Universidad de los Andes
Colombia

Departamento de Ingeniería de Sistemas y Computación

FOROS ISIS

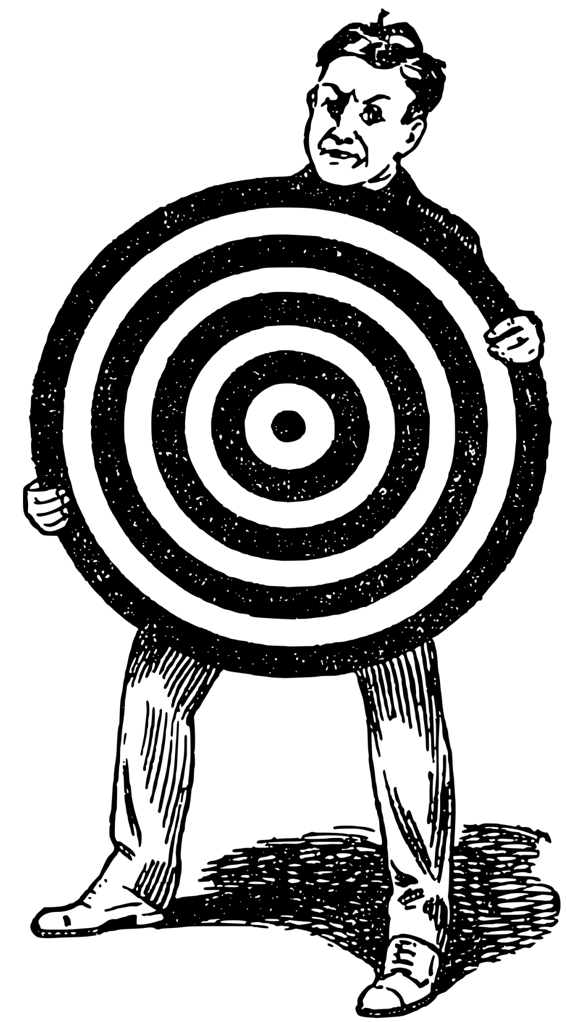
5to FORO
en Seguridad de la Información

RETOS Y SOLUCIONES
PARA LA PRIVACIDAD EN UN MUNDO CONECTADO

Location privacy

Martín Ochoa
Universidad del Rosario

joint work with Jorge Cuellar, Ruben Rios, Andrei Sabelfeld, Per Hallgren, Xiaolu Hou, Xueou Wang and Nils Tippenhauer



Universidad del Rosario

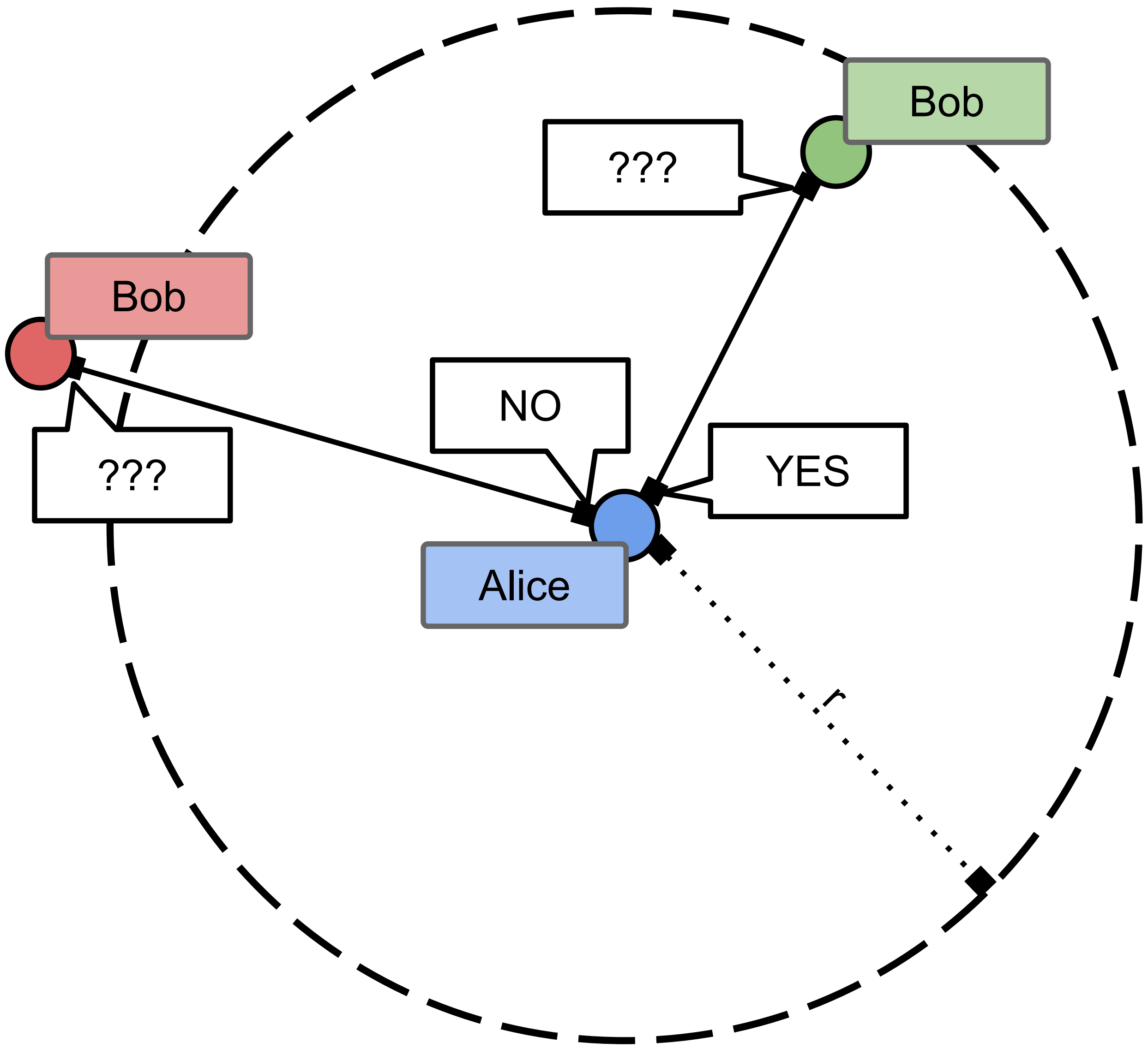
MACC
Matemáticas Aplicadas y Ciencias de la Computación

Motivation

- GPS enabled devices are ubiquitous
- Location-Based services are increasingly powerful
- Implementations of location-based services have been attacked
    - Include Security attack to locate any Tinder user, Feb 2014
    - "Girls around me" stalking app abusing Foursquare APIs, March 2012

Running example

- Finding friends



- Alice: is Bob close by (within r)?
  - Bob: yes/no

Problem

- How do we achieve utility and privacy?
- In other words, how do we share location securely?
    - *Exact location*: not private
    - *Distance*: triangulation attacks
    - *Obfuscated distance*: still possible to triangulate or loss of utility
    - *To third party*: Do we trust third party?

Outline

- Preliminaries
- One solution: **InnerCircle**
- An improvement: **BetterTimes**
- A further enhancement: **MaxPace**
- Triangulation: **Grids**
- Moving targets
- Work in Progress/Future Work

Secure Multi-party Computation

- Location proximity is an instance of a multi-party computation:

  f(location_A, location_B) = 1 ***if close,***
  0 ***otherwise***

- Very similar to original Millionaire's Problem (Yao).
- Solvable i.e. with Garbled Circuits, Fully Homomorphic encryption.

Homomorphic Encryption

- An encryption function [[ ]] is additively homomorphic if:

  $[[a]] + [[b]] = [[a + b]]$

- It follows:

  $[[a*m]] = [[a]]*m$

InnerCircle

- Note that:

$$[\![d^2]\!] = [\![(x_A - x_B)^2 + (y_A - y_B)^2]\!] = \ldots$$
$$= [\![x_A^2 + y_A^2]\!] \oplus [\![x_B^2 + y_B^2]\!] \ominus (([\![x_A]\!] \odot 2x_B) \oplus ([\![y_A]\!] \odot 2y_B))$$

- It follows:

$$[\![(d^2 - 0) \cdot r_0]\!], [\![(d^2 - 1) \cdot r_1]\!], \ldots, [\![(d^2 - r^2) \cdot r_{r^2}]\!]$$

contains a 0 iff d < r.

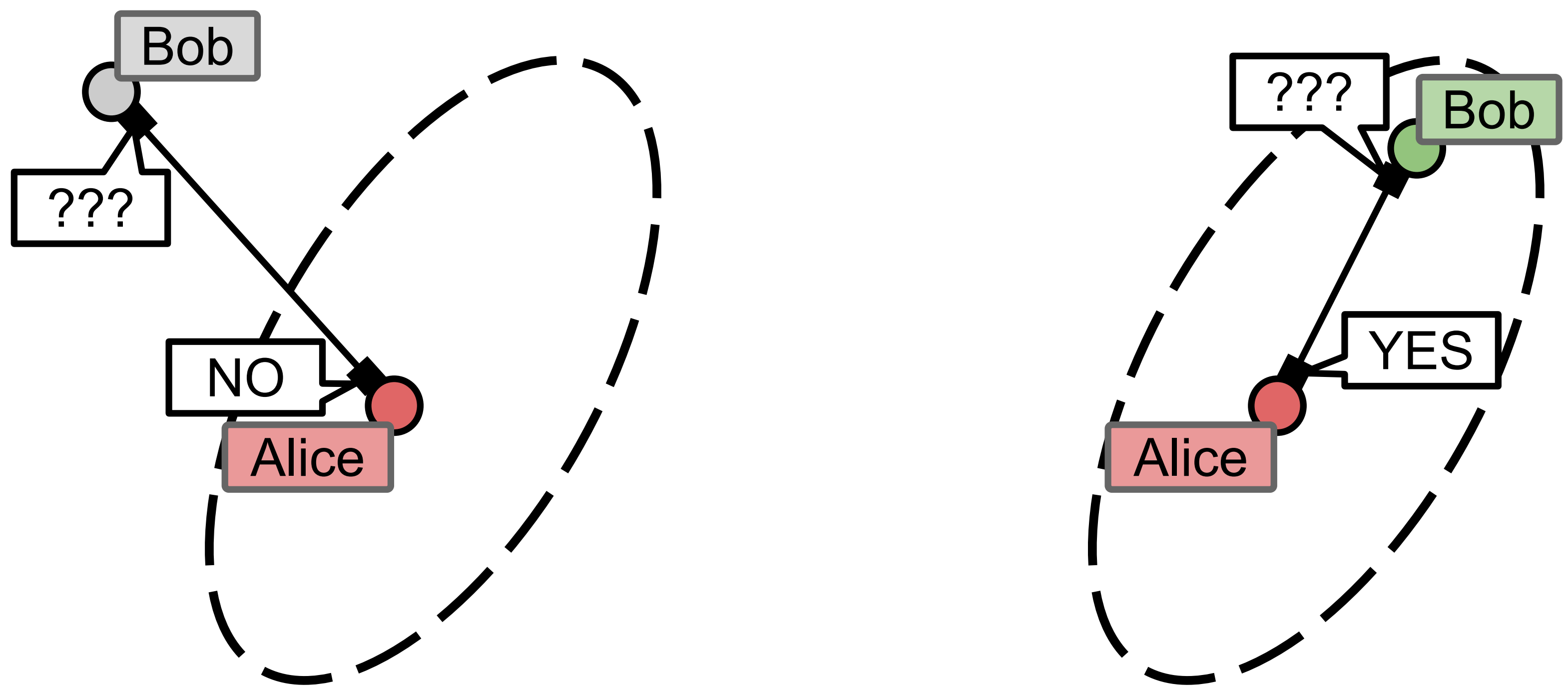
- InnerCircle is provably secure against semi-honest adversaries.

InnerCircle

- Results
  - Under one second
    - r=80 with 80 bits of security
    - r=30 with 112 bits of security
  - Faster than competing solutions
    - $r = 50$ for 80 bits of security
    - $r = 75$ for 112 bits of security
- Parallelization boosts performance almost linearly.

Malicious attackers

Malicious



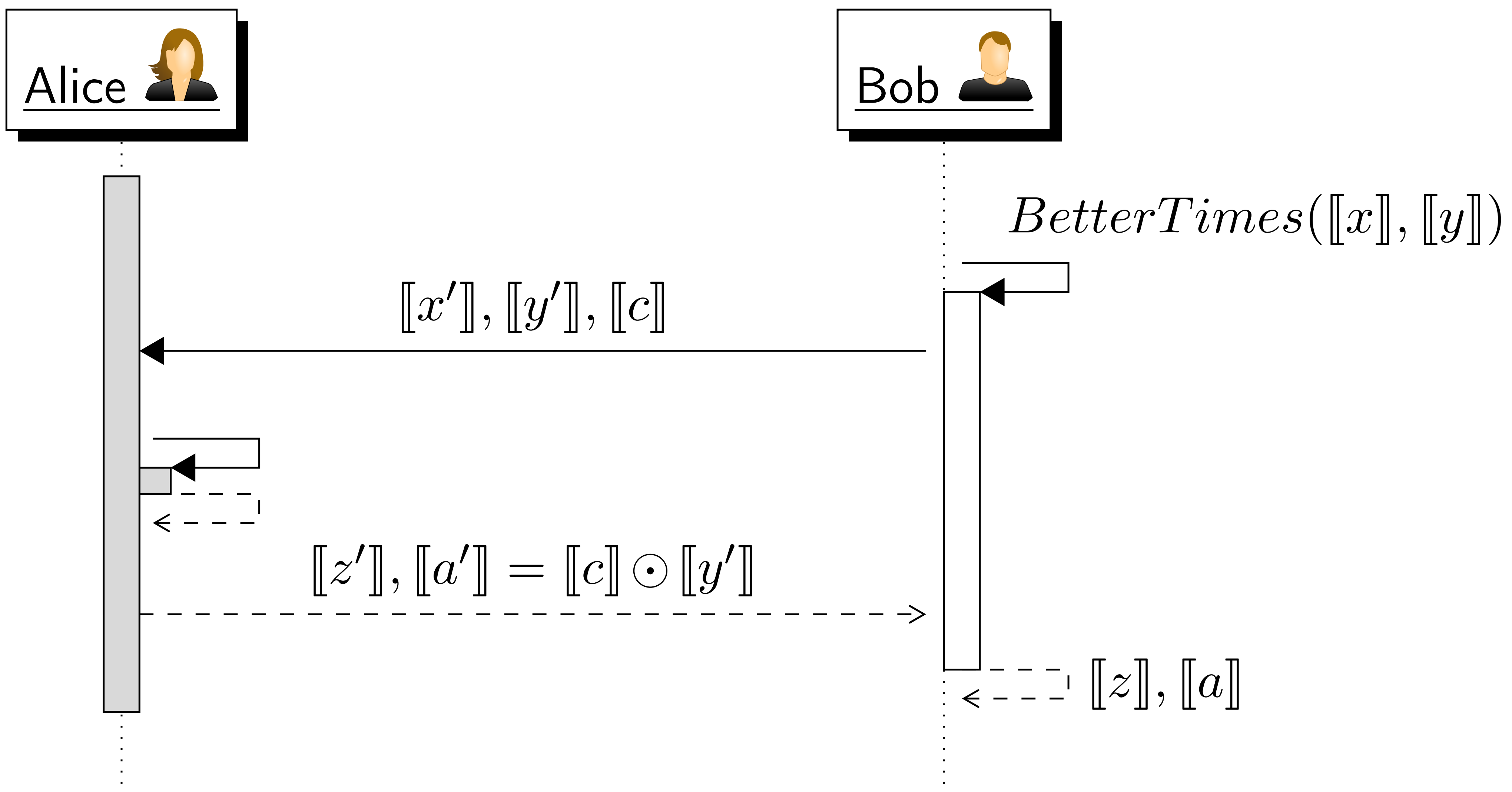
- $\alpha = x_A$
- $\beta = y_A$
- $\gamma = x_A^2 + y_A^2$

Alice sends α, β and γ s.t. $\gamma \neq \alpha^2 + \beta^2$

BetterTimes

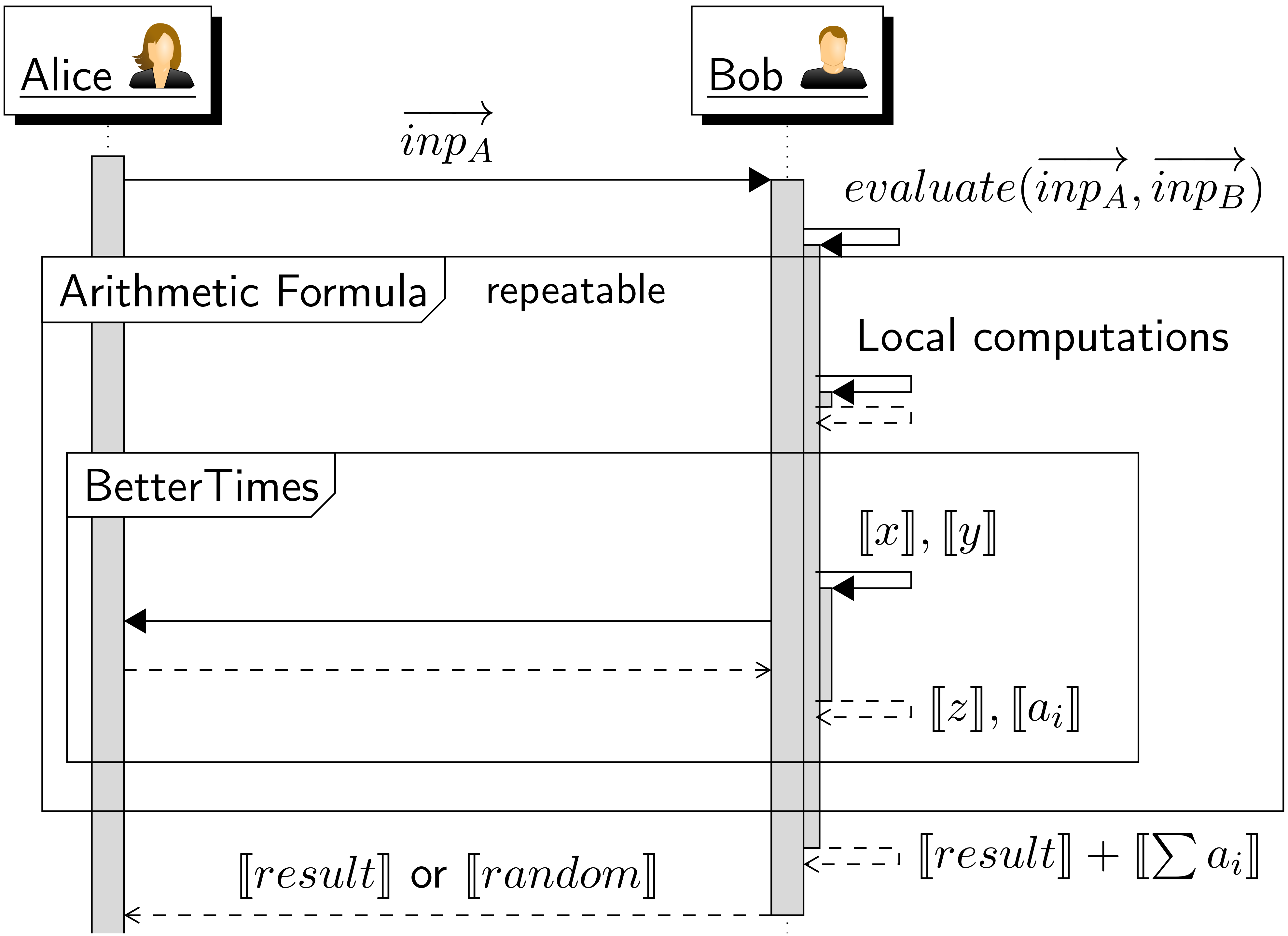
- From [[x]] we cannot compute [[x^2]].
- Missing operation: [[x]*[[y]].
- Idea: Outsource operation to Alice such that if result [[z]] != [[x*y]] then result of functionality is garbled.

BetterTimes

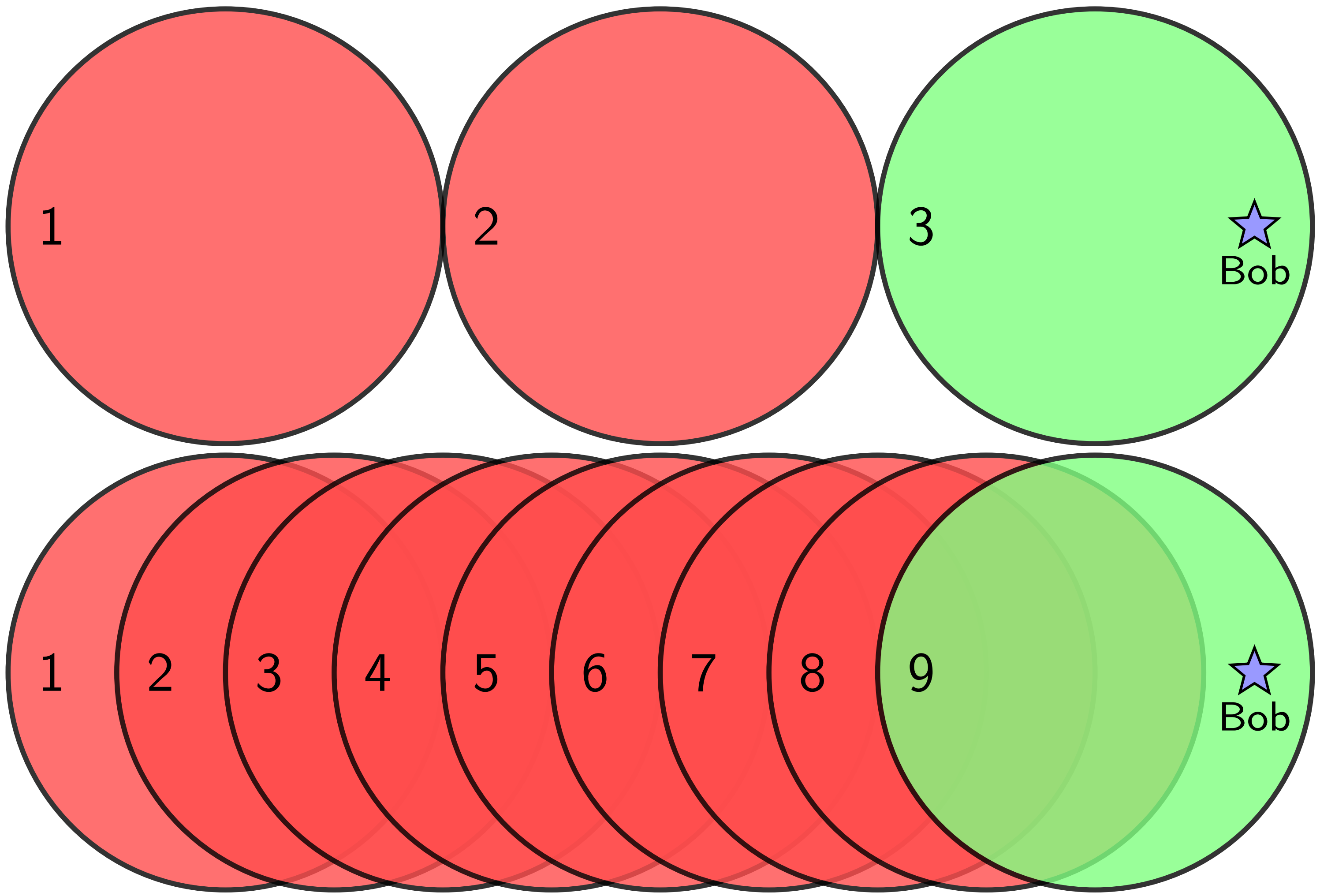


$\llbracket a \rrbracket = (\llbracket a' \rrbracket \ominus (\llbracket z' \rrbracket \oplus \llbracket y' \rrbracket \odot c_a) \odot c_m) \odot \rho,$ with ρ random

BetterTimes



Swiping the plane



MaxSpace

- Simple idea: force attacker to swipe the plane slower by limiting speed.
- **Key insight**: We can compute speed homomorphically and garbled output of proximity request if attacker moves too fast.

MaxSpace

TABLE I: Speeds in m/s and km/h for the used scenarios

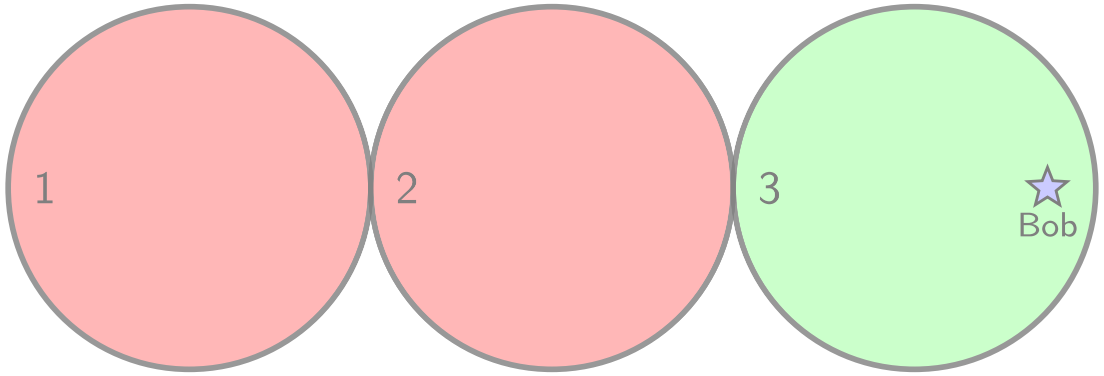
Activity	Walking	Running	Cycling	Bus	Car (highway)
m/s	2	3	5	14	33
km/h	7.2	10.8	18	50.4	118.8

TABLE II: Bounds for different speed radiuses

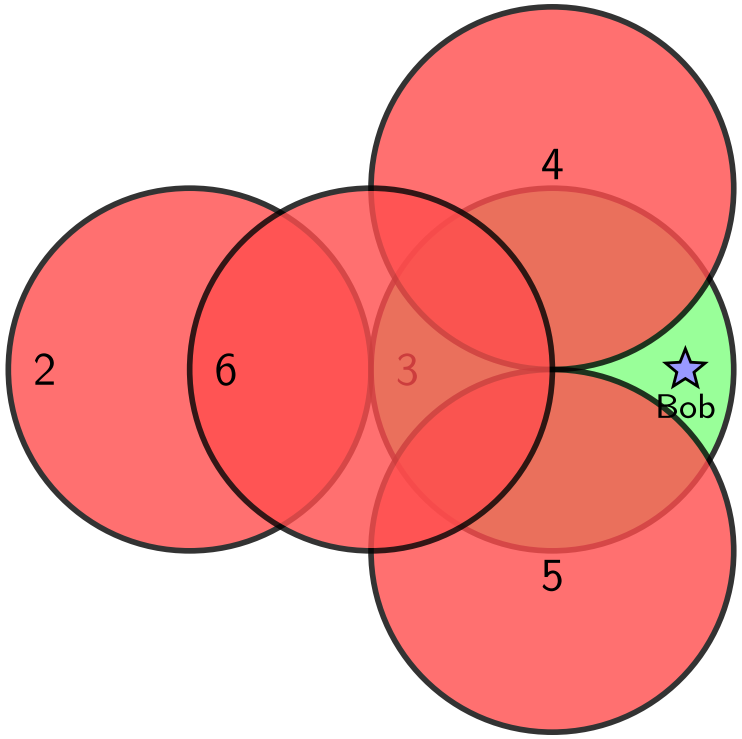
Speed	Radius			
	10	25	50	100
Walking	78.2	194.3	384.4	752.7
Running	52.2	130.0	258.1	508.8
Cycling	31.4	78.2	155.7	308.8
Bus	11.2	28.0	55.9	111.5
Car	4.8	11.9	23.8	47.5

Triangulation

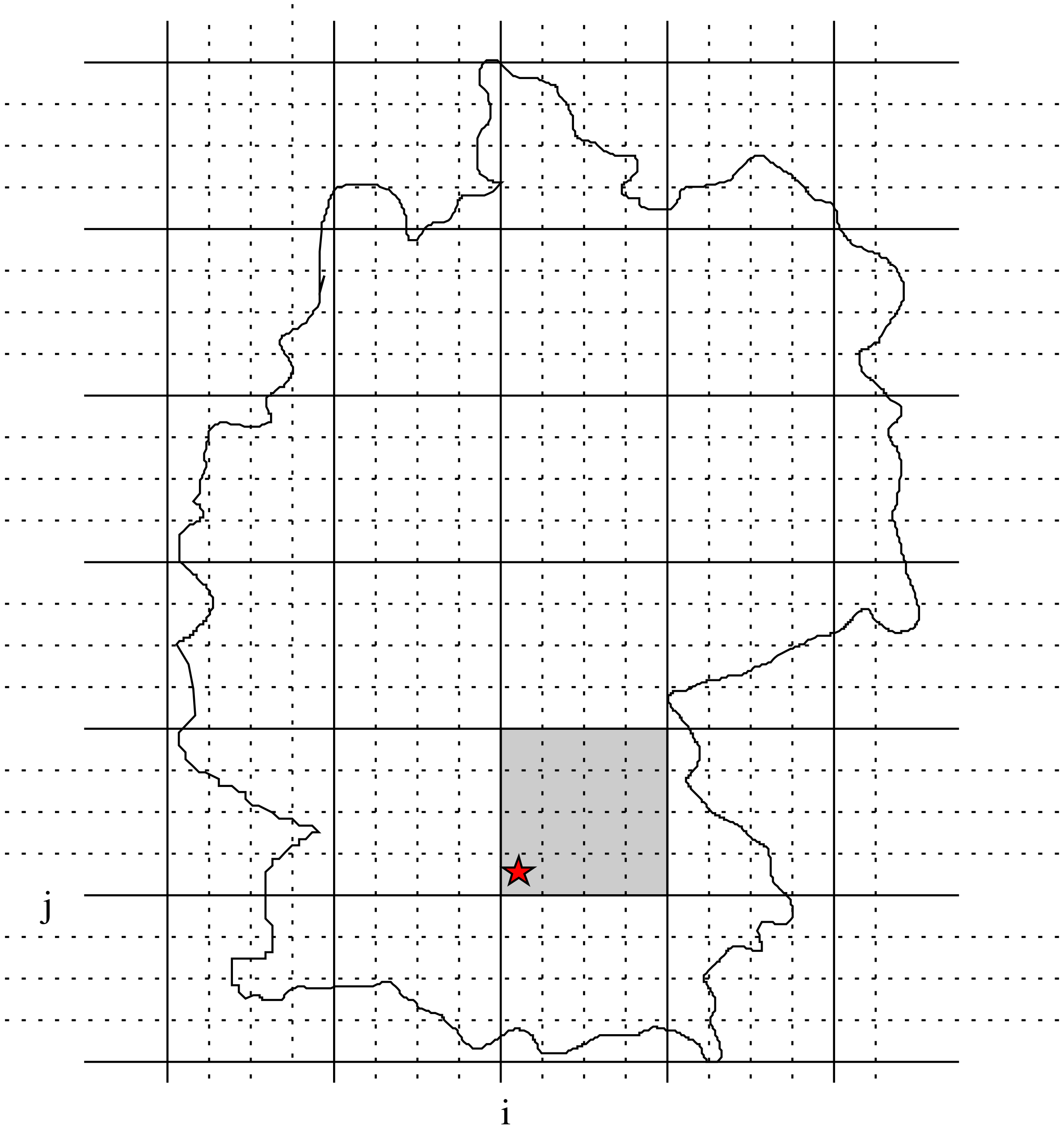
DiskCoverage



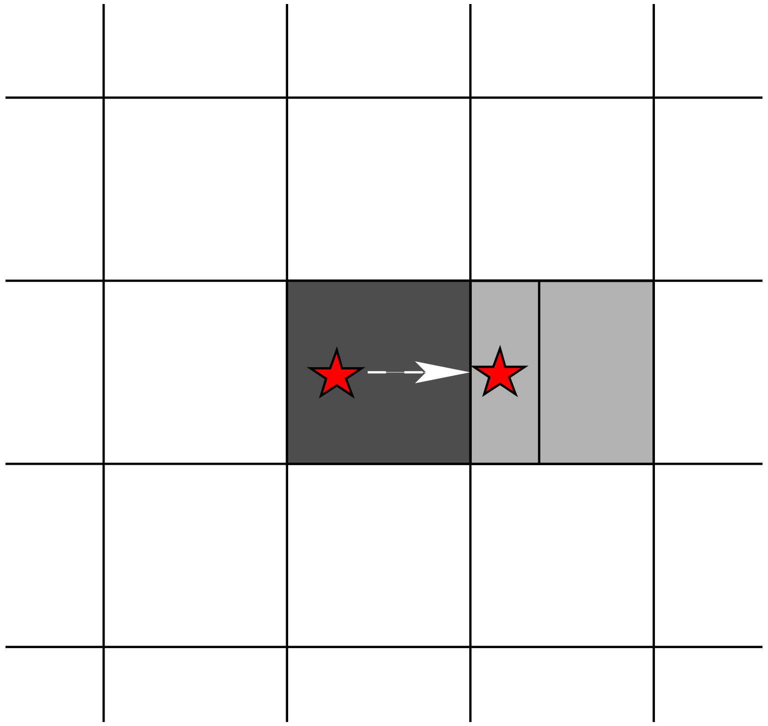
DiskSearch



Grids

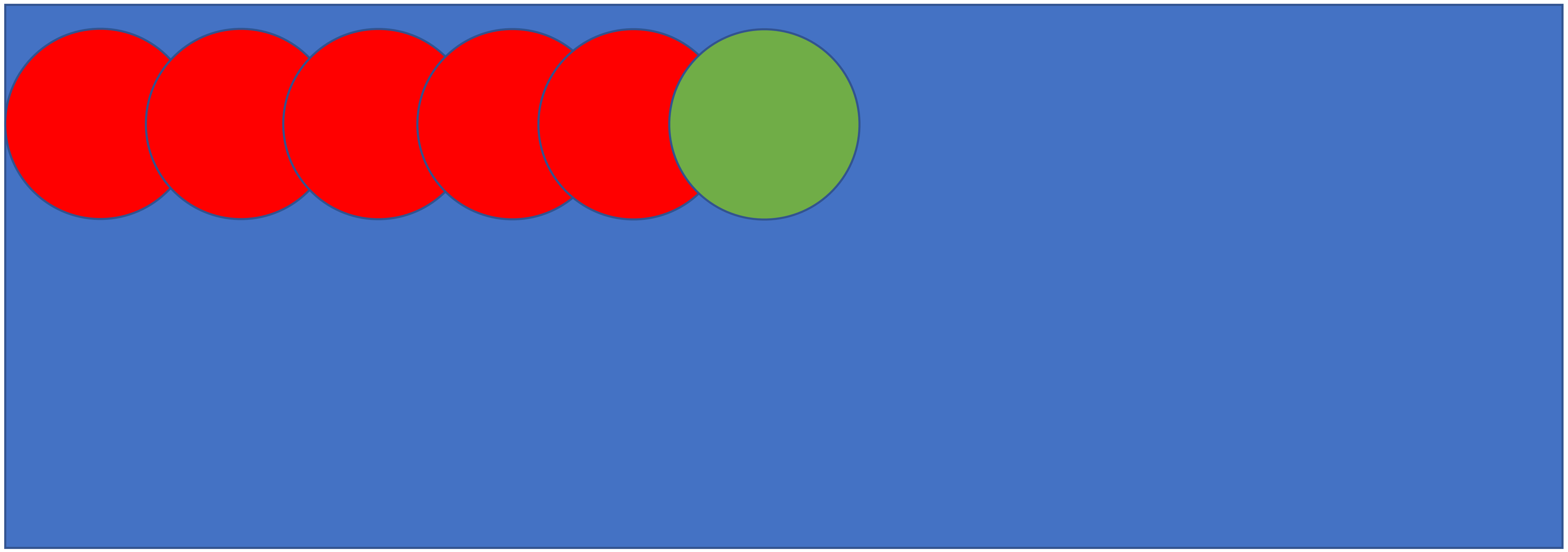


Problem:



Moving targets

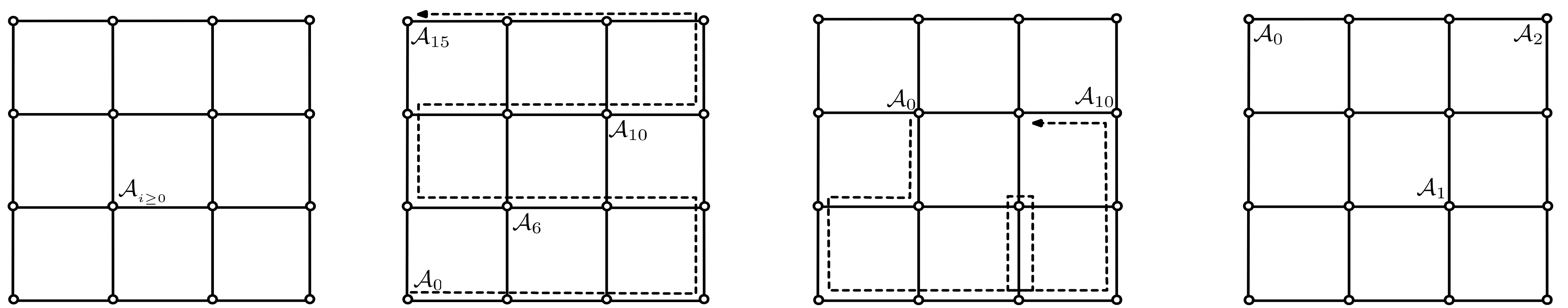
- Typically attacks in this setting involve "parsing" the plane, to then triangulate:



- But what if victim is moving? Should an attacker revisit some of previous guesses? What is his best strategy?

Moving targets

- We consider abstract attacks where both the target and the attacker move according to a particular mobility pattern



static linear random walk random jump

- Our goal is to determine the attacker effort to locate the target with a probability of at least p (usually p= ½).

Model

- We assume that many mobility models can be described by a transition matrix `P` where p_{ij} is the probability of moving from position `i` to `j` at any step

$$B^{(k+1)} = B^{(k)} \cdot P = \begin{pmatrix} 0 & \dots & \overset{i}{1} & \dots & 0 \end{pmatrix} \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,M} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M,1} & p_{M,2} & \cdots & p_{M,M} \end{pmatrix}$$

- Therefore we can calculate the probability of Bob (victim) being at a particular position after `k` steps by taking the `k`th power of `P`

Events of interest

- We are interested in the probability of two events:
  - **E_k** : is the event that Alice locates Bob within k steps (i.e., k + 1 queries)

$$E_k := \{\exists i \leq k \text{ s.t. } \mathcal{A}_i = \mathcal{B}_i\}$$

  - **F_j** : is the event that Alice locates Bob in exactly j steps

$$F_j := \{\mathcal{A}_j = \mathcal{B}_j\}.$$

Bounds

- An **upper bound** on $Pr(E_k)$ gives a **lower bound** on k :
  - If after k steps you have at **most** probability p => need at least k steps to reach p.
  - This is relatively easy to compute with the formula on previous slide.

- A **lower bound** on $Pr(E_k)$ gives an **upper bound** on k :
  - If after k steps you have at least probability p => need at most k steps to reach p.
  - This is harder, it needs a concrete attack strategy to realize an upper bound to p.

Lines vs. Planes

- We first tackle the problem when the space is linear and obtain (rigorous) bounds for *any* attacker and for *any* space size *n* when the victim moves in a random walk.
  - In this case the structure of the matrix P allows for easier algebraic bounds
  - We can test this also numerically.

- In the plane, it is much harder to analytically derive such bounds. Numerically we obtain similar bounds.
  - Matrix structure is more complex in this case!

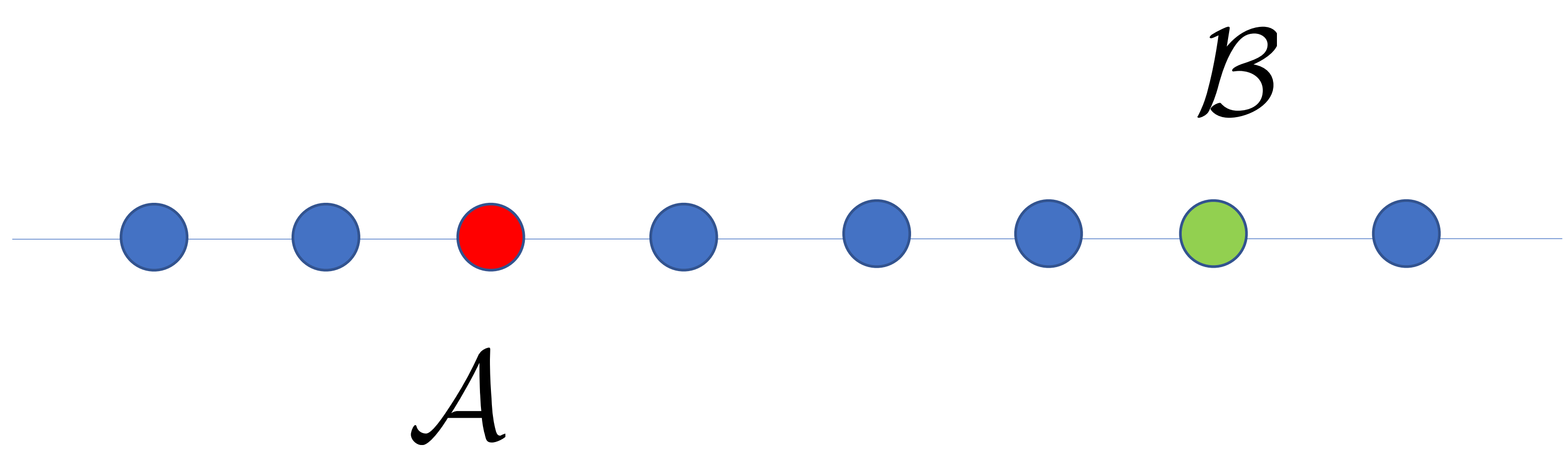
Random Walk Example

<u>Theorem</u>: Considering a random-walking victim, a search space of size `n` and a probability ½, we have that:

$$\sum_{i=0}^{k} \max_j B_j^{(i)}. \longrightarrow \left\lfloor \frac{n}{3} \right\rfloor - 1 \leq k_O \leq \left\lfloor \frac{n}{2} \right\rfloor \longleftarrow$$

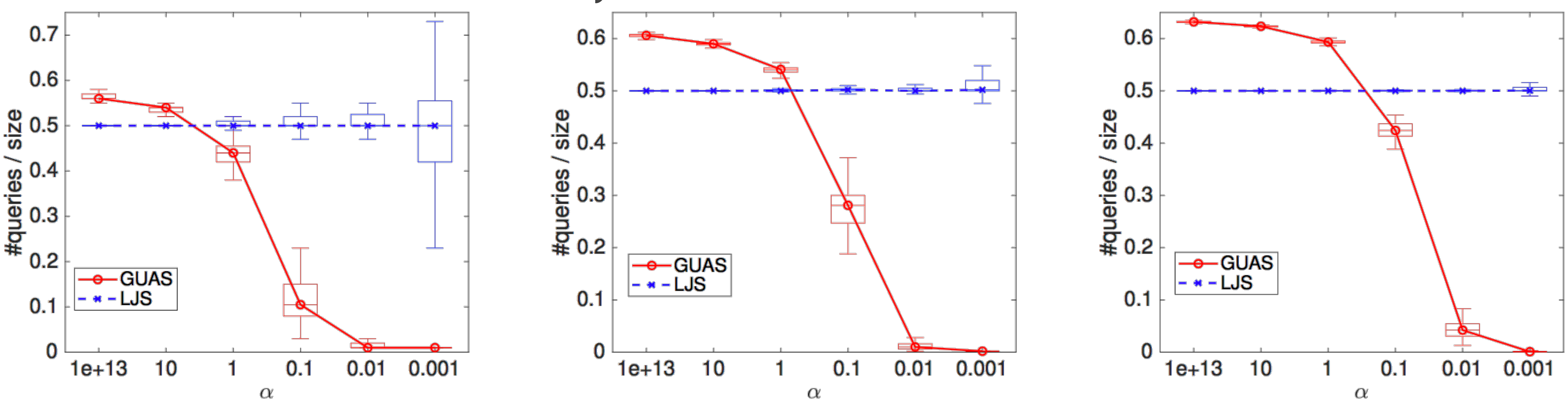
Linear Jump

for a linear search space.



Results on Random Walks

- *Linear Jumping Strategy* ***(LJS)***
  - Achieves the optimal lower bound when the victim's initial position distribution is almost uniform (i.e., large alpha)
- *Greedy Updating Attack Strategy* ***(GUAS)***
  - More effective than LJS for non-uniform initial distributions



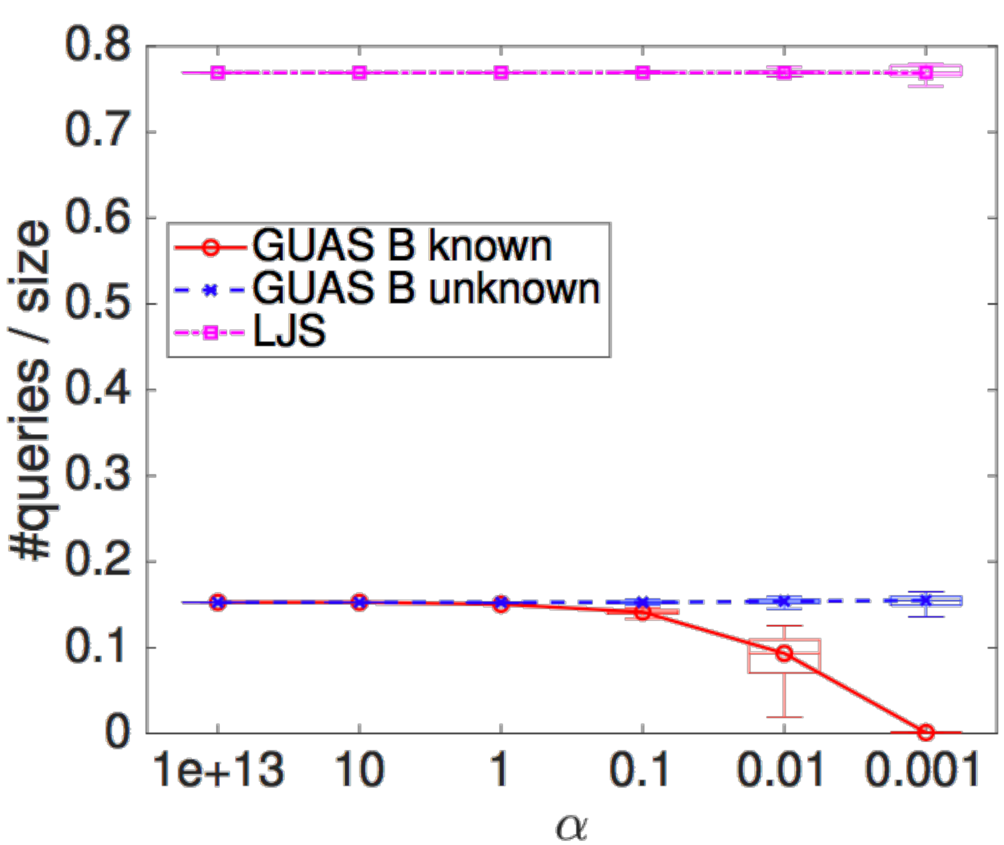
(a) Search space size = 100 (b) Search space size = 500 (c) Search space size = 2000

Evaluation with real mobility models

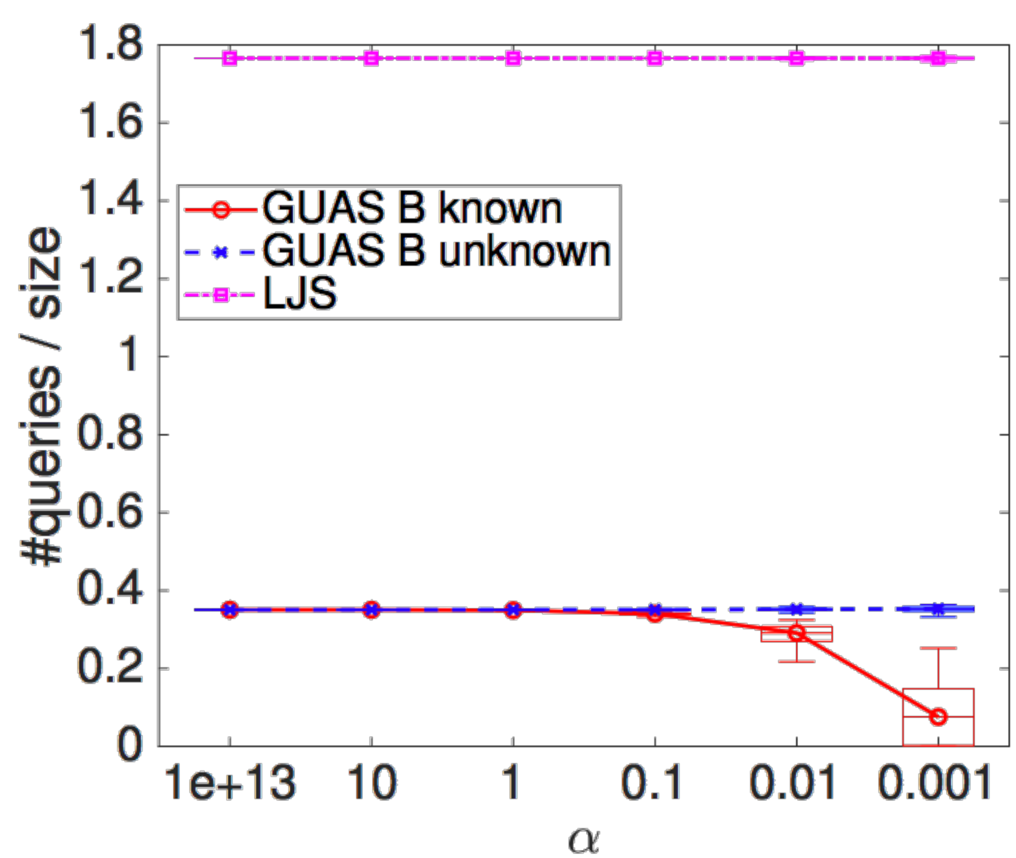
- Finally, we evaluated the performance of these strategies with a real-world dataset
- We derived a transition matrix P_{taxi} from the Beijing Dataset
  - GPS trajectories of taxis from city of Beijing (3rd ring).
  - The area is discretized into 884 locations of 500 x 500m
  - Average sampling interval is around 177 seconds

Results on realistic dataset

- Our results show that GUAS performs significantly better than LJS for more realistic mobility patterns
  - GUAS consistently requires less than N/6 queries for p=0.5
  - LJS requires more than 0.75N queries



(a) Performance with p=0.5



(b) Performance with p=0.8

Conclusions

- We establish a general formula for calculating the probability of the attacker finding the victim after any number of queries
- We give upper and lower bounds on the minimum number of queries to locate a victim with a given probability
  - An optimal attacker needs at most M/2 queries with probability ½
- We implement two attacker strategies (LJS, GUAS) and evaluated them in the case of
  - Random walk victim
  - Realistic mobility dataset
- GUAS strategy performs significantly better with realistic mobility patters
  - The attacker targets the victim in 134 steps (6.6 hours) with probability 1/2

Future Work

- We consider the evaluation of some countermeasures
  - The LBS probabilistically returns a wrong result
  - The LBS could verify that location claims conforms to some assumed transition matrix P
  - The LBS could impose limitations on the number of queries or the speed / frequency of queries

- Evaluation with different mobility models for different modes of transport

- Consider more powerful attackers (e.g., colluding)

- Devise new attacker “optimal” strategies

References

- *Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol*
  P Hallgren, M Ochoa, A Sabelfeld
  PST 2015
- *BetterTimes - Privacy-Assured Outsourced Multiplications for Additively Homomorphic Encryption on Finite Fields.*
  Per A. Hallgren, Martín Ochoa, Andrei Sabelfeld:
  ProvSec 2015
- *MaxPace: Speed-constrained location queries.*
  Per A. Hallgren, Martín Ochoa, Andrei Sabelfeld:
  CNS 2016
- *Indistinguishable regions in geographic privacy.*
  Jorge Cuéllar, Martín Ochoa, Ruben Rios:
  SAC 2012
- *Location Proximity Attacks Against Mobile Targets: Analytical Bounds and Attacker Strategies.*
  Xueou Wang, Xiaolu Hou, Ruben Rios, Per A. Hallgren, Nils Ole Tippenhauer, Martín Ochoa.
  ESORICS 2018