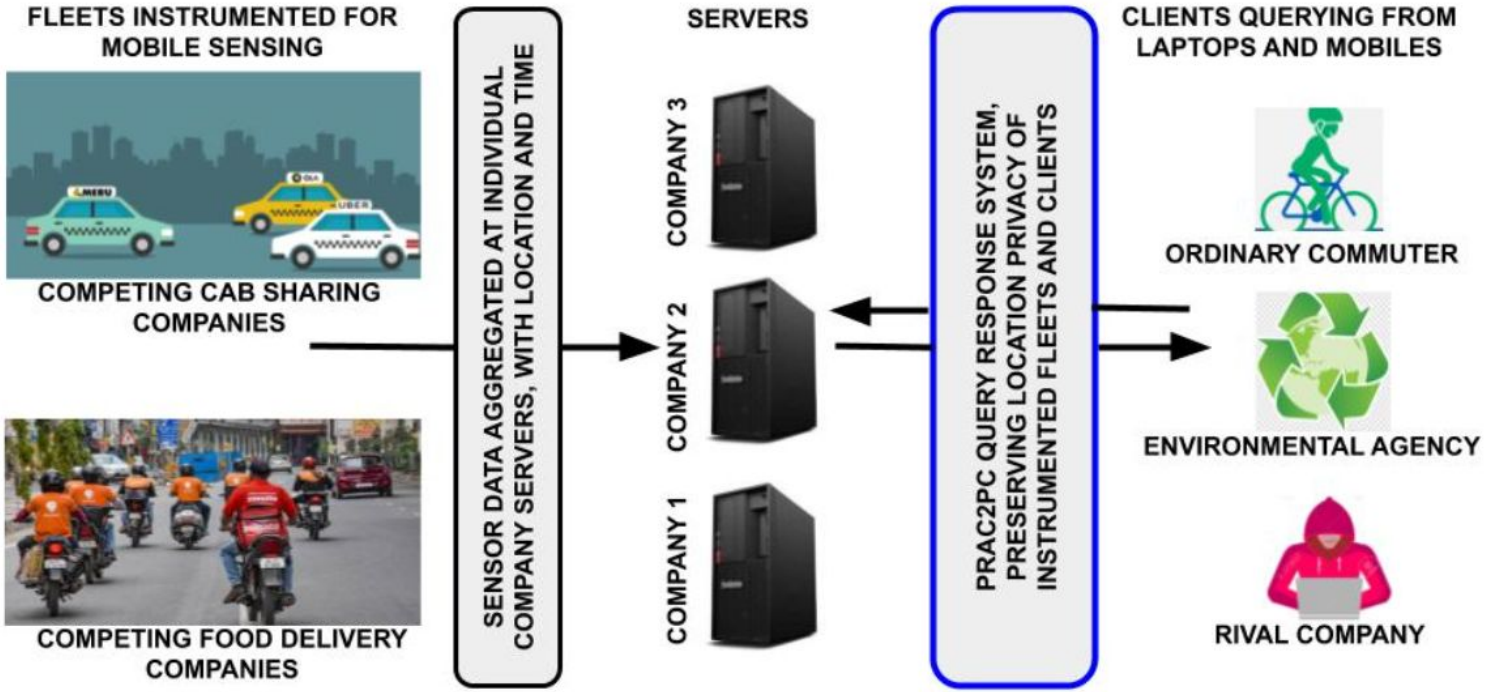
Prac2PC extension to MPC

Siddhant Mago, 2017CS50419
Ashish R Nair, 2017CS50521
Rahul Yadav, 2017CS50602

Under the supervision of Prof. Rijurekha Sen

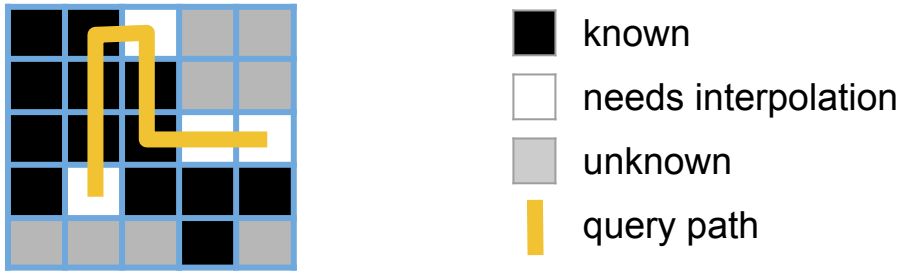
Problem Description



Prac2PC

Key points:

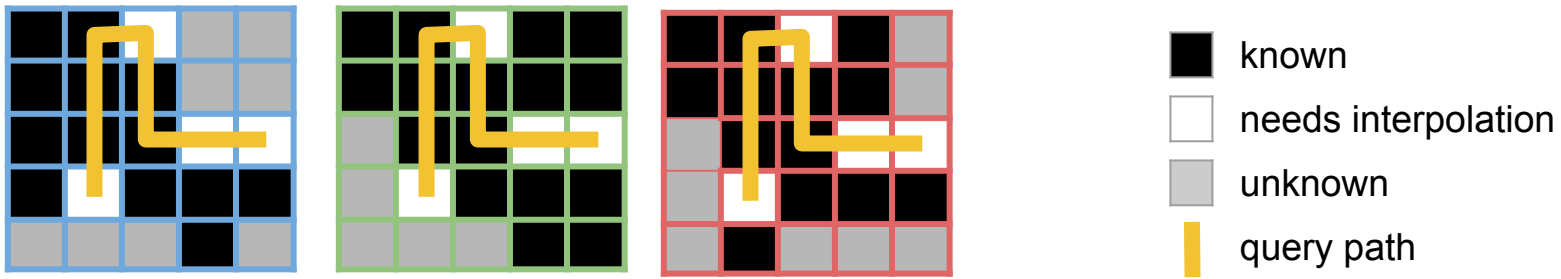
- based on Yao’s garbled circuit and oblivious transfer (2PC)
- a client could ask queries from only a single server
- server only performed interpolation on its own data
- queries accurate only in areas where one cab company's fleet is located



Prac2PC to PracMPC

Key points:

- 2PC protocols don’t work in MPC setting
- a client asks queries from multiple servers
- servers may perform joint interpolation
- queries answered using more data — more reliable predictions



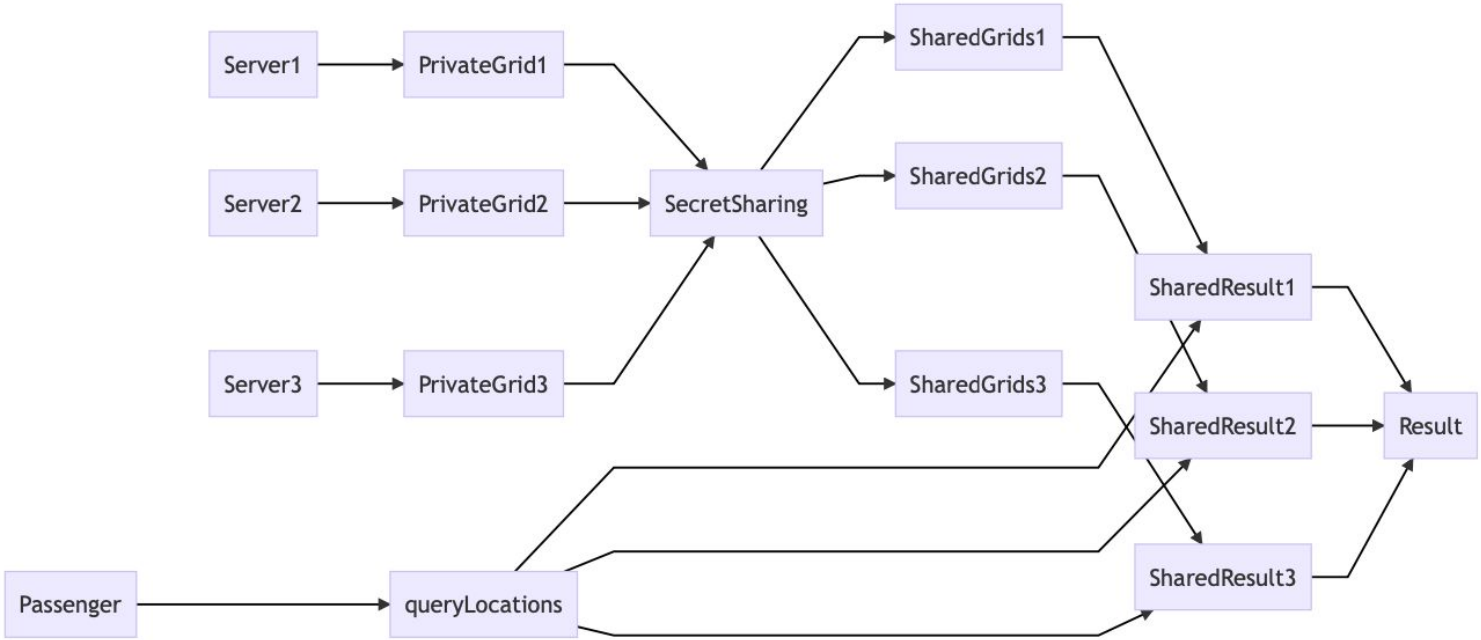
Phases of the Project

- Understanding and exploring secure MPC protocols and libraries
- Answering queries using merged interpolated grids
- Jointly interpolating missing pollution values

Answering queries using merged grids

(Interpolation happens independently on each server)

The Pipeline



Secure MPC for query processing with pre-computed private pollution grids

Merging Pollution Grids

- Need to merge values from different servers available for same locations
- We use the least variance estimate to combine values
- Each server shares it’s pollution value with others using secret sharing
- Secrets get merged at each server
- Each server has a secret of the combined grid in the end

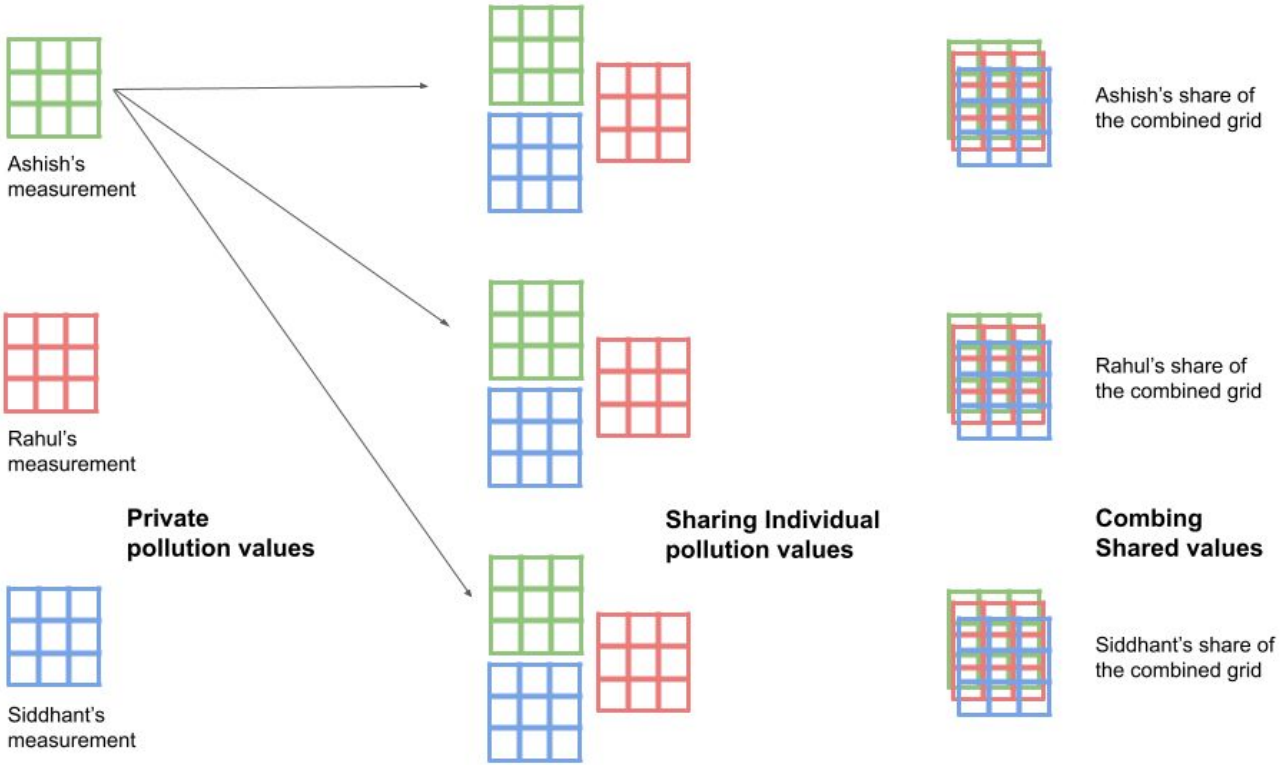
$$\hat{p}(x,y) = \frac{\sum_i^n p_i(x,y)^2/\sigma_i(x,y)^2}{\sum_i^n 1/\sigma_i(x,y)^2}$$

Combining pollution values from different sources

$$\hat{\sigma}^2(x,y) = \frac{1}{\sum_i^n 1/\sigma_i(x,y)^2}$$

Combining confidence values

Merging Pollution Grids



Private interpolated grids from different servers merged jointly

Answering Queries

comment: average pollution along path

for each $(x, y) \in query.subgrid$

do
$$\begin{cases} onQueryPath \leftarrow query.mask(x, y) \\ sum \leftarrow sum + onQueryPath * \hat{p}(x, y) \\ sumVariance \leftarrow sumVariance + onQueryPath * \hat{\sigma}(x, y)^2 \\ pathLength \leftarrow pathLength + onQueryPath \end{cases}$$

$avg \leftarrow sum/pathLength$

$avgConfidence \leftarrow \text{SQRT}(sumVariance)/pathLength$

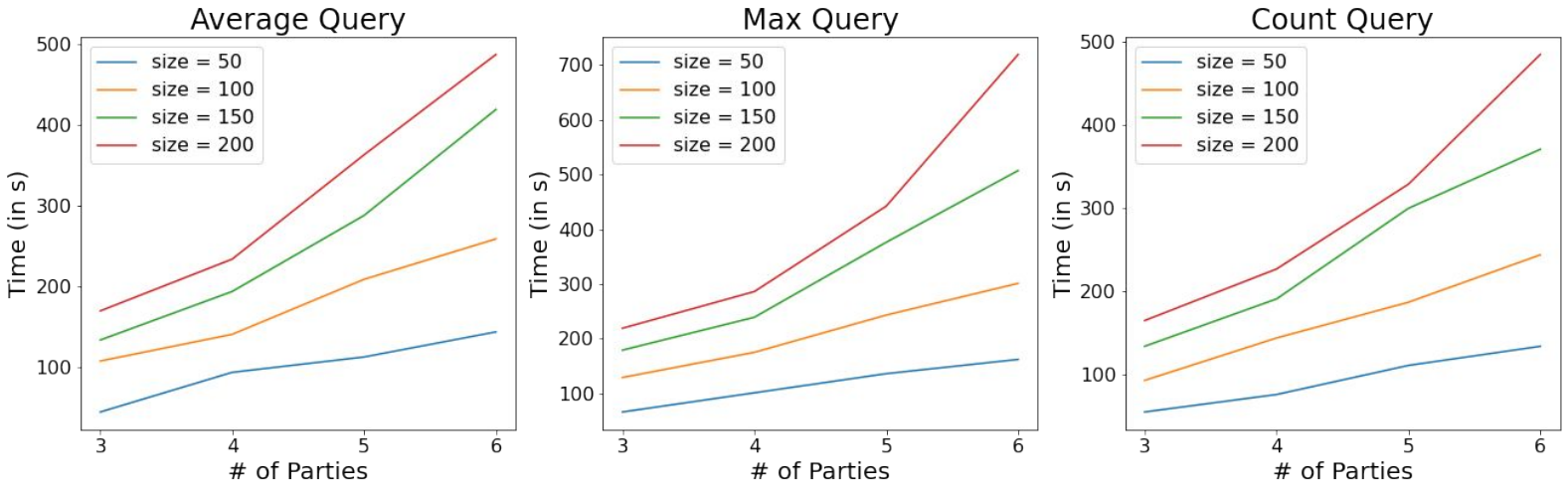
Algorithm for answering average query

Experiments

MPyC

- Implements BGW for addition and multiplication
- Also handles boolean operations using secret sharing
- Not suited for ML and gradient based learning
- Slow as fully implemented in Python
- Doesn’t use Beaver triples or other optimizations

Results

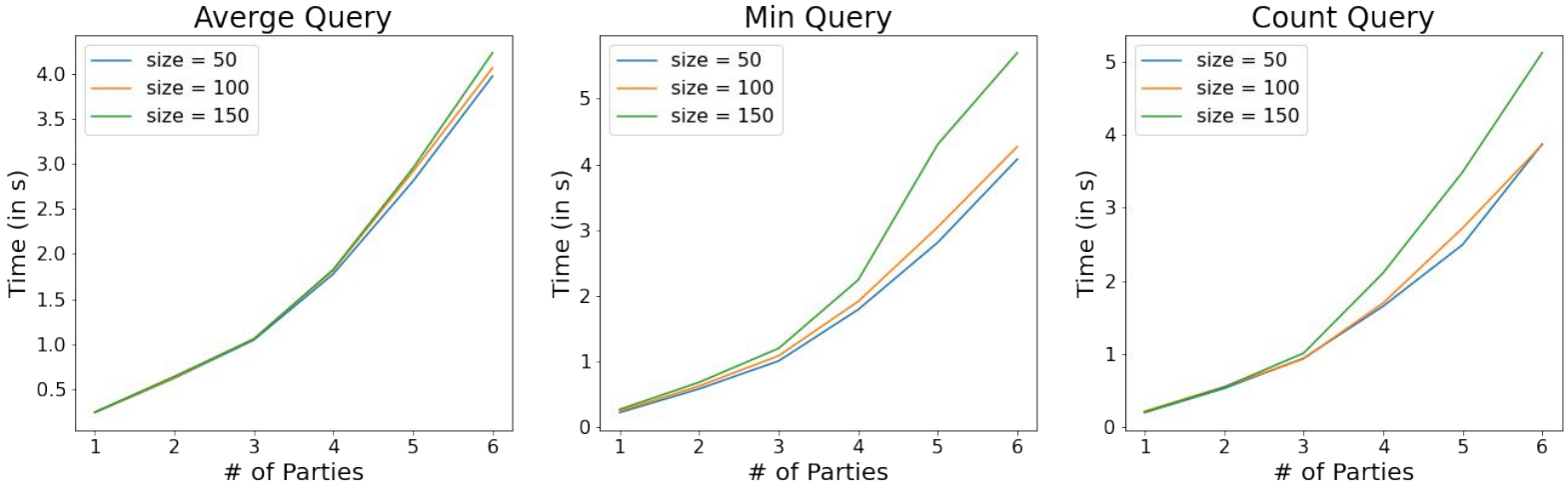


Timing Results

CrypTen

- Uses secret sharing protocols
- Private Addition: Parties sum their shares independently
- Private Multiplication: Using Beaver triples, requires trusted third party
- Linear Functions: Implemented using private additions and multiplications
- Non-Linear Functions: Implemented using standard approximations
- C++ implementation under the hood
- Cryten performance w.r.t other implementations is promising

Results



Timing Results

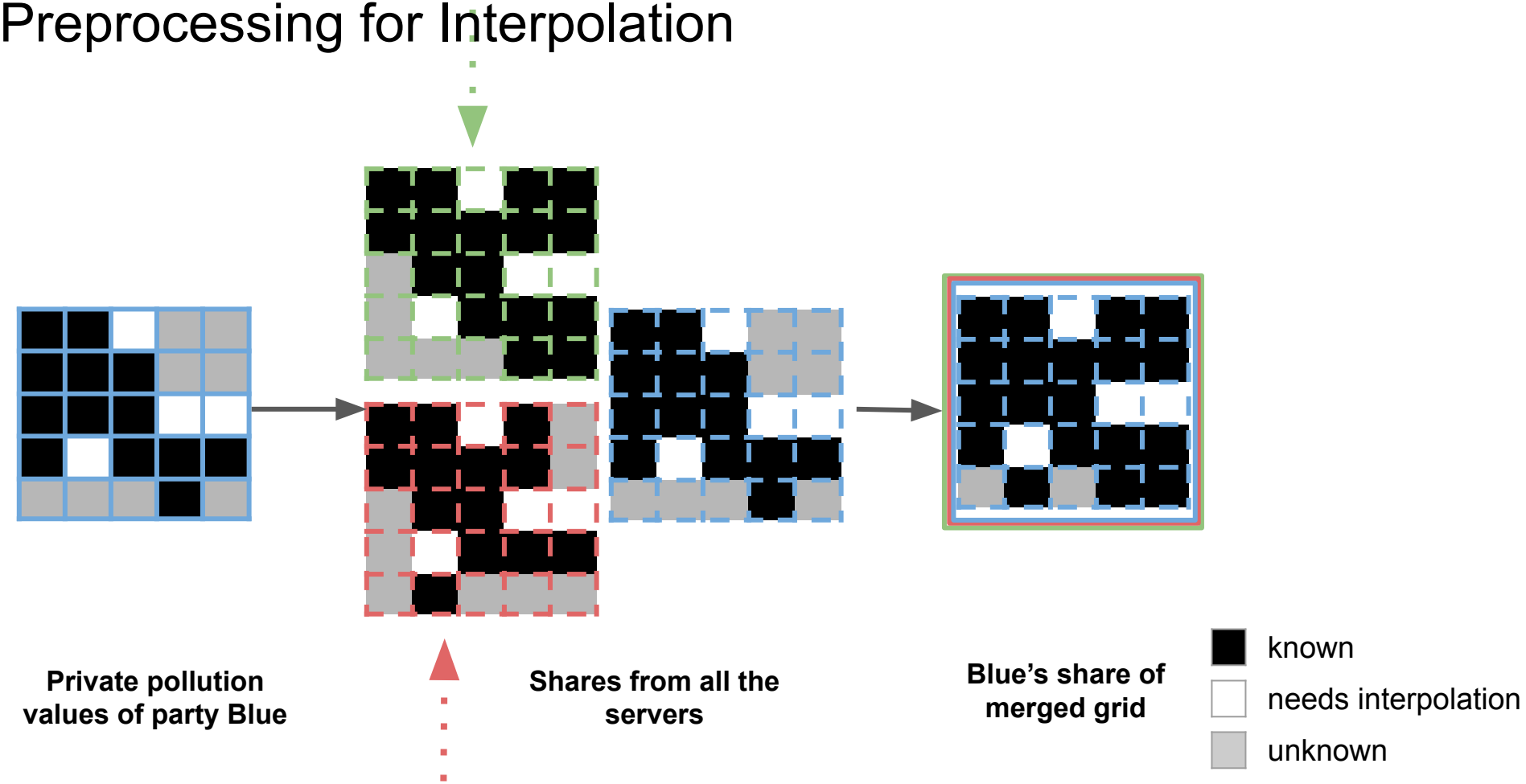
Relative error with
vs without SMPC

Average: $10^{-3} - 10^{-4}$ %

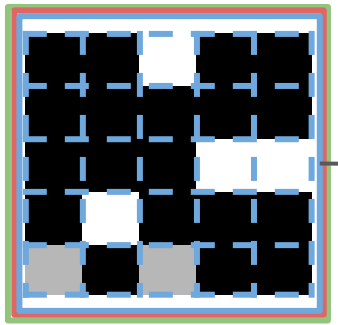
Min/Max/Count/Range: $10^{-5} - 10^{-6}$ %

GCN for Interpolation

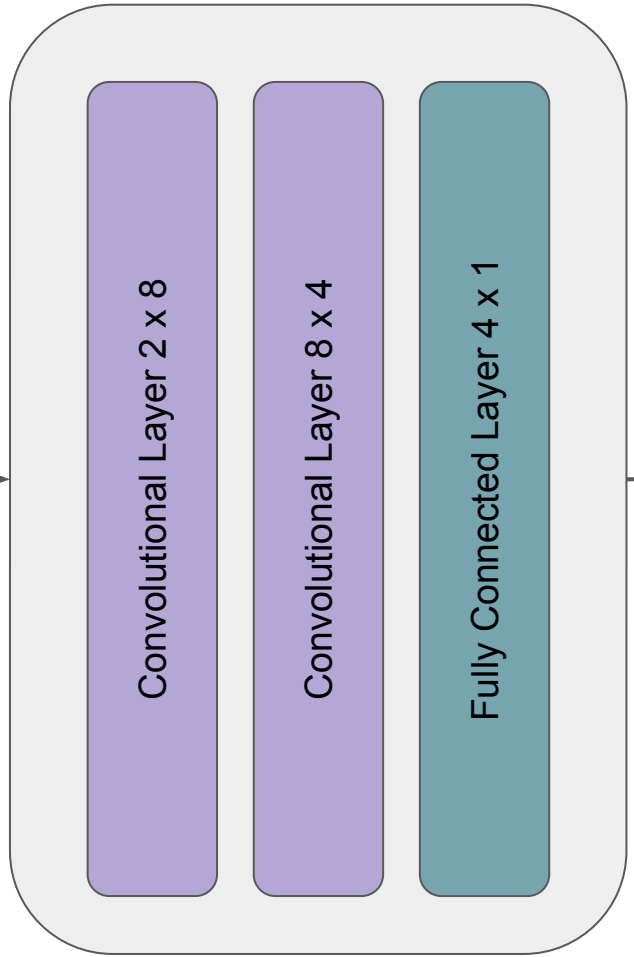
Preprocessing for Interpolation



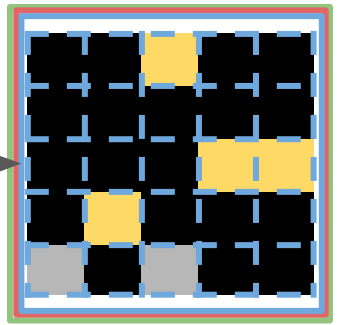
GCN



Blue's share of merged grid



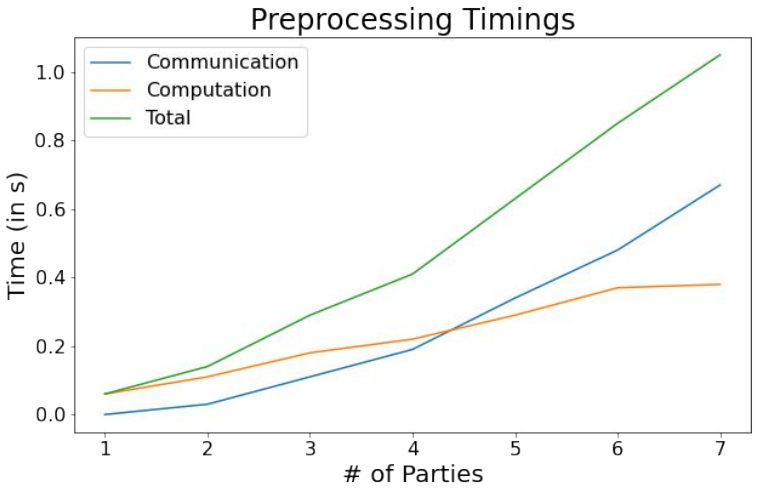
GCN Architecture

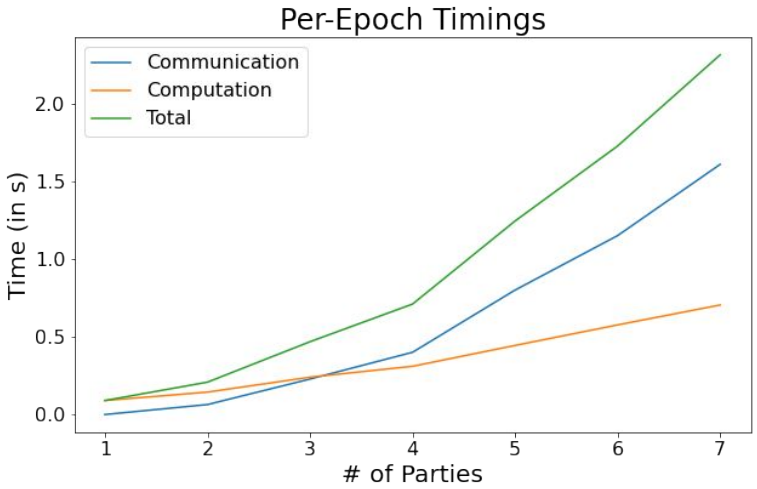


Blue's share of interpolated grid

known
predictions
unknown

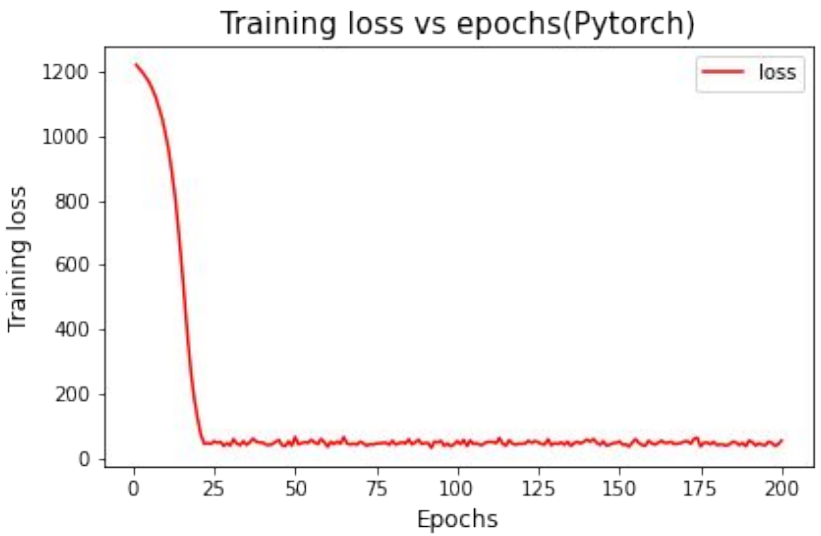
Timing Results

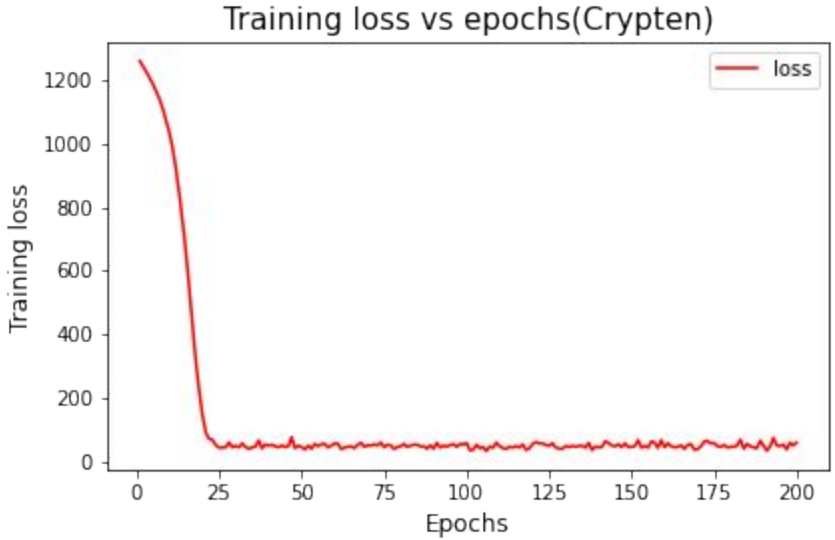




Computation and Communication time increases with # of Parties

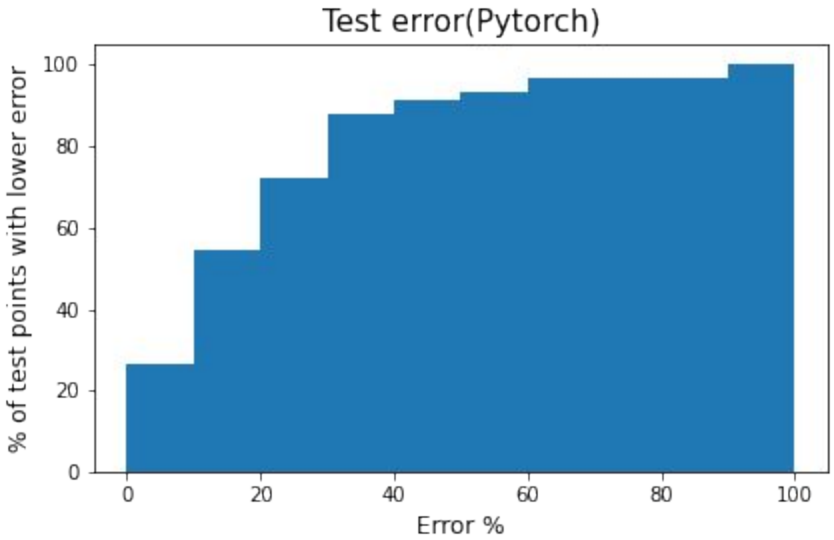
Training Loss

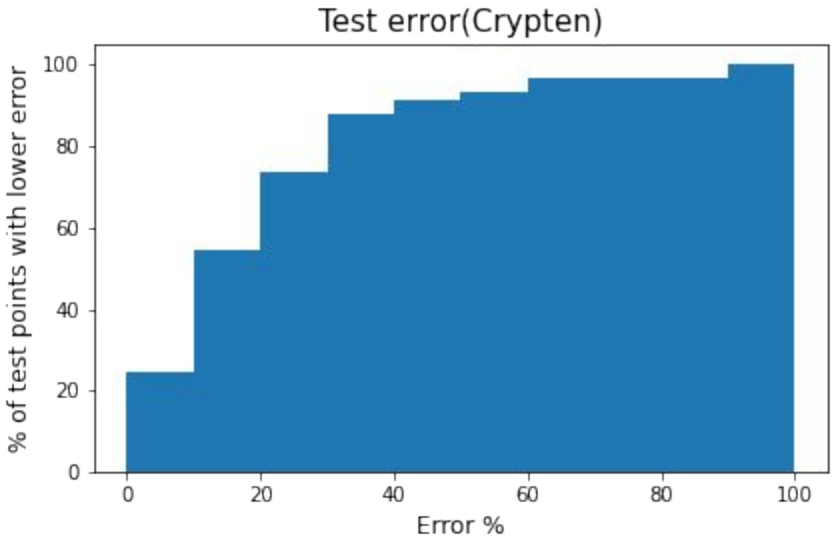




Similar graphs for both CrypTen and standard PyTorch indicates that the MPC implementation is correct

Test Error





75% points have less than 30% error

Issues

- Small training data
- GCN doesn’t return a confidence score
- Test points need to be neighbours of known points
- Crypten doesn’t support Adam optimizer which gave better result with Pytorch

GP for Interpolation

Libraries and Future Scope

- We explored different Variational GP libraries : gpytorch, gptorch and pyro
- GP provides confidence score with its predictions
- GP can make predictions on continuous input locations unlike GCN

Libraries and Future Scope

The key takeaways while trying to port different GP libraries to CrypTen were

- Libraries should be based on PyTorch to port to CrypTen
- Cholesky factorization was a roadblock in multiple libraries
- Large libraries like gpytorch have very large number of lines of code with specific classes defined on top of PyTorch which are not easy to port
- Other smaller libraries like gptorch and pyro did not give very accurate and reliable results even in a non-MPC setting

Thank You!

References

CrypTen

MPyC

Reference for merging values from different sources

A Systematic Comparison of Encrypted Machine Learning Solutions for Image Classification