Private Queries in Location-Based Services

“New technologies can pinpoint your location at any time and place. They promise safety and convenience but threaten privacy and security”

IEEE Spectrum, July 2003
Motivation

• Big and growing mobile Internet
  – 2.7 B mobile phone users (cf. 850 MM PCs)
  – 1.1 B Internet users, 750 MM access the Internet from phones
  – 419 M mobile phones sold in 1Q 2012 (Source: Gartner)
  – Africa has surpassed North America in numbers of users

• The mobile Internet will be location aware.
  – GPS, Wi-Fi-based, cell-id-based, Bluetooth-based, other
  – A very important signal in a mobile setting!
Location-Based Services (LBS)

- Location-based services
  - Location-based store finders
  - Location-based traffic reports
  - Location-based advertisements

- LBS users
  - Mobile devices with GPS capabilities

- Queries
  - Nearest Neighbor (NN) Queries

- Location-based services rely on the *implicit* assumption that users agree on revealing their *private* user locations
- Location-based services *trade* their services with privacy
Query Location Privacy

- A mobile user wants nearby points of interest.
- A service provider offers this functionality.
  - Requires an account and login
- The user does not trust the service provider.
  - The user wants location privacy.
Problem Statement

• Queries may disclose sensitive information
  – Query through anonymous web surfing service

• But user location may disclose identity
  – Triangulation of device signal
  – Publicly available databases
  – Physical surveillance

• How to preserve *query source anonymity*?
  – Even when exact user locations are known
Service-Privacy Trade-off

• Example:
  • Where is my nearest bus?
Spatial K-Anonymity: Spatial Cloaking

- $k$NN query ($k=1$)
- $K$ anonymity
- Range $k$NN query
  - Anonymizing spatial regions (ASR)
  - User hides among $K-1$ users
  - Probability of identifying user $\leq 1/K$

- Candidate set is $\{p_1, \ldots, p_6\}$
- Result is $p_1$
K-Anonymity in LBS: Architecture

K-Anonymity in LBS: Architecture

Location-based Database Server

Privacy-aware Query Processor

Location Anonymizer

1: Query + Location Information
2: Query + blurred Spatial Region
3: Candidate Answer
4: Candidate Or Exact Answer

Third trusted party that is responsible for \textit{blurring} the exact location information.
The New Casper

• Each mobile user has her own privacy-profile that includes:
  • $K$ – A user wants to be $k$-anonymous
  • $A_{min}$ – The minimum required area of the blurred area
  • Multiple instances of the above parameters to indicate different privacy profiles at different times

<table>
<thead>
<tr>
<th>Time</th>
<th>$k$</th>
<th>$A_{min}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>1</td>
<td>___</td>
</tr>
<tr>
<td>5:00 PM</td>
<td>100</td>
<td>1 sq mile</td>
</tr>
<tr>
<td>10:00 PM</td>
<td>1000</td>
<td>5 sq miles</td>
</tr>
</tbody>
</table>

Large $K$ and $A_{min}$ imply stricter privacy requirement
**Location Anonymizer: Grid-based Pyramid Structure**

- The system area is divided into grids at multiple levels in a quad-tree-like manner
  - Level $h$ (root at level 0) has $4^h$ grids;
  - Each cell is represented as $(cid, N)$ where $N$ is the number of mobile users in cell $cid$
- The Location Anonymizer incrementally keeps track of the *number of users* residing in each grid.

**Location update** $(uid, x, y)$

- If $cid_{old} = cid_{new}$ done
- else (a) update new cell identifier in hash table; (b) update counters in both cells; (c) propagate changes in counters to higher levels (if necessary)
- New user – (a) create new entry in hash table; (b) counters of all affected cells increased by 1
- User departs – (a) remove entry; (b) decrease counters by 1
Location Anonymizer: Grid-based Pyramid Structure

Cloaking Algorithm
- Blur the query location
- Traverse the pyramid structure from the bottom level to the top level, until a cell satisfying the user privacy profile is found.

- Let $K=2$
- If $u_3$ queries, ASR is $A_1$ or $A_2$
  (if the area > $A_{\text{min}}$) otherwise ...
Location Anonymizer: Grid-based Pyramid Structure

Cloaking Algorithm

• Traverse the pyramid structure from the bottom level to the top level, until a cell satisfying the user privacy profile is found.

Let \( K = 3 \)

• If any of \( u_1, u_2, u_3 \) queries, ASR is \( A_1 \)

• If \( u_4 \) queries, ASR is \( A_2 \)

• Disadvantages:
  • High location update cost
  • High cloaking cost
Adaptive Location Anonymizer

- Each sub-structure may have a different depth that is adaptive to the *environmental changes* and *user privacy requirements*
  - Stricter privacy requirements => higher level
    - All users at the higher level have strict privacy requirements that cannot be met by the lower level
Adaptive Location Anonymizer

- **Cell Splitting:** A cell $cid$ at level $i$ needs to be split into four cells at level $i+1$ if there is at least one user $u$ in $cid$ with a privacy profile that can be satisfied by some cell at level $i+1$.
  - Need to keep track of most relaxed user $u$ for each cell
  - If newly arrived user, $v$, to cell has a more relaxed profile than $u$
    - If splitting cell can satisfy $v$’s requirement, split and distribute content to the 4 children cells; otherwise, replace $u$ by $v$
  - If $u$ departs, need to find a replacement
- **Cell Merging:** Four cells at level $i$ are merged into one cell at a higher level $i-1$ only if all users in the level $i$ cells have strict privacy requirements that cannot be satisfied within level $i$.
  - Need to keep track of most relaxed user $u$ for the 4 cells of level $i$
  - If $u$ departs, find $v$ to replace $u$. If $v$’s requirement is stricter than can be handled by the 4 cells, then merge them
  - If $v$ enters cell at level $i$, we replace $u$ if necessary

Same cloaking algorithm applies at the lowest existent levels.
The Privacy-aware Query Processor

• Embedded inside the location-based database server
• Process queries based on cloaked spatial regions rather than exact location information
• Two types of data:
  – Public data. Gas stations, restaurants, police cars
  – Private data. Personal data records
• Three types of queries
  – Private queries over public data, e.g., What is my nearest gas station?
  – Public queries over private data, e.g., How many cars in the downtown area?
  – Private queries over private data, e.g., Where is my nearest friend?
• Focus on the first query type
Private Queries over Public Data: Naïve Approaches

• Complete privacy
  – The Database Server returns all (or a sufficiently large superset that contains the answer) the target objects to the Location Anonymizer
  – High transmission cost
  – Shifting the burden of query processing work onto the mobile user

• Nearest target object to center of the spatial query region
  – Simple but NOT accurate

Location Anonymizer
(The correct NN object is $T_{13}$.)
Private Queries over Public Data: The Casper Scheme

Basic idea:

- Find the smallest bounding region that contains the answer
- Return all points within the region
Private Queries over Public Data: The Casper Scheme

Step 1: Locate four filters

- The NN target object for each vertex
Private Queries over Public Data: The Casper Scheme

**Step 1:** Locate four filters
- The NN target object for each vertex

**Step 2:** Find the middle points
- The furthest point on the edge to the two filters
Private Queries over Public Data: The Casper Scheme

Step 1: Locate four filters

- The NN target object for each vertex

Step 2: Find the middle points

- The furthest point on the edge to the two filters

Step 3: Extend the query range
Private Queries over Public Data: The Casper Scheme

Step 1: Locate four filters
- The NN target object for each vertex

Step 2: Find the middle points
- The furthest point on the edge to the two filters

Step 3: Extend the query range

Step 4: Candidate answer
Private Queries over Public Data: Correctness

• Theorem 1
  – Given a cloaked area $A$ for user $u$ located anywhere within $A$, the privacy-aware query processor returns a candidate list that includes the exact nearest target to $u$.

• Theorem 2
  – Given a cloaked area $A$ for a user $u$ and a set of filter target object $t_1$ to $t_4$, the privacy-aware query processor issues the *minimum possible range query* to get the candidate list.
Casper may compromise location anonymity

- Quad-tree based
  - Fails to preserve anonymity for outliers
  - Unnecessarily large ASR size

Let $K=3$

- If any of $u_1$, $u_2$, $u_3$ queries, ASR is $A_1$
- If $u_4$ queries, ASR is $A_2$
- $u_4$’s identity is disclosed
SpaceTwist: No Cloaking Needed

• Cloaking
  – Requires servers to support “specialized” techniques for processing cloaked queries
  – High communication overheads

• Computes kNN query *incrementally* until client is guaranteed to have accurate results
  – Server supports R-tree, and INN (incremental nearest neighbor) retrieval
  – Simple client-server architecture, i.e., no trusted components

SpaceTwist Concepts

- **Anchor location** $q'$ (*fake* client location)
  - Defines an ordering on the data points
- **Client fetches points from server** (based on $q'$) incrementally
- **Supply space**
  - The part of space explored by the client so far
  - Known by both server and client
  - Grows as more data points are retrieved
- **Demand space**
  - Guaranteed to cover the actual result
  - Known only by the client
  - Shrinks when a “better” result is found
- **Terminate** when the supply space contains the demand space
SpaceTwist

• Input: user location q, anchor location q’ (NOTE: distance between q and q’ affects privacy)

• Client asks server to report points in ascending distance from anchor q’ iteratively
  – Note: server only knows q’ and reported points

• Supply space radius $\tau$, initially 0
  – Distance of the current reported point from anchor q’

• Demand space radius $\gamma$, initially $\infty$
  – Nearest neighbor distance to user (found so far)
  – Update $\gamma$ to $\text{dist}(q,p)$ when a point $p$ closer to $q$ is found

• Stop when $\text{dist}(q,q’) + \gamma \leq \tau$
  – Supply space covers demand space
  – Guarantee that exact nearest neighbor of q has been found
SpaceTwist Example

What client sees

The global view

What server sees
Privacy Analysis

• dist(q, q’) affects degree of privacy
  • If it is small, then few objects will be retrieved (and low cost), but less location privacy is achieved
• What does the server (malicious attacker) know?
  – The anchor location q’
  – The reported points (in reporting order): p_1, p_2, ..., p_{m\beta} where \beta is the number of points per packet and m is the number of packets transmitted
  – Termination condition: dist(q,q’) + dist(q,NN) ≤ dist(q’, p_{m\beta})
• Possible query location q_c
  – The client did not stop at point p_{(m-1)\beta} (else packet m is not needed (?))
    • dist(q_c, q’) + min{ dist(q_c, p_i) : i\in[1,(m-1)\beta] } > dist(q’, p_{(m-1)\beta})
  – Client stopped at point p_{m\beta}
    • dist(q_c, q’) + min{ dist(q_c, p_i) : i\in[1,m\beta] } ≤ dist(q’, p_{m\beta})
• *Inferred* privacy region \Psi: the set of all possible q_c
• Quantification of privacy
  – Privacy value: \Gamma(q, \Psi) = the average dist. of location in \Psi from q
  – NOTE: Only user can compute this
Visualization of $\Psi$

- Visualization with different types of points
- Characteristics of $\Psi$ (i.e., possible locations $q_c$)
  - Roughly an irregular ring shape centered at $q'$
  - Radius approx. $\text{dist}(q,q')$
  - $\Gamma(q, \Psi)$ is at least $\text{dist}(q,q')$

- Coarser granularity (low data density)
Privacy Analysis

• By carefully selecting the distance between \( q \) and \( q' \), it is possible to guarantee a privacy setting specified by the user.

• SpaceTwist extension: Instead of terminating when possible, request additional query points.
  – This makes the problem harder for the adversary.
  – It makes it easier (and more practical) to guarantee a privacy setting.
Granular Search

• What if the server considers searching on a small sample of the data points instead of all?
  – Lower communication cost
  – $\Psi$ becomes large at low data density
  – But less accurate results

• Accuracy requirement
  – User specifies an error bound $\varepsilon$
  – A point $p \in P$ is a relaxed NN of $q$ iff
    \[ \text{dist}(q, p) \leq \varepsilon + \min \{\text{dist}(q, p') : p' \in P\} \]

• Granular search
  – Goal: Search at coarser granularity
  – Reduces communication cost; yet guarantees accuracy bound of results
Granular Search

• Given an error bound $\varepsilon$, impose a grid in the space with cell length $\lambda = \frac{\varepsilon}{\sqrt{2}}$

• Slight modification of the incremental NN search
  – Points are still reported in ascending distance order from anchor $q'$
  – But the server discards a data point $p$ if it falls in the same cell of any reported point (never reports more than one data point $p$ from the same cell)

• Incremental granular searching at anchor $q'$
  – Server reports $p_1$, client updates its NN to $p_1$
  – Server discards $p_2, p_3$
  – Server reports $p_4$, client updates its NN to $p_4$

• Outcome: reduced communication cost (from 4 points to 2 points), yet with guaranteed result accuracy
How users choose appropriate parameter values?

• Error bound $\varepsilon$
  – Set $\varepsilon = v_{\text{max}} \cdot t_{\text{max}}$
    • $t_{\text{max}}$: maximum time delay acceptable by user
    • $v_{\text{max}}$: maximum travel speed (walking, cycling, driving)

• Anchor point $q'$
  – Decide the anchor distance $\text{dist}(q, q')$
    • Based on privacy value, i.e., privacy value at least $\text{dist}(q, q')$
    • Based on acceptable value of $m$ (communication)

\[
N_{\varepsilon} = \min\{N, 2k \cdot (U/\varepsilon)^2\} \quad \text{dist}(q, q') = \frac{U}{\sqrt{\pi} \cdot N_{\varepsilon}} \cdot (\sqrt{m\beta} - \sqrt{k})
\]

– $U$ is the extent of the space; $U/(\lambda) = \sqrt{2} \times U/\varepsilon$ is the length of each grid cell; so total number of cells $= 2 \times (U/\varepsilon)^2$; each cell returns at most $k$ points, so we have $N_{\varepsilon}$

– Set the anchor $q'$ to a random location at distance $\text{dist}(q, q')$ from $q$
LBS Privacy with Computational Private Information Retrieval (cPIR)

- Limitations of existing solutions
  - Assumption of trusted entities
    - anonymizer and trusted, non-colluding users
  - Considerable overhead for sporadic benefits
    - maintenance of user locations
  - No privacy guarantees
    - especially for continuous queries (same user issuing the same query in different areas – correlation attack possible for cloaking methods)

- cPIR
  - Two-party cryptographic protocol
    - No trusted anonymizer required
    - No trusted users required
  - No pooling of a large user population required
    - No need for location updates
  - Location data completely obscured
cPIR Overview

- Computationally hard to find $i$ from $q(i)$
- Bob can easily find $X_i$ from $r$ (trap-door)
cPIR Theoretical Foundations

- Let \( N = q_1 \cdot q_2 \), \( q_1 \) and \( q_2 \) large primes

\[
\mathbb{Z}_N^* = \{ x \in \mathbb{Z}_N | \gcd(N, x) = 1 \}
\]

\[
QR = \{ y \in \mathbb{Z}_N^* | \exists x \in \mathbb{Z}_N^* : y = x^2 \mod N \}
\]

- E.g. \( N=5 \cdot 7=35 \), 11 is \( QR \) \((9^2=11 \mod 35)\), 3 is \( QNR \) (no \( y \) exists for \( y^2=3 \mod 35 \))

- Let \( \mathbb{Z}_N^{+1} = \{ y \in \mathbb{Z}_N^* | \left( \frac{y}{N} \right) = 1 \} \) where \( \left( \frac{y}{N} \right) \) is the Jacobi symbol

then exactly half of the numbers are in \( QR \) and the other half in \( QNR \)

- **Quadratic Residuosity Assumption (QRA)**
  - QR/QNR decision computationally hard (if \( q_1 \) and \( q_2 \) are not given)
  - Essential properties:

\[
QR \cdot QR = QR
\]
\[
QR \cdot QNR = QNR
\]
cPIR Protocol for Binary Data

\( N = 35 \)
QNR = \{3, 12, 13, 17, 27, 33\}
QR = \{1, 4, 9, 11, 16, 29\}

public data size: \( n = 16 \)  \( \text{let } t = \sqrt{n} \)

Organize data in a \( t \times t \) (4x4) binary matrix \( M \)

Get \( M_{2,3} \)

Server computes (Server knows \( N \)):

\[
z_i = \prod_{j=1}^{t} y_j \cdot y_j^{1 - M_{i,j}} \pmod{N}
\]

\( M_{i,j} = 0 \)

\( y_j^2 \)

\( M_{i,j} = 1 \)

\( y_j \)

\[
z_2 = 4^2 \times 16 \times 17 \times 11^2 \pmod{35} = 17
\]
cPIR Protocol for Binary Data

\( N=35 \)
\( \text{QNR} = \{3,12,13,17,27,33\} \)
\( \text{QR} = \{1,4,9,11,16,29\} \)

Public data size: \( n = 16 \) \hspace{1cm} \text{let } t = \sqrt{n} \)

Organize data in a \( t \times t \) \((4\times4)\) binary matrix \( M \)

Get \( M_{2,3} \)

Server computes:
\[
z_i = \prod_{j=1}^{t} y_j \cdot y_j^{1-M_{i,j}}
\]

Client computes:
\[
\left( z_a \frac{q_1-1}{2} = 1 \mod q_1 \right) \wedge \left( z_a \frac{q_2-1}{2} = 1 \mod q_2 \right)
\]

If expression is true, then \( Z \) is in QR.

\[
z_2 = \begin{cases} \text{QNR} & \Rightarrow M_{2,3} = 1 \\ \text{QR} & \Rightarrow M_{2,3} = 0 \end{cases}
\]
cPIR protocol for objects

• Same idea for binary data can be easily extended
  • Organize collection of objects as a matrix
  • Conceptually, this is like having \( m \) matrices (assuming each object is represented by \( m \) bits)
  • Server applies the computation on each of these matrices, and \( m \) answer messages will be returned
  • Communication overhead is \( m \) times larger (\( m \cdot \sqrt{n} \))

• \( \text{PIR}(p_i) \) denote user retrieving object \( p_i \) using this protocol
Exact Nearest Neighbor Queries

• Preprocess the data
  – Compute Voronoi tessellation of the set of objects
    • NN of any point within a Voronoi cell is the point enclosed in that cell
  – Superimpose a regular G x G grid on top of the Voronoi diagram
    • For each cell C, determine all Voronoi cells that intersect it; C keeps track of the corresponding objects
    • C contains all potential NNs of every location inside it
Exact Nearest Neighbor

A3: $p_1, p_2, p_3$

A4: $p_1, --, --$
Exact NN

• Query processing
  – User u initiates query
  – Server returns the granularity of the grid \( \sqrt{n} \)
  – u can figure out the cell of the current location, and corresponding column, say b
  – u issues PIR(b) (which is essentially y)
    
    \[
    y = [y_1 : y_{\sqrt{n}}], \quad y_b \in QNR, \quad \text{and } \forall j \neq b, y_j \in QR
    \]
  – From the answers returned, NN of u can be determined
A3: $p_1, p_2, p_3$
A4: $p_1, --, --$

Answer: p4
Exact NN

- Cells may be associated with different number of points
  - “Object” of each cell has different size!
  - Need to “force” them to be the same size, otherwise, server will know which cell $u$ is targeting.
  - Fix the size to the maximum number of data objects, and pad with dummy those cells that have fewer than $P_{\text{max}}$
Exact NN

• Concern
  – Since information of entire column b is returned, potentially reveals to user $\sqrt{n \times P_{\text{max}}}$ points!
  – However, many of these are also duplicates, e.g., D1, D2, D3 and D4 contains only P4
    • Compression can be used to reduce overheads of sending duplicates to user

• Effect of grid size
  – As number of grids increases, communication cost reduces (since $P_{\text{max}}$ decreases); however, beyond certain point, it starts to increase again since it reaches the lower bound (and replication effect kicks in)
  – CPU cost increases with number of grids
Rectangular PIR Matrix

$r < s$ may be beneficial:
- Since “object” size is larger
- For exact NN, user learns fewer other objects
Summary

• LBS services is here to stay
• User privacy needs to be preserved
• Various methods have been developed for user location privacy
  – Spatial K-Anonymity
  – SpaceTwist
  – cPIR
• What else?
  – Continuous queries
  – Road networks
  – ...
