Privacy-Preserving Location-based Services

X. Sean Wang

Department of Computer Science
University of Vermont, Vermont, USA

Corso di Dottorato, Milano, Italiano, Maggio 2008

Some materials based on slides provided by Prof. Claudio Bettini

Location based service (LBS)

Location based service:

- request includes location information
- (service) information response is a function of the given location

Example

"Give me the closest vegetarian restaurant to this location"

where "this location" can be filled in by a GPS device and/or the cell operator.

Generally speaking

Objective:

- Allow the use of location service without giving up (too much) privacy

Two privacy concerns:

- Location of the user is private information
- The request (or the fact there is a request from the user) is private information

Two methods:

- "Obfuscate" private information
- "Disown" private information

Basics first: near/nearest neighbor search

Problem:

- Given an n-dimensional point q, find the nearest neighbor among a set S of (n-dimensional) points.
- We assume $n = 2$ here.

Distance function: Usually Euclidean.

- Each point is a vector: $X = \langle x_1, \ldots, x_n \rangle$.
- Given two points X and Y, $D(X, Y) = \sqrt{\Sigma_{i=1}^{n}(x_i - y_i)^2}$.
- Triangulation property, $D(X, Y) \leq D(X, Z) + D(Z, Y)$, holds for all points X, Y, Z.

Two versions

- Near neighbors of X within distance δ: $\{Y \in S | D(X, Y) \leq \delta\}$
- Nearest neighbors of X: $\{Y | \forall Y' \in S : D(X, Y') \geq D(X, Y)\}$

PR-Quadtree and KD-tree

PR-Quadtree

The PR-quadtree partitions the 2-dimensional region (a node) into four equal quadrants (4 children nodes), recursively, until the region for each node (leaf) contains no greater than the prescribed number of points.
Online demo: `http://donar.umiacs.umd.edu/quadtree/points/prquad.html`

KD-tree

A kd-tree is similar to PR-Quadtree, i.e., it recursively split the region (subregions form the children nodes). Kd-tree, however, only uses a splitting plane that is perpendicular to one of the coordinate system axes. Each time it chooses a particular axes so that the splitting is most even in terms of the number of points in the two subregion.

Nearest neighbor search

NN search algorithm

- Given a query point q and a PR-quadtree, go down the tree from the root to the *most promising* branch (depth first search)
- Once in a leaf node, find the nearest point (called candidate NN) to q, and remember the point and the distance δ from q to this point.
- Now go on with the depth first search but prune the search space by the distance δ to q, i.e., if a region is farther away from q than δ, then we do not need to go into that region.
- Every time we reach a leaf node, we try to modify the candidate NN and δ and continue the search.

Privacy preserving NN search – disowning the request

- The idea is to obtain some kind of k-indistinguishability, or k-anonymity.
  - Remove the obvious ID values from request, but can't remove location information
  - HOWEVER, Location information may be used to link back to the user! [1]
- Then?
  - Find $k-1$ other users around you, and use their location to ask for nearest neighbors for each of *for them*. (Group nearest neighbor search.)
  - Then from all the answers, find the real target
  - Note that the actual NN must be within the returned set
  - The service provider would not just have location of one user, but locations of k users, then k-anonymity is obtained.

But wait... which other $k-1$ users to choose (or "use")?

[1] Adversary model! In this case, we assume the adversary can somehow link the location to user.

Privacy preserving NN search – disowning the request (1)

- If the adversary does not have any idea how the other $k-1$ users are chosen, then any other $k-1$ users work. For efficiency, perhaps just choose the nearest $k-1$ neighbors. (Why more efficient this way?)
- However, if the adversary knows the algorithm[2], then the adversary may be able to figure out the location of the user who issued the request!
- Perché? Let's try use the "nearest $k-1$ neighbors" method as an example. Consider $k=2$. Give an example when the privacy protection fails.

[2] Adversary model!

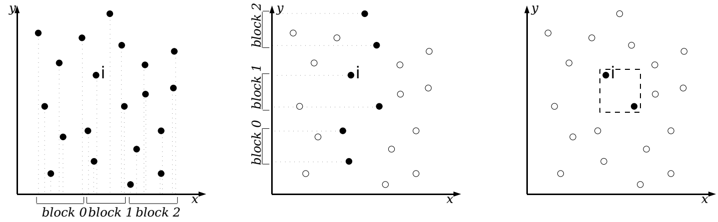
Privacy preserving NN search – disowning the request (2)

- The lesson from the above example: under the assumption that the adversary knows the algorithm of choosing the other $k-1$ users,
- Then if we use a group G of k users for k-anonymity for query point q and a particular algorithm $f(q) = G$ to choose this group, then we need to make sure that $f(p) = G$ as well for each p in G.

Privacy preserving NN search – disowning the request (3)

- So we need a method to find this $f(q) = G$.
- A remark is that its always the better if the points in G are all close to q.
  - This is why we tried to use $(k-1)$-NN for q... but it does not satisfy the above condition (namely, $f(p) = G$ for each $p \in G$)
- Sergio Mascetti is an expert in finding this f function!

The Grid algorithm



(a) First iteration

(b) Second iteration

(c) Third iteration

Achieving location uncertainty

- What if I am only concerned with my location information being revealed?
  - That is, I don't really care if adversary knows that I issued the request...
  - But I DO care if they find out from *WHERE* I issued the request...
- No anonymity is needed because in this case, they can know that it's me who issued the request
- We need location uncertainty

First attempt

- What about just randomly pick a position near me as the location information in my request? Like in the next building, or next block, or next town, depending on the user privacy concern.
- What if the adversary knows the algorithm to choose the fake location? (Homework for you! *Hint: randomization*.)

Achieving location uncertainty (2)

- Fake location information works to hind the location of the issuer...
- But the NN query will not give a right answer
- Dilemma:
  - To get more precise answer, we need to choose a location near the original/true location... but this provides less uncertainty
  - To provide more uncertainty, less precise query result will be given
- Any method to have best of both?

SpaceTwist

Here is an idea (SpaceTwist algorithm):

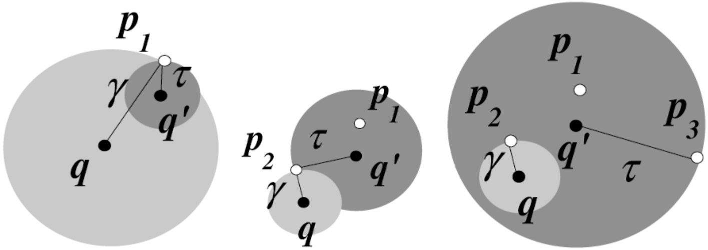
- I use a fake point to ask to server for nearest neighbor
- But keep on asking for the next nearest neighbor, and next, and next...
- Until I see the nearest neighbor of my original query point.

Two questions:

- How do I know the NN of my original point has arrived?
- How much space uncertainty does this method give?

SpaceTwist (2)

To answer the first question, let us consider the following diagram:



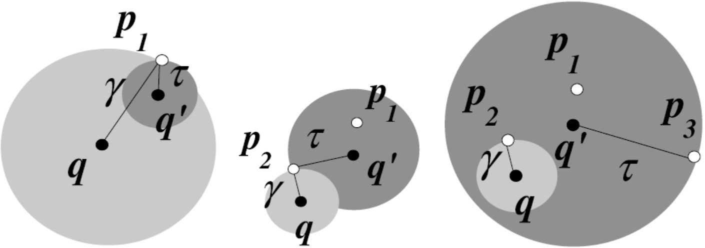
(a) first point (b) second point (c) third point

The darker circle is called the *supply region*, and the lighter one is called the *demand region*.

- When the supply region covers the demand region, then my nearest neighbor has arrived.

SpaceTwist (3)

How are the two regions defined?



(a) first point (b) second point (c) third point

- The *supply region* is the circle with the fake point q' as the center and the distance to the current next nearest neighbor of q' as radius.
  - Property: all the points in the supply region have been delivered to me.
- The *demand region* is the circle with my query point q as the center and the distance to the nearest neighbor to q among all I have received as the radius.

SpaceTwist (4)

How much location uncertainty does this provide?

- Assume the adversary can listen to the traffic, i.e., receives all the replies the server sends $p_1, \ldots, p_n$.
- Assume also that the adversary knows the algorithm.
- Let's try to guess where q could be.

Where can q be?

Assume I stopped the serve at the n-th point it sends me, which is when the supply region covers my demand region.

- Fact: I didn't stop at $(n-1)$-th point. It means at that time, the supply region did not cover the demand region.

So q must satisfy the following conditions:

- $D(q, q') + min_{1 \leq i < n} D(q, p_i) > D(q', p_{n-1})$
- $D(q, q') + min_{1 \leq i < n} D(q, p_i) \leq D(q', p_n)$

PIR-based method

Private Information Retrieval

- Assume a server maintains a database of n items.
- A user wants to query the content of the i-th item.
- However, the user doesn't want the server know which item he's querying!

How do we do this?

- Assume database has $\langle x_1, \ldots, x_n \rangle$, and each $x_i = 0, 1$.
- Assume we have a set:

$$QR = \{y \in Z_N^* | \exists x \in Z_N^* : y = x^2 \bmod N\}$$

  where $Z_N^* = \{x \in Z_N | gcd(N, x) = 1\}$, and $N = p \cdot q$ (product of two large primes)
- Items in QR is called quadratic residuals.
- Denote QNR the complementary of QR (i.e.,

PIR

- A fact about QR and QNR: It's computationally hard to distinguish where a number is in QR or QNR without knowing p and q. Easy if p and q are known.
- Now assume a query wants content (0 or 1) at the i^*-the position.
- It sends over $y_1, \ldots, y_n$, where y_{i^*} is in QNR and all other y_i are in QR.
- The database computes and sends $z = \Pi_{i=1}^{n} w_i$, where $w_i = y_i^2$ if $x_i = 0$ and $w_i = y_i$ otherwise.
- The user looks at z: if $z \in QNR$ then $x_i = 1$, otherwise $x_i = 0$.

PIR (2)

Examples

- database $= \langle 0, 1, 1, 0 \rangle$
- user wants 2nd item, and sends $\langle qr, qnr, qr, qr \rangle$, where $qr \in QR$ and $qnr \in QNR$.
- database sends back $z = (qr)^2 \times (qnr) \times (qr) \times (qr)^2$, and hence $z \in QNR$, and the user knows the answer is 1.

- database $= \langle 0, 1, 1, 0 \rangle$
- user wants 4th item, and sends $\langle qr, qr, qr, qnr \rangle$, where $qr \in QR$ and $qnr \in QNR$.
- database sends back $z = (qr)^2 \times (qr) \times (qr) \times (qnr)^2$, and hence $z \in QR$ and the user knows the answer is 0.

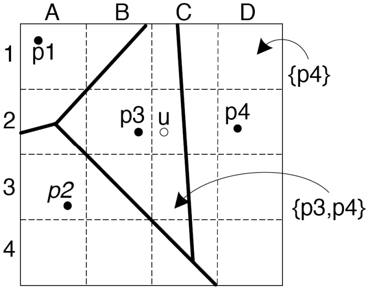
Making it more efficient

- We fold the database n-bits into $m \times m$ matrix (if n is not a square, pad with 0s)
- User asks for content (i^*, j^*), but only send $y_1, \ldots, y_m$ as if the database only has one row.
- Database computes m z numbers as if there are m databases (rows)
- User gets these m numbers, and pick out the i^* row (throw away everything else), and then use the same trick as before to get (i^*, j^*) content.
- Why more efficient? Communication is $O(\sqrt(n))$ instead of n.

The above can be extended to ask for k-bit strings easily (just ask one bit at a time).

PIR-based nearest neighbor search

- Divide the region of interest into a Voronoi diagram.
  - Property: the nearest neighbor of any query point in a Voronoi region is in the same region.
- Superimpose a grid onto it.
- For each grid cell, store all the points of the Voronoi regions that intersect with the cell.



- The user just (privately) asks for the cell content where the user's position is in, and then compute the NN.

References

- Ghinita et al. Private Queries in Location Based SErvices: anonymizers are not necessary. ICDE 2008.
- Yiu et al. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. ICDE 2008.
- Bettini, Mascetti, Wang. Privacy Protection through Anonymity in Location-based Services. In **Digital Privacy: Theory, Technologies, and Practices**, Taylor and Francis, 2007.
- Bettini, Mascetti, Wang. Privacy Issues in Location-based Services. In **Encyclopedia of Geographical Information Science**, Springer, 2007.
- Sergio Mascetti. Privacy Protection through Anonymity in Location-based Services. PhD Dissertation, DICo, Università di Milano, 2007.