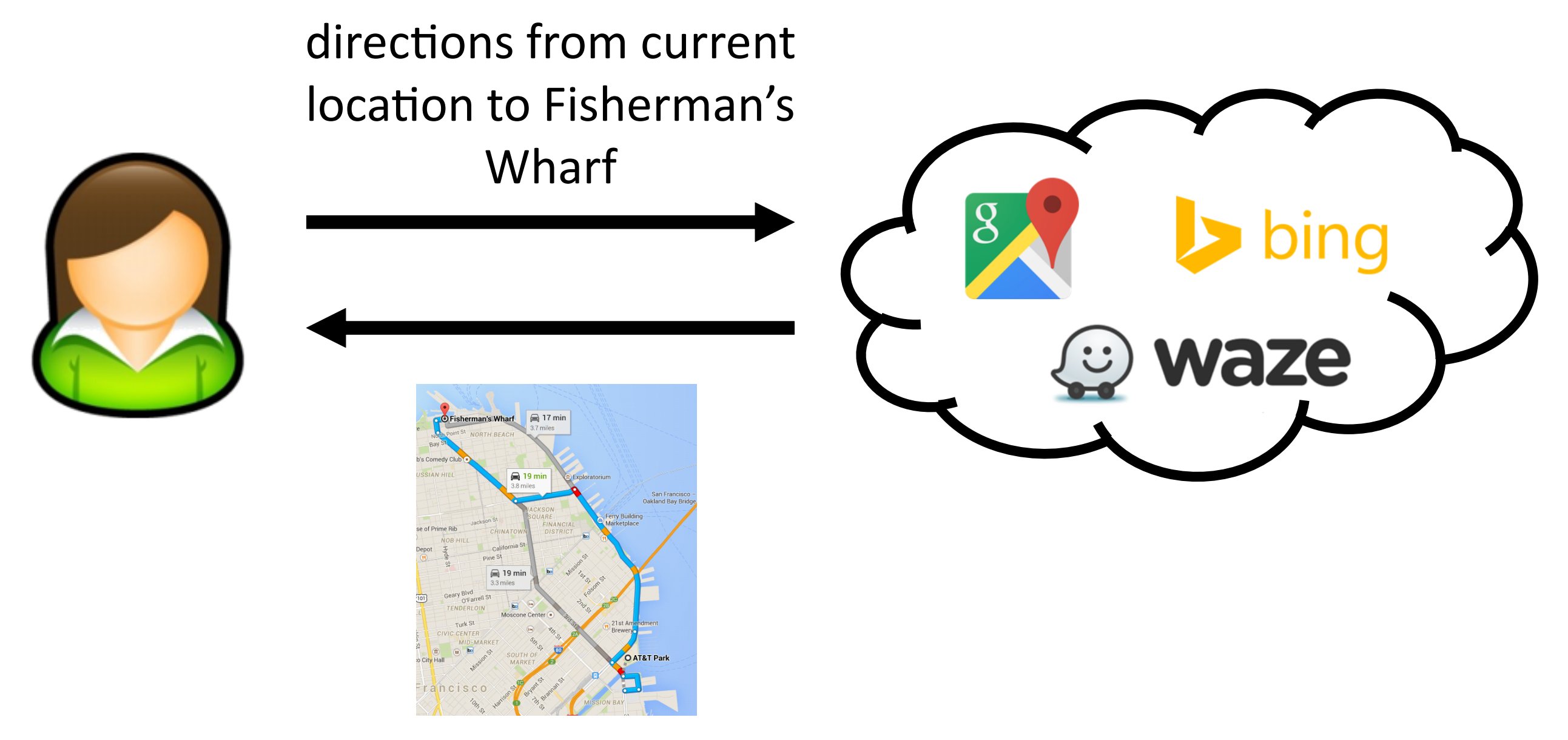
Privacy-Preserving Shortest Path Computation

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

Navigation and Location Privacy

Location privacy is a major concern among smartphone users and numerous controversies have come up due to companies surreptitiously tracking users' locations. Among the various apps that require location information, navigation is one of the most popular. For example, companies such as Google, Apple, and Waze, have built traffic-aware navigation apps to provide users with the most up-to-date routing information. But to use these services, users must reveal their location to learn the fastest route to their destination. In doing so, they also reveal other sensitive information about their personal lives, such as their health condition, their shopping habits, and more to the map provider.



If we just desire privacy for the user's location, a simple approach is for the user to download the entire map and compute the shortest paths herself. Even discounting performance issues, map providers are not incentivized to simply give away their routing information for free. In other scenarios, the mapping information might contain sensitive or confidential information that should not be freely given out. Thus, in our work, we examine the problem of fully private shortest path computation, that is, privacy should hold for both the client's location and the server's routing information.

A Strawman Protocol

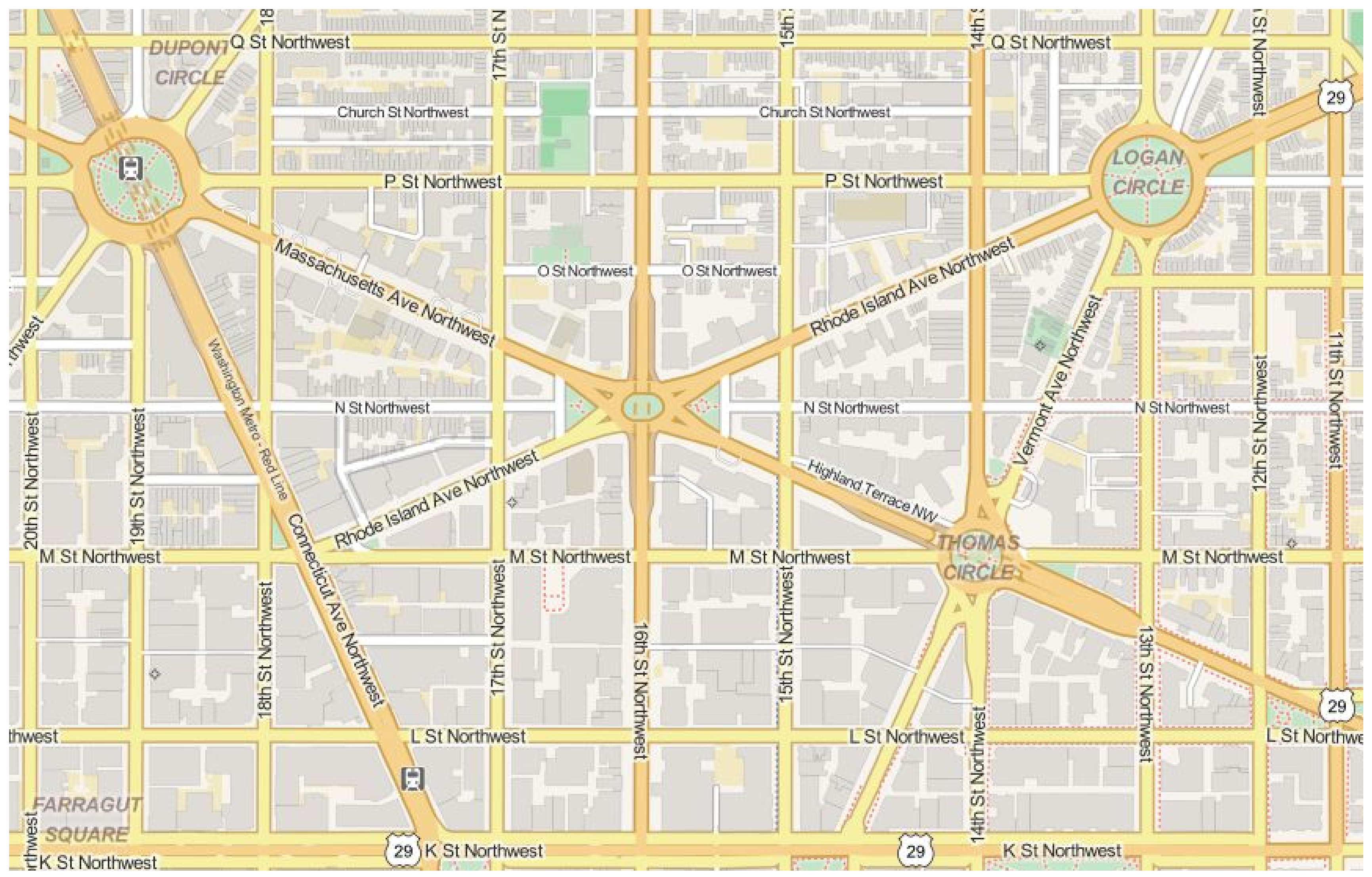
Suppose the road network has only n nodes. Then, we can construct an n-by-n database of shortest paths

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

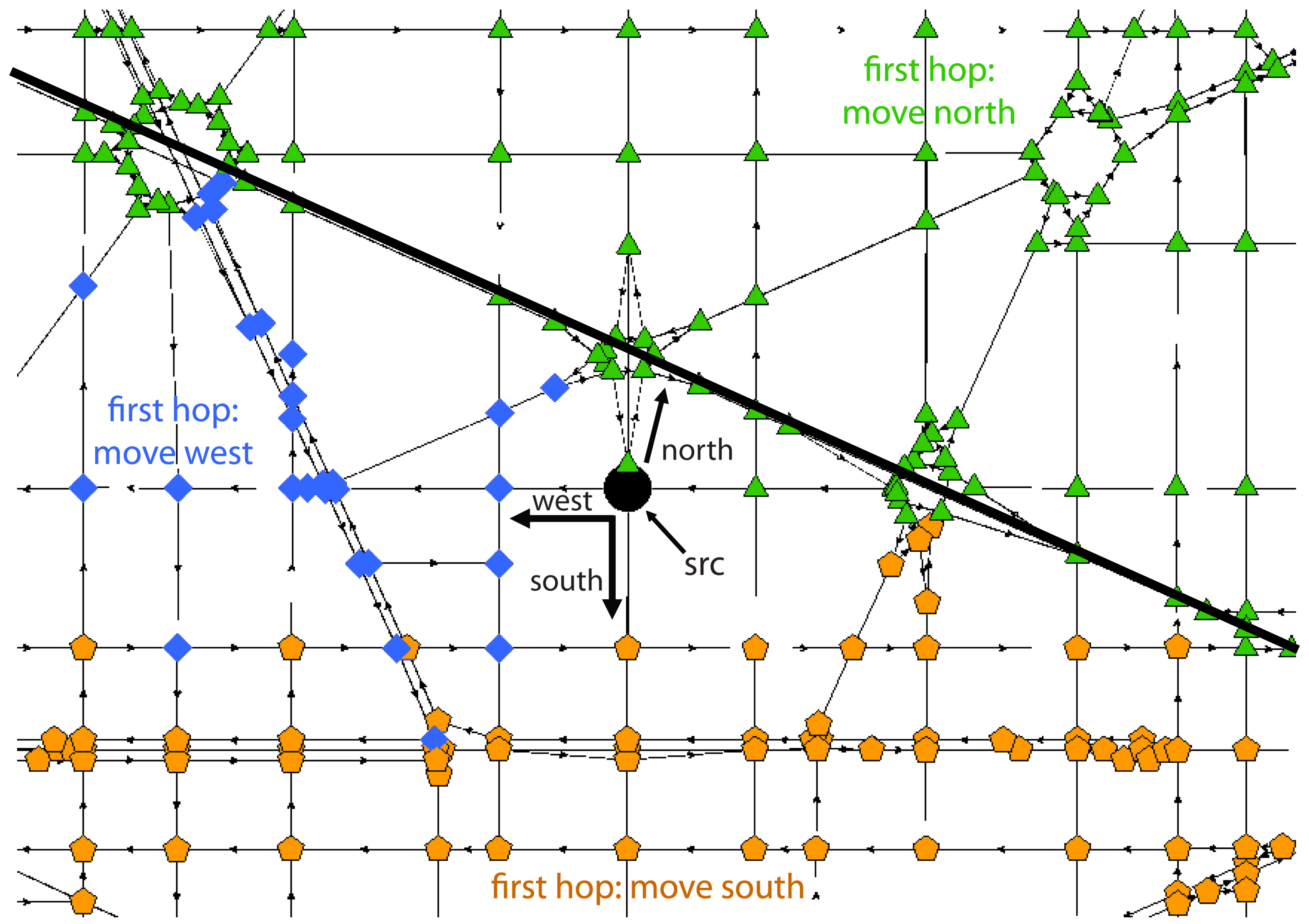
record r_{st} contains the shortest path from s to t

Given the database of shortest paths, private shortest path computation reduces to symmetric private information retrieval (SPIR). However, the size of the database grows quadratically in the number of nodes in the graph.

Map Preprocessing and Compression



Subsection of road network for Washington D.C. taken from OpenStreetGraph.



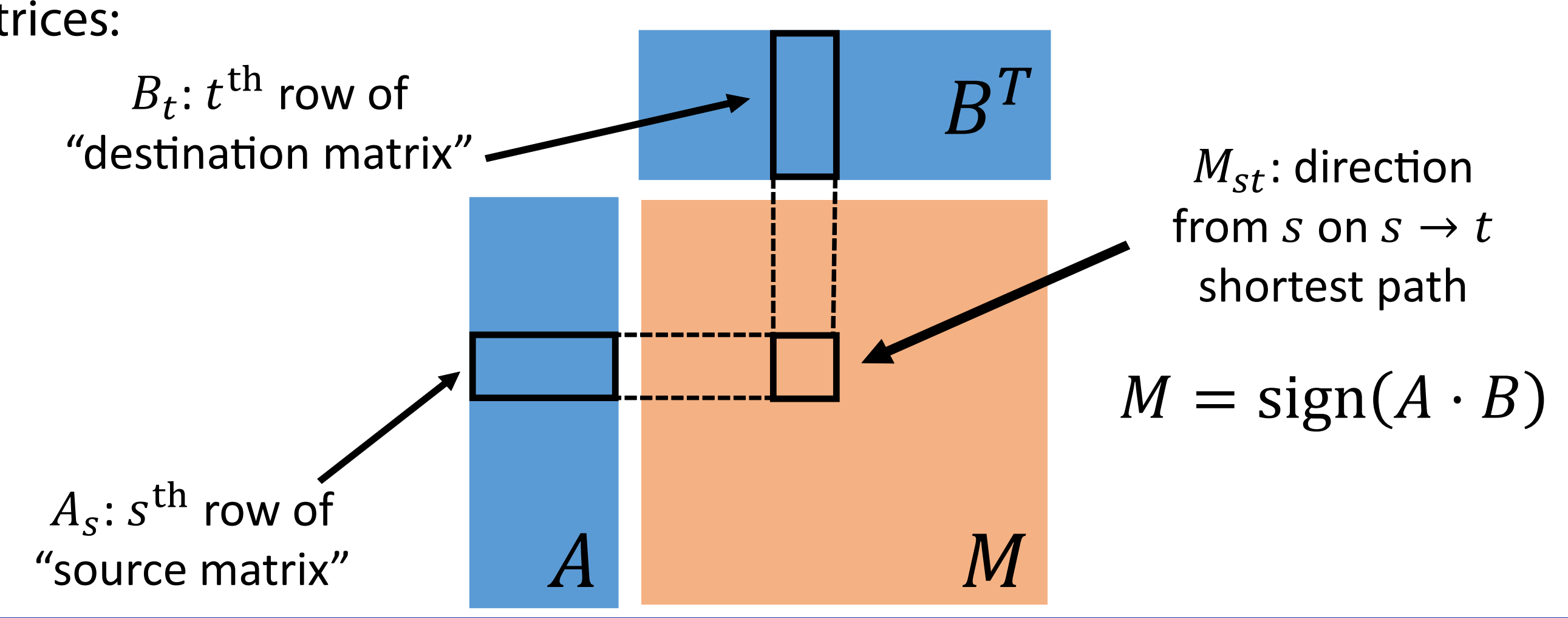
Visualization of preprocessed road network.

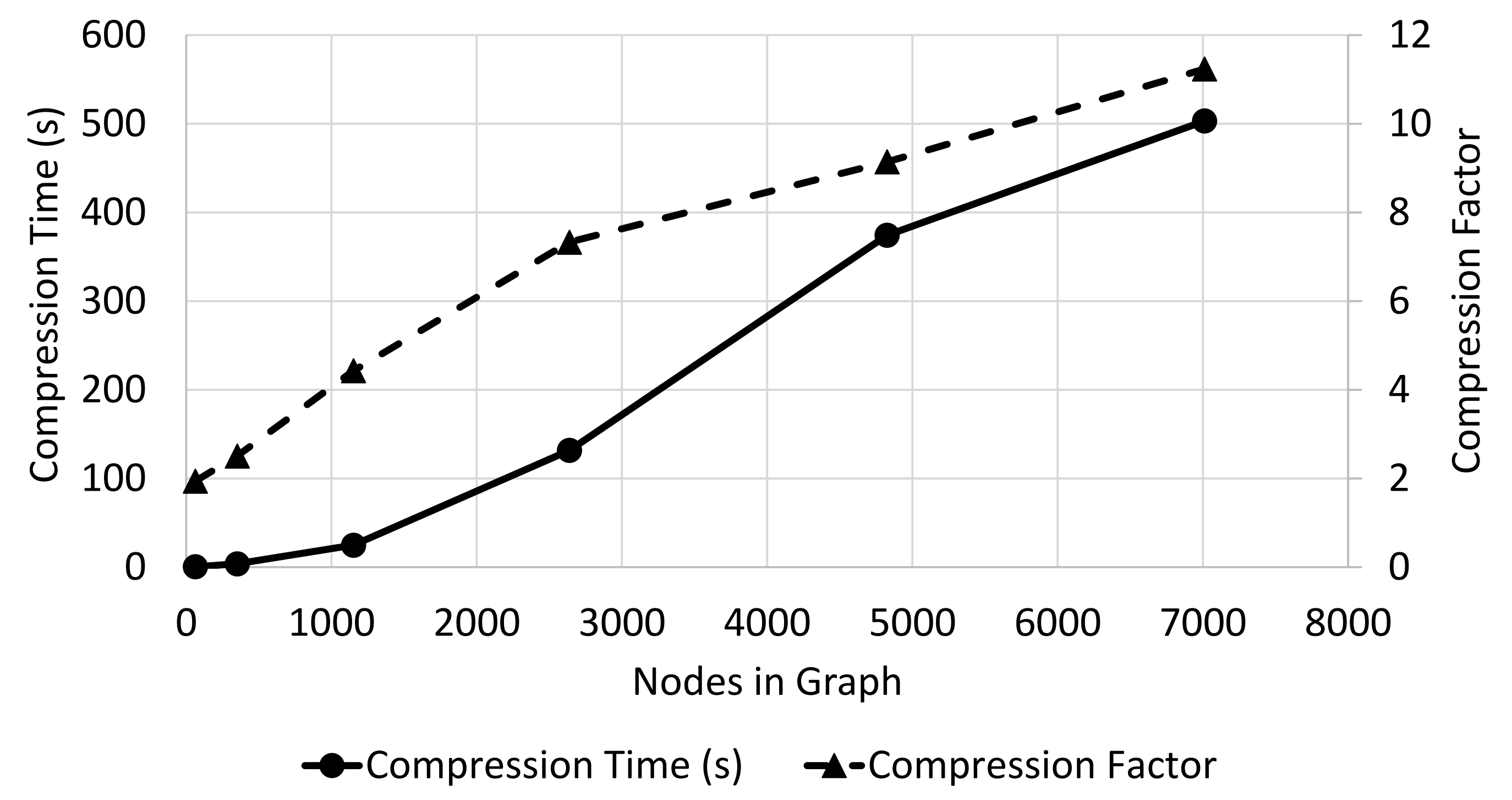
Observation 1: Nodes in road networks have low (constant) degree.

For each node in the network, we can associate each of its neighbors with a direction (unique index). Then, we can replace the shortest path matrix with the next-hop routing matrix. To compute the shortest paths, we iteratively retrieve the next-hop from the next-hop routing matrix.

Observation 2: Road networks have compressible geometric structure.

We preprocess the graph so that each node has at most four neighbors, and we associate a direction with each neighbor. The index of each neighbor can be encoded with two bits: a component along the NW/SE axis and a component along the NE/SW axis. We formulate an optimization problem to find a compact representation of the next-hop routing matrices:


B_t: tth row of "destination matrix"
B^T
M_{st}: direction from s on $s \to t$ shortest path
$M = \mathrm{sign}(A \cdot B)$
A_s: sth row of "source matrix"
A
M



Average time needed to compress the next-hop routing matrix and the resulting compression factor for networks constructed from subgraphs of the road network of Los Angeles.

A Private Navigation Protocol

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

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

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

While this protocol only requires SPIR on databases with *n* records, the rows and columns of *A* and *B* reveal additional information about the server's routing information.

Nonetheless, we have effectively reduced the shortest-path problem to computing a thresholded inner product. We now develop a method to privately evaluate this function.

Affine encodings. To compute the inner product $\langle A, B\rangle$ without revealing *A* or *B*, we can use a garbled arithmetic circuit. In fact, it suffices to use the affinization gadgets from [AIK14].

For example, suppose we wanted to garble an addition circuit

$$f(a, b) = a + b \in \mathbb{F}_p$$

The encoding of *a*, *b* is given by $(a + r, b - r)$ for some random $r \in \mathbb{F}_p$. Then, given only the encoding, one can only learn the sum $a + b$ and nothing else.

In our protocol, rather than using SPIR to retrieve the actual source and destination vectors, the client instead uses SPIR to obtain the *encodings* of the vectors. This allows the client to learn only the inner product and nothing more.

Garbled circuits. In Step 3 of our protocol, we only want to reveal the sign, and not the actual value of the inner product.

To hide the inner product, we modify the arithmetic circuit to instead compute a *blinded* inner product, that is, instead of computing $\langle A, B\rangle$, we compute $\alpha\langle A, B\rangle + \beta$ for random $\alpha, \beta \in \mathbb{F}_p$.

To complete the protocol execution, we construct a garbled circuit for the following unblind-and-threshold functionality:

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

Here, the client provides as input the blinded inner product z, and the server provides as input the blinding factors α, β. Security of the garbled circuit ensures the client only learns the thresholded value, and nothing else.

This yields a protocol secure against *semi-honest* clients.

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).

To hide the length of the shortest path, we pad the number of rounds to the maximum number needed to answer any shortest path query. For these road networks, we require between 95 and 170 rounds.