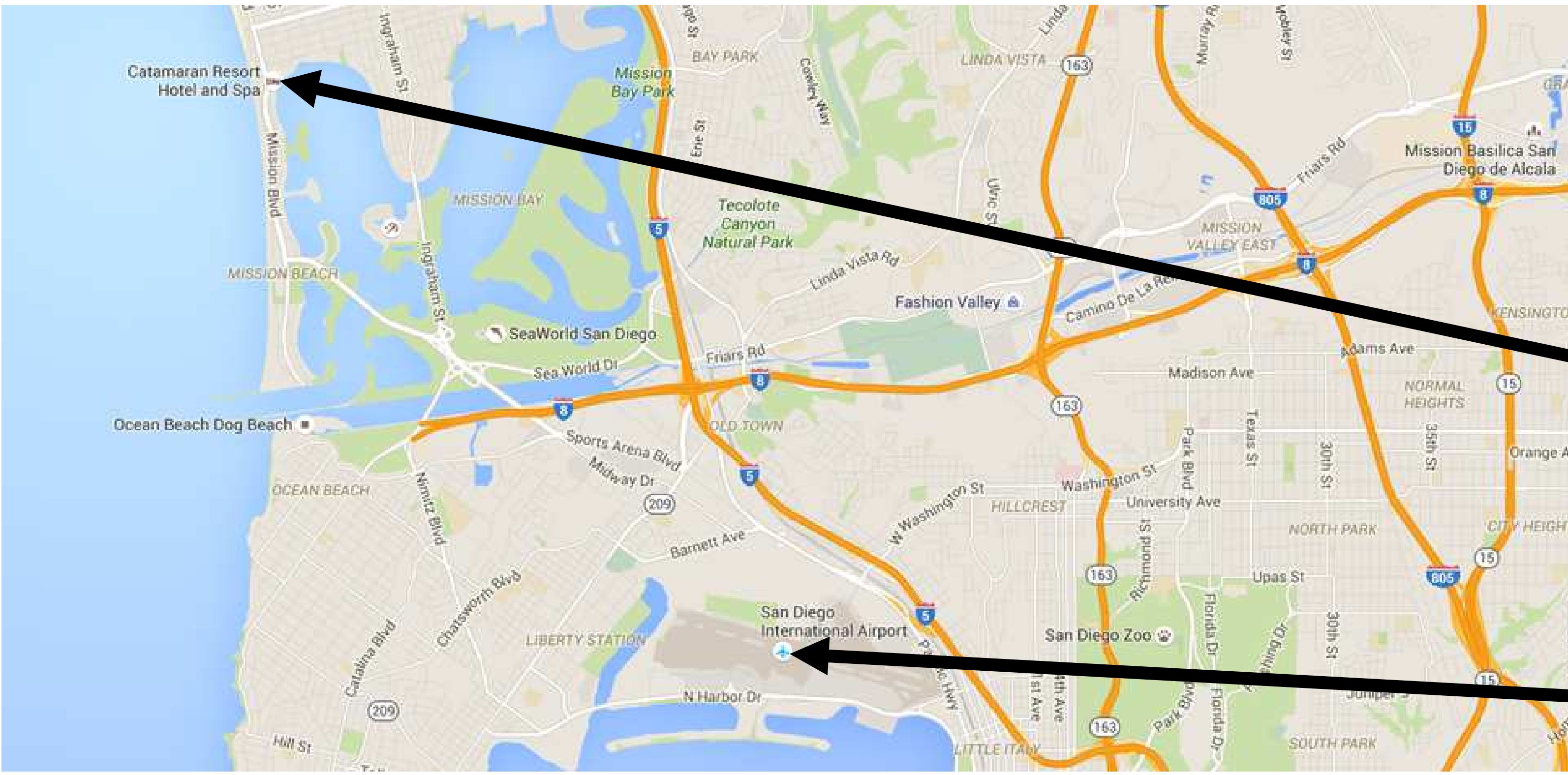
Privacy-Preserving Shortest Path Computation

David J. Wu, Joe Zimmerman, Jérémy Planul, and John C. Mitchell

Stanford University

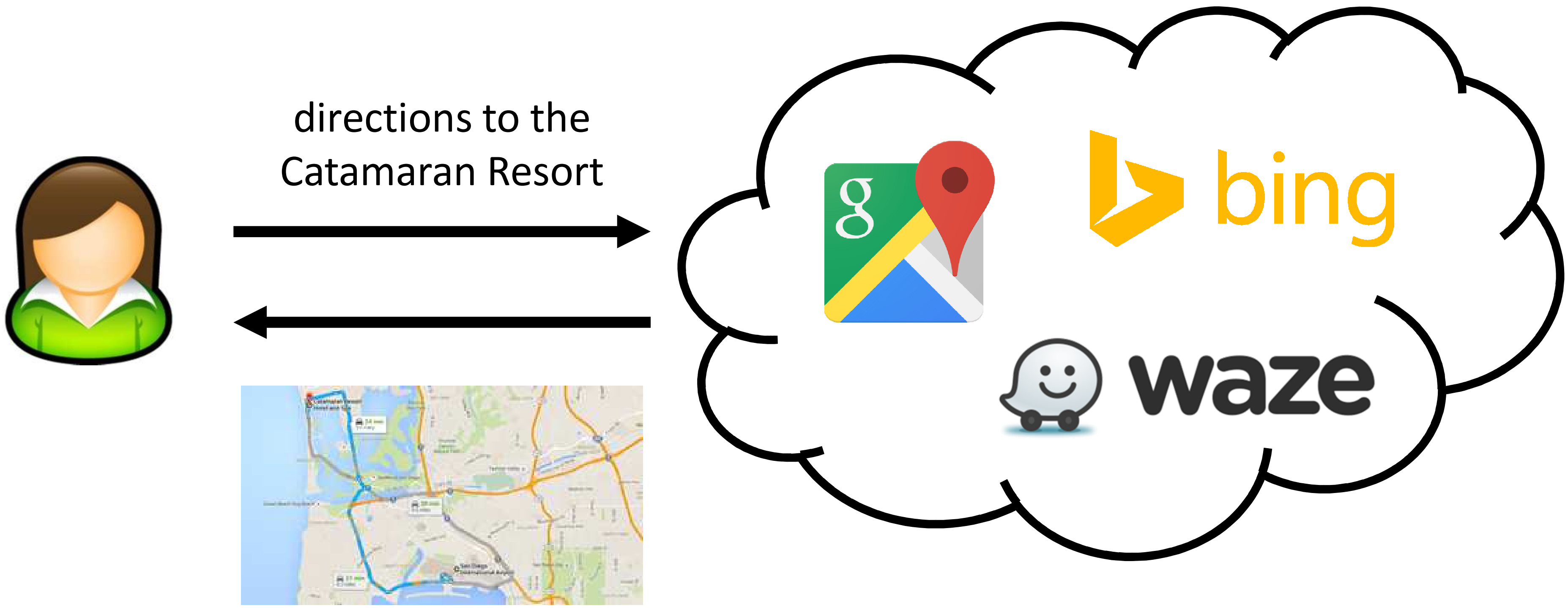
Navigation



desired destination

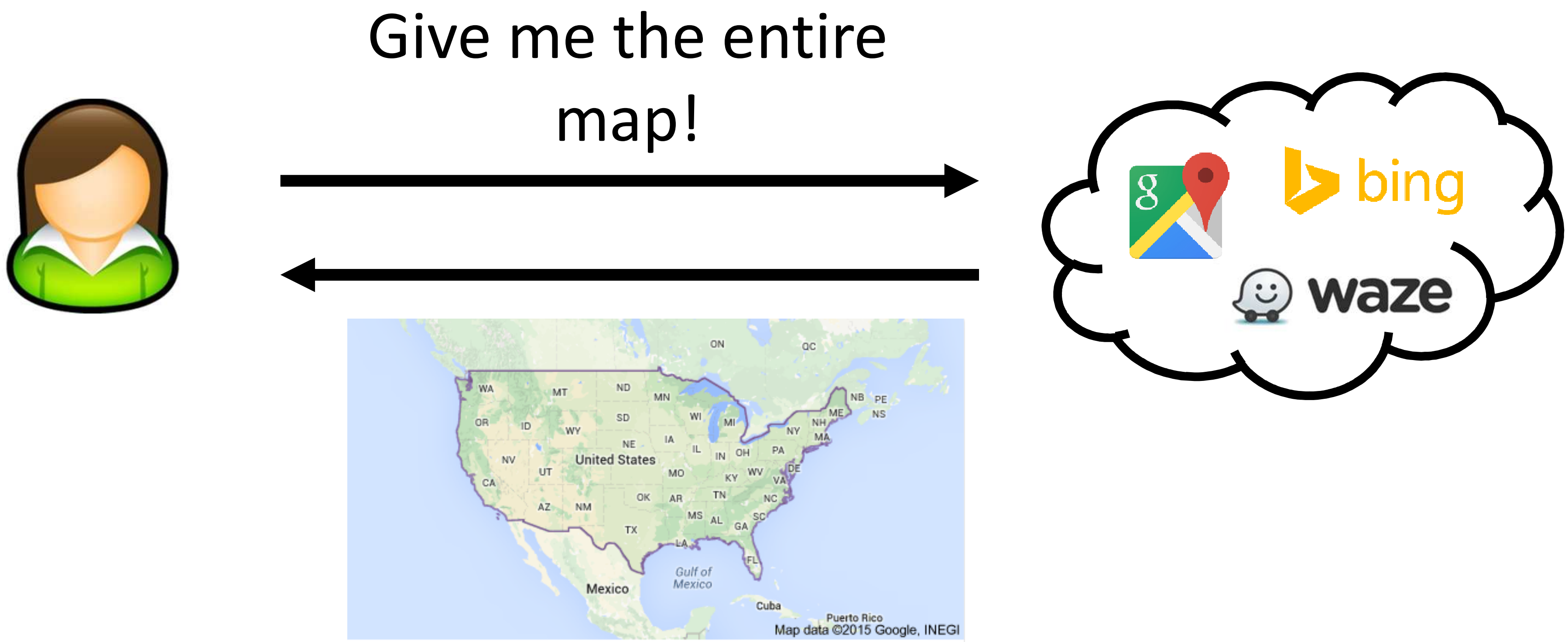
current position

Navigation: A Solved Problem?

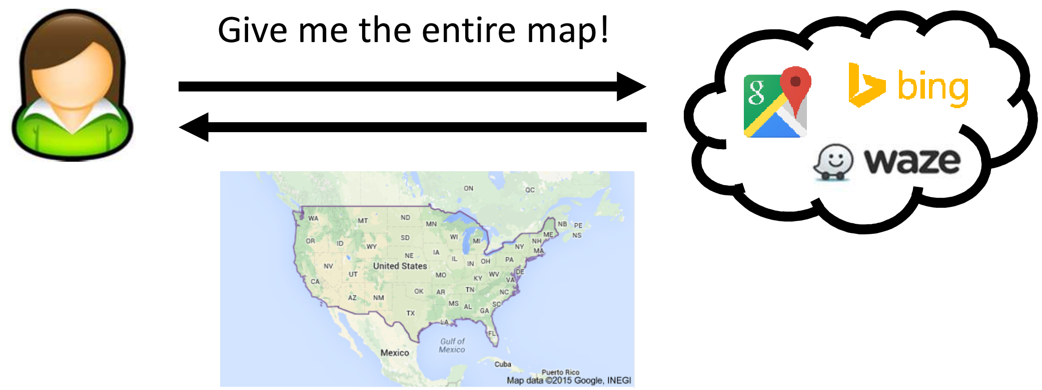


Issue: cloud learns where you are and where you are going!

“Trivial” Solution



“Trivial” Solution

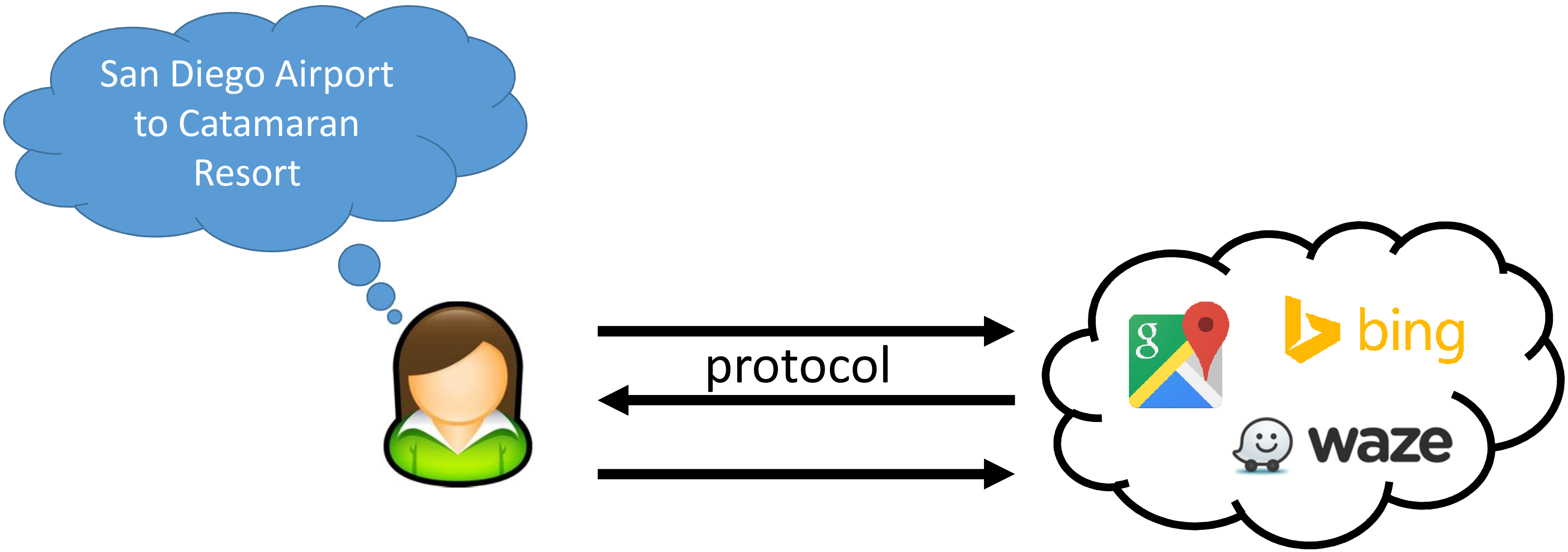


Cons:

- routing information constantly changing
- map provider doesn’t want to give away map for “free”

Pros: lots of privacy (for the client)

Private Shortest Paths



Client Privacy: server does not learn source or destination

Server Privacy: client only learns route from source to destination

Private Shortest Paths

Model: assume client knows topology of the network (e.g., road network from OpenStreetMap)

Weights on edges (e.g., travel times) are **hidden**

Client Privacy: Server does not learn client's source s or destination t

Server Privacy: Client only learns $s \to t$ shortest path and nothing about weights of other edges not in shortest path

Straw Man Solution

Suppose road network has n nodes

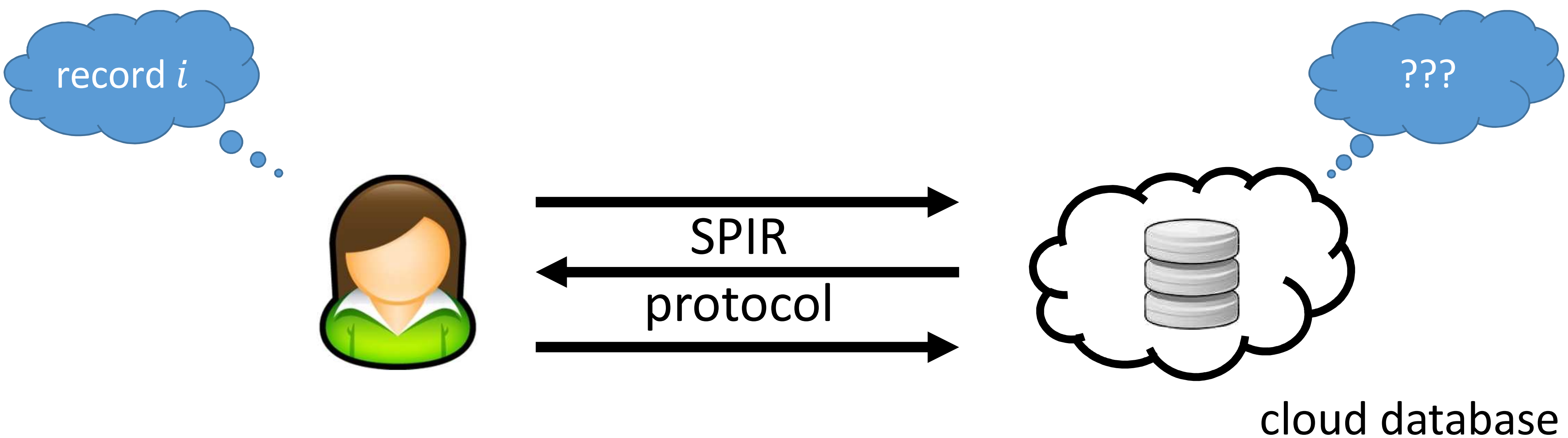
Construct $n \times n$ database:

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{bmatrix}$$

record r_{st}: shortest path from node s to node t (e.g., $s \to v_1 \to v_2 \to t$)

Shortest Path Protocol: privately retrieve record r_{st} from database

Symmetric Private Information Retrieval (SPIR)



Client Privacy: server does not learn i

Server Privacy: client only learns record i

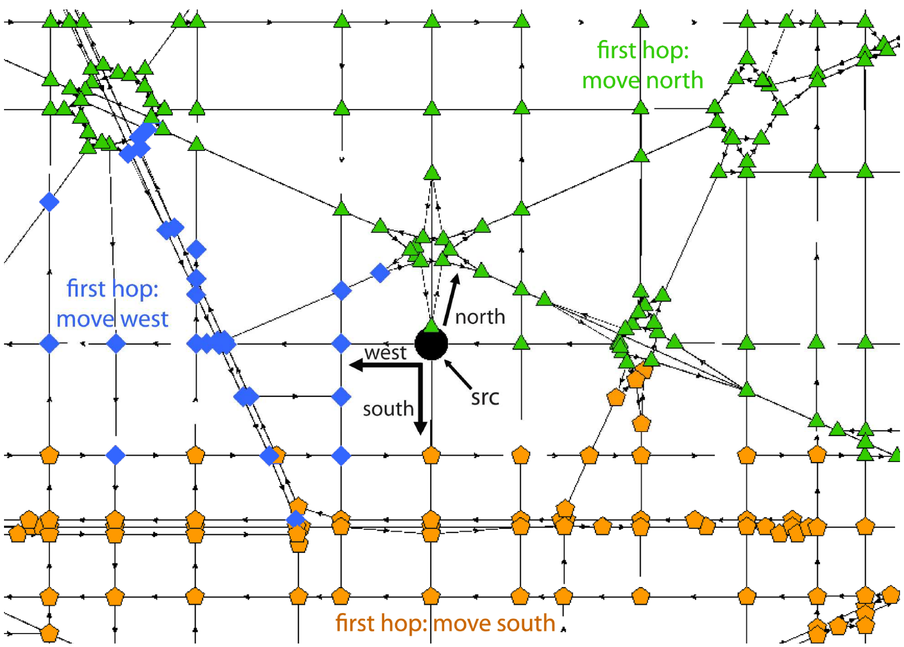
Finding Structure

Straw man solution requires SPIR on databases with $\boldsymbol{n^2}$ records – quadratic in number of nodes in the graph – rather impractical!

Observation 1: Nodes in road networks tend to have low (constant) degree

Finding Structure

Typically, an intersection has up to four neighbors (for the four cardinal directions)



For each node in the network, associate each neighbor with a direction (unique index)

Finding Structure

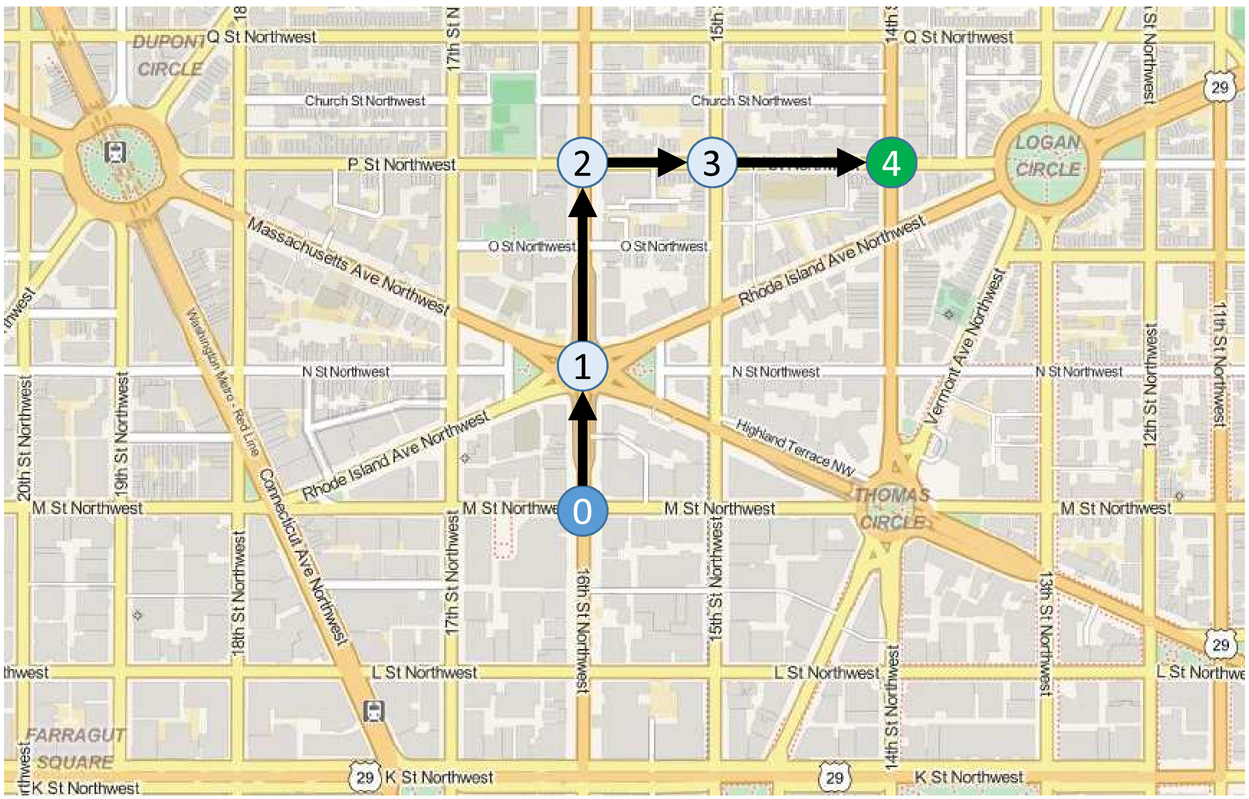
Next-hop routing matrix for graph with n nodes:

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{bmatrix}$$

r_{st}: index of neighbor to take on first hop on shortest path from node s to node t

shortest path protocol: iteratively retrieve the next hop in shortest path

Finding Structure

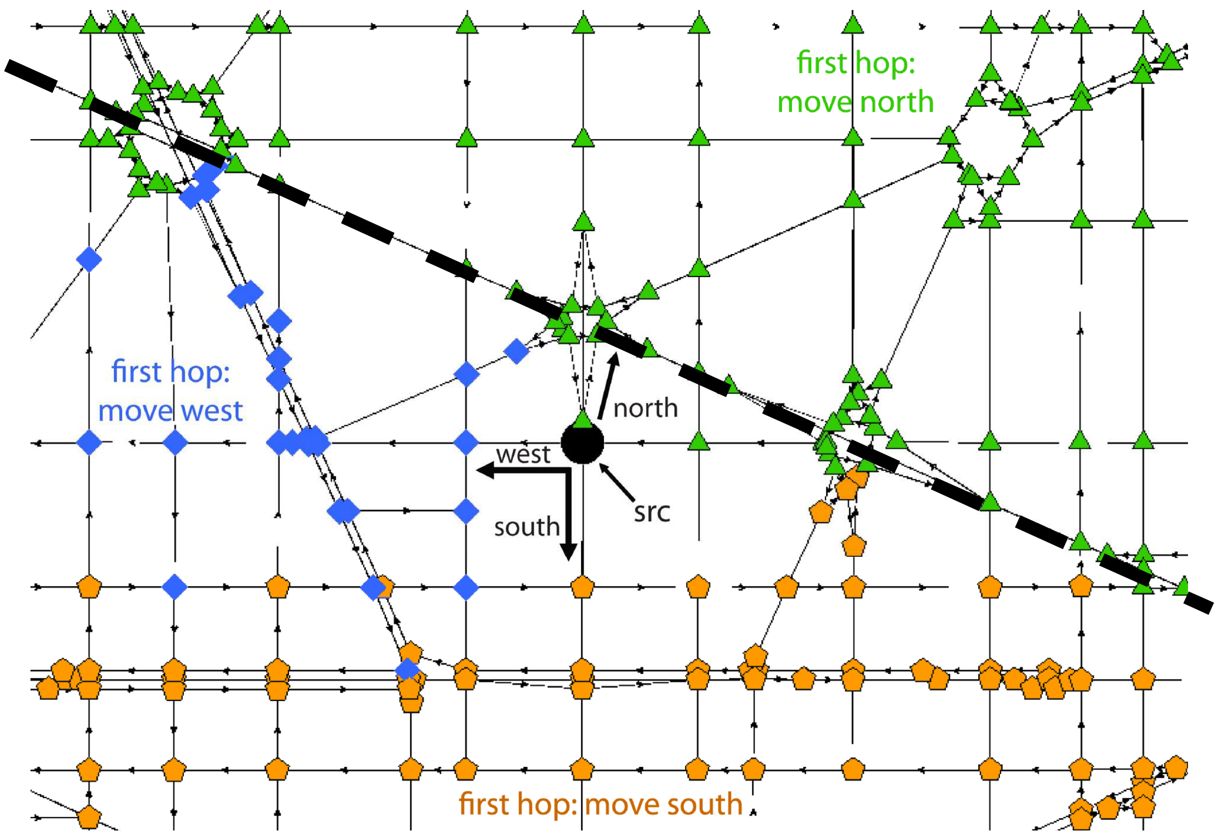


Routing from 0 to 4:

1. Query r_{04}: North
2. Query r_{14}: North
3. Query r_{24}: East
4. Query r_{34}: East

But same problem as before: SPIR on database with n^2 elements

Finding Structure

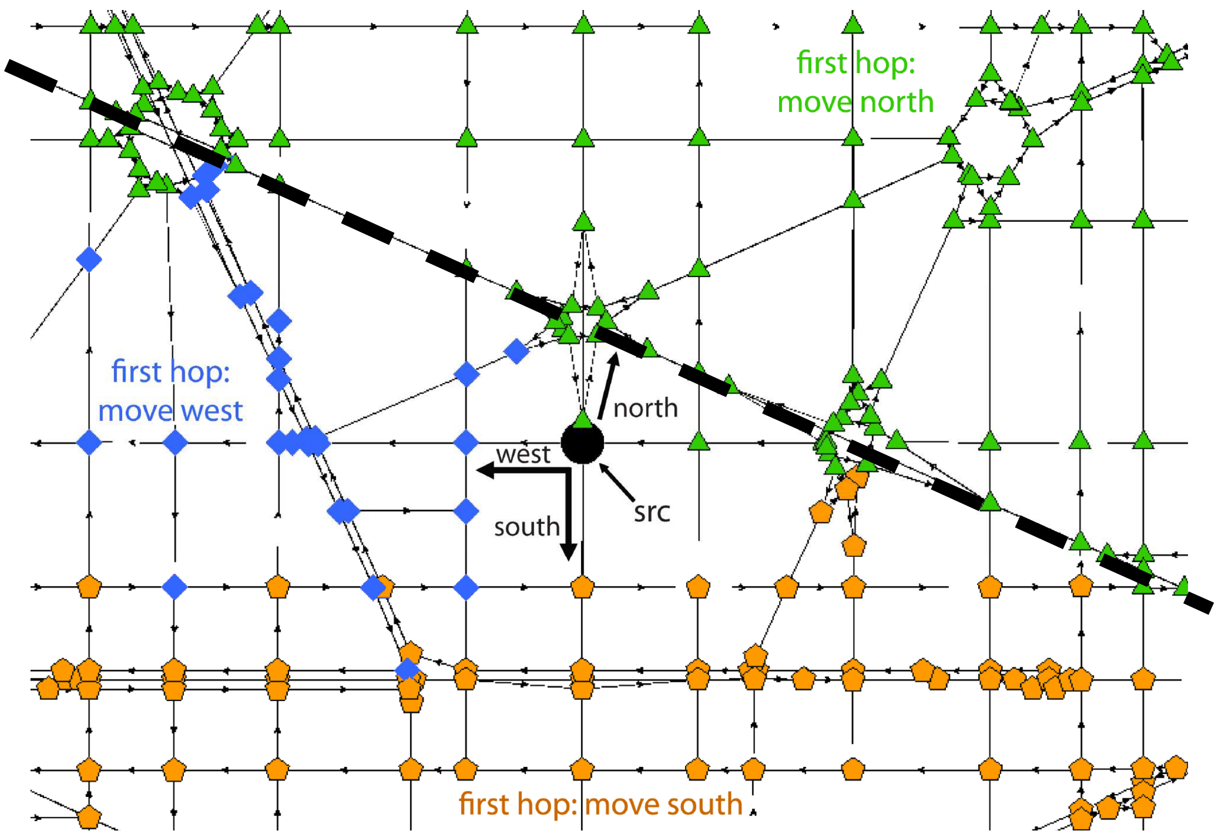


Observation 2: Road networks have geometric structure

Nodes above hyperplane: first hop is north or east

Nodes below hyperplane: first hop is south or west

Finding Structure



If each node has four neighbors, can specify neighbors with **two** bits:

- 1st bit: encode direction along NW/SE axis
- 2nd bit: encode direction along NE/SW axis

A Compressible Structure

Let $M^{(\mathrm{NE})}$ and $M^{(\mathrm{NW})}$ be next-hop matrices along NE and NW axis (entries in $M^{(\mathrm{NE})}$ and $M^{(\mathrm{NW})}$ are bits)

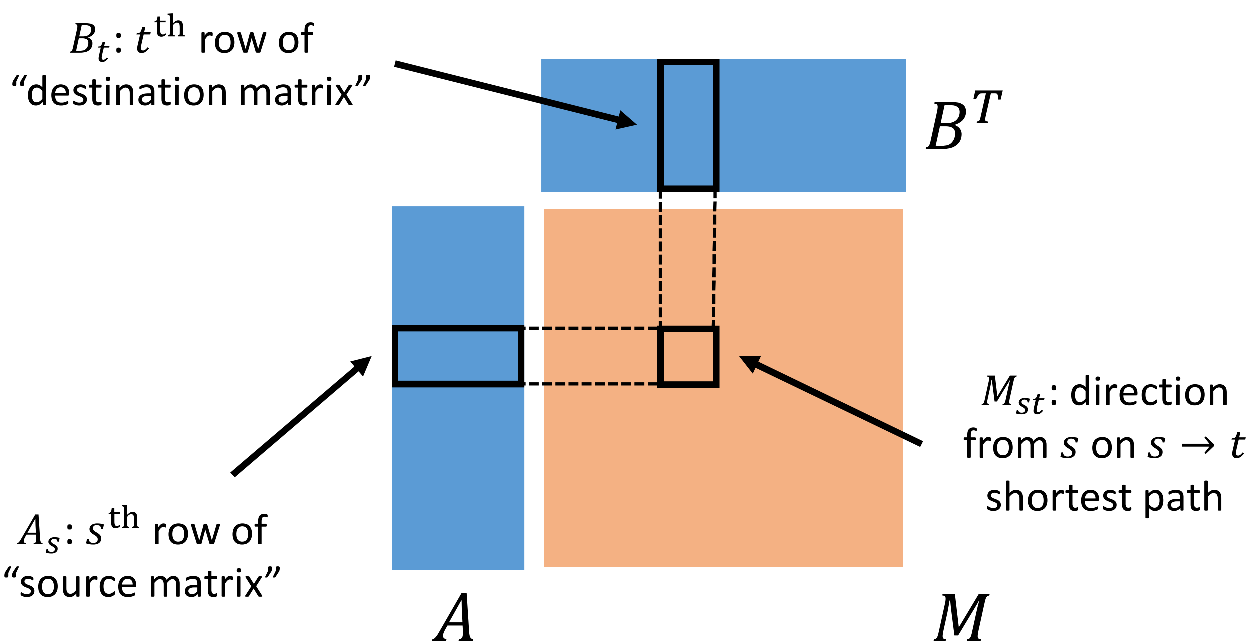
Objective: for $i \in \{\mathrm{NE}, \mathrm{NW}\}$, find matrices $A^{(i)}, B^{(i)}$ such that

$$M^{(i)} = \mathrm{sign}\left(A^{(i)} \cdot \left(B^{(i)}\right)^T\right)$$

A Compressible Structure

Objective: for $i \in \{\mathrm{NE}, \mathrm{NW}\}$, find matrices $A^{(i)}, B^{(i)}$ such that

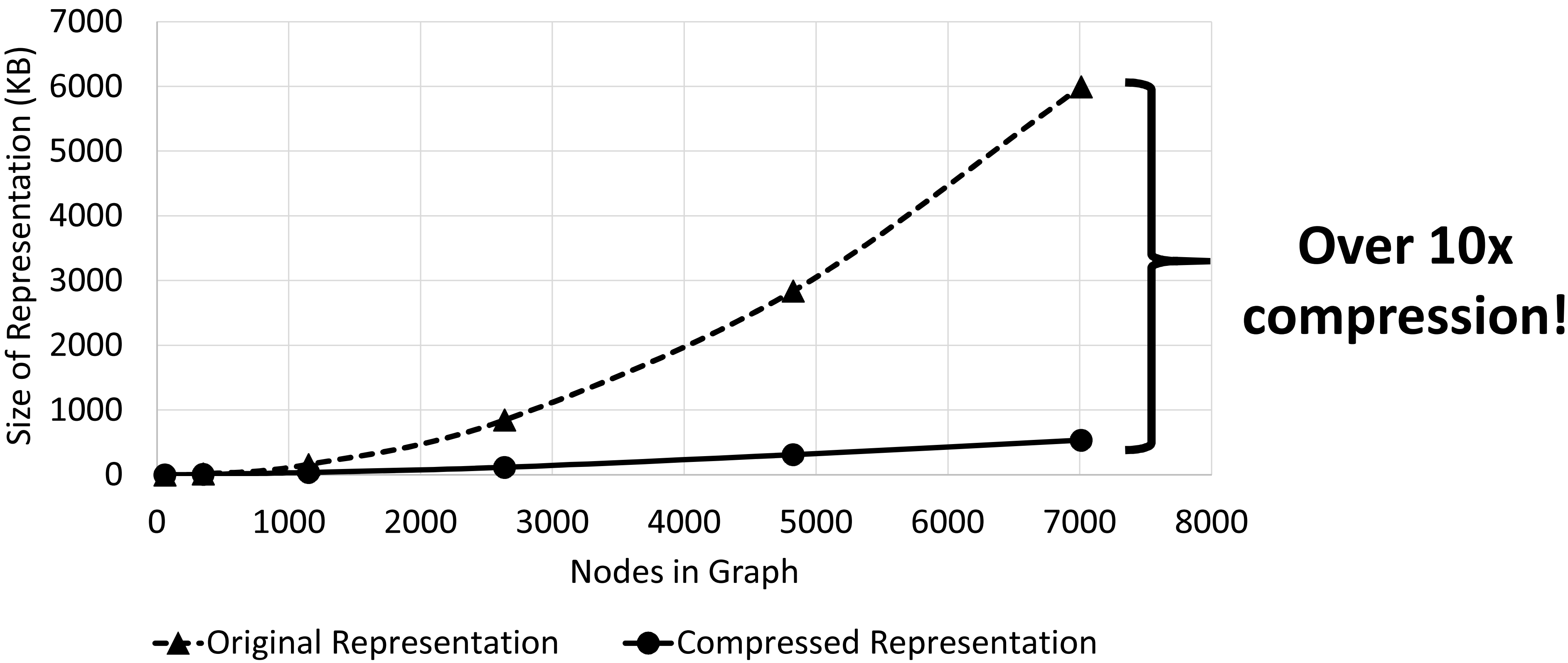
$$M^{(i)} = \mathrm{sign}\left(A^{(i)} \cdot \left(B^{(i)}\right)^T\right)$$



Computing next-hop reduces to computing inner products

Index of row in A only depend on *source*, index of row in B only depend on *destination*

A Compressible Structure



An Iterative Shortest-Path Protocol

To learn next-hop on $s \to t$ shortest path:

1. Use SPIR to obtain s^{th} row of $A^{(\text{NE})}$ and $A^{(\text{NW})}$
2. Use SPIR to obtain t^{th} row of $B^{(\text{NE})}$ and $B^{(\text{NW})}$
3. Compute

   $M_{st}^{(\text{NE})} = \text{sign}\left\langle A_s^{(\text{NE})}, B_t^{(\text{NE})} \right\rangle$ and $M_{st}^{(\text{NW})} = \text{sign}\left\langle A_s^{(\text{NW})}, B_t^{(\text{NW})} \right\rangle$

SPIR queries on databases with $\boldsymbol{n}$ records

Problem: rows and columns of A, B reveal more information than desired

Affine Encodings and Arithmetic Circuits

Goal: Reveal inner product without revealing vectors

Idea: Use a "garbled" arithmetic circuit (affine encodings) [AIK14]

- Encodings reveal output of computation (inner product) and nothing more

Solution: SPIR on arithmetic circuit *encodings*

An Iterative Shortest-Path Protocol

To learn next-hop on $s \to t$ shortest path:

1. Use SPIR to obtain encodings of s^{th} row of $A^{(\text{NE})}$ and $A^{(\text{NW})}$
2. Use SPIR to obtain encodings of t^{th} row of $B^{(\text{NE})}$ and $B^{(\text{NW})}$
3. Evaluate inner products $\left\langle A_s^{(\text{NE})}, B_t^{(\text{NE})} \right\rangle$ and $\left\langle A_s^{(\text{NW})}, B_t^{(\text{NW})} \right\rangle$
4. Compute $M_{st}^{(\text{NE})}$ and $M_{st}^{(\text{NW})}$ (signs of inner products)

Affine encodings hide source and destination matrices, but inner products reveal too much information

Thresholding via Garbled Circuits

Goal: Reveal only the *sign* of the inner product

Solution: Blind inner product and evaluate the sign function using a garbled circuit [Yao86, BHR12]

- Instead of $\langle x, y \rangle$, compute $\alpha\langle x, y \rangle + \beta$ for random $\alpha, \beta \in \mathbb{F}_p$
- Use garbled circuit to unblind and computing the sign

An Iterative Shortest-Path Protocol

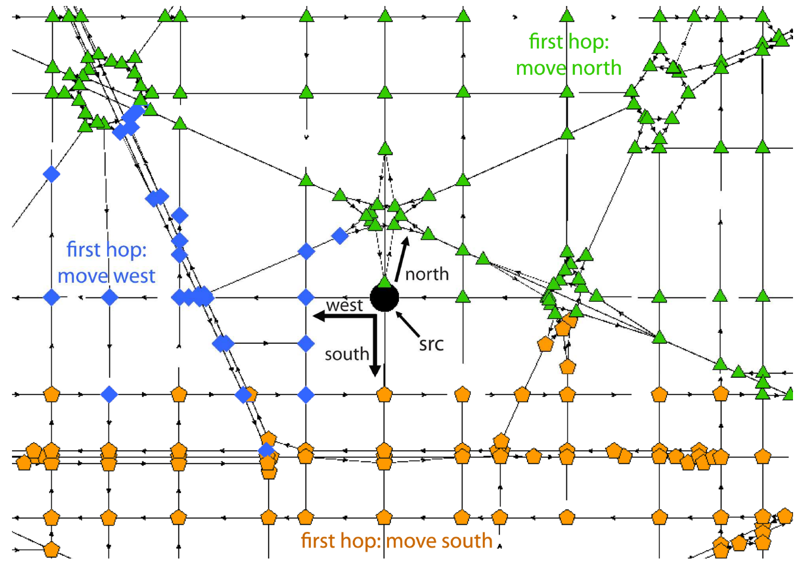
To learn next-hop on $s \to t$ shortest path:

1. Use SPIR to obtain encodings of s^{th} row of $A^{(\text{NE})}$ and $A^{(\text{NW})}$
2. Use SPIR to obtain encodings of t^{th} row of $B^{(\text{NE})}$ and $B^{(\text{NW})}$
3. Evaluate to obtain blinded inner products $z^{(\text{NE})}$ and $z^{(\text{NW})}$
4. Use garbled circuit to compute $M_{st}^{(\text{NE})}$ and $M_{st}^{(\text{NW})}$

Semi-honest secure!

See paper for protection against malicious parties

Benchmarks



Preprocessed city maps from OpenStreetMap

Online Benchmarks

City	Number of Nodes	Time per Round (s)	Bandwidth (KB)
San Francisco	1830	1.44 ± 0.16	88.24
Washington D.C.	2490	1.64 ± 0.13	90.00
Dallas	4993	2.91 ± 0.19	95.02
Los Angeles	7010	4.75 ± 0.22	100.54

Timing and bandwidth for each round of the online protocol (with protection against malicious clients)

End-to-End Benchmarks

City	Number of Rounds	Total Online Time (s)	Online Bandwidth (MB)
San Francisco	97	140.39	8.38
Washington D.C.	120	197.48	10.57
Dallas	126	371.44	11.72
Los Angeles	165	784.34	16.23

End-to-end performance of private shortest paths protocol (after padding number of rounds to maximum length of shortest path for each network)

Conclusions

Problem: privacy-preserving navigation

Routing information for road networks are compressible!

- Optimization-based compression technique achieves over 10x compression of next-hop matrices

Compressed routing matrix lends itself to iterative shortest-path protocol

- Computing the shortest path reduces to computing sign of inner product
- Leverage combination of arithmetic circuits + Boolean circuits

Questions?