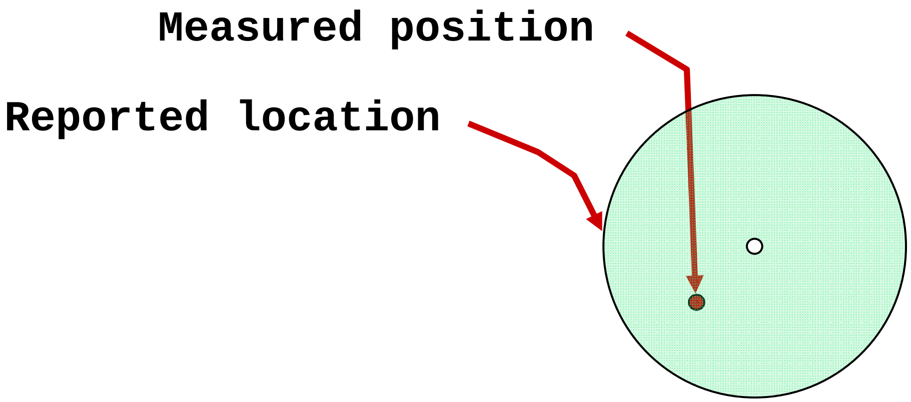
Geopriv: Privacy Preferences for Location Information

draft-ietf-geopriv-policy-23

Jorge Cuellar

Location Obscuring: Want to present location with given Uncertainty



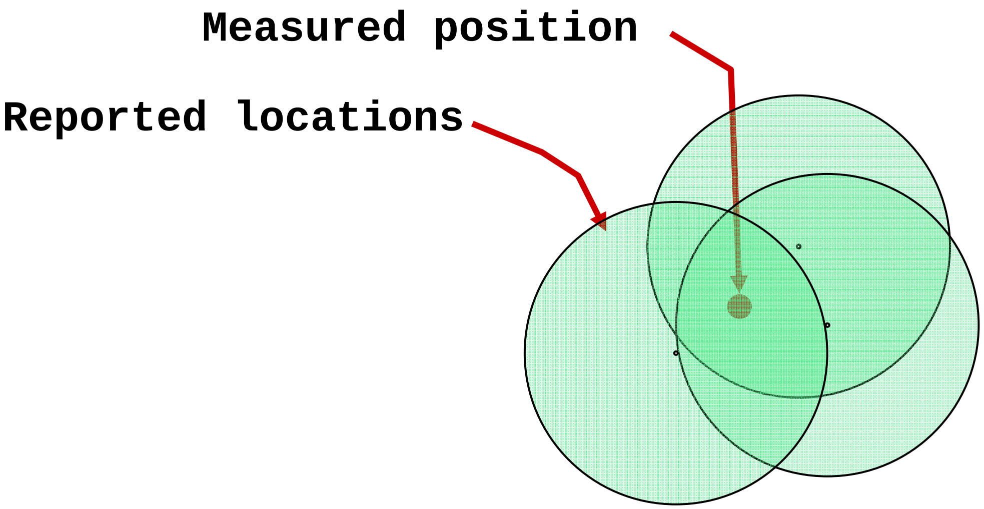
Goal: given a point, the “*measured position*” (say, precise)
find a circle, the “*reported location*”, of radius d (uncertainty)
that contains the point

Assume: we do not force the user to lie

Location Obscuring:
Want to present location with given Uncertainty

1. **Question**: shall each run of the protocol render always a different output (with same input)?

If yes: the intersections provide high precision



Measured position

Reported locations

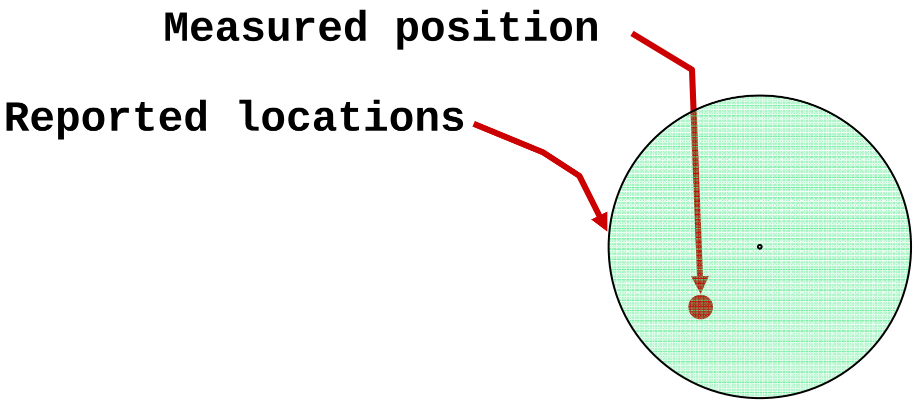
Location Obscuring:
Want to present location with given Uncertainty

1. **Question**: shall each run of the protocol render always a different output (with same input)?

If not: what about small movements?

What about several devices providing location?

How, where to keep state?

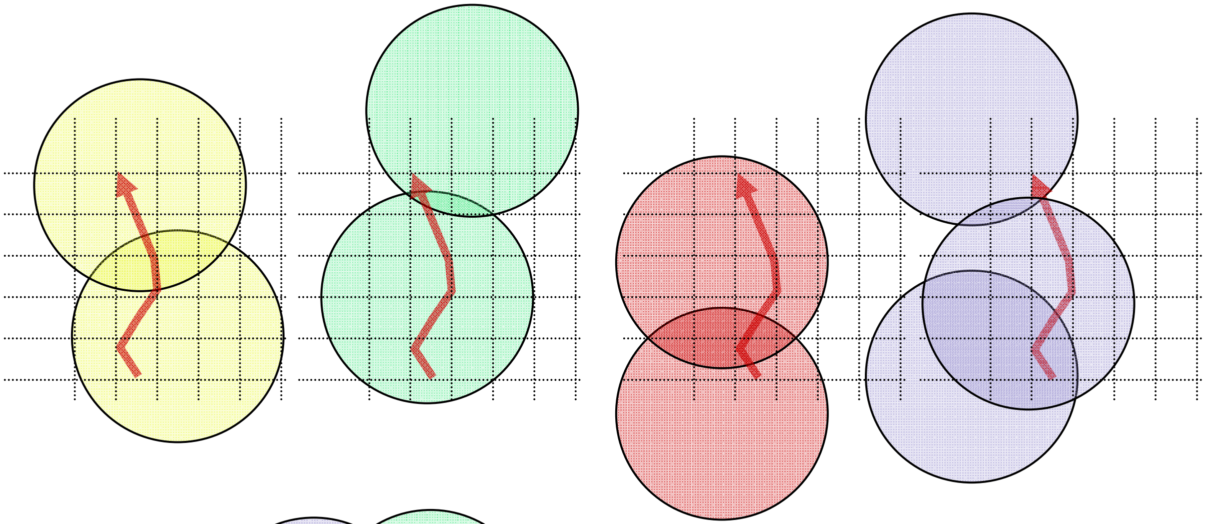


Measured position

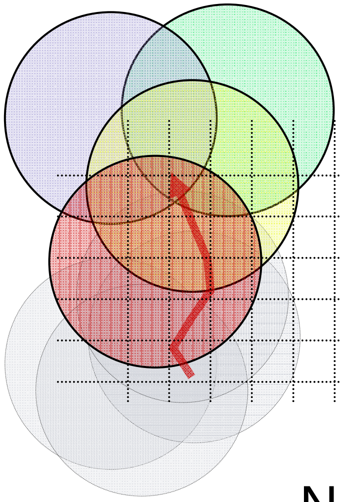
Reported locations

Location Obscuring:
Want to present location with given Uncertainty

Assume you always produce different outputs. What happens if every evening you go to the same **place**?



Look at intersection of the *last* reported locations (& other statistics).



The final destination is leaked with high precision.

Not good!

Distinguishability

2. **Question**: How can you objectively compare solutions ?
   Can you measure how good a protocol is?

Indeed!

- Is the target *here*? Is it *there*?
- We say that the two locations are ***distinguishable*** via the algorithm
- *Any* algorithm partitions the space into ***indistinguishability regions***
  - Two points are in the same region if they are indistinguishable

$$\text{If } \; A_{prov} = \text{Area of location provided}$$

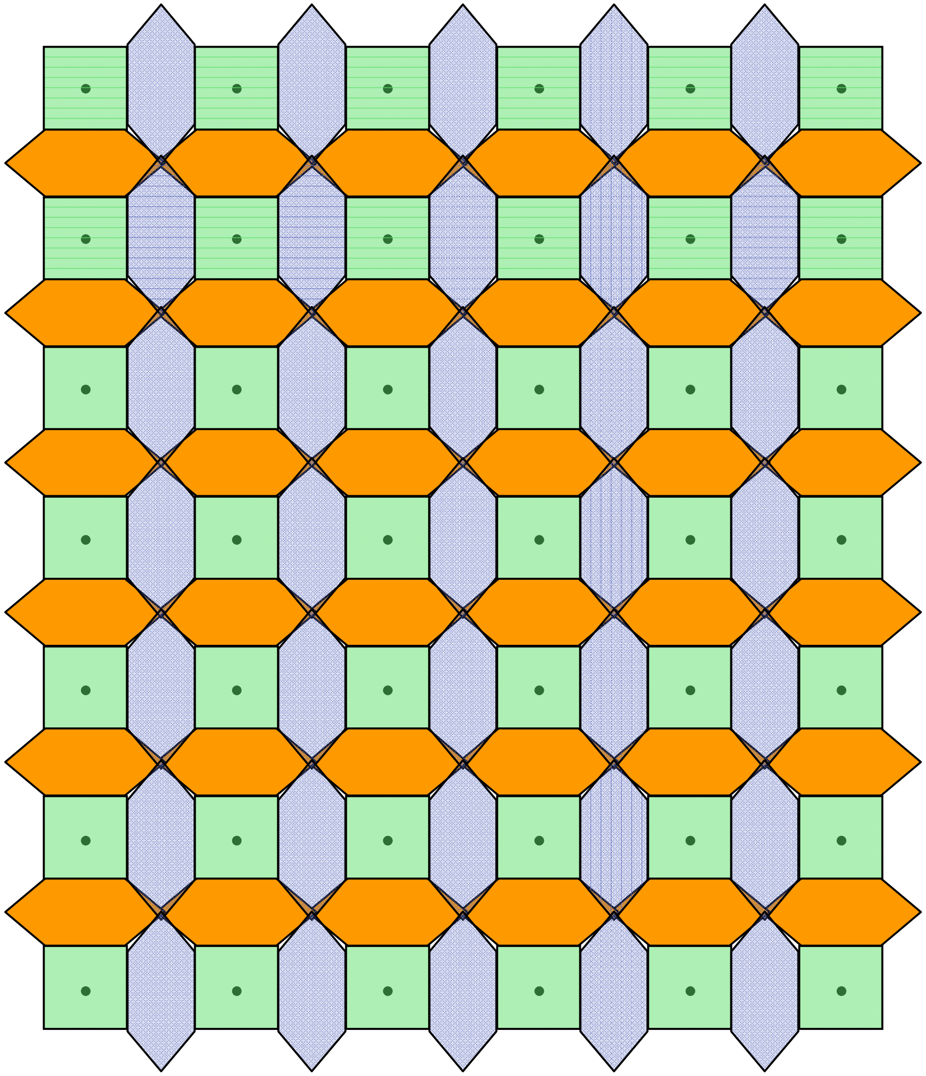
$$\text{and } \; A_{block} = \text{Area of a block}$$

$$\text{then } \; 1-(A_{block}/A_{prov}) = \text{Leakage of algorithm}$$

Solution

- Construct fixed blocks as big as possible
    - make the reported location depend on the block,
    - not on the point within the block
- Necessary to introduce transition intermediate blocks, in order to diffuse the area when moving from one block to another
- What ever method is chosen, it MUST be standardized in detail, because the intersection of the outputs of different algorithms will otherwise provide a high information leakage
- A simple version of the algorithm based on a rectangular grid is in draft-ietf-geopriv-policy-23. It offers:
    - protection for static targets,
    - (limited) protection for moving targets, and
    - protection for targets that regularly visits a certain location

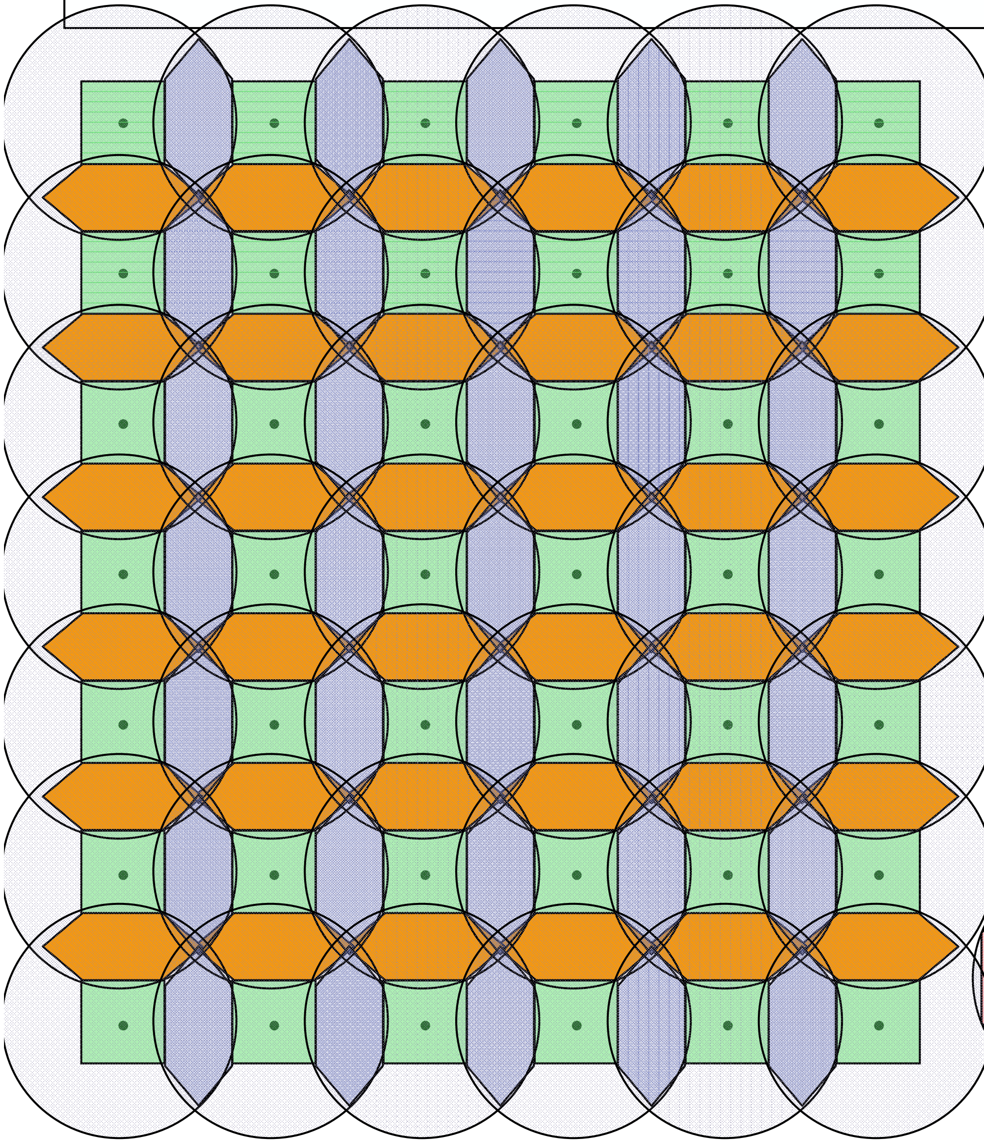
An Algorithm based on a grid

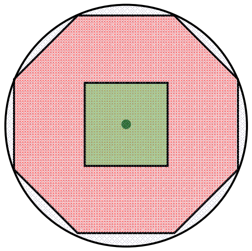


Construct a grid of points

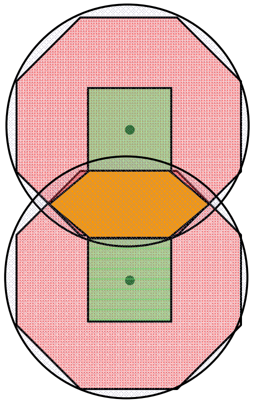
Find octagons around the points
- hexagons = intersection of octagons
- squares = areas in only one octagon
- such that all “blocks” (squares and hexagons) have the same area

An Algorithm based on a grid



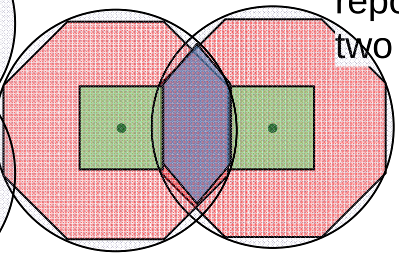


If you are in a green square,
report the smallest circle that
contains the whole octahedrons
(the circumcircle of the octahedrons)



If you are in a orange hexagon,
report the circumcircle of any of the
two octahedrons, north or south

If you are in a violet hexagon,
report the circumcircle of any of the
two octahedrons, west or east



Solution ok, but is it easy to calculate?

```
  o : real  (choose from a small table)
  P := sqrt(3)/6
  q := 1 - p
  prob:  real

Function choose(Ma, Mb: real * real): real * real;
  {rand:= Random[0,1];

  If     prev-M1 == Ma Then
         If rand < prob Then choose := Ma;
                        Else choose := Mb;  EndIf
  Elseif prev-M1 == Mb Then
         If rand < prob Then choose := Mb;
                        Else choose := Ma;  EndIf
  Else   If rand < 0.5  Then choose := Ma;
                        Else choose := Mb;  EndIf }
```

```
{ d := radius/1000;
  d1:= (d * 180) / (pi*M*cos(o));
  d2:= d / 110.6;

  l := d1*floor(m/d1)
  r := l+d1;
  b := o+d2*floor(n-o/d2);
  t := b+d2;
  x := (m-l)/(r-l);
  y := (n-b)/(t-b);

  SW := (l,b);
  SE := (r,b);
  NW := (l,t);
  NE := (r,t);

  If     x < p and y < p      Then M1 := SW;
  Elseif x < p and q <= y     Then M1 := NW;
  Elseif q <= x and y < p     Then M1 := SE;
  Elseif q <= x and q <= y    Then M1 := NE;
  Elseif p <= x and x < q and y < x  and y < 1-x
         Then M1 := choose(SW,SE);
  Elseif p <= y and y < q and x <= y and y < 1-x
         Then M1 := choose(SW,NW);
  Elseif p <= y and y < q and y < x  and 1-x <= y
         Then M1 := choose(SE,NE);
  Elseif p <= x and x < q and x <= y and 1-x <= y
         Then M1 := choose(NW,NE);
  Endif                                           }
```