Location Privacy in LBS (Part II)

Ken (Man Lung Yiu)

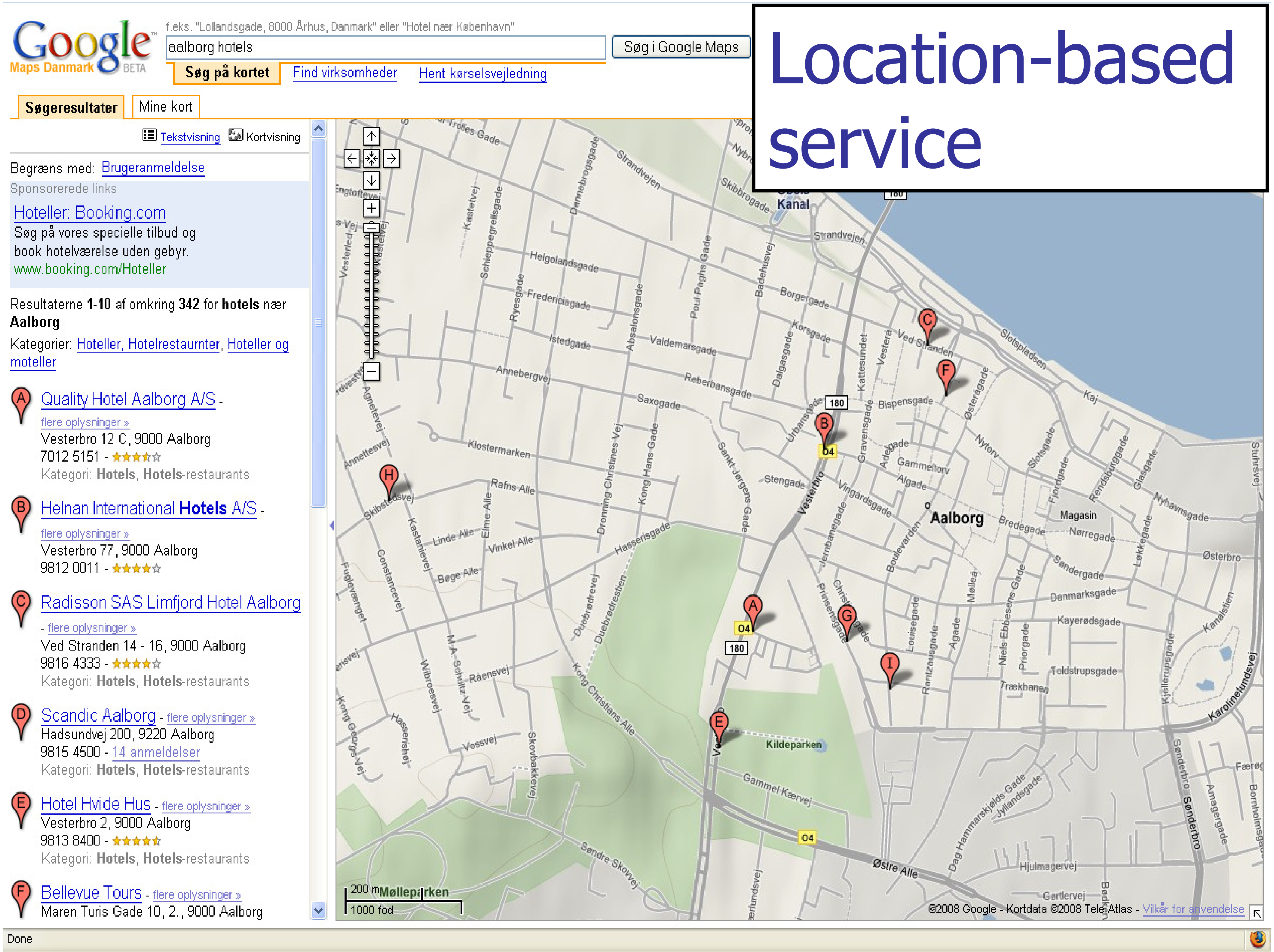
Department of Computer Science
Aalborg University

AALBORG UNIVERSITET

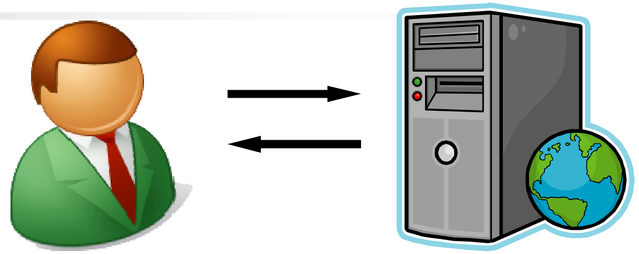
Outline

- Motivation of location privacy
- Privacy model
- K-anonymity
- Transformation-based matching
- SpaceTwist

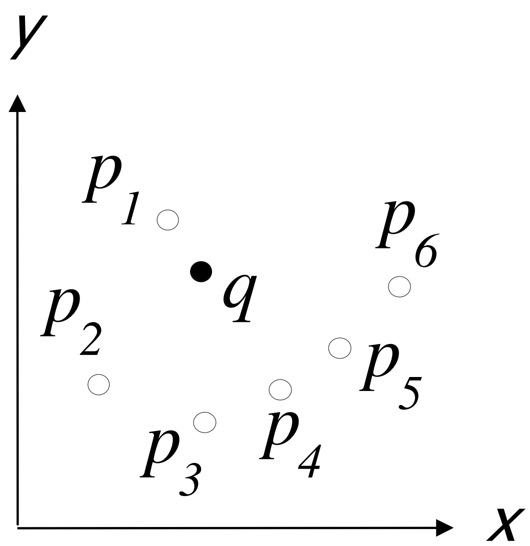
Location-based service



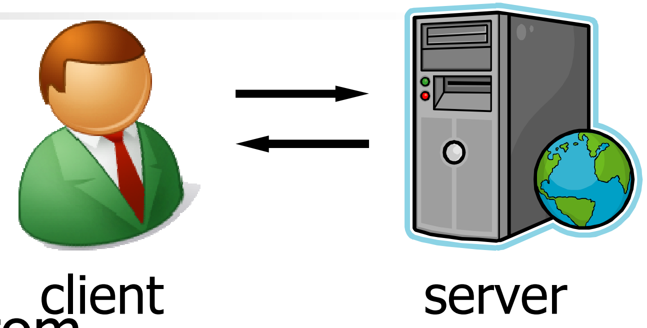
Why location privacy?



- Queries in location-based services (LBS)
  - POI Points-of-interest (e.g., cinema locations)
  - Nearest neighbor (NN) query
    - Find the closest POI to user location q
- Client-server architecture
  - Client (user) sends the point q to the LBS server
  - Server reports the result (i.e., p_1) back to client
- Danger: server may not be trusted



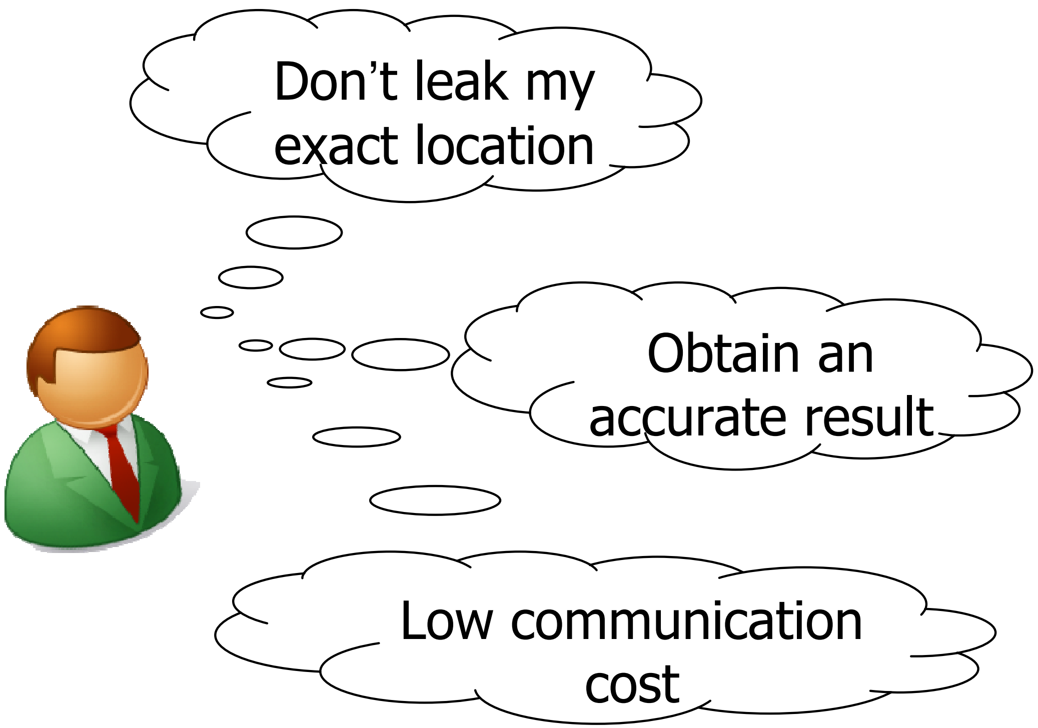
Baseline solutions

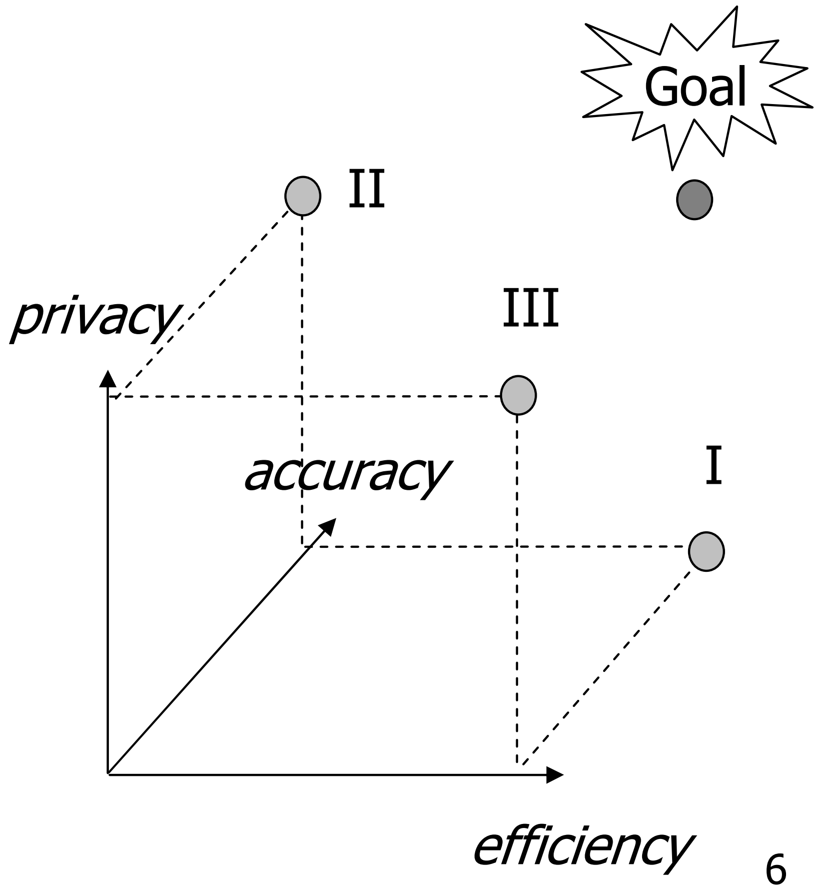


- Baseline I: original query
  - Idea: issue the original query to the LBS
  - Good: Low (optimal) amount of data received from the server
  - Problem: the server knows the user location directly
- Baseline II: brute-force data transfer
  - Idea: request the LBS to send all data points
  - Good: the server has no information of the user's location
  - Problem: high communication cost

Baseline solutions

- Baseline III: sample data transfer
  - Idea: request the LBS to send only a sample of data points
  - Good: low communication cost, the server has no information of the user's location
  - Problem: inaccurate result





Privacy model

- Someone proposes a location privacy solution (say, method X)
- How much privacy does X provide?
- Need a privacy model to answer this question
- Privacy model
  - Assumption(s) of what the attacker knows
    - E.g., knowledge of user locations
  - The “amount” of privacy
    - E.g., number of “other” users in a region

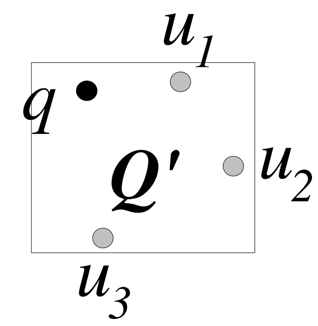
Attacker's knowledge

- Knowledge of user locations
  - A powerful attacker such as Telecom company, government
  - K-anonymous region [Mokbel et al. 2006]
  - K-sharable region [Kalnis et al. 2007], in case the attacker knows the exact anonymization method
  - Full domain anonymity [Khoshgozaran et al., 2007], in which the user can be anywhere in the domain (e.g., no location information)
- No knowledge of user locations, only knows the query issued by the user
  - A weak attacker such as a hacker exploiting a server
  - Analysis of possible query locations constrained by the method [Yiu et al. 2008]

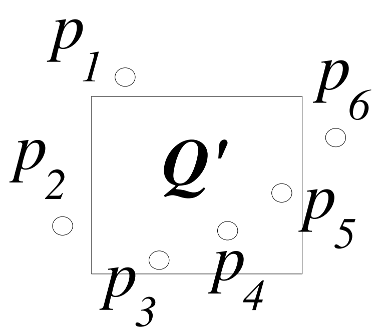
K-anonymity

Spatial cloaking

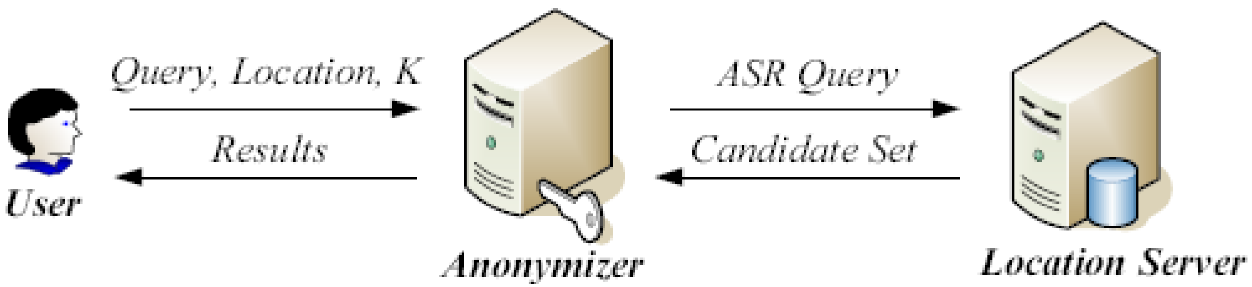
- **K-anonymous region**: a region that contains the query user location q at least (K–1) other user locations
- Spatial cloaking
  - Typical architecture: trusted anonymizer
  - Step 1: Anonymizer computes a K-anonymous region Q′ (cloaked region) of the query point q
  - Step 2: Anonymizer sends Q′ to the location server
  - Step 3: Server computes a candidate result set that contains the result of <u>any possible</u> query location in Q′
    - Example: candidate set: {p_1, p_2, p_3, p_4, p_5, p_6}
  - Step 4: Anonymizer computes the actual result from the candidate result set returned from the location server



Anonymizer

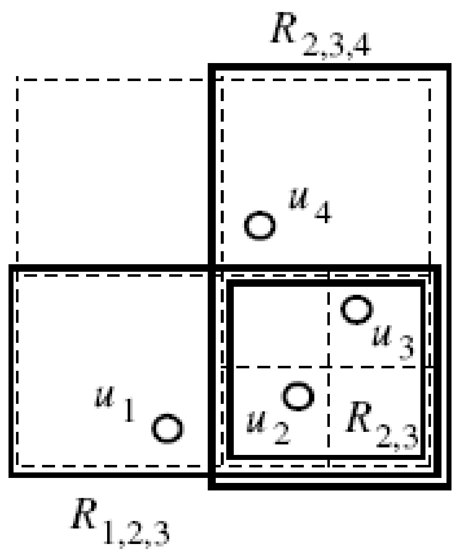


LBS server



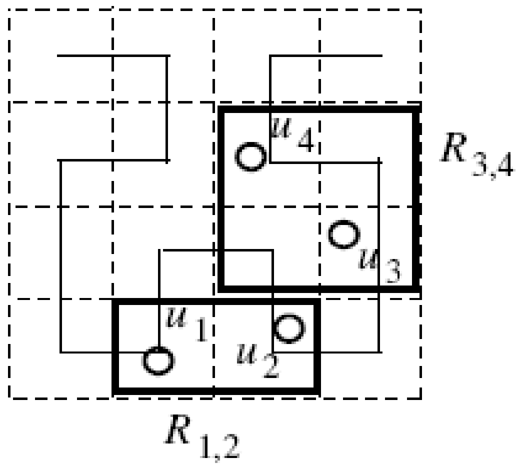
Spatial cloaking

- Most of the solutions in this category focus on Step 1, i.e., computing the cloaked region
- [Mokbel et al. 2006] uses a quadtree to index user locations at anonymizer
- When a user q issues a query, the anonymizer finds a quadtree node (or two adjacent nodes) that contains q and at least K-1 users
- Consider that K=2 in this example
  - The user u_1 obtains the cloaked region $R_{1,2,3}$
  - Both users u_2 and u_3 obtains the cloaked region $R_{2,3}$
  - Problem: the attacker knows that u_1 is the only one using the region $R_{1,2,3}$



Spatial cloaking

- **K-sharable region**: a cloaked region R is shared by at least K users
  - Better privacy protection than K-anonymous region
- [Kalnis et al. 2007] proposes to rearrange user locations at anonymizer in ascending order of their Hilbert values $\mathbb{H}(p)$
  - 1^{st} – K^{th} users form a group
  - $(K+1)^{st}$ – $(2K)^{th}$ users form a group
  - ……
  - cloaked region of a user: minimum bounding rectangle of cells in the group
- Consider that K=2 in the example of Fig. a
  - Both u_1 and u_2 share the same cloaked region $R_{1,2}$
  - Both u_3 and u_4 share the same cloaked region $R_{3,4}$



Spatial cloaking

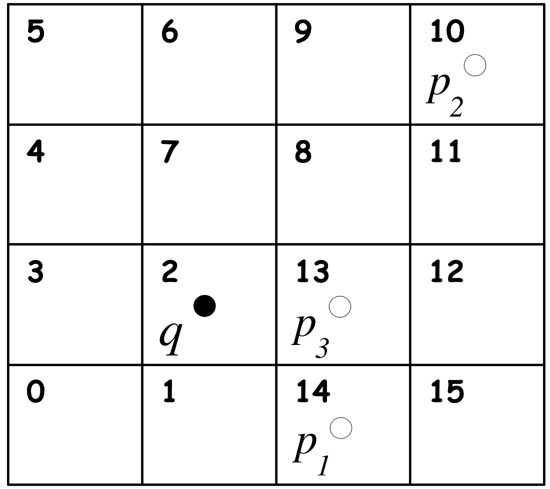
- Advantage
  - Provides strong privacy guarantee even if the attacker knows all user locations in the space
- Disadvantages
  - Drawbacks of using a trusted anonymizer
    - Single point of failure, performance bottleneck
    - How do we know that the anonymizer can be trusted?
  - Location update
    - Even if users are not issuing queries, they need to report their locations constantly to the anonymizer
  - Query processing
    - High processing and communication cost at the server
    - Complex algorithms, not readily implemented in LBS servers

Transformation-based matching

Transformation-based matching

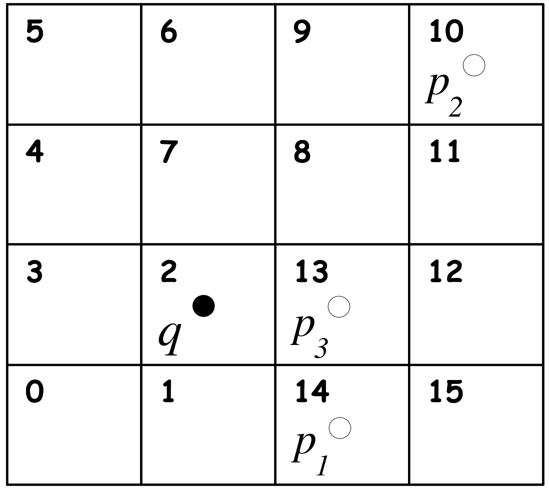
- Avoid drawbacks of using a trusted anonymizer (discussed before)
- Transformation-based matching
  - Typical architecture: client-server model only
  - Trusted entities can be used by data owner and query users
    - For transformation 2D points into "meaningless" 1D values
      - E.g., location (3,5) → value 18 ; location (4,6) → value 13
  - Let the server evaluates the query blindly (without seeing any points)
    - Challenge: the server needs to compute "distances" between those values such that they reflect the distances between their original locations
  - **Full domain anonymity**: if the transformation function is irreversible by the attacker, then the attacker cannot distinguish significant difference between the mapped values of two different locations

Transformation-based matching

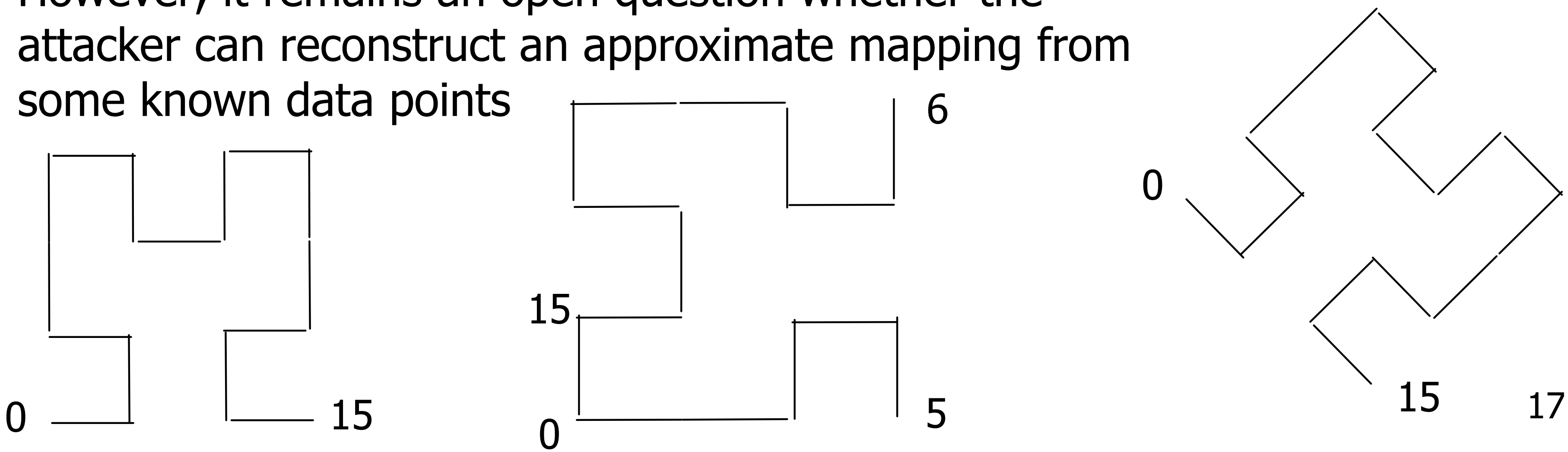


- Hilbert transformation [Khoshgozaran et al., 2007]
  - Hilbert curve: a space filling curve
  - $\mathbb{H}$(q): computes the Hilbert value of the location q
- Preprocessing step
  - a **trusted entity** converts each point p (e.g., restaurant) to the value $\mathbb{H}$(p), uploads it to server
  - $p_1 \rightarrow 14$, $p_2 \rightarrow 10$, $p_3 \rightarrow 13$
- Query time
  - **client** sends $\mathbb{H}$(q) to server, which reports the closest Hilbert value to $\mathbb{H}$(q); then client decodes the reported value into the result location
  - $q \rightarrow 2$; the server retrieves the closest value (10)
  - The client applies the inverse function $\mathbb{H}^{-1}$ to decode the value 10 back to the location p_2
- Features: low result size, but no accuracy guarantee

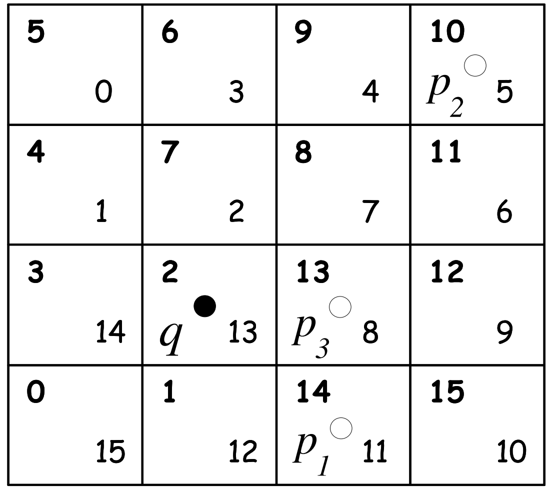
Why we need a key?



- **Danger**: If the same function H(q) is always used, then the attacker will eventually find out this
- In practice, the function is used together with a key value *SK* , known only by client and a trusted entity
- This key consists of these parameters:
  - starting point, curve orientation, scale factor, ……
- The authors claim that there is exponential combinations of parameters to obtain the exact key
  - However, it remains an open question whether the attacker can reconstruct an approximate mapping from some known data points



Double Hilbert Curve



- Using a single Hilbert curve (default)
  - The returned object p_2 is far from the actual result p_3
- Using double (orthogonal) Hilbert curves
  - Preprocessing step is done for each function
    - E.g., p_1 is converted to the values 14 and 11
  - Query step is performed for each function
    - E.g., q is converted to the values 2 and 13
    - Get the nearest value (10) of 2, i.e., obtain p_2
    - Get the nearest value (11) of 13, i.e., obtain p_1
    - The client choose the closest point (p_1) to be the final result
  - Better accuracy, but still no guarantee of finding the exact result

Transformation-based matching

- Advantages
  - No need to use trusted anonymizer
  - The attacker only sees some unreadable 1D values, but not any locations
- Disadvantages
  - Need a preprocessing step
  - No guarantee the return of exact results
  - The attacker may be able to deduce an approximation of the function if the distribution of data points in the dataset is known

SpaceTwist

A Realistic Question

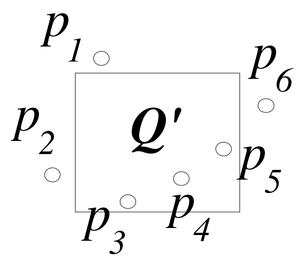
- Does the service provider want to implement these functionalities?
  - High cost on execution
  - Do not want others to upload meaningless 1-d values
  - Burden on implementation/testing

- We need to find an acceptable solution for both users and service providers!

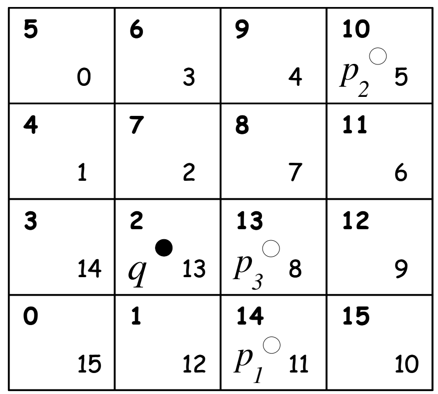
Google Maps

No way!

Cloaked query processing



Transformed query processing

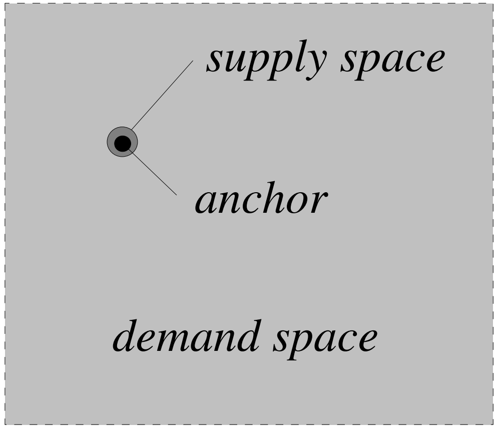


Features of our solution

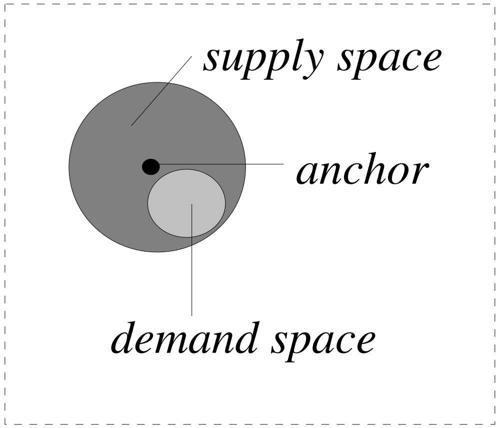
- Our solution: SpaceTwist [Yiu et al. 2008]
  - retrieves POI's from the server *incrementally*
  - until the client is guaranteed to have accurate results
- Fundamental differences from previous approaches
  - *No cloaked region* (unlike spatial cloaking)
  - Query evaluated in the *original space* (unlike transformation approaches)
- Readily applicable on existing systems
  - Simple client-server architecture (i.e., NO trusted components)
  - Simple server-side query processing: incremental nearest neighbor search [Hjaltason et al. 1999]

SpaceTwist: overview

- Anchor location (*fake* client location)
  - Define an ordering of points in the space
- Client fetches points from server *incrementally*
- Supply space (color: ♦)
  - The space of objects retrieved from the server
    - Supply space known by both server and client
  - Grows as more objects retrieved
- Demand space (color: ♦)
  - The target space guaranteed to cover the actual result
    - Demand space known only by client
  - Shrinks when a "better" result is found
- Termination: supply space contains the demand space



the beginning

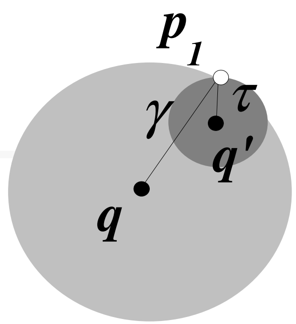


the end

Transmission of points

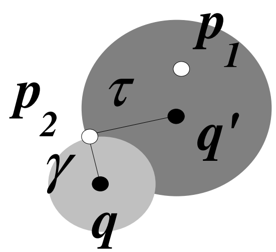
- Communication cost (via the Web)
  - Points are sent from server to client through (TCP/IP) packets
  - Cost: number of packets sent from the server
- Each packet can store up to β points
- Value of the packet capacity β?
  - Depends on Maximum Transmission Unit (MTU)
  - Our experiments: MTU=576 bytes, and β=67

SpaceTwist: example

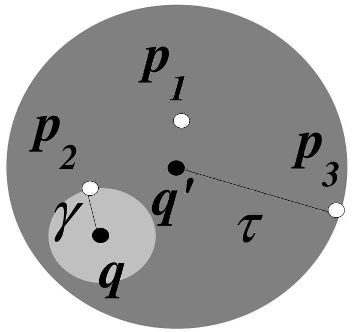


1st point

- Input: user location q, anchor location q'
- Client asks server to report points in ascending distance from anchor q' iteratively [Hjaltason et al. 1999]
  - Note: server only knows q' and reported points
- Supply space radius τ, initially 0
  - Distance of the current reported point from anchor q'
- Demand space radius γ, initially ∞
  - Nearest neighbor distance to user (found so far)
  - Update γ to dist(q,p) when a point p closer to q is found
- Stop when dist(q,q') + $\gamma \leq \tau$
  - Supply space covers demand space
  - **Guarantee** that exact nearest neighbor of q has been found

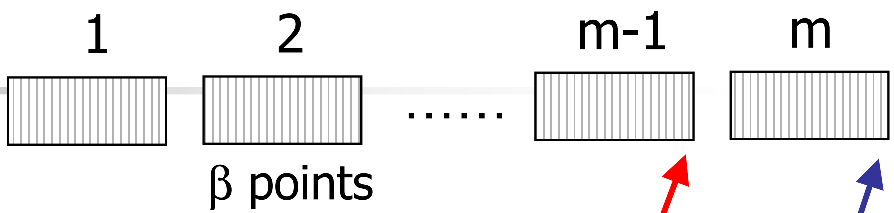


2nd point



3rd point

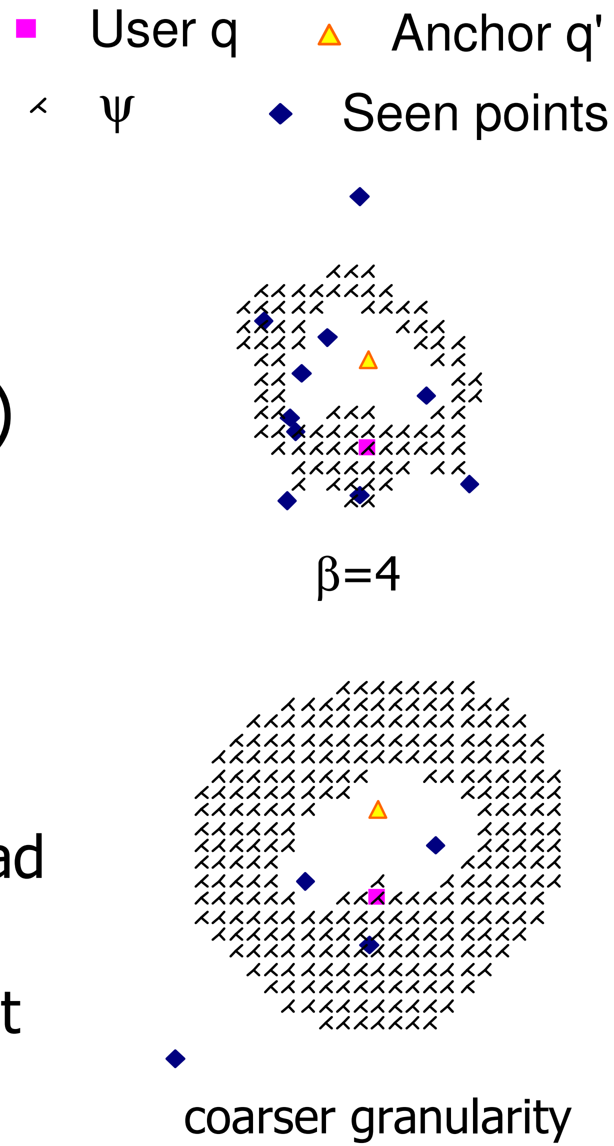
Privacy analysis



- What does the server (malicious attacker) know?
  - Anchor location q’
  - Reported points (in reported order): $p_1, p_2, \ldots, p_{m\beta}$
  - Our termination condition: $dist(q,q') + \gamma \leq \tau$
- A possible query location q_c must satisfy both:
  - Client did not stop at the point $p_{(m-1)\beta}$
    - $dist(q_c, q') + \min\{ dist(q_c, p_i) : i \in [1,(m-1)\beta] \} > dist(q', p_{(m-1)\beta})$
  - Client stops at the point $p_{m\beta}$
    - $dist(q_c, q') + \min\{ dist(q_c, p_i) : i \in [1,m\beta] \} \leq dist(q', p_{m\beta})$
- *Inferred* privacy region Ψ: the set of all possible q_c

Visualization of Ψ

- Quantification of privacy
  - Privacy value: $\Gamma(q, \Psi)$ = average dist. of location in Ψ from q
- Features of Ψ (i.e., possible locations q_c)
  - A ring with center at q'
  - Radius approximately equal to dist(q,q')
- Trade-off: improve the communication cost by reducing the result accuracy
  - E.g., the server searches on *a* ***sample*** instead of the whole dataset
  - Challenge: control the **accuracy** of the result

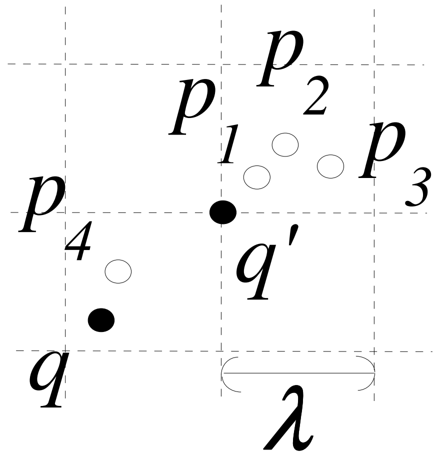


Granular search requirement

- Accuracy requirement
  - User specifies an error bound ε
  - A point $p \in P$ is a relaxed NN of q if
    $$dist(q, p) \leq \varepsilon + \underbrace{\min \{ dist(q, p') : p' \in P \}}_{\textit{Actual NN distance}}$$
- Granular search (optional server-side functionality)
  - Goal: search POI's at coarser granularity
  - Reduces communication cost and yet guarantees accuracy bound of results
    - Spatial cloaking incurs high communication cost at the server
    - Transformation approach does not offer result accuracy guarantees

Granular search

- Given an error bound ε, impose a grid in the space with cell length $\lambda = \varepsilon / \sqrt{2}$
- Slight modification of the incremental NN search [Hjaltason et al. 1999]
  - 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
- 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, yet with guaranteed result accuracy



regular grid

Parameter tuning guide

- Determine appropriate parameter values for the user
- Error bound ε
  - Set $\varepsilon = v_{max} \cdot t_{max}$ based on
    - t_{max}: maximum time delay acceptable by user
    - v_{max}: maximum travel speed (walking, cycling, driving)
- Anchor point q'
  - Decide the anchor distance dist(q, q')
    - Based on privacy value, i.e., privacy value at least dist(q, q')
    - Or, based on acceptable value of m (communication cost)

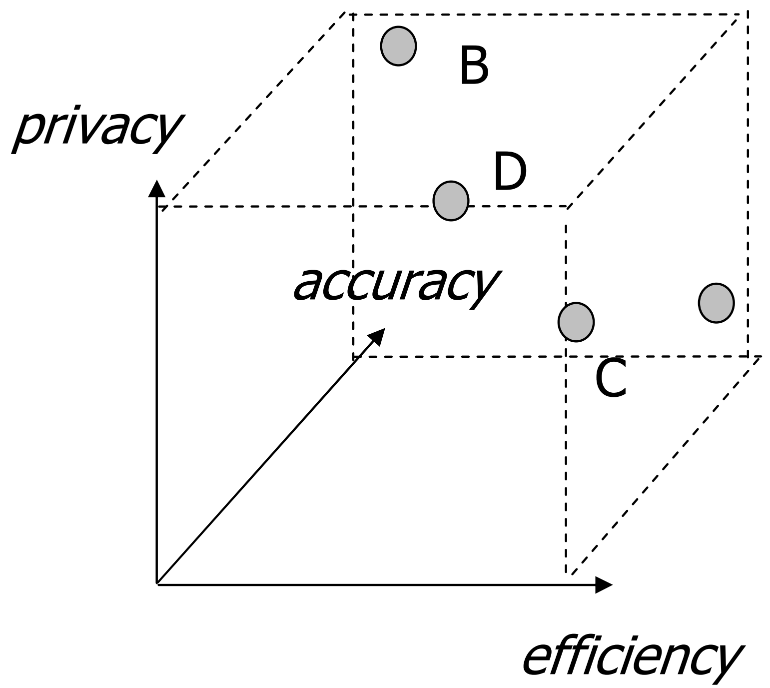
$$N_\epsilon = \min\{N, 2k \cdot (U/\epsilon)^2\} \qquad dist(q, q') = \frac{U}{\sqrt{\pi \cdot N_\epsilon}} \cdot (\sqrt{m\beta} - \sqrt{k})$$

  - Set the anchor q' to a random location at distance dist(q, q') from q

Tradeoff in SpaceTwist

- Error bound: ε
- Anchor distance: dist(q',q)

- A: low ε, low dist(q',q)
- B: low ε, high dist(q',q)
- C: high ε, low dist(q',q)
- D: high ε, high dist(q',q)



Experimental study

- Our solution: Granular SpaceTwist (GST)
  - Client-side: SpaceTwist client algorithm
  - Server-side: Granular search algorithm
- Performance metrics (workload size=100)
  - Communication cost (in number of packets)
  - Measured Result error (result NN distance – actual NN distance)
  - Privacy value of *inferred* privacy region Ψ
- Real spatial data: SC (172K points), TG (556K points)
- Default parameter values
  - Anchor distance dist(q,q'): 200
  - Error bound ε: 200

GST vs. transformation approach

- Hilbert transformation [Khoshgozaran et al., 2007]
  - SHB: single Hilbert curve
  - DHB: two orthogonal Hilbert curves
- GST computes result with low error
  - Low error on real data (skewed) distribution
- Communication cost (not shown here)
  - DHB transfers 2k Hilbert values (fit in one packet)
  - GST needs 1-3 packets for most of the tested cases (see later)

kNN search:
k is the number of required results

	Error (metre)								
	UI, N=0.5M			*SC*			*TG*		
k	**SHB**	**DHB**	**GST**	**SHB**	**DHB**	**GST**	**SHB**	**DHB**	**GST**
1	7.1	2.2	51.3	1269.3	753.7	2.5	1013.9	405.8	16.1
2	9.3	4.0	49.0	1634.3	736.2	2.6	1154.6	548.7	16.7
4	13.2	6.0	47.6	1878.5	810.9	2.6	1182.3	596.5	17.0
8	19.0	7.3	42.0	2075.6	864.5	2.6	1196.2	599.7	16.3
16	27.0	10.3	36.3	2039.6	985.7	2.6	1199.6	603.2	14.5

Domain length = 10000

result error, at ε=200

GST vs. spatial cloaking

- Our problem setting: no trusted third-party middleware/components
- Competitor: client-side spatial cloaking (CLK)
  - CLK: enlarge q into a square with side length 2*dist(q,q'), i.e., its extent is comparable to inferred privacy region Ψ of GST
- GST produces result at low communication cost
  - Low cost even at high privacy
- Result accuracy (not shown here)
  - CLK always provides exact results
  - Result error of GST bounded by ε, and much lower than ε in practice

varying dist(q,q')

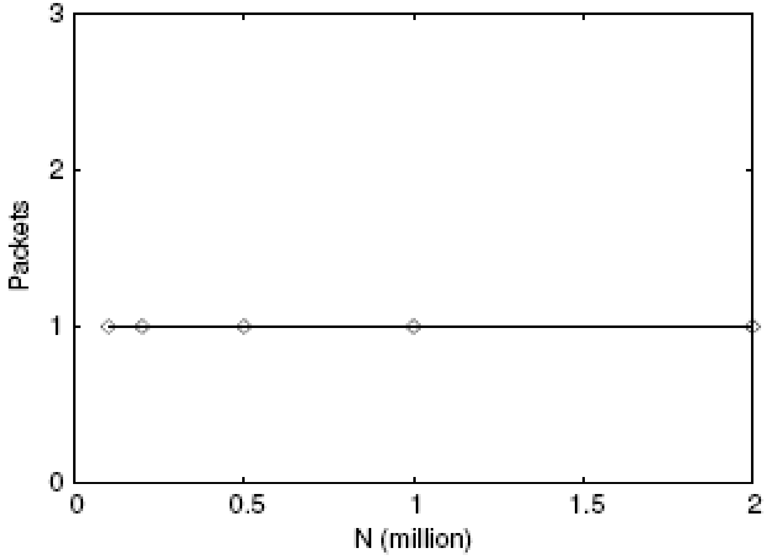
$dist(q,q')$	*SC* CLK	*SC* GST	*TG* CLK	*TG* GST
50	1.3	1.0	1.9	1.0
100	2.0	1.0	4.6	1.0
200	6.2	1.0	15.0	1.0
500	33.5	1.1	72.8	1.3
1000	107.0	1.4	282.0	2.6

N (million)	*UI* CLK	*UI* GST
0.1	3.0	1.0
0.2	5.1	1.0
0.5	12.2	1.0
1	23.9	1.0
2	47.5	1.0

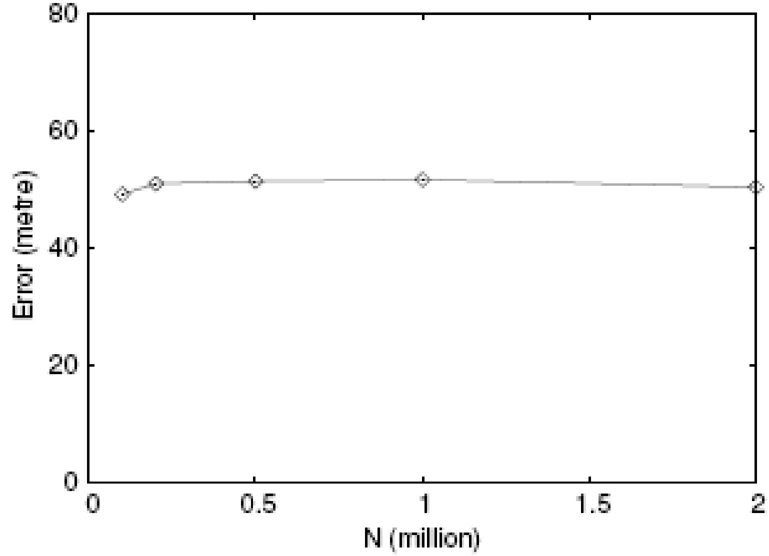
varying data size N

communication cost (# of packets)

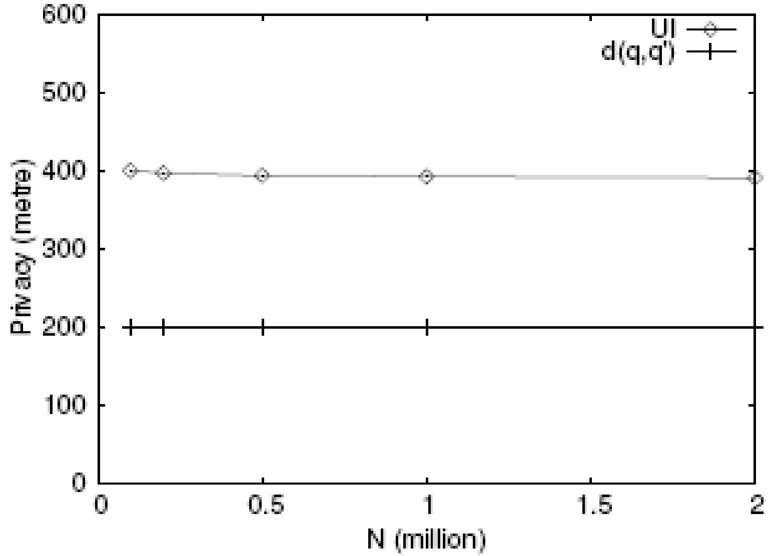
Effect of data size N (million) [Synthetic uniform data]



communication cost



result error



privacy value

SpaceTwist Summary

- Advantages
  - Readily applicable on existing systems (e.g., no trusted anonymizer, no transformation of points)
  - Allow the user to control result error (with guarantee)
  - Enable tradeoff among result error, communication cost, privacy value
- Disadvantage
  - The privacy model is not as strong as K-anonymity

Conclusion

- Privacy model
- K-anonymity
- Transformation-based matching
- SpaceTwist

References

M. F. Mokbel, C.-Y. Chow, and W. G. Aref.
The New Casper: Query Processing for Location Services without Compromising Privacy.
In *VLDB*, 2006.

P. Kalnis, G. Ghinita, K. Mouratidis, D. Papadias,
Preventing Location-Based Identity Inference in Anonymous Spatial Queries.
IEEE TKDE, 19(12), 1719-1733, 2007.

A. Khoshgozaran and C. Shahabi.
Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy.
In *SSTD*, 2007.

M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu.
SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services.
In I*CDE, 2008.*

G. R. Hjaltason and H. Samet.
Distance Browsing in Spatial Databases.
ACM TODS, 24(2):265–318, 1999.