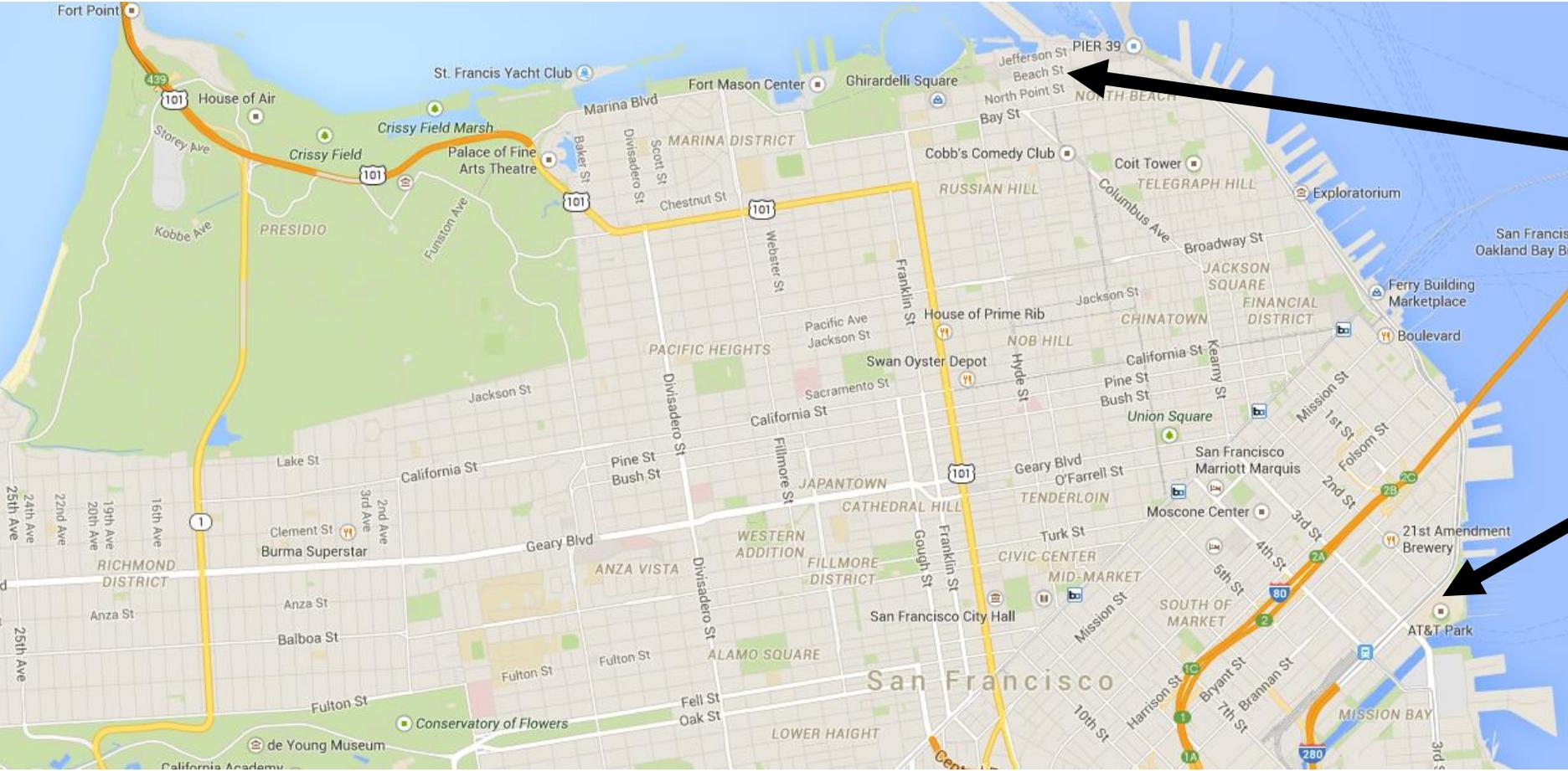


Privacy-Preserving Shortest Path Computation

David Wu

Joint work with Joe Zimmerman, Jeremy Planul, and
John Mitchell

Navigation



desired destination

current position

Navigation: A Solved Problem?



directions from current
location to Fisherman's
Wharf



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

“Trivial” Solution

Give me the entire
map!



“Trivial” Solution



Give me the entire map!



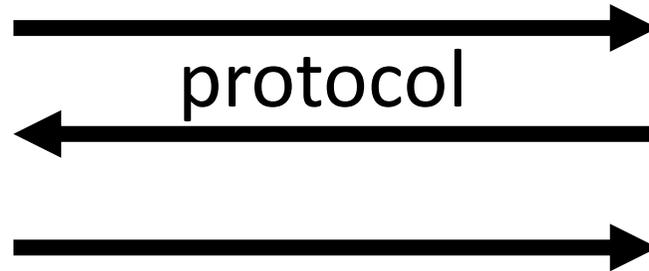
Pros: lots of privacy (for the client)

Cons:

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

Private Shortest Paths

AT&T park to
Fisherman's
Wharf



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 $s \rightarrow t$ shortest path and nothing about weights of other edges not in shortest path

Straw Man Solution

Suppose road network has n nodes

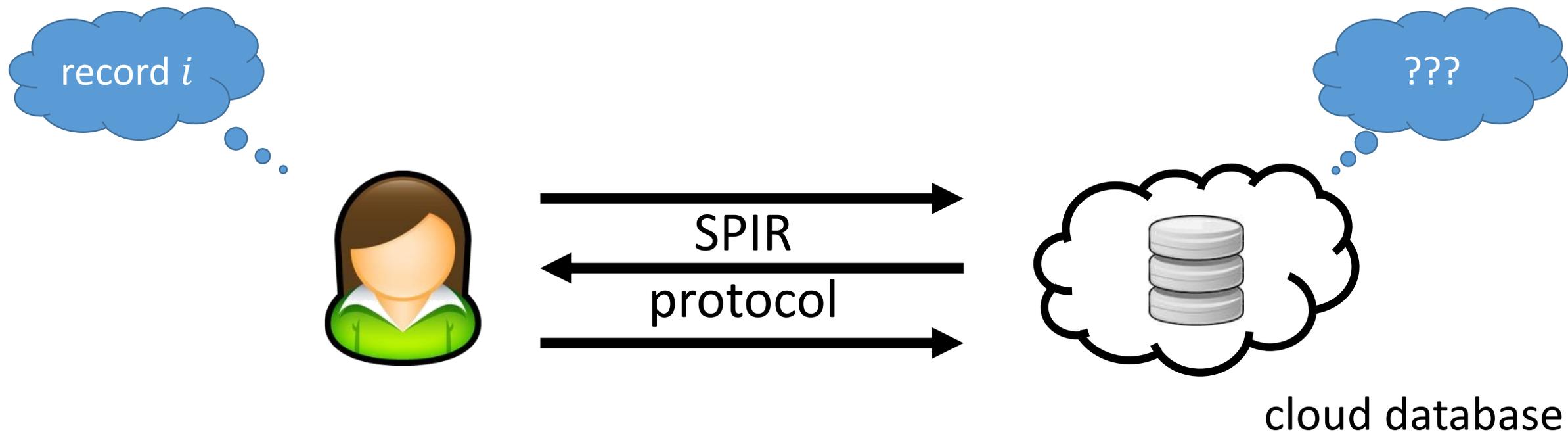
Construct $n \times n$ database:

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

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

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

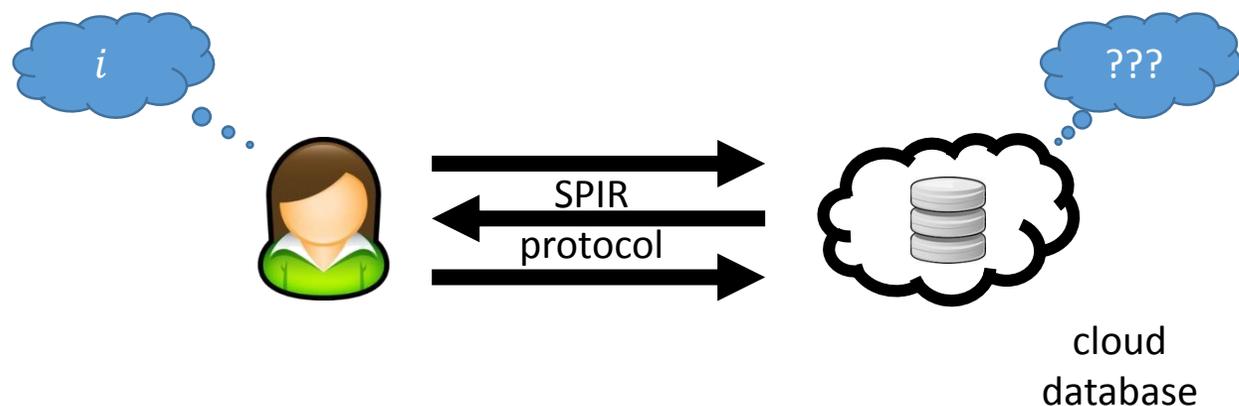
(Strong) Private Information Retrieval (SPIR)



Client Privacy: server does not learn i

Server Privacy: client only learns record i

(Strong) Private Information Retrieval (SPIR)



- single-server PIR: solutions exist from additive homomorphism [KO97]
- SPIR: construction from PIR + OT on short secrets [NP05]
- computation lower bound: *linear* in size of database

query on 10^6 records = 10^6 public key operations = several minutes of (single-threaded) computation

Finding Structure

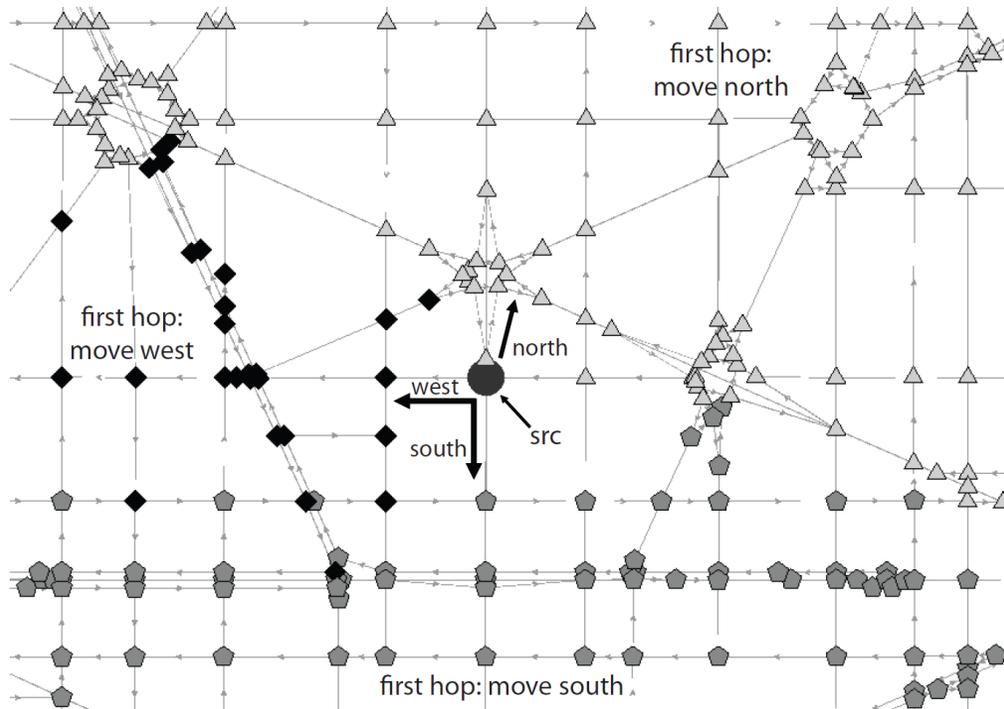
Straw man solution requires SPIR on databases with n^2 records – quadratic in number of nodes in the graph – rather impractical!



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

Finding Structure

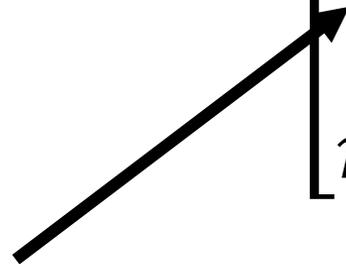
Typically, an intersection has up to four neighbors (for the four cardinal directions)



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

Finding Structure

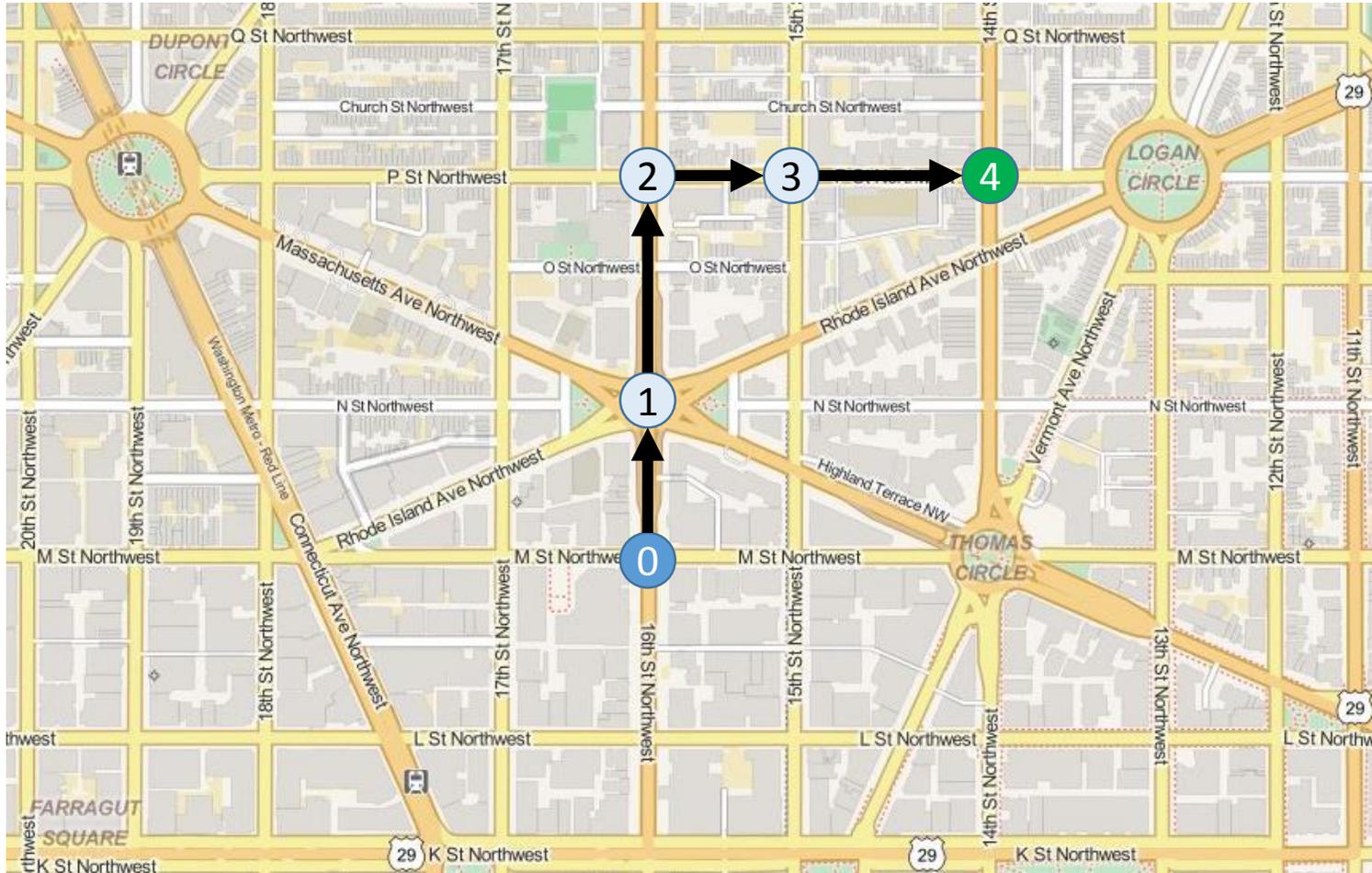
Next-hop routing matrix for graph with n nodes:

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


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

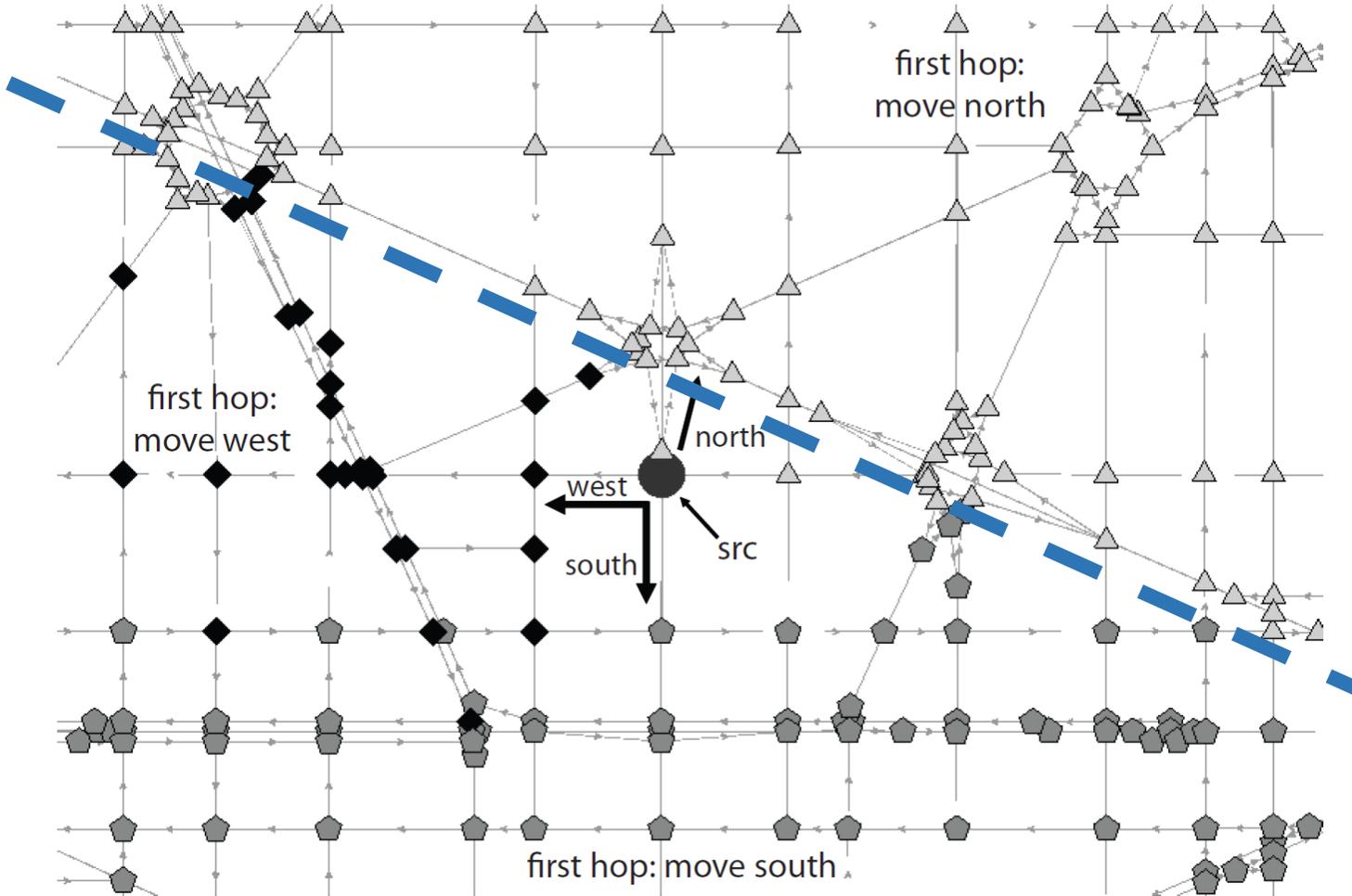


Routing from 0 to 4:

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

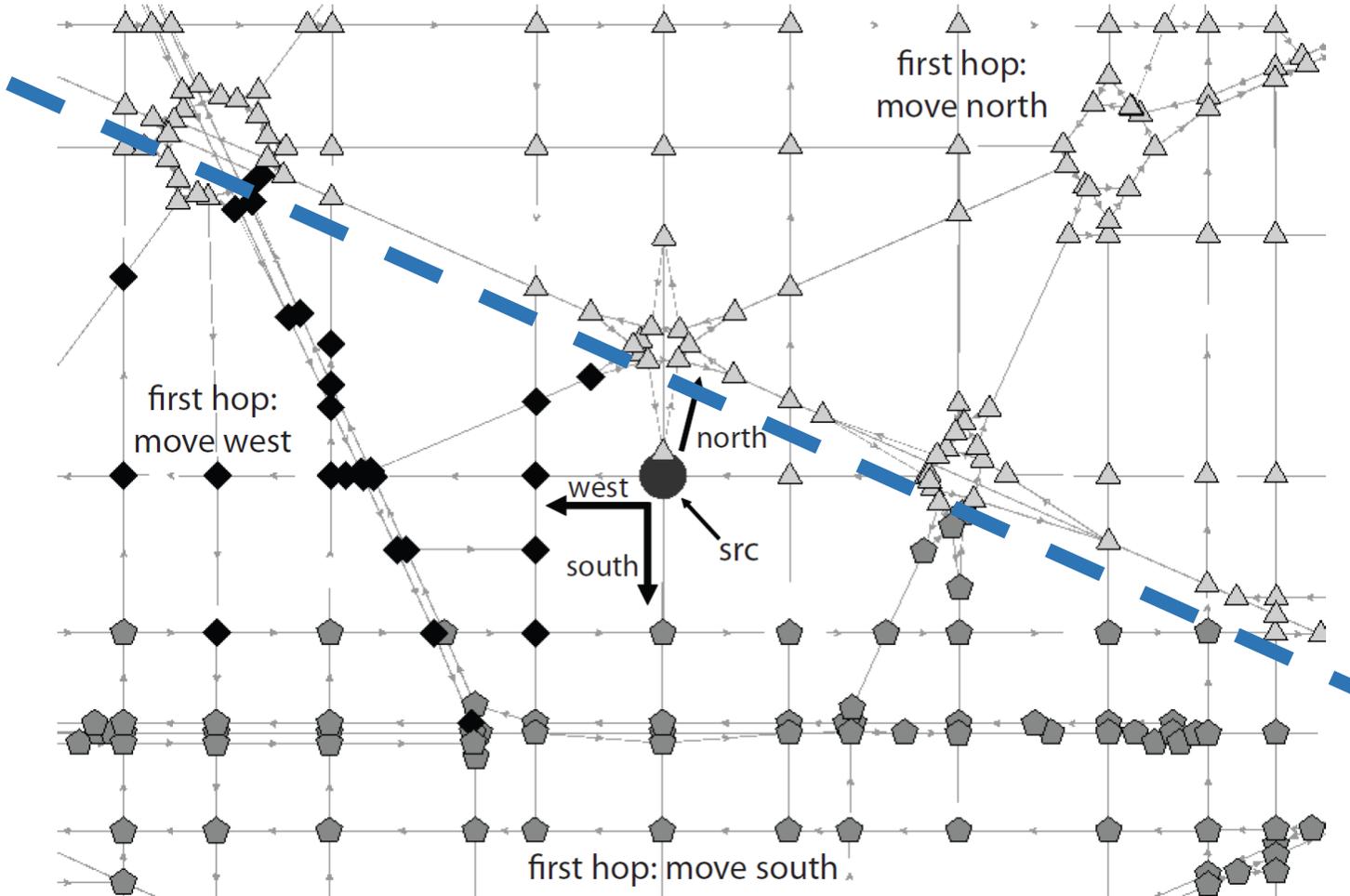


Observation 2: Road networks have geometric structure

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

Examples:

- North: 00
- East: 10
- South: 11
- West: 01

A Compressible Structure

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

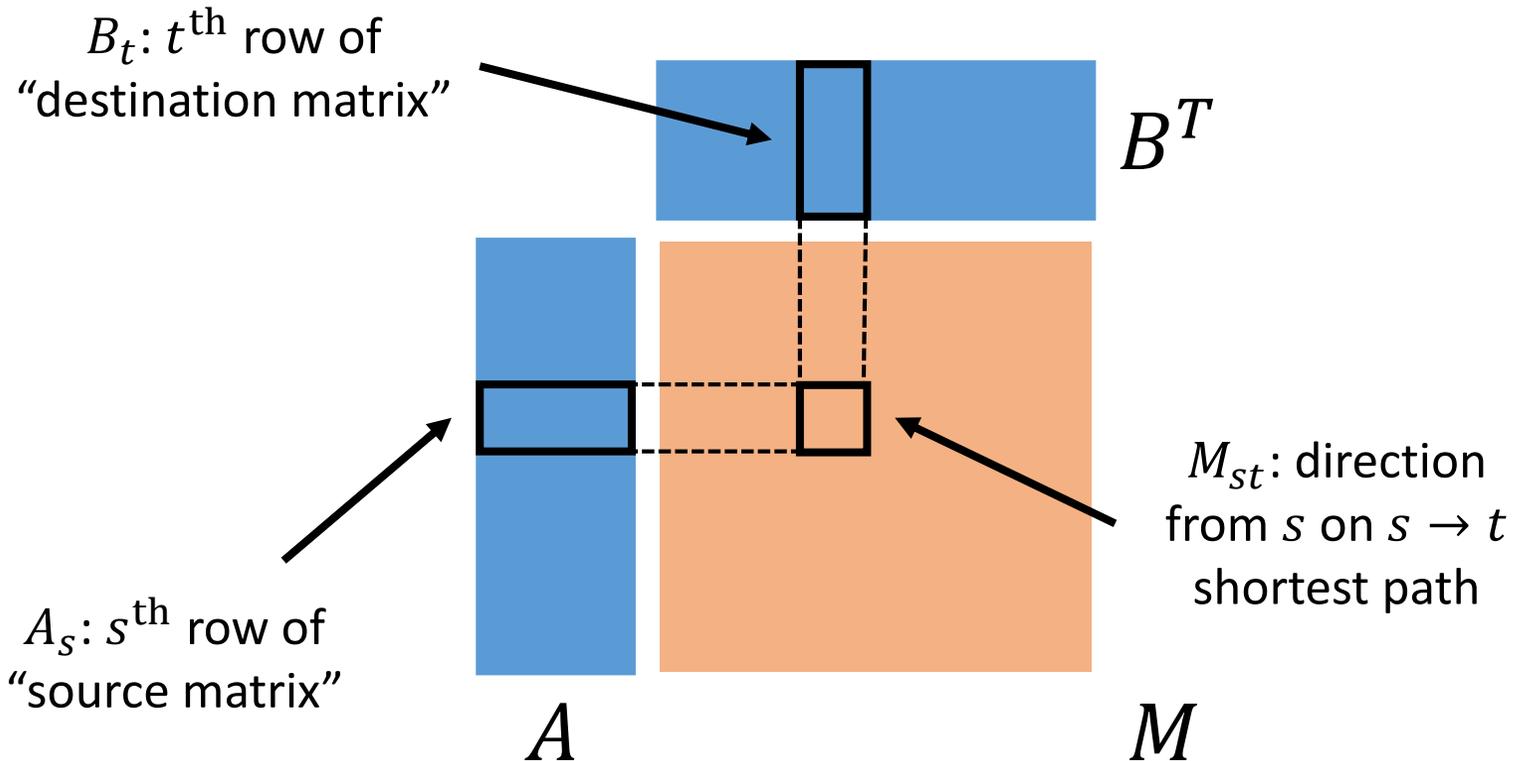
Objective: for $i \in \{\text{NE}, \text{NW}\}$, find matrices $A^{(i)}, B^{(i)}$ such that

$$M^{(i)} = \text{sign}(A^{(i)} \cdot B^{(i)})$$

A Compressible Structure

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

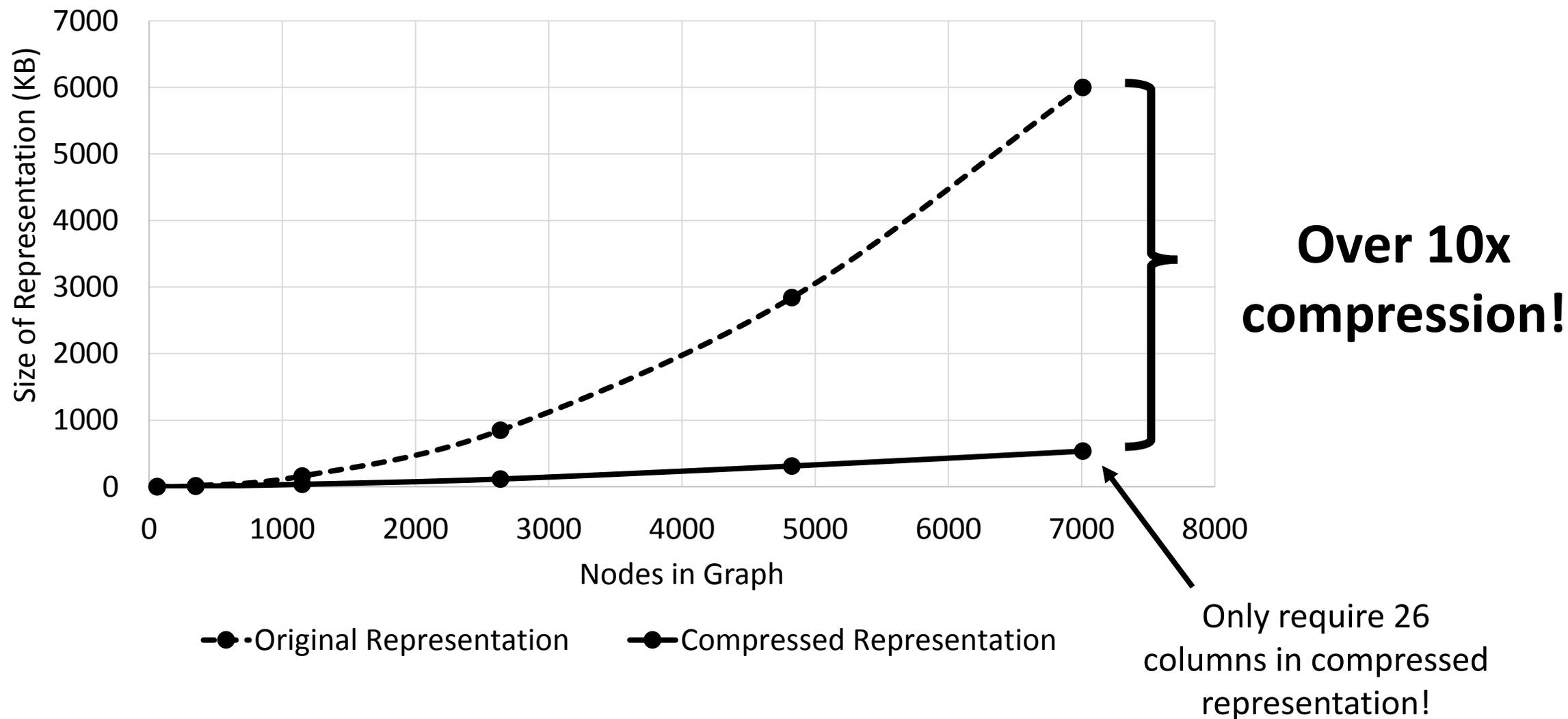
$$M^{(i)} = \text{sign}(A^{(i)} \cdot B^{(i)})$$



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 $s \rightarrow t$ 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

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

SPIR queries on databases
with 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]

Example: Encoding of addition circuit $f(a, b) = a + b \in \mathbb{F}_p$:

- Encoding of a, b given by $(a + r, b - r)$ for random $r \in \mathbb{F}_p$
- Encodings $(a + r, b - r)$ reveal $a + b$ and nothing more

Solution: SPIR on arithmetic circuit *encodings*

An Iterative Shortest-Path Protocol

To learn next-hop on $s \rightarrow t$ 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 $\langle A_s^{(\text{NE})}, B_t^{(\text{NE})} \rangle$ and $\langle A_s^{(\text{NW})}, B_t^{(\text{NW})} \rangle$
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 evaluate function

$$g(z, \alpha, \beta) = \text{sign}(\alpha^{-1}(z - \beta) \bmod p)$$

Client input: z

Server input: α, β

Input privacy of garbled circuits hide α, β

An Iterative Shortest-Path Protocol

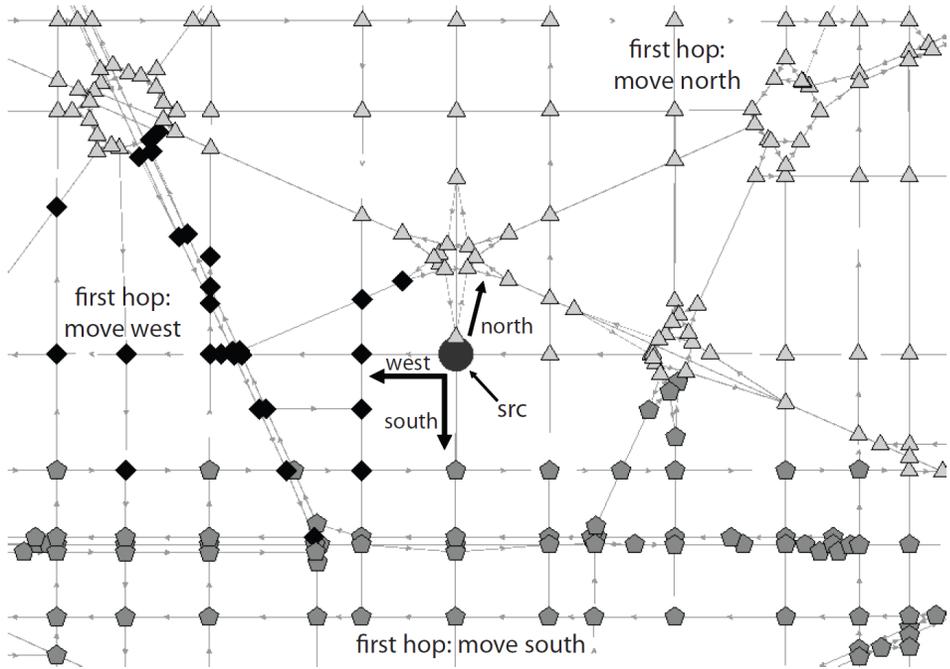
To learn next-hop on $s \rightarrow t$ shortest path:

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 circuits** to compute $M_{st}^{(\text{NE})}$ and $M_{st}^{(\text{NW})}$

Semi-honest secure!

But malicious client can make
inconsistent queries...

Benchmarks



Preprocessed city maps from OpenStreetMap

Benchmarks

City	Number of Nodes	Time per Round (s)	Bandwidth (KB)
San Francisco	1830	1.42 ± 0.09	88.24
Washington D.C.	2490	1.69 ± 0.22	90.00
Dallas	4993	2.91 ± 0.18	95.02
Los Angeles	7010	4.75 ± 0.14	100.54

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

Benchmarks

Most expensive component of protocol is sending garbled circuits (≈ 520 KB per circuit), but this can be done prior to the online (navigation) phase

Each round of the protocol completes in a few seconds (bottleneck is PIR protocol); fast enough for real-time navigation if it takes more than a few seconds between intersections (generally true)

Modest amount of bandwidth (around 100 KB) per round

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?