Privately Querying Location-based Services with SybilQuery

Pravin Shankar, Vinod Ganapathy, and Liviu Iftode
Department of Computer Science
Rutgers University

{spravin, vinodg, iftode} @ cs.rutgers.edu
Location-based Services (LBSes)

Implicit assumption:
- Users agree to reveal their locations for access to services
• With two weeks of GPS data from a user’s car, we can infer home address (median error < 60 m) [Krumm ‘07]
• 5% of people are uniquely identified by their home and work locations even if it is known only at the census tract level [Golle and Partridge ‘09]
Querying an LBS

Client

Home

loc_1

loc_2

. .

loc_n

LBS

Work
Basic Idea

Client

Home

loc_1, loc_1', loc_1''

loc_2, loc_2', loc_2''

loc_n, loc_n', loc_n''

Home'

Work

loc_1, loc_1', loc_1''

loc_2, loc_2', loc_2''

loc_n, loc_n', loc_n''

Home''

Work'

Work''

LBS
What the LBS sees

Which of these is the real user?
Outline

- Introduction
- **SybilQuery Overview**
- Design Challenges
- Implementation
- Evaluation and Results
- Conclusions and Future Work
• **Basic Idea:** Achieves privacy using synthetic (Sybil) queries

  • For each real user trip, the system generates
    – k-1 Sybil start and end points (termed endpoints)
    – k-1 Sybil paths

  • For each real query made, the system generates
    – k-1 Sybil Queries
Outline

• Introduction
• SybilQuery Overview
• Design Challenges
• Implementation
• Evaluation and Results
• Conclusions and Future Work
SybilQuery Challenges

• Endpoint generation:
  – How to automatically generate synthetic endpoints similar to a pair of real endpoints?

• Path generation:
  – How to choose the waypoints of the Sybil path?

• Query generation:
  – How to simulate motion along the Sybil path?
Basic design of SybilQuery

Value of $k$

Source & Destination

Endpoint Generator

Original and $k-1$ synthetic endpoints

Path Generator

Path $P$ and $k-1$ synthetic paths

Query Generator

Client location $l$

Location $l$ & $k-1$ synthetic locations

Location based service

Response
• Produces synthetic endpoints that resemble the real source and destination

• High-level idea:
  – Tag locations with features
  – Identify clusters of locations that share similar features

• Feature used in SybilQuery: traffic statistics
Tagging locations with traffic statistics (1/2)

• Naïve approach: Annotate locations with descriptive tags
  – Eg. “parking lot”, “downtown office building”, “freeway”
  – Laborious manual task

• Our approach: Automatically compute features using a database of regional traffic statistics
  – Dataset: Month-long GPS traces from the San Francisco Cabspotter project - 530 unique cabs; 529,533 trips
  – Compute traffic density $\tau_l$ for each location from dataset
Tagging locations with traffic statistics (2/2)

• Locations represented as QuadTree
  – Balances precision with scalability

San Francisco Airport.
Black blocks have higher densities
Finding suitable endpoints using reverse geocoding

- Real endpoints do not start in non-driveable terrain

Random point in geographic location

Reverse Geocoding

Street address closest to the random point
Path Generator

- Consults an off-the-shelf navigation service
  - Our implementation uses Microsoft Multimap API to obtain waypoints
- Users may not always follow the shortest path to destination
  - Detours, road closures, user intention
- Computes multiple paths to the destination (with varying lengths)
- Uses a probability distribution to choose path
• Triggered each time the user queries the LBS
• Simulates the motion of users along the Sybil paths
• Uses current traffic conditions to more accurately simulate user movement
  – Eg. Simulate slower movement if traffic is congested
Outline

• Introduction
• SybilQuery Overview
• Design Challenges
• **Implementation**
• Evaluation and Results
• Conclusions and Future Work
• An interface akin to navigation systems

• Input:
  – The source and destination address for the trip
  – A security parameter $k$
    • Number of Sybil users

• Query interface:
  – Integrated with Yahoo! Local Search
Outline

- Introduction
- SybilQuery Overview
- Design Challenges
- Implementation
- Evaluation and Results
- Conclusions and Future Work
1. Privacy
   • How indistinguishable are Sybil queries from real queries?

2. Performance
   • Can Sybil queries be efficiently generated?
Evaluation: Privacy

• User Study
  – Give the working system to adversarial users, who would try to break the system by find real user paths hidden between Sybil paths
  – 15 volunteers

• Methodology
  – Pick real paths from the Cabspotter traces
  – Use SybilQuery to generate Sybil paths with different values of $k$
# Results from user study

<table>
<thead>
<tr>
<th>$k$</th>
<th># Questions</th>
<th># Correct</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>75</td>
<td>20</td>
<td>0.26</td>
</tr>
<tr>
<td>6</td>
<td>75</td>
<td>14</td>
<td>0.19</td>
</tr>
</tbody>
</table>
User approaches to distinguish queries

• Contrasting rationale to guess real users
  – “Circuitous paths”
  – “Prominent start/end location”
  – “Odd man out”
Evaluation: Performance

• Setup:
  – Server:
    • 2.33 GHz Core2 Duo, 3 GB RAM, 250 GB SATA (7200 RPM)
  – Client:
    • 1.73 GHz Pentium-M laptop, 512 MB RAM, Linux 2.6
  – Privacy parameter $k = 4$ (unless otherwise specified)
  – Each experiment repeated 50 times

• Micro-benchmarks
  – One-time and once-per-trip costs
  – Query-response latency of SybilQuery

• Comparison with Spatial Cloaking
One-time and once-per-trip costs

- One-time cost – preprocessing of traffic database
  - 2 hours 16 mins (processed 529,533 trips)
- Once-per-trip costs – endpoint generation and path generation

<table>
<thead>
<tr>
<th>Task</th>
<th>Average Time (sec)</th>
<th>St dev (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint generation</td>
<td>5.47</td>
<td>1.02</td>
</tr>
<tr>
<td>Path generation *</td>
<td>0.36</td>
<td>0.15</td>
</tr>
</tbody>
</table>

* Includes network latency to query the Microsoft MultiMap API
Query-response latency of SybilQuery

- Scales linearly with $k$ (number of Sybil users)
- Sub-second latency for typical values of $k$
• Spatial Cloaking – k-anonymity solution that uses anonymizers

• Users send their location to anonymizer
• Anonymizer computes cloaked region
  – Region where at least k users are present
Performance Comparison with Spatial Cloaking

Response Size as users travel

- Cloaked regions grow as users travel
- SybilQuery overhead constant
Related Work

- Synthetic Locations for Privacy [Krumm ’09, Kido ‘05]
- Spacial Cloaking [Gruteser and Grunwald ’03, and others]
- Peer-to-peer Schemes [Chow ’06, Ghinita ‘07]
- Private Information Retrieval (PIR) [Ghinita ’08]

Detailed list is available in paper
Conclusions and Future Work

- SybilQuery: Efficient decentralized technique to hide user location from LBSes
- Experimental results demonstrate:
  - Sybil queries can be generated efficiently
  - Sybil queries resemble real user queries
- Future Work
  - Enhance SybilQuery to achieve stronger privacy guarantees, such as l-diversity, t-closeness and differential privacy
Thank You!