SCIENCE CHINA
Information Sciences

• **Supplementary File** •

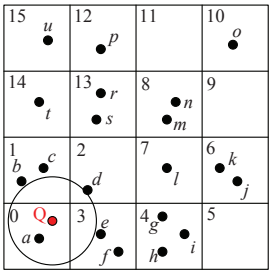
Secure and Efficient k-Nearest Neighbor Query for Location-based Services in Outsourced Environments

Haiqin WU[1], Liangmin WANG[1*] & Tao JIANG[2]

[1]*Department of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang*212013*, China;*
[2]*School of Cyber Engineering, Xidian University, Shaanxi* 710071*, China*

Appendix A An exmaple

For better illustration of our proposed framework, we show an example in Fig. A1 with n=20 POIs ($a \sim u$), where k=4 and τ=3. Clearly, we have $m = \lceil \frac{n}{\tau} \rceil$=7. Suppose that we adopt 2-order Hilbert curve in the location transformation and the query point Q lies in the bottom left corner of the whole region.



ID	$\varepsilon_{k_2}(SGI)$	$\varepsilon_{k_2}(EGI)$
1	$\varepsilon_{k_2}(0)$	$\varepsilon_{k_2}(1)$
2	$\varepsilon_{k_2}(2)$	$\varepsilon_{k_2}(3)$
3	$\varepsilon_{k_2}(4)$	$\varepsilon_{k_2}(4)$
4	$\varepsilon_{k_2}(6)$	$\varepsilon_{k_2}(7)$
5	$\varepsilon_{k_2}(8)$	$\varepsilon_{k_2}(10)$
6	$\varepsilon_{k_2}(12)$	$\varepsilon_{k_2}(13)$
7	$\varepsilon_{k_2}(14)$	$\varepsilon_{k_2}(15)$

ID	$E_{k_1}(P)$	$\varepsilon_{k_2}(R)$	$E_{k_1}(S)$
1	$E_{k_1}(a,b,c)$	$\varepsilon_{k_2}(R_{a,b},R_{b,c},R_{a,c})$	$E_{k_1}(S_{a,b},S_{b,c},S_{a,c})$
2	$E_{k_1}(d,e,f)$	$\varepsilon_{k_2}(R_{d,e},R_{e,f},R_{d,f})$	$E_{k_1}(S_{d,e},S_{e,f},S_{d,f})$
3	$E_{k_1}(g,h,i)$	$\varepsilon_{k_2}(R_{g,h},R_{h,i},R_{g,i})$	$E_{k_1}(S_{g,h},S_{h,i},S_{g,i})$
4	$E_{k_1}(j,k,l)$	$\varepsilon_{k_2}(R_{j,k},R_{k,l},R_{j,l})$	$E_{k_1}(S_{j,k},S_{k,l},S_{j,l})$
5	$E_{k_1}(m,n,o)$	$\varepsilon_{k_2}(R_{m,n},R_{n,o},R_{m,o})$	$E_{k_1}(S_{m,n},S_{n,o},S_{m,o})$
6	$E_{k_1}(p,r,s)$	$\varepsilon_{k_2}(R_{p,r},R_{r,s},R_{p,s})$	$E_{k_1}(S_{p,r},S_{r,s},S_{p,s})$
7	$E_{k_1}(t,u)$	$\varepsilon_{k_2}(R_{t,u})$	$E_{k_1}(S_{t,u})$

Figure A1 An example of 4-NN query processing where $\tau = 3$. (a) Distribution of POIs and query point Q. (b) Encoded HAI. (c) Encrypted DII.

In the preprocessing phase, DO mainly constructs DII and HAI based on the 1-D Hilbert value, then it sends the encrypted DII (Fig. A1(c)) and encoded HAI (Fig. A1(b)) to the storage and proxy server, respectively. As for the client, we have $h(x_q, y_q)$=0 as shown in Fig. A1(a). The client issues the kNN query by sending the encoded location $\varepsilon_{k_2}(0)$ to the proxy server. Subsequently, the proxy server retrieves encoded HAI and finds the corresponding HAI record (ID=1) containing the grid cell 0. Observe that, since $\tau < k$, the proxy server needs to expand the HAI record to the neighbor ID, here the second record is considered and we have $\mathbb{C}$={1,2}, which is next sent to the storage server. After receiving $\mathbb{C}$, the storage server returns the encrypted data information of POIs in grid cell 0, 1, 2, 3 (Fig. A1(c)). The following processing is to perform SDCP between the proxy server and the client. The candidate POIs $\{a, b, c, d, e, f\}$ are considered in SDCP. Moreover, to further reduce the comparison time, we can perform SDCP [1] only among d, e and f to find the fourth nearest neighbor d, because the 3-NN are a, b and c with high probability.

With respect to the issue of data updates, we mainly consider small-range motions (changes) of POIs, which is appropriate due to the expensive cost of relocation of POIs. In this example, if POI a moves in the range of grid cell 0 or 1, i.e., its corresponding Hilbert value is 0 or 1. In this case, HAI is not required to be updated since a still lies in the first record of HAI. However, the position of a and the slopes of the edges $S_{a,b}$, $S_{a,c}$ should be changed and re-encrypted with AES, and the right-hand in DII must be re-encoded with mOPE. Hence, the update time is $O((\tau - 1) * 2)$. If a moves out of grid cell 0 or 1, it is of high possibility that a is in their neighbor cells (2 or 3). In this case, HAI will be updated and a will be rearranged into the neighbor records of the original one (record 2). Accordingly, DII must be updated for the first two records to which b, c, d and a, e, f belong, respectively. The update time is $O((\tau - 1) * 4)$.

* Corresponding author (email: wanglm@ujs.edu.cn)

Appendix B Security analysis

The detailed security analysis for our scheme is given as follows.

Theorem 1. For the ciphertext-only attack, in our HkNN, the attacker learns nothing about the distribution of POIs except for the existence of densely distributed area.

Proof. Given the encrypted DII (Msg_1), the attacker learns nothing about the exact locations of POIs and the relative locations among them, as both locations and slopes are encrypted with AES. For the encoded HAI (Msg_2), by comparing if ε_{k_2}(SGI) equals to ε_{k_2}(EGI) in one or multiple records, the attacker may identify if there exist densely distributed areas in the whole region. However, the specific information about dense areas is not revealed to the attacker, which is regarded secure as POIs are unevenly distributed in the real world. In addition, based on the security guarantee of mOPE, It is intractable for attackers to deduce the exact locations of clients. Thus, our HkNN shows robust resilience against the ciphertext-only attacks.

Theorem 2. For the estimation attack, it is computationally intractable to estimate the locations of other original POIs and query point in our HkNN.

Proof. As we have introduced before, Hilbert curve transformation retains the proximity property of original data. Given a subset of original POIs $O \subseteq P$, and its transformed set O', the estimation average error (EAE) [3] between P and its estimated set P^* is computed by

$$\begin{aligned} EAE(P, P^*) &= \frac{1}{|O|}\sum_{i=1}^{|O|}\|p_i - p_i^*\| \\ &= \frac{1}{|O|}\sum_{i=1}^{|O|}\sqrt{(x_i - x_i^*)^2 + (y_i - y_i^*)^2} \end{aligned} \tag{B1}$$

where $p_i \in O$, p_i^* is the estimated location of p_i, and $\|p_i - p_i^*\|$ denotes the Euclidean distance between p_i and p_i^*. Clearly, a secure transformation scheme requires a high EAE value. In our HkNN, all transformed POIs and query point are further encoded with mOPE, which hides the proximity property of original data. Therefore, the EAE is pretty high in our method, which is not vulnerable to the estimation attack.

By Theorem 1 and 2, it is shown that our HkNN achieves the security goal of protecting both data confidentiality and clients' query privacy.

Appendix C Experiments

Table C1 presents the specific parameter settings in our experiments, where S_0, N, σ, Θ belong to Hilbert transformation parameter (HTP). In addition, we also implement mOPE using 32-bit encoding and use AES scheme with a key of 128 bits.

Table C1 Parameter Settings
(Default Values Are Shown in Bold Face)

Parameter	**Values**
S_0 (start point)	(0,0)
N (curve order)	8
σ (curve orientation)	D_1
Θ (scale factor)	1
τ (threshold)	5
n (# of POIs)	20000, 40000, 60000, 80000, **100000** (synthetic data) 175813 (real data)
k (# of query POIs)	4, **8**, 16, 32, 64
l (size of data packet)	16 Bytes

Preprocessing Cost at DO. For the DO, there are three major steps in preprocessing period: indices creation based on Hilbert transformation, indices encryption and encrypted indices transmission to the proxy server. Note that, the actual POIs distribution has little effect on preprocessing time in this period, therefore, we omit the results in other distributions and mainly consider the uniform distribution with different number of POIs.

Fig. C1 shows the preprocessing time for TkNN [1] and HkNN [2] with varying n ranging from 20000 to 100000. In TkNN, the data generation time (i.e., generating Delaunay triangulation) is $O(n\log n)$, and data encryption time is $O(8n)$ ($5n$ mOPE encryption and $3n$ AES encryption), so the overall preprocessing time is $O((\log n{+}8)n)$. However, for our HkNN, the indices creation time is $O(n)$. In addition, DO must encrypt the n POIs (i.e., the x and y coordinate), as well as the slopes in $\lceil \frac{n}{\tau} \rceil$ (i.e., m) records with AES. For mOPE encoding, it must encode the HAI and the right side in $\lceil \frac{n}{\tau} \rceil$ records, so there are $2n{+}n(\tau\text{-}1)$ AES encryption and $2\lceil \frac{n}{\tau} \rceil{+}n(\tau\text{-}1)$ mOPE encoding, Therefore, the preprocessing time is

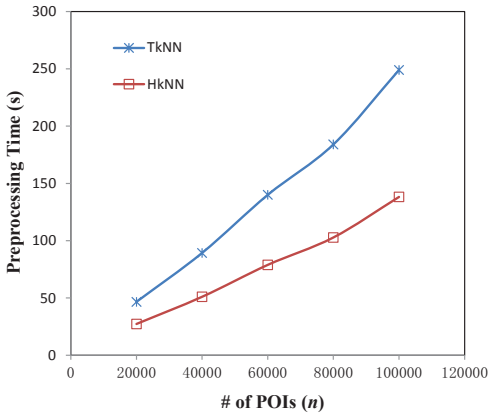


Figure C1 Preprocessing cost at DO with varying n

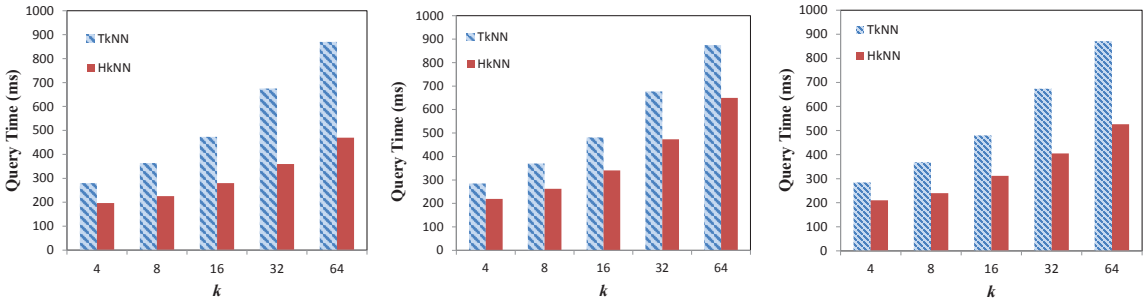


Figure C2 kNN query time with varying k. (a) Uniform, (b) Real-world, (c) Gaussian.

$O((2+(\tau^2+1)/\tau)n)$. Fig. C1 captures this advantage that HkNN outperforms TkNN. As we can see, the preprocessing time both grows linearly with n for two schemes, however, the gap between them gets larger as n increases, and our protocol reduces 44.5% time cost compared to TkNN, when n is 100000.

***k*NN Query Processing Time.** We define the query processing time as the duration from the time the query is issued until the k results are received after decrypting at the client.

Fig. C2 presents the kNN query processing time of both schemes as k increases under different datasets. As we can see, our protocol is superior to TkNN on all three datasets and reduces 45% time mostly. This is because TkNN needs to find the candidate POIs with $O(n)$ time overhead before performing SDCP, while our scheme reduces this overhead effectively by creating indices, which facilitates the POIs retrieval.

Interestingly, for TkNN, the different POIs distributions have little effect on query time, since all values are treated in a similar way in encrypted form. However, our scheme HkNN exhibits the best performance on the uniform dataset (Fig. C2(a)), which performs up to 1.8 times better than the TkNN when k equals 64.

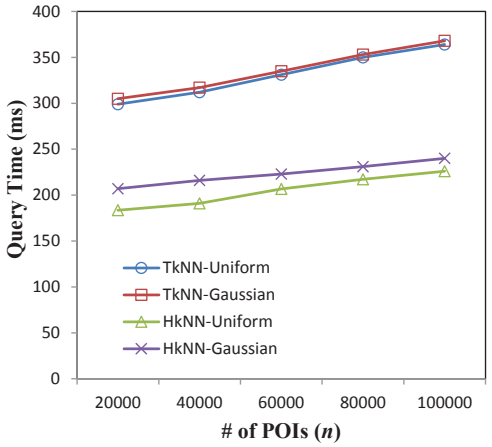


Figure C3 kNN query time with varying n

Fig. C3 shows the query time for both methods on two synthetic datasets with varying n. As expected, HkNN presents

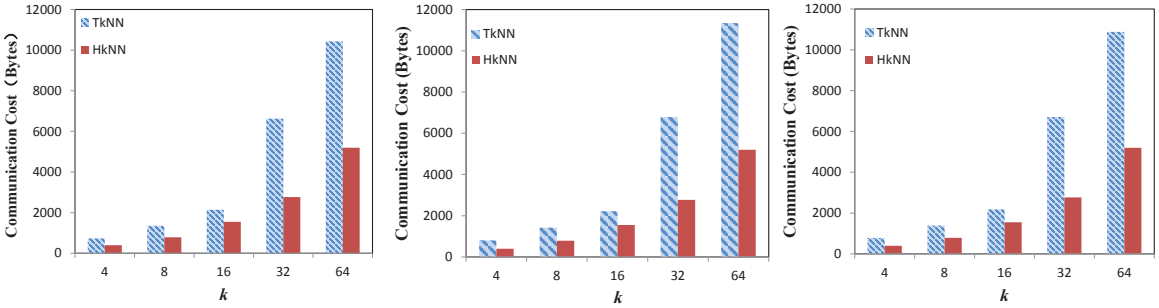


Figure C4 kNN communication cost at the client with varying k. (a) Uniform, (b) Real-world, (c) Gaussian.

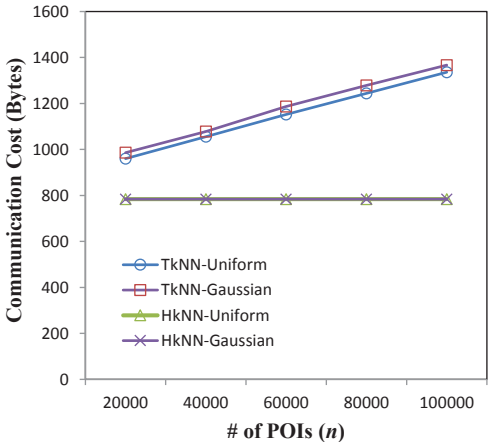


Figure C5 kNN communication cost with varying n

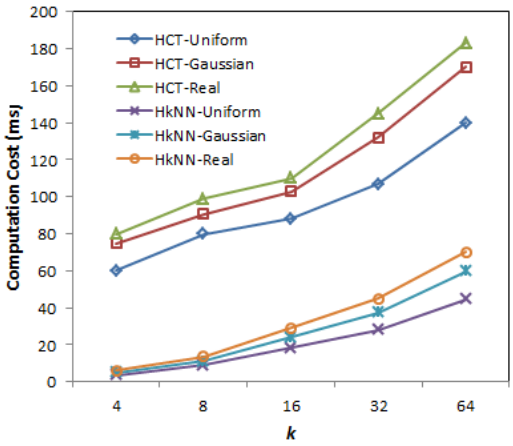


Figure C6 kNN computation cost with varying k

a better query performance when compared to TkNN for both datasets. In addition, the difference on POIs distributions does not result in a remarkable change in query time for both schemes, but the uniform dataset exhibits slightly less time.

Communication Cost at Client. In our experiment, the communication cost is measured in data bytes transferred between the proxy server and the client. Fig. C4 presents the communication cost as k increases on different datasets. It is observed that the amount of communication grows with k, due to a proportionally larger number of results returned to the clients. In addition, when k increases, TkNN is considerably costlier than HkNN in three cases, due to the fact that client needs to send encoded query square (four data values) to the server and may receive more slopes returned from server in TkNN, whereas HkNN only needs one Hilbert value when issuing a query.

As illustrated in Fig. C5, the increases in dataset size do not lead to a significant increase in communication cost for both two methods. Moreover, the different data distributions also have little effects on it, with TkNN subtly more expensive in Gaussian distribution, and HkNN keeping the same. The reasons are similar to those explained above in Fig. C2.

Computation Cost at Client. In particular, we further evaluate the computation cost at the client. Note that, the computation time at the client is included included in the query time. And we have illustrated that our HkNN outperforms TkNN above. Hence, we omit the comparison between them and use another HCT as benchmark.

Fig. C6 shows the impact of k on the client's computation cost for both schemes on three dataset. We can see that the computation cost of HCT increases rapidly than HkNN as k increases, and the fomer is also much higher than the latter. The reason is that the index retrieval and false-positive filtering are both performed at the client in HCT. In contrast, our scheme reduces the computation overhead dramatically by assigning the most of computation to the proxy server, and the client only needs to perform a small part in the query processing.

References

1 Choi S, Ghinita G, Lim H S, et al. Secure knn query processing in untrusted cloud environments. IEEE Trans. Knowledge and Data Engineering, 2014, 26(11): 2818-2831.

2 Kim H I, Hong S, Chang J W. Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data. Data & Knowledge Engineering, 2016, 104: 32-44.

3 Yiu M L, Ghinita G, Jensen C S , et al, Enabling search services on outsourced private spatial data. The VLDB Journal, 2010, 19(3): 363-384.