Technical Report: Authenticating Top-k Queries in Location-based Services with Confidentiality

Qian Chen, Haibo Hu, Jianliang Xu
Department of Computer Science, Hong Kong Baptist University
Kowloon Tong, Hong Kong
{qchen, haibo, xujl}@comp.hkbu.edu.hk

In this technical report, we offer more details about several issues in the paper. Sec. 1 discusses more about the the ranking function we adopted in the paper. Sec. 2 analyzes the cost of two private ranking comparison methods: PPB and PLB. Sec. 3 gives out the complete proof of Lemma 6.7 in the paper. Sec. 4 briefly introduces how to extend the PLB method to 3D. Followed by Sec. 5, which shows how to further encrypt area A_2 in PLB method.

1. DISCUSSION ON RANKING FUNCTION

In this section, we'd like to clarify that the top-k ranking function adopted in this paper is not tailored. In the following, we first explain our rationale behind adopting the ranking function, then show it is a weighted function, and finally argue that this function can be easily reduced to the sum ranking function with guaranteed bound of overhead.

1. Our top-k ranking function, $rank(r,q) = ||r.\lambda - q.\lambda||^2 + r.\omega = ||r.\lambda - q.\lambda||^2 + (\sqrt{r.\omega})^2$ (known as *Euclidean scoring function* [2]), is an alternative to the sum scoring function and has been widely considered in the previous work (e.g., [2, 3, 6, 7]).[1] We adopt this Euclidean scoring function for location-based top-k queries since it treats the non-spatial and spatial dimensions equally. It is essentially a (squared) Euclidean distance between a query point $q = (q.\lambda_x, q.\lambda_y, 0)$ and a result object $r = (r.\lambda_x, r.\lambda_y, \sqrt{r.\omega})$ in 3D space.[2] To make it clearer, in this revision we have rewritten the ranking function as $rank(r,q) = ||r.\lambda - q.\lambda||^2 + r.\omega^2$, where the result object is represented by $r = (r.\lambda_x, r.\lambda_y, r.\omega)$, and modified all related descriptions accordingly.

2. Our ranking function does take weights into account, though we omit an explicit weight between the spatial distance and the non-spatial score. In fact, we normalize the non-spatial score ω to reflect the relative weighting on these two factors. Suppose there is a constant weight α in the ranking function:

$$rank(r,q) = \alpha \cdot \|r.\lambda - q.\lambda\|^2 + (1-\alpha) \cdot r.\omega^2.$$

This ranking function can be transformed to:

$$\frac{rank(r,q)}{\alpha} = \|r.\lambda - q.\lambda\|^2 + \frac{1-\alpha}{\alpha} \cdot r.\omega^2,$$

which can be rewritten in the form of our ranking function by normalizing $r.\omega$ to $r.\omega' = \sqrt{\frac{1-\alpha}{\alpha}} \cdot r.\omega$ in preprocessing:

$$rank'(r,q) = \|r.\lambda - q.\lambda\|^2 + r.\omega'^2$$

Similar techniques of hiding explicit weights were also adopted in [5, 1] based on the rationale that the query results will only be affected by the relative importance, rather than the absolute magnitude, in the query vector.

3. Our ranking function differs from a sum ranking function $rank(r,q) = \alpha \cdot ||r.\lambda - q.\lambda|| + (1-\alpha) \cdot r.\omega$ **in the same way Euclidean distance differs from Manhattan distance**. Therefore, the top-k results of a sum ranking function can be derived from the top-k' results of our ranking function. As shown in Figure 1, all top-k results of the sum ranking function are in Zone I (the grey triangle), bounded by the solid line that crosses the top-kth object r_k. On the other hand, the top-k' results of our ranking function form a quadrant of an ellipse. To guarantee all top-k results are in the top-k' results, we make this ellipse enclose Zone I; and to achieve the minimum k', this ellipse should be the minimum circumscribed ellipse (the solid arc) of Zone I. Note that enclosing Zone I is just a sufficient condition of enclosing all top-k results — in a less worse case a smaller ellipse, such as the dashed arc, may already enclose them.

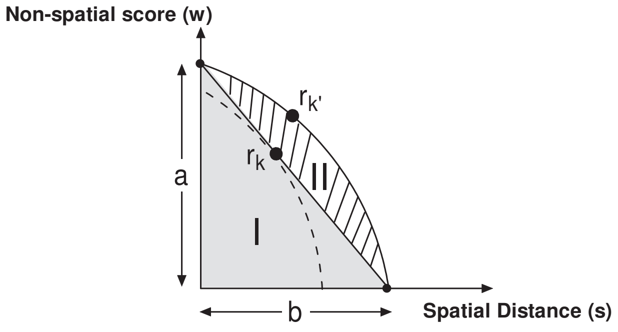

Figure 1: Top-k in Sum Ranking vs. Top-k in Euclidean

[1]It is termed as *non-linear* function in [3, 7] and *semi-monotone* function in [6].
[2]That is, the (squared) Euclidean distance is $(r.\lambda_x - q.\lambda_x)^2 + (r.\lambda_y - q.\lambda_y)^2 + (\sqrt{r.\omega})^2 = ||r.\lambda - q.\lambda||^2 + (\sqrt{r.\omega})^2$.

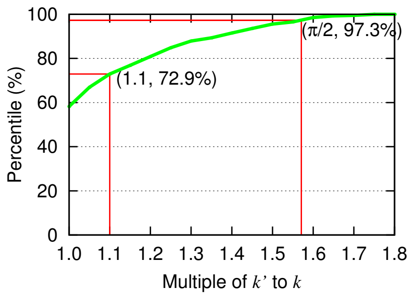

Figure 2: Cumulative Probability Distribution of k' (as multiple of k, k=16)

Next we show how to use our top-k' query to derive the top-k query results. Besides normalizing ω by multiplying $\frac{1-\alpha}{\alpha}$, we set k' for our ranking function as follows:

$$k' = k \cdot \left(1 + \frac{\iint_{II} f(s,\omega) ds\, d\omega}{\iint_{I} f(s,\omega) ds\, d\omega}\right), \tag{1}$$

where $f(s,\omega)$ is the probabilistic density function (pdf) of spatial distance s and non-spatial score ω. As s and ω are independent, $f(s,\omega)$ can be decomposed into $g(s) \cdot h(\omega)$. According to [M12], $g(s)$, the pdf of the Euclidean distance of two random points in a circle, is approximately constant when they are very close (as in the top-k query). If we further assume $h(\omega)$ is constant, i.e., the non-spatial scores are uniformly distributed, the fraction in Eqn. (1) will be the ratio of the areas of Zone I and II:

$$k' \simeq k \cdot \left(1 + \frac{Area(II)}{Area(I)}\right) = k \cdot \frac{\pi ab/4}{ab/2} = \frac{\pi}{2}k.$$

The above equation suggests that in the worse case, by setting $k' = \frac{\pi}{2}k$, the top-k' results of our ranking function will enclose all top-k results of a sum ranking function.

To verify this result, we issue top-k queries of sum ranking functions with a variety of α on the Gowalla dataset and find the minimum k' for each query that encloses all top-k results. Figure 2 plots the cumulative probability distribution (cdf) of k' in terms of a multiple of k. It can be observed that in more than 70% of cases, k' is as low as $1.1k$. And by setting $k' = \frac{\pi}{2}k$ as we derived above, this percentile can reach as high as 97.3%. (The remaining 2.7% cases could be caused by failing to conform to the constant pdf assumption of $g(s)$ when k is large.) Further by setting $k' = 1.8k$, all top-k results are enclosed.

We have now added a footnote (Footnote 2 on Page 3) to explain the difference between the sum and Euclidean ranking functions.

In summary, our ranking function is indeed a general linear (weighted) Euclidean function, which is justified for location-based top-k queries. Also, if some applications really want a linear (weighted) sum ranking function, the solution can be approximated by our ranking function with bounded overhead.

2. COST ANALYSIS OF PRIVATE RANKING COMPARISON

In this section, we analyze the costs of the PPB and PLB methods as follows. Let M_{enc}, M_{sign}, and M_{digest} denote the lengths of a Paillier ciphertext, a signature, and a digest, respectively. Let C_{mul}, C_{sign}, and C_{hash} denote the CPU costs of a modular multiplication, a signature verification, and a hash operation, respectively. Following the notations in the paper, let $m = log_B(U)$, where B is the base and U is the upper bound of the data domain.

- **Cost analysis for PPB**. The total number of encrypted seeds $\{E(\delta_i \cdot B^i) \mid i = 1, 2, \cdots, m, \delta_i \in [0, 1, \cdots, B-1]\}$ stored on the server is $B \cdot m$. The number of encrypted seeds needed for one comparison is m. Thus, the communication involves: (1) $E(2x), E(2y), E(x^2), E(y^2), E(\omega^2)$ of two points; (2) m encrypted seeds $\{E(\delta_i \cdot B^i) \mid i = 1, 2, \cdots, m\}$; and (3) an aggregated signature. So its total cost is:

$$M_{PPB} = (10 + m) \cdot M_{enc} + M_{sign}.$$

This is also the client space cost. To verify the comparison result, the client: (1) reconstructs $E(\delta)$ with the encrypted seeds, which costs $m \cdot C_{mul}$; (2) tests Eqn. (5), whose cost is bounded by $(2m + 9) \cdot C_{mul}$; and (3) verifies the signature, which costs C_{sign}. So the total client CPU cost is:

$$C_{PPB} = (3m + 9) \cdot C_{mul} + C_{sign}.$$

Note that the client does not perform any Paillier encryption in PPB.

- **Cost analysis for PLB**. The communication involves: (1) two points o'_1, o'_2, (2) digests for $g(A_1)$ and the signature for $g(A_3)$. According to [19], the length of the digests for $g(A_1)$ is $(m+4+\lceil log_2 m \rceil) \cdot M_{digest}$. So the total communication cost is:

$$M_{PLB} = m/2 + (m + 4 + \lceil log_2 m \rceil) \cdot M_{digest} + M_{sign}.$$

This is also the client space cost. To verify the comparison result, the client: (1) calculates area A_2, whose CPU cost can be omitted; (2) reconstructs $g(A_3)$ from the digests of $g(A_1)$ with the help of A_2, which costs $[B(m + 1) + \lceil log_2 m \rceil + 2] \cdot C_{hash}$; and (3) verifies the signature. So the total client CPU cost is:

$$C_{PLB} = (B(m + 1) + \lceil log_2 m \rceil + 2) \cdot C_{hash} + C_{sign}.$$

Note that the PLB method only works for the pairs that have been pre-signed. Nonetheless, once the PLB method is enabled, its cost will not be affected by the total number of pairs that are pre-signed.

3. PROOF OF LEMMA 6.7 IN SECURITY ANALYSIS OF POWER DIAGRAM BASED SCHEME

In this section, we give out the detailed proof of Lemma 6.7 in the paper as follows.

Lemma 6.7: *Let $u \notin R$, $P(u = a, \omega = b) = P(u = a, \omega = b \mid rank(u, q) \geq rank(r_k, q) \bigwedge u \in \mathcal{VN}(r_i))$.*

PROOF.

$$\begin{aligned}
&P(u=a,\omega=b \mid rank(u,q) \geq rank(r_k,q) \bigwedge u \in \mathcal{VN}(r_i)) \\
&= \frac{P(rank(u,q) \geq rank(r_k,q) \bigwedge u \in \mathcal{VN}(r_i) \mid u=a,\omega=b)}{P(rank(u,q) \geq rank(r_k,q) \bigwedge u \in \mathcal{VN}(r_i))} \\
&\quad \cdot P(u=a,\omega=b) \\
&= \frac{P(rank(u,q) \geq rank(r_k,q) \bigwedge u \in \mathcal{VN}(r_i) \bigwedge u=a \bigwedge \omega=b)}{P(rank(u,q) \geq rank(r_k,q) \bigwedge u \in \mathcal{VN}(r_i))} \\
&= P(u=a,\omega=b)
\end{aligned}$$

Here the third equality is due to the fact that both $u \in \mathcal{VN}(r_i)$ and $rank(u,q) \geq rank(r_k,q)$ are independent of $u=a$ and $\omega=b$ as the locations and scores of r_i and r_k are unknown to the client. □

4. EXTENSION OF PLB METHOD TO 3D SPACE

We briefly show the 3D solution as follows. Given two 3D objects s,t, we define a **score-shifted plane** of s,t, on which s,t have equal ranking values. Let $\mathcal{SP}(s,t)$ denote this plane, which can be found by shifting the perpendicular bisector of s,t by their score difference:

$$\mathcal{SP}(s,t) = \{p \mid (t-s)^T p = \frac{1}{2}(\|t\|^2 - \|s\|^2 + \omega_t^2 - \omega_s^2)\}.$$

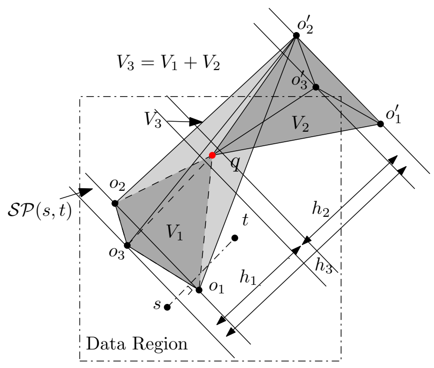

Figure 3: Presigned Line in 3D Space

Similar to the 2D case, we choose three arbitrary points o_1, o_2, o_3 on $\mathcal{SP}(s,t)$, as shown in Figure 3. To avoid disclosing $\mathcal{SP}(s,t)$, we find another far away plane $pl(o_1', o_2', o_3')$ parallel to $pl(o_1,o_2,o_3)$ and let triangular areas $A_{o_1o_2o_3} = A_{o_1'o_2'o_3'}$. Let V_1 denote the *directed* volume of tetrahedron $qo_3o_2o_1$, V_2 denote the *directed* volume of tetrahedron $qo_1'o_2'o_3'$, and V_3 the *directed* volume of tetrahedron $o_2'o_3o_2o_1$, we have:

$$\begin{aligned}
V_3 &= \tfrac{1}{3}A_{o_1o_2o_3} \cdot h_3 \\
&= \tfrac{1}{3}A_{o_1o_2o_3} \cdot (h_1 + h_2) \\
&= \tfrac{1}{3}A_{o_1o_2o_3} \cdot h_1 + \tfrac{1}{3}A_{o_1'o_2'o_3'} \cdot h_2 \\
&= V_1 + V_2.
\end{aligned}$$

The above equation resembles Eqn. (8) in the paper for the 2D case. As such, the 3D PLB authentication scheme follows.

5. ENCRYPTION ENHANCEMENT OF AREA A_2 COMPUTATION IN PLB METHOD

This section shows how A_2 is further encrypted in the PLB method, in order not to disclose o_1' and o_2' to the client.

Hiding only o_1' and o_2' can't solve the problem of disclosing o_1' and o_2' to the client essentially, if the client conducts four or more queries with various query locations, it can compute out the coordinates of o_1' and o_2' by simultaneous equations after receiving enough values of A_2. As thus, we further encrypt the value A_2 to avoid this attack, meanwhile the PLB method still works.

The PLB method works based on a function proposed and optimized by Pang et al. in [4], which is for verifying $q \geq \alpha$ without the client knowing the value of α. The optimization is inspired by the *canonical* representation of integer. Let B be the base, then

$$\delta = \sum_{i=0}^{m} \delta_i \cdot B^i,$$

where $\delta_i \in [0,B)$. Applied with the representation, the optimized function for verifying $q \geq \alpha$ works as follows. The SP gets the representation $(q-\alpha)_i$, and sends components $g((q-\alpha)_i)$ to the client, who then gets representation $(\alpha-L)_i$ and computes $g((q-L)_i) = g((q-\alpha)_i) \otimes g((\alpha-L)_i)$, where $\otimes$ is a well-defined operation on the digest. At last, client verifies $q \geq \alpha$ by comparing every jointly computed $g((q-L)_i)$ value with the $g((q-L)_i)$ value signed by the DO.

Similarly in PLB method, to hide the value of A_2, the SP returns only $A_{2,i}$, the canonical representation of A_2, to the client, since the client only need the $A_{2,i}$ to jointly compute out $g(A_{3,i}) = g(A_{1,i}) \otimes g(A_{2,i})$. Without knowing the base B, the client can not infer what value A_2 is[3].

On the other hand, the client without knowing A_2 has to verify the correctness of A_2. Let x_p and y_p denote the x and y coordinate of a point p, respectively. By expanding the formula calculating the area of $\triangle qo_1'o_2'$, we can get:

$$x_{o_1'}y_q + y_{o_2'}x_q + x_{o_2'}y_{o_1'} = x_{o_2'}y_q + y_{o_1'}x_q + x_{o_1'}y_{o_2'} + 2A_2.$$

If both sides of this equation are encrypted by Pailler, according to properties of Pailler, and A_2 is rewrote as its canonical representation $A_{2,i}$, it is equivalent to proving following equation instead:

$$\begin{aligned}
&E(x_{o_1'})^{y_q} E(y_{o_2'})^{x_q} E(x_{o_2'}y_{o_1'}) \\
&\equiv E(x_{o_2'})^{y_q} E(y_{o_1'})^{x_q} E(x_{o_1'}y_{o_2'}) \prod_{i=0}^{m} E(B^i)^{2A_{2,i}} \mod n^2,
\end{aligned} \tag{2}$$

where except for x_q, y_q and $2A_{2,i}$, all the rest items can be pre-computed and signed by the DO offline. And since only the DO possesses the private key of Pailler, these items can not be decrypted by the client. Thus, the client can verify Eqn. 2 without knowing o_1', o_2' or A_2.

The following summaries the whole procedure of PLB method without disclosing o_1', o_2' or A_2 to the client. During the service initialization, the DO prepares seeds $E(B_i)$ for the SP, so that the SP can construct any future A_2 for the client. It also encrypts $E(x_{o_1'})$, $E(y_{o_1'})$, $E(x_{o_2'})$ and $E(y_{o_2'})$ (donated as $E()$ values), $g(A_{3,i})$ and a signature for every selected pairs of points. Upon a comparison on $rank(s,q) \geq rank(t,q)$, besides sending $E()$ values of o_1', o_2' and $g(A_{1,i})$, the SP also sends back $A_{2,i}$ and corresponding $E(B_i)$ to the client. By verifying every jointly computed $g(A_{3,i})$ is the same as $g(A_{3,i})$ signed by the DO, the client can verify $A_1 \geq 0$; by verifying Eqn. 2 holds, it can verify

[3]To be more securer, the *canonical* representation can be replaced with other representation.

the returned representation $A_{2,i}$ is correct; and at last, by verifying the signature, the client can verify all the returned values are not tampered with.

6. REFERENCES

[1] Y. K. A. Vlachou, C. Doulkeridis and K. Norvag. Reverse top-k queries. In *ICDE*, 2010.

[2] S. Chaudhuri and L. Gravano. Evaluating top-k selection queries. VLDB, pages 397–410, 1999.

[3] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 635–646, 2006.

[4] H. Pang, A. Jain, K. Ramamritham, and K. lee Tan. Verifying completeness of relational query results in data publishing. In *Proc. SIGMOD*, 2005.

[5] P. Tsaparas, N. Koudas, Y. Kotidis, T. Palpanas, and D. Srivastava. Ranked join indices. In *ICDE*, pages 277–288, 2003.

[6] D. Xin, J. Han, and K. C. Chang. Progressive and selective merge: computing top-k with ad-hoc ranking functions. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 103–114, 2007.

[7] L. Zou and L. Chen. Pareto-based dominant graph: An efficient indexing structure to answer top-k queries. *IEEE Trans. on Knowl. and Data Eng.*, pages 727–741, 2011.