Shortest Path Computation with No Information Leakage
(K. Mouratidis and Man Lung Yiu VLDB 2012)

Evripidis Paraskevas (ECE Dept. UMD)
03/12/2014
Introduction (1)

- Location Based Services (LSB) have become increasingly popular (Google Maps, Waze, MapQuest etc)
Common query from mobile users: Shortest path on road net
Motivation

- Disclosure of source and destination to LBS reveals personal information:
  - Social Habits
  - Health Status
  - Shopping needs/preferences
  - Lifestyle choices

- Possible misuse for commercial profiling, government surveillance, advertising etc
Related Work (1)

- Obfuscation Method (Lee et al. CIKM’ 09)
  - LBS obtains knowledge of a finite set of alternatives for s and t
  - Computes the SP for each s and t pair
  - LBS have a clue about the composition of the SP and the query
Related Work (2)

- Private Information Retrieval (PIR)
  - Retrieve data items hosted to a server without the server learning anything about the clients’ access patterns
  - Privacy relies on reduction to problems that are either computationally infeasible or theoretically impossible to solve
Objective and General Approach

- Our Objective is **strong privacy**
  - The LBS learns nothing about the query, including any knowledge about s, t and the path

- Approach:
  - Use PIR building blocks, provide cryptographic privacy guarantees

- Hardware-aided PIR protocols: practical and secure
Hardware-aided PIR

- Practical PIR = hardware-aided PIR [Williams & Sion: *Usable PIR*. NDSS’ 08]

- Secure co-processor (SCP):
  - Interface to securely fetch specific disk images (i.e. make retrievals) from a database
  - Constant Communication and amortized polylogarithmic computation cost $O(\log^2 N)$, i.e. approx. 1 sec to fetch 1GB database
System Model

• Architecture:
  • Road Network is modeled as a weighted graph $G=(V,E)$
  • LBS hosts the road network (graph) and stores it on disk
  • LBS knows (or even owns) the road network data

• Adversary
  • Location Based Service (LBS) (e.g. Google Maps)
  • Adversary is curious but not malicious

• Security Objective: LBS processes and answers user queries but learns nothing about the queries
Methodology

- Establish SSL connection and use PIR interface (e.g. SCP) to fetch disk pages
- Make sure that every query follows the same query plan, i.e. same number of rounds, in each round accesses the same files in the same order and from each file it retrieves the same number of pages.
- PIR security guarantees + same query plan [S. Papadopoulos et al., IEEE TKDE 2010] ⇒ queries indistinguishable from each other
Design Considerations for PIR

- Accesses still 2 orders of magnitude slower than (unprotected) disk reads
  - Calls for smart data organization
  - Keep the maximum no. of pages required for any possible SP query as small as possible (processing cost reduction)

- PIR protocol of Williams & Sion imposes limit on maximum file size
  - In paper setting is 2.5GB
  - Essential to keep space requirements small in order PIR to operate
Approach and Performance Evaluation

- Create different Schemes based on the described methodology that will differ to:
  - Data organization (files that store the relevant data for SP)
  - Sequence of Retrievals that need to be made (using PIR interface)

- Performance trade-off to examine:
  - Query response time (the less retrievals the better)
  - Space Overhead (e.g. size of files)
Shortcomings of Common SP methods

- Consider Dijkstra’s Algorithm or A* search (for shortest path computation) and that LBS stores raw network data

- Using the methodology described:
  1. Identical query plan requirement ⇒ every query has same cost as costliest possible shortest path query
  2. For some source-destination pairs, the above algorithms access (almost) entire road net.

- \(1+2\) bound all queries to incur cost equivalent to accessing all pages in the database.
Solution

- Pre-processing steps:
  - Network Partitioning
  - Pre-computation of SPs !!! (extremely important)

- Smart Data Organization Schemes to store the results

- Efficient Query Plan followed by all the queries

- DISK SPACE vs PROCESSING COST
Baseline Schemes

- **Landmark (LM) [A.V. Goldberg et al., SODA 2005] adaptation:**
  - Partition G into regions.
  - For each region $R_i$ we allocate one disk page for information regarding all its nodes.
  - Fetch pages for $R_s$ and $R_t$ in the first round.
  - Initializes $A^*$ search to find shortest path and at each round fetch the page of the next node encountered.
  - Large number of page Retrievals $\Rightarrow$ Large Response Time

- **Arc-flag (AF) [E. Kohler et. al, DIMACS 2006] adaptation:**
  - Partition G into regions
  - For each region $R_i$ we allocate a fixed number of pages (retrieved together during query processing).
  - Space needed to store all the paths computed exceeds the max. supported by PIR interface.
Concise Index Scheme (CI)

- **Key Idea:**
  - partition network into regions
  - materialize information for region-to-region paths (for each pair or source/destination region, store which the possible intermediate regions are)

- **Database comprises of 4 files:**
  - Region Data file (information about region of the network)
  - Network Index file (pre-computation information)
  - Look-up file (index of Network file)
  - Header (query plan and map of source and destination)
CI: Network Partitioning

- KD-tree partitioning
  - Space-partitioning data structure for organizing points in a k-dimensional space
  - KD-tree packing method with guarantees about space utilization
  - Network information for each region = one page of Region Data File
CI: Pre-Computation

- **Border nodes**: intersections of edges (road segments) with KD-tree splitting lines

- For each pair $R_i, R_j$:
  - compute *all* SPs from BNs of $R_i$ to those of $R_j$ and…
  - store IDs of intermediate regions *in any* of these SPs into region set $S_{ij}$

- Any path from $R_i$ to $R_j$ may pass only via regions in $S_{ij}$

- Region sets are stored in **Network Index file**
CI: Pre-Computation Example

- $S_{1,8}$ will include (among others) $R_3$, $R_4$, $R_7$
CI: Look-up file and Header file

- **Look-up file** = dense index on Network Index file

(i,j) value  |  page no.
---|---
(1,1)|1  |  (1,2)|1  |  (1,3)|1  |  (1,4)|1  |  (1,5)|2  |  ...

Look-up File

Page 1  |  Page 2  |  Page 3
---|---|---
$S_{1,1}$,$S_{1,2}$,$S_{1,3}$,$S_{1,4}$  |  $S_{1,5}$,$S_{1,6}$,$S_{1,7}$  |  $S_{1,8}$,$S_{2,1}$  ...

**Header file**: stores KD-tree, mapping from regular IDs to pages in Region Data file, metadata.
CI: Query Processing

- Download entire Header (no need for PIR)
  - Client learns source and destination regions $R_s, R_t$

- Read page in Look-up for entry $(s,t)$
  - Client learns which page in Network Index file hosts region set $S_{st}$

- Read that page from Network Index file
  - Client learns $S_{st}$, i.e., all possible intermediate regions

- From Region Data file, read pages for $R_s, R_t$, and all regions in $S_{st}$
  - Client learns sub-graph guaranteed to contain the SP and computes it using Dijkstra’s algorithm in this subgraph.
CI: Performance

- Let $m$ be the maximum cardinality between the region sets $S_{ij}$
- Suppose that $S_{ij}$ fits in one page of Network Index file
- No. of PIR accesses =
  - 1 in Look-up
  - 1 in Network Index
  - $2 + m$ in Region Data [dummy requests if needed to fill up the number of PIR accesses]

- Observations:
  - Network Index File is concise $\Rightarrow$ small space overhead
  - On the other hand for large networks, $m$ can be large $\Rightarrow$ PIR cost may be an issue (large number of PIR accesses)
Passage Index Scheme (PI)

- Key Idea: materialize more information to improve performance in terms of query response time

- Files and pre-processing (network partitioning and pre-computation) similar to CI, but
  - **Network Index file** keeps for every $R_i, R_j$ an **edge set** $G_{i,j}$ (instead of region set $S_{i,j}$)
  - **Edge set $G_{i,j}$** stores all **edges** in SPs between the border nodes of $R_i, R_j$
PI: Example of Network Index File

- $G_{1,8}$ will include (among others) red and blue edges
PI: Performance

- Let $h$ be the **maximum number of pages** that a $G_{ij}$ takes in the Network Index file.

- No. of PIR accesses =
  - 1 in Look-up
  - $h$ in Network Index
  - 2 in Region Data (*for source and destination region*)

- In the case of a small network $h = 1$ and only 4 PIR accesses are needed.

- Observations:
  - Drastic Reduction in the number of pages needed (compared to CI) and thus the response time
  - Space overhead is larger than CI (due to the Network Index File size)
Schemes for Large road Networks

- In large road networks space exceeds maximum space supported by PIR protocol
- Propose two modified schemes to reduce the space size:
  1. HYBRID:
     - Start with the index of CI
     - Iteratively identify the $S_{i,j}$ with maximum cardinality and replace it with the respective edge set $G_{i,j}$
     - Stop when space limit is reached
     - Network Index File and Region Data File should concatenate to one file (LBS cannot distinguish if the query was answered by region set or subgraph)
  2. PI*: allocate more than one page per region
     - This reduces no. of regions and no. of Border Nodes ⇒ Smaller Network Index file
Experiments

<table>
<thead>
<tr>
<th>Location</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany (Ger.)</td>
<td>28,867</td>
<td>30,429</td>
</tr>
<tr>
<td>Argentina (Arg.)</td>
<td>85,287</td>
<td>88,357</td>
</tr>
<tr>
<td>Denmark (Den.)</td>
<td>136,377</td>
<td>143,612</td>
</tr>
<tr>
<td>India (Ind.)</td>
<td>149,566</td>
<td>155,483</td>
</tr>
<tr>
<td>North America (Nor.)</td>
<td>175,813</td>
<td>179,179</td>
</tr>
</tbody>
</table>

Table 2: System specifications

<table>
<thead>
<tr>
<th>System parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk page size</td>
<td>4 KByte</td>
</tr>
<tr>
<td>Disk seek time</td>
<td>11 ms</td>
</tr>
<tr>
<td>Disk read/write rate</td>
<td>125 MByte/s</td>
</tr>
<tr>
<td>SCP read/write rate</td>
<td>80 MByte/s</td>
</tr>
</tbody>
</table>
Performance on Different road Networks

- Implement all the schemes LM, AF, CI and SI
- Observe that CI is 3 times faster than baseline schemes
- PI is 5 times faster than baseline schemes
- PI exceeds space limit for larger nets
Performance on Very Large Networks

- PI* is more efficient method than the hybrid approach (HY) in terms of Response time.

### (a) Response time (s)

- CI
- HY
- PI*

### (b) Space requirements (MB)

- CI
- HY
- PI*
Conclusion and Future Work

- Proposed the first schemes for private shortest path computation that
  - Offer strong privacy guarantees
  - Are readily deployable

- Future work:
  - Although performance is reasonable for that degree of privacy, overheads are still significant
  - How to reduce overheads?
    - Network compression (lossless/lossy), taking into account their structure
    - Approximate SP schemes?
Thank you!!
Questions???