近傍ロケーションテーブルの分割によりサーバ側の計算量を低減するプライバシ保護 k-Points-of-Interest 検索手法

宇都宮 靖人 †　　豊田 健太郎 †　　笹瀬 巌 †

† 慶應義塾大学理工学部情報工学科
223-8522 神奈川県横浜市港北区日吉 3-14-1
utsunomiya@sasase.ics.keio.ac.jp

あらまし 近年，ユーザの所在地をサーバに明かすことなく，近傍のロケーション（POIs：Points of Interest）を検索するサービスが注目されている．Lien らは，準同型暗号を用いることでユーザの位置をサーバから秘匿し，POI を検索可能な方式を提案しているが，サーバが所有する全ての POI に対して行列演算を行うため，サーバの計算量が増大する問題がある．そこで本論文では，POI テーブルを分割し，準同型性を用いて統合することで，計算量を低減する POI 検索方式を提案する．特性評価により，提案方式は，従来方式の安全性及び検索精度を保ちつつ，サーバの計算量を大幅に削減できることを示す．

Low-Complexity Privacy-Preserving k-POIs Search Scheme by Dividing and Aggregating POI-Table

Yasuhito Utsunomiya†　　Kentaroh Toyoda†　　Iwao Sasase†

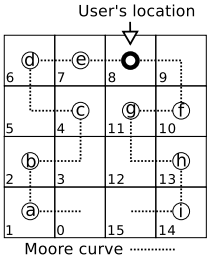
†Department of Information and Computer Science, Keio University.
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa, 223-8522 JAPAN
utsunomiya@sasase.ics.keio.ac.jp

Abstract In recent years, privacy-preserving Location-Based Services (LBSs) are getting much attention. In a privacy-preserving LBS, a user searches k-Points-of-Interest (k-POIs) in the vicinity of it without revealing the user's location to a LBS server. Lien *et al.* propose a privacy-preserving k-POIs search scheme that applies homomorphic encryption. However, it requires heavy multiplications against all POI-table entries and this causes intolerant burden on a server. To tackle this problem, in this paper, we propose a low-complexity k-POIs search scheme by dividing and aggregating a POI-table. Our scheme divides a POI-table into some sub-tables and hides the calculated sub-table by aggregating them with homomorphism. The experimental results show that our proposed scheme reduces computational complexity on the server without sacrificing both the security and accuracy.

1 Introduction

A Location-Based Service (LBS) becomes one of major services in recent mobile communication trends. A LBS is to find k-Points-of-Interest (k-POIs), *e.g.* cafes or drag stores, in the vicinity of a user. A LBS consists of two entities, a user who wants to find k-POIs and a LBS server which returns k-POIs. A LBS server manages POI information such as the name or the location of POI. For instance, nine POIs are located on the map in Fig. 1(a). These POI information are stored on the LBS server. When a user is in the cell numbered as eight in Fig. 1(a) and wants to find POI near the user, he/she informs his/her current location, *i.e.* eight, to the LBS server. Then, the

server knows where the user is now and thus it can return information regarding POI near the user, e or g. For searching POI near a user, a LBS server requires the user's location information. However, the information a LBS requires is sensitive and thus it might harm the privacy of the user. Therefore, a privacy-preserving search scheme is required in LBSs and many schemes have been recently proposed. They are categorised into three types; cloaking-based [1, 2], transformation-based [3, 4], and private-information-retrieval (PIR) based methods [5, 6]. The cloaking-based methods try to hide a user's location by generating a cloaking region that a LBS server cannot distinguish him/her from other $K-1$ users. The transformation-based methods utilize a trusted third party (TTP) to protect a user's location. The PIR-based methods are based on the theory of PIR. PIR enables a user to retrieve POI from a database on a server without revealing which item is retrieved. The cloaking-based and the transformation-based methods achieve high-accuracy of k-POIs search, but they require a TTP to hide a user's location in general. In contrast, the PIR-based methods achieve high-level security without a TTP, but they require high computational cost for searching POI. The computational cost on a LBS server can be dispersed toward many high-performance computers on the clouds, whereas a TTP continues to require costs to protect sensitive information against various attacks. For this reason, we pay attention to Private Circular Query Protocol (PCQP) which is a PIR-based scheme proposed by Lien *et al.* [5]. They adopt Moore curve [7] and Paillier cryptosystem to securely rotate the POI-table to preserve a user's location. No adversary (even a LBS server) knows a user's location in PCQP because the user's query and the search results are encrypted, *i.e.,* it is robust to so-called the correlation attack and the background knowledge attack. However, it takes high computational complexity since it involves a large number of multiplications against a matrix on the server side and this must be reduced to realise a realtime LBS.



(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 Moore curve (a), POI-table (b), and lookup-table (c).

In this paper, we propose a low-complexity privacy-preserving k-POIs search scheme. We notice that PCQP multiplies an encrypted matrix by the entire POI-table for circularly shifting although most POIs are not required in 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 PCQP in order to obtain partial entries of a POI-table. This reduces the computational complexity in the matrix multiplication on the server. In addition, we use a homomorphic cryptosystem with the properties of additive and multiplicative homomorhism to perform the above process in the encryption domain. As a result, our scheme achieves low-complexity and high-accuracy of k-POIs search without losing security.

The rest of this paper is constructed as follows. Section 2 describes conventional scheme. Comprehensive description regarding the proposed scheme is described in Section 3. Simulation result and evaluation are discussed in Section 4. Finally we conclude our discussion in Section 5.

2 Conventional Scheme

PCQP is a privacy-preserving k-POIs search scheme. A user's location is hidden by circularly shifting the entries of a POI-table through k-POIs search in PCQP . The shifted amount is randomly chosen by a user whenever he/she queries to a LBS. Then, the user requires k-POIs of the

shifted location. In order to securely shift the POI-table, PCQP uses additive homomorphism. Additive homomorphism offers that a LBS server can directly compute the addition of two messages with them encrypted. Lien *et al.* use Paillier cryptosystem as a building tool of additive homomorphism. Shifting is securely performed in the encryption domain, and thus no user's location is revealed to the server. The architecture of PCQP is described below.

2.1 Initialization Process

In the initialization process, a LBS server constructs Moore curve. 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 and start cell and end cell are adjoined. The number on the corner of each cell represents an index in the curve, called *H-value*. Thus, the two-dimensional coordinates of POI can be converted into the one-dimensional one, *i.e. H-value*. *H-value* is efficient as a index for k-POIs search for the following reasons: (i) calculating the distance between *H-values* 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 that. 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 1(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 a user belongs to from the Moore curve and then what his/her *H-index* is from the lookup-table. Meanwhile, users generates their key pair, *i.e.* a public-key and a private-key, of Paillier cryptosystem and send their public-key to the server.

2.2 Query Process

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

$$\boldsymbol{P}^t = [P_{i,j}]_{0 \le i,j \le n_p - 1},$$
$$P_{i,j} = \begin{cases} 1 & (j = (i + n_p - t) \bmod n_p) \\ 0 & (otherwise) \end{cases}, \quad (1)$$

where n_p denotes the number of all entries of a POI-table. In order to hide the amount of shifts, the user encrypts $\boldsymbol{P}^t$ before sending it. However, it requires high communication cost for sending encrypted whole $\boldsymbol{P}^t$. Here, $(i+1)$-th row of $\boldsymbol{P}^t$ can be obtained by circularly shifting the element of i-th row by one. Therefore, the user encrypts only the first row of $\boldsymbol{P}^t$ and sends it. Let $\mathcal{E}_{k_{enc}}(m, r)$ and $\mathcal{D}_{k_{dec}}(\mathcal{E}_{k_{dec}}(m, r))$ denote the encryption of a message m with an encryption key k_{enc} and a pseudo-random number r and the decryption of that with a decryption key k_{dec}, respectively. For a given set of pseudo-random numbers $\boldsymbol{r} = \{r_0, \cdots, r_{n_p-1}\}$ and his/her public-key k_{pub}, the encrypted first row of $\boldsymbol{P}^t$ is represented as

$$\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t_{0,j}, \boldsymbol{r}) = \left[\mathcal{E}_{k_{pub}}(P_{0,j}, r_j)\right]_{0 \le j \le n_p - 1}, \quad (2)$$

where $\boldsymbol{P}^t_{0,j}$ denotes the first row of $\boldsymbol{P}^t$. After encrypting $\boldsymbol{P}^t_{0,j}$, the user sends $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t_{0,j}, \boldsymbol{r})$ and 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/her current location as a key. The server receives $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t_{0,j}, \boldsymbol{r})$ and obtains the encrypted $\boldsymbol{P}^t$, *i.e.* $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t, \boldsymbol{r})$, by circularly shifting the element of $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t_{0,j}, \boldsymbol{r})$ by one consecutively. Then, the server multiplies $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t, \boldsymbol{r})$ by a POI-table. By multiplying $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t, \boldsymbol{r})$ with the homomorphic properties

of Paillier cryptosystem, *i.e.* homomorphic addition of two ciphertexts and multiplication of a ciphertext and a plaintext, the POI-table is shifted by t with it encrypted. That is, for given entries of POIs $\boldsymbol{I} = \{I_1, \cdots, I_{n_p}\}$, the i-th entry of the shifted POI-table I_i^t is defined as:

$$I_i^t \;=\; \prod_{j=0}^{n_p-1} \mathcal{E}_{k_{pub}}(P_{i-1,j},\, r_j)^{I_{j+1}}. \qquad (3)$$

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

$$\mathcal{D}_{k_{pri}}(I_i^t) \;=\; I_{((i+n_p-t) \bmod n_p)}. \qquad (4)$$

Therefore, only the user can decrypt shifted POIs and obtain 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_i^t, is encrypted by user's public-key as shown in Eq. (3). After receiving encrypted k results from the server, the user decrypts them by using its private-key.

2.3 Drawback of PCQP

Although PCQP achieves high-level security and high-accuracy, it takes high computational complexity in the matrix multiplication on a LBS server. For a given k-POIs query, the server requires $k \times (n_p - 1)$ additions and $k \times n_p$ multiplications in the plaintext domain; it requires $k \times (n_p - 1)$ multiplications and $k \times n_p$ exponentiations in the Paillier domain for the query. The high-cost computation may keep users waiting for a long time in each query. Obviously, the multiplication against the entire POI-table is too wasteful to find at most k entries of POI.

3 Proposed Scheme

Here, we propose a low-complexity k-POIs search scheme by dividing and aggregating a

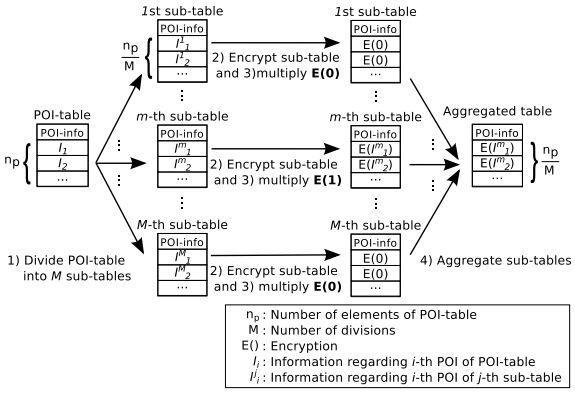


Fig. 2: Operations on LBS server in our scheme.

POI-table. Our scheme is based on PCQP. Our idea is to shift the partial POI-table whereas PCQP shifts the entire one. Figure 2 indicates our technique for partial shifting. In our scheme, a server divides a POI-table into M sub-tables and a user chooses the sub-table that involves the POI he/she wants, *i.e.* m-th sub-table in Fig. 2. Then the user sends M encrypted elements with his/her public-key, where the m-th element is the encrypted '1' and the others are the encrypted '0'. 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 $\frac{n_p}{M}$ by aggregating them into a table. Note that the server cannot see which sub-table the user wants since the processes above are performed in the encrypted domain with additive and multiplicative homomorphism. Thus, our scheme reduces computational complexity on the server without sacrificing both the security and accuracy. We describe how to divide and aggregate a POI-table in Section 3.2 and 3.3 in detail.

3.1 Preliminary

Before presenting our proposed scheme, we show the properties of additive and multiplicative homomorphism. Let $+_c$ and $\times_c$ denote the operator of additive homomorphism and that of multiplicative homomorphism, respectively. Here, for given plaintexts m_1 and m_2, pseudo-random numbers r_1 and r_2, a public-key k_{pub}, and a

private-key k_{pri}, Eq. (5) and (6) are satisfied.

$$\mathcal{D}_{k_{pri}}(\mathcal{E}_{k_{pub}}(m_1, r_1) +_c \mathcal{E}_{k_{pub}}(m_2, r_2)) = m_1 + m_2. \tag{5}$$

$$\mathcal{D}_{k_{pri}}(\mathcal{E}_{k_{pub}}(m_1, r_1) \times_c \mathcal{E}_{k_{pub}}(m_2, r_2)) = m_1 \times m_2. \tag{6}$$

In other words, the server can execute addition and multiplication without decryption. For example, NTRU cryptosystem [8] is one of the cryptosystems with additive and multiplicative homomorphism.

3.2 Initialization Process

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

$$\begin{aligned} \boldsymbol{t}^j_{POI} &= \left[\ I^j_i\ \right]_{1 \le i \le n_{p_0}}, \\ I^j_i &= I_{(n_{p_0} \times (j-1) + i)}, \end{aligned} \tag{7}$$

where n_{p_0}, I^j_i, and I_i denote the number of all entries of a sub-table calculated as $\lceil \frac{n_p}{M} \rceil$, 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 2 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. Note that a POI-table may contain overlapped entries. The reasons are as follows. In our proposed scheme, all sub-tables are aggregated into a table and thus the number of entries of the aggregated table is raised to the largest number of entries in sub-tables. For instance, when $n_p = 8$ and $M = 3$, eight POIs are divided into two sub-tables with three entries and a sub-table with two entries without redundancy. In this case, a user can obtain at most two POIs from the aggregated table when the user requires POIs in the sub-table with two entries. It is inefficient to obtain two POIs nevertheless the aggregated table accommodates three POIs. For this reason, we fix the number of entries in each sub-table to n_{p_0} with redundancy.

Now, let ID_{table} denote an index of a sub-table in which POI is contained. ID_{table} is added to the lookup-table so that a user knows which sub-table he/she should search. The lookup-table of our scheme, therefore, contains ID_{table}, $H\text{-}index$, and $H\text{-}value$ 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.

3.3 Query Process

In the query process, a user first generates a pseudo-random number t and a $n_{p_0} \times n_{p_0}$ t-offset circular shift permutation matrix $\boldsymbol{P'}^t$, which is defined as follows:

$$\begin{aligned} \boldsymbol{P'}^t &= \left[P'_{i,j}\right]_{0 \le i,j \le n_{p_0}-1}, \\ P'_{i,j} &= \begin{cases} 1 & (j = (i + n_{p_0} - t) \bmod n_{p_0}) \\ 0 & (otherwise) \end{cases}. \end{aligned} \tag{8}$$

Before sending $\boldsymbol{P'}^t$ to the server, the user encrypts it with his/her public-key and pseudo-random numbers $\boldsymbol{r} = \{r_0, \cdots, r_{n_{p_0}}\}$ as $\mathcal{E}_{k_{pub}}(\boldsymbol{P'}^t, \boldsymbol{r})$ 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

$$\boldsymbol{q}^m_M = [q_i]_{1 \le i \le M},\ q_i = \begin{cases} 1 & (i = m) \\ 0 & (otherwise) \end{cases}. \tag{9}$$

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

$$\begin{aligned} \boldsymbol{t}^{j'}_{POI} &= \mathcal{E}_{k_{pub}}(q_j,\ r'_j) \times_c \mathcal{E}_{k_{pub}}(\boldsymbol{t}^j_{POI}) \\ &= \left[\ \mathcal{E}_{k_{pub}}(q_j,\ r'_j) \times_c \mathcal{E}_{k_{pub}}(I^j_i)\ \right]_{1 \le i \le n_{p_0}}. \end{aligned} \tag{10}$$

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 $\boldsymbol{t}_{POI}^{j'}$ into a table $\boldsymbol{t}_{POI}^{'}$ as shown below:

$$\begin{aligned}\boldsymbol{t}_{POI}^{'} &= \boldsymbol{t}_{POI}^{1'} +_c \boldsymbol{t}_{POI}^{2'} +_c \cdots +_c \boldsymbol{t}_{POI}^{M'} \\ &= \left[\ I_i'\ \right]_{1\le i\le n_{p_0}}, \\ I_i' &= \mathcal{E}_{k_{pub}}(q_1,\, r_1') \times_c \mathcal{E}_{k_{pub}}(I_i^1) \\ &\quad +_c \mathcal{E}_{k_{pub}}(q_2,\, r_2') \times_c \mathcal{E}_{k_{pub}}(I_i^2) \\ &\quad +_c \cdots +_c \mathcal{E}_{k_{pub}}(q_M,\, r_M') \times_c \mathcal{E}_{k_{pub}}(I_i^M). \end{aligned} \tag{11}$$

Here, I_i' is still encrypted and satisfies

$$\mathcal{D}_{k_{pri}}(I_i') \;=\; I_i^m. \tag{12}$$

Therefore, each entry in an aggregated table is indicated by them of m-th 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 different random numbers against each element. In our scheme, the server multiplies $\mathcal{E}_{k_{pub}}(\boldsymbol{P'}^t, \boldsymbol{r})$ instead of $\mathcal{E}_{k_{pub}}(\boldsymbol{P}^t, \boldsymbol{r})$ by $\boldsymbol{t}_{POI}^{'}$ for circularly shifting. Therefore, as compared with Eq. (1) in PCQP, the server can reduce the number of multiplications to n_{p_0}. Hence, our scheme reduces computational complexity without sacrificing security.

3.4 Computational Complexity

In this section, we discuss the computational complexity on a server of our proposed scheme. The computational complexity is represented as that in the plaintext domain in the following since it depends on which encryption system is used. In other words, in this section, the terms 'addition' and 'multiplication' mean $+_c$ and $\times_c$, respectively. In the initialization process, our scheme requires the dividing process in addition to that of PCQP. It is negligible since its cost significantly low as compared with 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 $n_{p_0} \times (M-1)$ additions. The server, finally, shifts the aggregated table by multiplying $\boldsymbol{P'}^t$ and thus it requires $min(k, n_{p_0}) \times (n_{p_0} - 1)$ additions and $min(k, n_{p_0}) \times n_{p_0}$ multiplications, where $min(a, b)$ denotes the function returns the smaller one of a or b. TABLE 1 summarizes the computational complexity of PCQP and our scheme, respectively.

3.5 Search Accuracy

In our scheme, n_p POIs are aggregated into a table with n_{p_0} POIs. This causes losing information regarding $(n_p - n_{p_0})$ POIs. The more tables are divided, the less accurate the search results get. Therefore, we choose a certain number M for high-accuracy and low-complexity k-POIs search. We evaluate the relationship between search accuracy and the number of sub-tables in the following section.

4 Simulation Results

We evaluate the k-POIs search of our scheme in terms of computational complexity and search accuracy. Our scheme is implemented in Java language and performed on a laptop computer, which has Intel Core i5 1.8 GHz processor and 4 GB memory. We use Sequoia dataset [1] as a real-world dataset. We randomly chose 10,000 POIs from 65,536 POIs in Sequoia dataset to evaluate the performance metrics against Sequoia dataset.

Let $\boldsymbol{G}$ and $\boldsymbol{R}$ denote the ground-truth result set and the query result set obtained by k-POIs search, respectively. The accuracy rate of k-POIs search is defined as

$$r_{accuracy} \;=\; \frac{|\boldsymbol{R} \cap \boldsymbol{G}|}{|\boldsymbol{G}|}, \tag{13}$$

where $|\boldsymbol{R}|$ denotes the cardinality of the set $\boldsymbol{R}$. Then, for the given accuracy rate of our proposed scheme, $r_{accuracy_{PROP}}$, and that of PCQP, $r_{accuracy_{PCQP}}$, we define the ratio α as:

$$\alpha \;=\; \frac{r_{accuracy_{PROP}}}{r_{accuracy_{PCQP}}}. \tag{14}$$

As the definition above, $\alpha \in [0, 1]$ and $\alpha = 0$ means that the proposed scheme is not accurate at all

[1] http://www.chorochronos.org/

TABLE 1: Computer Complexity of PCQP and Our Scheme

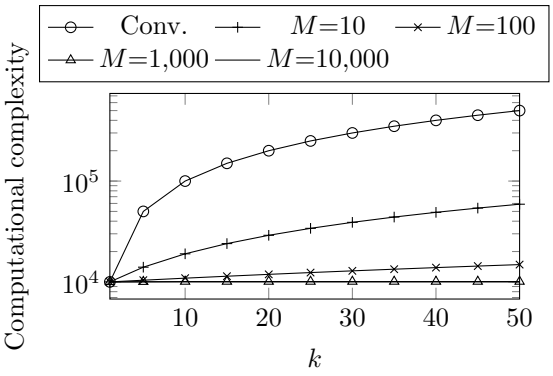
	Addition ($+_c$)	**Multiplication ($\times_c$)**
PCQP	$k \times (n_p - 1)$	$k \times n_p$
Our Scheme	$n_{p_0} \times (M-1) + min(k, n_{p_0}) \times (n_{p_0} - 1)$	$n_p + min(k, n_{p_0}) \times n_{p_0}$

while $\alpha = 1$ means that ours achieves the same accuracy as that of the conventional one achieves. Our objective is to keep α to be close to one while reducing computational complexity on the server.

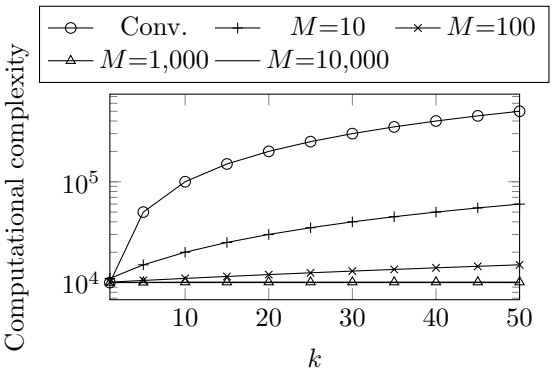
We experimentally evaluate the performance of our k-POIs search scheme for different values of M and k. M and k vary from 1 to 10,000 and from 1 to 50, respectively. Khoshgozaran *et al.* show that the average number of POIs which are assigned to the same *H-value* should be two or less for high-accuracy k-POIs search [3]. Thus, we set the curve order of Moore curve to eight that satisfies the above condition against our dataset. In each experiment, queries are issued at randomly chosen 1,000 locations on the map and the results are averaged.

4.1 Computational Complexity

Figure 3(a) and 3(b) indicate the number of additions and that of multiplications in the plaintext domain that are required in the schemes versus k per query, respectively. In Fig. 3, 'Conv.' indicates the conventional scheme. From Fig. 3, we can see that our schemes in any M significantly decrease the complexity of both addition and multiplication. Furthermore, the each number of additions and multiplications gets fewer as M gets larger. In particular, our scheme can reduce 98% of that required in the conventional scheme when $M = 10,000$ and $k = 50$. Even if $M = 10$, our scheme can reduce 90% of that. From these results, we can see that it is effective to divide a POI-table except when k is significantly small, *e.g.* $k = 1$, since our scheme requires multiplying every entry by encrypted '0' or '1' for the query.



(a) Addition ($+_c$)



(b) Multiplication ($\times_c$)

Fig. 3: Each number of additions and multiplications in the plaintext domain required in the schemes versus k.

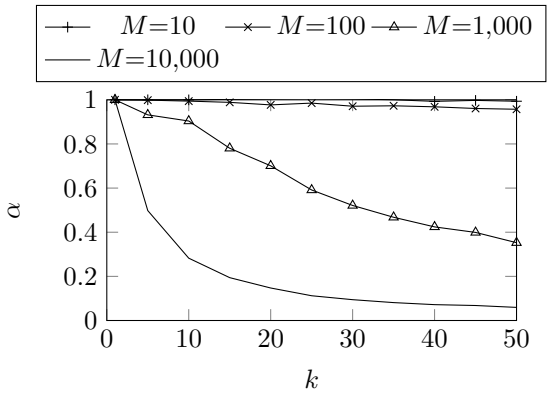


Fig. 4: α versus k.

4.2 Search Accuracy

Figure 4 indicates α versus k. From Fig. 4, we can see that α decreases when M or k gets large. In particular, our scheme achieves $\alpha \geq 0.99$ when $M = 10$ even if $k = 50$. However, α drops as M gets larger. The reason is that some entries of POI located near the user are lost by aggregating as mentioned in Section 3.5. Therefore, we must choose the minimum M that satisfies $\alpha \geq 0.95$ to provide a high-accuracy and low-complexity k-POIs search. Then, the each number of additions and multiplications required in our scheme is reduced 97% of that required in the conventional scheme under the condition of $n_p = 10,000$ and $k = 50$, respectively, and it is more reduced as k gets larger.

5 Conclusion

In this paper, we have proposed a low-complexity privacy-preserving k-POIs search scheme by dividing and aggregating a POI-table with homomorphism cryptosystem. Our scheme is as secure and accurate as the conventional scheme. We evaluate the computational complexity and search accuracy of our scheme and show that our scheme significantly reduces computational cost of k-POIs search without sacrificing the search accuracy.

Acknowledgment

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

Reference

[1] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," *ACM Trans. Database Syst.*, vol. 34, no. 4, pp. 24:1–24:48, Dec. 2009.

[2] K. Tan, Y. Lin, and K. Mouratidis, "Spatial cloaking revisited: Distinguishing information leakage from anonymity," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, N. Mamoulis, T. Seidl, T. Pedersen, K. Torp, and I. Assent, Eds. Springer Berlin Heidelberg, 2009, vol. 5644, pp. 117–134.

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

[4] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, April 2008, pp. 366–375.

[5] I.-T. Lien, Y.-H. Lin, J.-R. Shieh, and J.-L. Wu, "A novel privacy preserving location-based service protocol with secret circular shift for k-nn search." *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 863–873, 2013.

[6] A. Khoshgozaran and C. Shahabi, "Private information retrieval techniques for enabling location privacy in location-based services," in *Privacy in Location-Based Applications*, ser. Lecture Notes in Computer Science, C. Bettini, S. Jajodia, P. Samarati, and X. Wang, Eds. Springer Berlin Heidelberg, 2009, vol. 5599, pp. 59–83.

[7] E. H. Moore, "On certain crinkly curves," *Transactions of the American Mathematical Society*, vol. 1, no. 1, pp. 72–90, 1900.

[8] J. Hoffstein, J. Pipher, and J. Silverman, "Ntru: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, J. Buhler, Ed. Springer Berlin Heidelberg, 1998, vol. 1423, pp. 267–288.