# 优化方案

初始方案暂且不谈，先说优化方案。

一：地图分块，减少计算量

二：把路网嵌入向量打包起来，使得加密等操作可以一步完成

## 步骤：Setup、Location-update、Ride-request、Ride-matching

为了保证分块后的精度，我们首先寻找与乘客所处区域α的最近司机，再以乘客到该司机的距离为半径来判断区域α周围的区域是否划入范围（就是看以该距离为半径的圆是否“触”到了周围区域）

## 使用技术：

路网嵌入（ROAD NETWORK ENBEDDING），用于提高最短距离计算效率

同态加密 （Homomorphic），用于提供密文状态下的加法（及乘法）同步到明文中

混淆电路（Garbled Circuit），用于提供CP（Crypto Provider）和ORH服务器的安全两方计算、有点像零知识证明

## 一、初始化 Setup

1.ORH 地图分块(如图1所示)

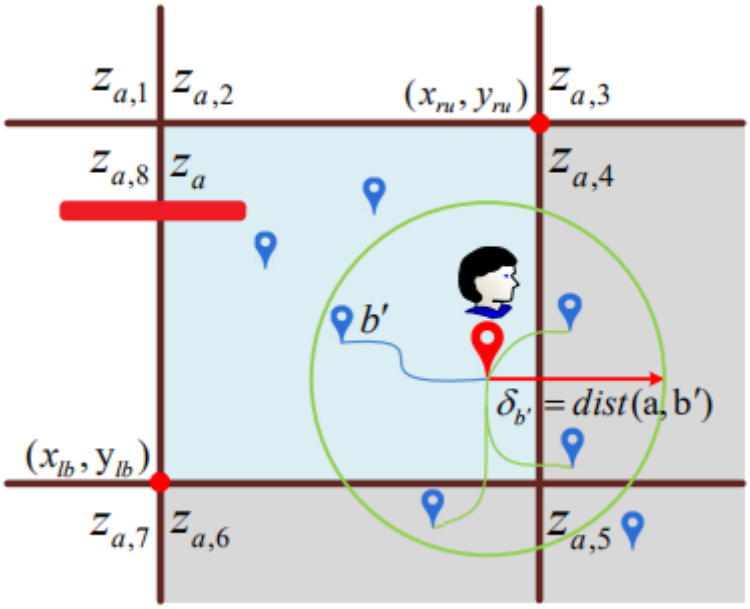


图 1

2.ORH 计算 地图道路的 路网嵌入向量

3.CP 初始化同态加密公钥并分发

## 二、司机 位置更新 Location-update

此步骤主要用于司机上传自己的位置信息（在pRide里是上传自己的路网嵌入信息）

1.司机端　计算自己的路网嵌入变量S

根据从服务器获取的地图道路路网嵌入向量、来在本地计算自己的路网嵌入向量S（这在路网嵌入已经讲过了）

2.司机端　把S打包成一个变量P

就是像进制位一样叠加维度S(如图2)

***Data packing:*** let $S(u) = (S_1(u), \ldots, S_\kappa(u))$ be a $\kappa$-dimensional sketch vector, we can pack it through a polynomial given by

$$P(u) = \sum_{j=1}^{\kappa} S_j(u) x^{j-1},$$

where the base $x = 2^\Phi$ and $\Phi$ is a parameter that is large enough to separate two dimension values at the bit level.

图 2

3.司机端　把P用从CP获得的公钥加密成[P(b)]，把这个[P(b)]和所处区域Zα上传到ORH

4.ORH　随机生成一个路网嵌入向量u，并加密之成[P(u)]

5.ORH　使用[P(b)]-[P(u)]=>[P(b')]

## 三、乘客 乘车请求 Ride-request

1.乘客端　把自己的坐标x、y加密为[x]、[y]

2.乘客端　计算自己的路网嵌入向量S、打包、加密之得到[P(a)]

3.乘客端 上传[P(a)]、[x]、[y]、所处区域Zα给ORH

4.ORH　随机生成两随机数ηx、ηy，把其当做一个坐标值，计算其的网嵌入向量S、打包、加密之得到[P(η)]

5.ORH　计算　[P(η)]-[P(a)]=》[P(a')]　　;　　[x]-[ηx]=>[x']　　;　　[y]-[ηy]=>[y']

《第五步是为了在4.2提高匹配精度的阶段判断坐标，所以把坐标也放到了距离向量中[P(a)]》

## 四、ORH 乘车匹配 Ride-matching

### 4.1.乘客所属区域zα匹配 Local Zone matching

1.ORH　计算　[P(d)']=[P(a)]-[P(b')]=[P(a)]-[P(b)]+[P(u)]

2.ORH　将 [P(d)']发给CP

3.CP　解密 [P(d)']得到 P(d)'、并且获得其混淆值 ~P(d)'　《这里的混淆电路还不太明白怎么操作的》

4.ORH　通过1-out-of-2 OT protocol 获得[P(u)]的混淆值　~[P(u)]

5.通过算法2(如图3)判断两个司机哪个离乘客近

算法注释：

第一、二步是计算第一个司机到该乘客的最近距离(把P打包、第二步是找到两个P之间的最大值)

第三、四步同上计算第二个司机到该乘客的最近距离

第五步是判断哪个司机离乘客近

**Algorithm 2** Distance Comparison Circuit

**Input:** $\widehat{P(d_{b_1})'}$, $\widehat{P(d_{b_2})'}$, $\widehat{P(\mu_{b_1})}$ and $\widehat{P(\mu_{b_2})}$

1: Unpack $P(d_{b_1})'$ and $P(\mu_{b_1})$ into $d'_{b_1,1}, \ldots, d'_{b_1,\kappa}$ and $\mu_{b_1,1}, \ldots, \mu_{b_1,\kappa}$, respectively

2: Set $\delta_{b_1} = \max_{1 \leqslant j \leqslant \kappa} \left( \left| d'_{b_1,j} - \mu_{b_1,j} \right| \right)$

3: Similar, unpack $P(d_{b_2})'$ and $P(\mu_{b_2})$, and obtain $d'_{b_2,1}, \ldots, d'_{b_2,\kappa}$ and $\mu_{b_2,1}, \ldots, \mu_{b_2,\kappa}$

4: Set $\delta_{b_2} = \max_{1 \leqslant j \leqslant \kappa} (|d'_{b_2,j} - \mu_{b_2,j}|)$

5: Return $\operatorname{argmin}(\delta_{b_1}, \delta_{b_2})$

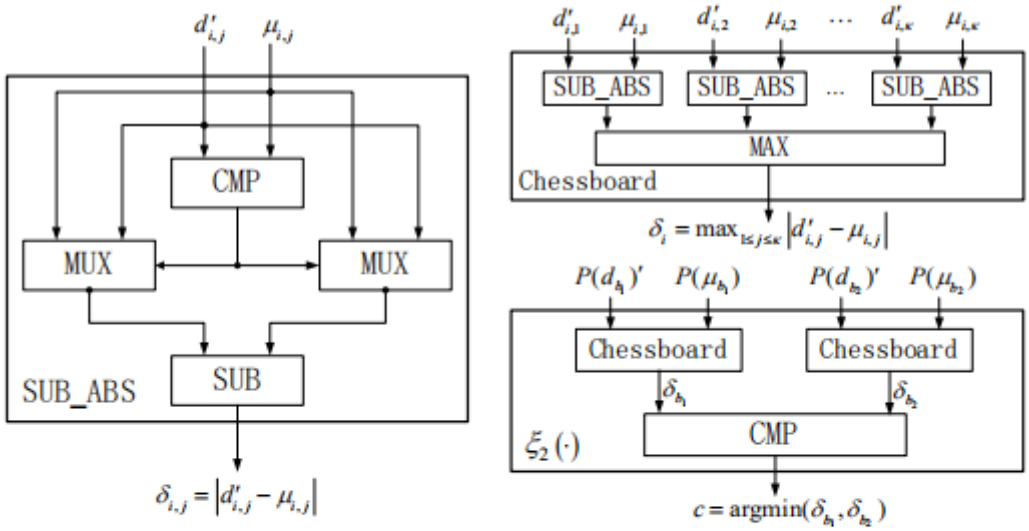


Fig. 3: The construction details of Distance Comparison Circuit $\xi_2(\cdot)$, where *CMP*, *MUX*, *SUB* and *MAX* denote a comparator circuit, a multiplexer circuit, a subtractor circuit and a maximum circuit, respectively.

图 3 上为算法、下为混淆电路

## 4.2.提高匹配精度 Refinement

1.ORH 把4.1中计算出来的，乘客所处区域zα与最近司机的[P(d)'] (4.1.1中计算)、

与[x]-[ηx]=>[x'] ； [y]-[ηy]=>[y']都发给CP

2.CP 将其解密并获得混淆值 ~P(d)'、~[x']、~[y']

3.ORH 通过1-out-of-2 OT protocol 获得~[P(η)] ； ~[ηx] ； ~[ηy]

4.通过如下图算法3(如图4)判断其他八块区域za1, za2, za3, za4, za5, za6, za7,za8（具体见图1）是否有需要计算的必要

算法注释：

第四步是计算乘客的坐标x、y

第五、六步是计算第一个司机到该乘客的最近距离(把P打包、第二步是找到两个P之间的最大值)

第七步是计算Γ 「是一个8位的二进制数，每一位指示一个区域是否该计算（就是看以该距离为半径的圆是否“触”到了周围区域，具体看图1）

第八步是判断 区域zα4是否该计算，后面9到15步同理，是很简单的距离判断（见图一）

**Algorithm 3** Refinement Circuit

**Input:**

1: 1) The location: $\widehat{x'_a}$, $\widehat{y'_a}$, $\widehat{\eta_x}$, $\widehat{\eta_y}$
2: 2) The distance: $\widehat{P(d_{b'})'}$, $\widehat{P(\eta_a)}$
3: 2) The zone: $x_{lb}$, $y_{lb}$, $x_{ru}$, and $y_{ru}$

**Output:** An array of zones Γ that a nearer driver may exist.

4: Compute $x_a = x'_a - \eta_x$ and $y_a = y'_a - \eta_y$
5: Unpack $P(d_{b'})'$ and $P(\eta_a)$ into $d'_{b',1}, \ldots, d'_{b',\kappa}$ and $\eta_{a,1}, \ldots, \eta_{a,\kappa}$, respectively.
6: Set $\delta_{b'} = \max_{1 \leqslant j \leqslant \kappa}(|d'_{b',j} - \eta_{a,j}|)$
7: Initialize an empty array Γ
8: If $x_a + \delta_{b'} > x_{ru}$, then add $z_{a,4}$ into Γ
9: If $x_a - \delta_{b'} < x_{lb}$, then add $z_{a,8}$ into Γ
10: If $y_a + \delta_{b'} > y_{ru}$, then add $z_{a,2}$ into Γ
11: If $y_a - \delta_{b'} < y_{lb}$, then add $z_{a,6}$ into Γ
12: If $(x_a - x_{lb})^2 + (y_a - y_{lb})^2 < \delta^2_{b'}$, then add $z_{a,7}$ into Γ
13: If $(x_a - x_{lb})^2 + (y_a - y_{ru})^2 < \delta^2_{b'}$, then add $z_{a,1}$ into Γ
14: If $(x_a - x_{ru})^2 + (y_a - y_{ru})^2 < \delta^2_{b'}$, then add $z_{a,3}$ into Γ
15: If $(x_a - x_{ru})^2 + (y_a - y_{lb})^2 < \delta^2_{b'}$, then add $z_{a,5}$ into Γ
16: Return Γ

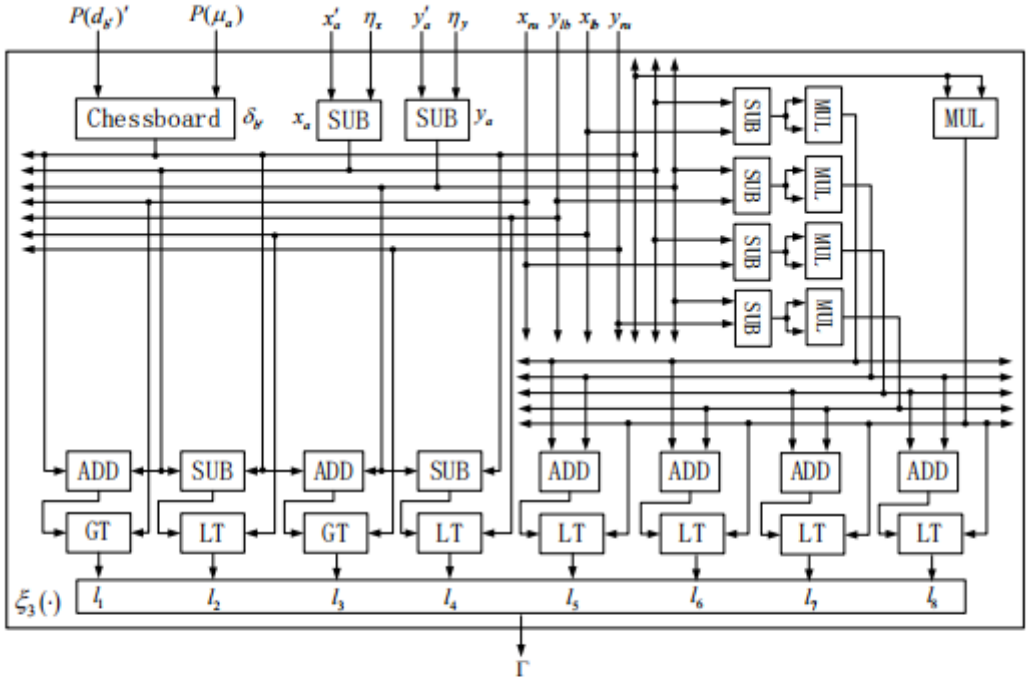


Fig. 4: The construction details of Refinement Circuit $\xi_3(\cdot)$, which outputs a bit array that indicates the indexes of the adjacent zones that a nearer driver may exist. The *ADD*, *GT*, *LT* and *MUL* denote *addition* circuit, *greater than* circuit, *less than* circuit and *multiplication* circuit, respectively.

图 4 上为算法、下为混淆电路

5.通过算法3获得有需要计算得必要的其他区域、

然后通过4.1Local Zone matching算法计算该区域中离乘客最近的司机，若该司机到乘客的距离B比乘客到本区域最短距离A小，就刷新最短距离A。最终我们得到距离最近的司机，并且将乘客位置信息发送给他.....