascal, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, *Advances in Cryptology – EUROCRYPT '99*, Lecture Notes in Computer Science, Vol.1592, pp.223–238, Springer Berlin Heidelberg (1999).

[4] Hoffstein, J., Pipher, J. and Silverman, J.: NTRU: A ring-based public key cryptosystem, *Algorithmic Number Theory*, Lecture Notes in Computer Science, Vol.1423, pp.267–288, Springer Berlin Heidelberg (1998).

[5] López-Alt, A., Tromer, E. and Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, *Proc. 44th Annual ACM Symposium on Theory of Computing*, pp.1219–1234, ACM (2012).

[6] Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices, *Proc. the Forty-first Annual ACM Symposium on Theory of Computing* (*STOC' 09*), STOC '09, pp.169–178, ACM (2009).

[7] Um, J.-H., Kim, H.-D. and Chang, J.-W.: An Advanced Cloaking Algorithm Using Hilbert Curves for Anonymous Location Based Service, *Proc. IEEE 2nd International Conference on Social Computing* (*SocialCom 2010*), pp.1093–1098 (2010).

[8] Tan, K., Lin, Y. and Mouratidis, K.: Spatial Cloaking Revisited: Distinguishing Information Leakage from Anonymity, *Advances in Spatial and Temporal Databases*, Lecture Notes in Computer Science, Vol.5644, pp.117–134, Springer Berlin Heidelberg (2009).

[9] Kalnis, P., Ghinita, G., Mouratidis, K. and Papadias, D.: Preventing Location-Based Identity Inference in Anonymous Spatial Queries, *IEEE Transactions on Knowledge and Data Engineering*, Vol.19, No.12, pp.1719–1733 (2007).

[10] Chow, C.-Y., Mokbel, M.F. and Aref, W.G.: Casper*: Query Processing for Location Services Without Compromising Privacy, *ACM Transactions on Database Systems*, Vol.34, No.4, pp.24:1–24:48 (2009).

[11] Hu, H. and Xu, J.: Non-Exposure Location Anonymity, *Proc. IEEE 25th International Conference on Data Engineering 2009* (*ICDE '09*), pp.1120–1131 (2009).

[12] Khoshgozaran, A., Shirani-Mehr, H. and Shahabi, C.: Blind evaluation of location based queries using space transformation to preserve location privacy, *GeoInformatica*, Vol.17, No.4, pp.599–634 (2013).

[13] Puttaswamy, K., Wang, S., Steinbauer, T., Agrawal, D., El Abbadi, A., Kruegel, C. and Zhao, B.: Preserving Location Privacy in Geosocial Applications, *IEEE Trans. Mobile Computing*, Vol.13, No.1, pp.159–173 (2014).

[14] Yiu, M.L., Jensen, C.S., Huang, X. and Lu, H.: SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services, *Proc. IEEE 24th International Conference on Data Engineering 2008* (*ICDE '08*), pp.366–375 (2008).

[15] Papadopoulos, S., Bakiras, S. and Papadias, D.: Nearest Neighbor Search with Strong Location Privacy, *Proc. the VLDB Endowment*, Vol.3, No.1–2, pp.619–629 (2010).

[16] Khoshgozaran, A., Shahabi, C. and Shirani-Mehr, H.: Location Privacy: Going Beyond K-anonymity, Cloaking and Anonymizers, *Knowledge and Information Systems*, Vol.26, No.3, pp.435–465 (2011).

[17] Khoshgozaran, A. and Shahabi, C.: Private Information Retrieval Techniques for Enabling Location Privacy in Location-Based Services, *Privacy in Location-Based Applications*, Lecture Notes in Computer Science, Vol.5599, pp.59–83, Springer Berlin Heidelberg (2009).

[18] Gentry, C. and Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme, *Advances in Cryptology – EUROCRYPT 2011*, Lecture Notes in Computer Science, Vol.6632, pp.129–148, Springer Berlin Heidelberg (2011).

[19] Stehlé, D. and Steinfeld, R.: Faster Fully Homomorphic Encryption, *Advances in Cryptology – ASIACRYPT 2010*, Lecture Notes in Computer Science, Vol.6477, pp.377–394, Springer Berlin Heidelberg (2010).

[20] Byun, J., Rhee, H., Park, H.-A. and Lee, D.: Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data, *Secure Data Management*, Lecture Notes in Computer Science, Vol.4165, pp.75–83, Springer Berlin Heidelberg (2006).

[21] Islam, M.S., Kuzu, M. and Kantarcioglu, M.: Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation, *Proc. 19th Annual Network and Distributed System Security Symposium* (*NDSS '12*) (2012).

[22] Smart, N. and Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes, *Proc. 13th International Conference on Practice and Theory in Public Key Cryptography* (*PCK 2010*), Lecture Notes in Computer Science, Vol.6056, pp.420–443, Springer Berlin Heidelberg (2010).

Yasuhito Utsunomiya was born in Japan in 1991. He received his B.E. degree from Tokyo University of Technology in 2013 and M.S. degree from Keio University in 2015. His research interest is security and privacy.

Kentaroh Toyoda was born in Tokyo, Japan in 1988. He received his M.S. degree from Keio University in 2013. He is a Ph.D. student at Keio University and a researcher at security research center in Kanagawa Institute of Technology. His research interest is security & privacy for RFID, IoT, and CPS (Cyber Physical Systems). He was a research associate at Keio University from 2013 to 2015. He received a Telecom System Technology Encouragement Award in 2015 and IEICE communication society encouragement awards in 2012 and 2015. He is a member of IEEE, IPSJ, and IEICE.

Iwao Sasase was born in Osaka, Japan in 1956. He received his B.E., M.E., and D. Eng. degrees in Electrical Engineering from Keio University, Yokohama, Japan, in 1979, 1981 and 1984, respectively. From 1984 to 1986, he was a Post Doctoral Fellow and Lecturer of Electrical Engineering at University of Ottawa, ON, Canada. He is currently a Professor of Information and Computer Science at Keio University, Yokohama, Japan. His research interests include modulation and coding, broadband mobile and wireless communications, optical communications, communication networks and information theory. He has authored more than 278 journal papers and 419 international conference papers. He granted 43 Ph.D. degrees to his students in the above field. Dr. Sasase received the 1984 IEEE Communications Society (ComSoc) Student Paper Award (Region 10), 1986 Inoue Memorial Young Engineer Award, 1988 Hiroshi Ando Memorial Young Engineer Award, 1988 Shinohara Memorial Young Engineer Award, 1996 Institute of Electronics, Information, and Communication Engineers (IEICE) of Japan Switching System Technical Group Best Paper Award, and WPMC2008 Best Paper Award. He is now serving as Vice President of IEICE (2014–). He was Board of Governors Member-at-Large (2010–2012), Japan Chapter Chair (2011–2012), Director of the Asia Pacific Region (2004–2005), Chair of the Satellite and Space Communications Technical Committee (2000–2002) of IEEE ComSoc., President (2013–2014), Vice President (2004–2006), Chair of the Network System Technical Committee (2004–2006), Chair of the Communication System Technical Committee (2002–2004) of IEICE Communications Society, Director of the Society of Information Theory and Its Applications in Japan (2001–2002). He is a fellow of IEICE, senior member of IEEE, member of IPSJ, respectively.