Empowering SMPC: Bridging the Gap Between Scalability, Memory Efficiency and Privacy in Neural Network Inference

Ramya Burra[1] Anshoo Tandon[2] Srishti Mittal[2]

[1]Center for Brain Research, IISc

[2]IUDX, SID, IISc

Table of Contents

Table of Contents

SMPC

Aim

To compute $f(x, y)$ where Party 0 has no knowledge of y and Party 1 has no knowledge of x.

$\boxed{\text{Party 0}}$ $\boxed{\text{Party 1}}$

Private input: x Private input: y

INDIA URBAN DATA EXCHANGE

SMPC

Aim

To compute $f(x, y)$ where Party 0 has no knowledge of y and Party 1 has no knowledge of x.

Secret Share creation $x := x_0 + x_1$ $y := y_0 + y_1$

Party 0	Party 1
Private input: x	Private input: y

SMPC

Aim

To compute $f(x, y)$ where Party 0 has no knowledge of y and Party 1 has no knowledge of x.

Secret Share creation $x := x_0 + x_1$ $\quad$ $y := y_0 + y_1$

$$\text{Party 0} \underset{y_0}{\overset{x_1}{\rightleftarrows}} \text{Party 1}$$

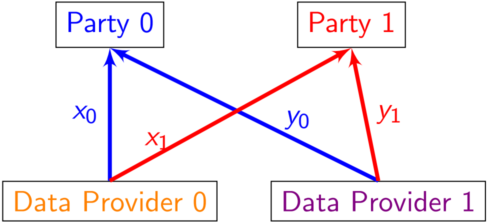
Private input: x $\quad$ Private input: y

Received: y_0 $\quad$ Received: x_1

Local computations $\quad$ Local computations

Data Provider Model

- Separation of data providers from compute servers
- Helps to scale the same algorithm to multiple(>2) data providers
- None of the compute servers ever know the private inputs



Private input: x | Private input: y

Secret Share creation $x := x_0 + x_1$ $y := y_0 + y_1$

ABY 2.0 Secret sharing mechanism

Data Provider 0	Data Provider 1
Owner of private data x	Owner of private data x
Creates two random numbers $[\delta_x]_0, [\delta_x]_1$	Creates two random numbers $[\delta_y]_0, [\delta_y]_1$
Calculates Δ_x as $\Delta_x := x + [\delta_x]_0 + [\delta_x]_1$	Calculates Δ_y as $\Delta_y := y + [\delta_y]_0 + [\delta_y]_1$
Broadcasts $(\Delta_x, [\delta_x]_0)$ to Party 0 Broadcasts $(\Delta_x, [\delta_x]_1)$ to Party 1	Broadcasts $(\Delta_y, [\delta_y]_0)$ to Party 0 Broadcasts $(\Delta_y, [\delta_y]_1)$ to Party 1

Available data at Party 0 : $(\Delta_y, [\delta_y]_0), (\Delta_x, [\delta_x]_0)$
Available data at Party 1 : $(\Delta_x, [\delta_x]_1), (\Delta_y, [\delta_y]_1)$

IUDX
INDIA URBAN DATA EXCHANGE

Table of Contents

MOTION2NX

- A Framework for Generic Hybrid Two-Party Computation
- It can implement GMW, ABY 2.0, Yao protocols
- Provided handles to implement secure operations
- Assumes data providers are a part of compute servers
- No intermediate values are reconstructed
- Assumes either of the compute servers as output owners
- Output is reconstructed in clear and shared with the output owner

Short comings of MOTION2NX

- ✗ No provision to implement data provider model
- ✗ No provision implement Secure Argmax operation in the optimized tensor version
- ✗ Output is always reconstructed
- ✗ Memory issues : A simple 2 layer neural network inference requires about 3.2GB of RAM

Table of Contents

Secure Inferencing task - 2 layers

We implemented the 2 layer neural network with the following parameters[1]

- Layer 1 : Weights (256×784), bias (256×1)
- Layer 2 : Weights (10×256), bias (10×1)

Summary

Memory requirement: 3.2 GB
Time required for execution on 16GB machines: 86.88 sec

[1] Dimensions of W^T have been mentioned

Optimizing memory requirement

- Realized that layer-1 matrix multiplication is the bottle neck
- Break that one matrix multiplication into multiple matrix multiplications called splits
- No compromise on privacy
- Observed linear scale down of memory requirement with number of splits

Summary

Splits	Memory requirement	Time of execution
2 splits	1.6419 GB	75.68 seconds
4 splits	0.888 GB	89.91 seconds
8 splits	0.4527 GB	100.30 seconds
64 splits	0.0988 GB	121.53 seconds

A sample secure computation on MOTION2NX

SERVER 0	SERVER 1
16280432298212562843	14052653260407326067
7233090451320594075	9299030725943297803
8394278896146359934	11079105942734303167
12605804215631305748	18497853776721253696
14468922559315646082	9902646149782746607
11932412653977310625	11656944753379615164
15966898911639313689	4299638653364405072
5644686034858105469	10747697829954897842
16214254855426078338	15152134660921080468
9299032496203297992	3218511257450256594

$$\text{9299032496203297992} \; \underset{\text{9299032496203297992}}{\overset{\text{3218511257450256594}}{\leftrightarrows}} \; \text{3218511257450256594}$$

Server 0	Server 1
Reconstruction Ans = 3	Reconstruction Ans = 3

INDIA URBAN DATA EXCHANGE

Modularize

SERVER 0		SERVER 1
16280432298212562843		14052653260407326067
7233090451320594075		9299030725943297803
8394278896146359934		11079105942734303167
12605804215631305748	14468922559315646082	18497853776721253696
14468922559315646082	9902646149782746607	9902646149782746607
Reconstruction Ans = 2.5		Reconstruction Ans = 2.5
11932412653977310625		11656944753379615164
15966898911639313689		4299638653364405072
5644686034858105469		10747697829954897842
16214254855426078338	3218511257450256594	15152134660921080468
9299032496203297992	9299032496203297992	3218511257450256594
Reconstruction Ans = 3		Reconstruction Ans = 3

Modularize

SERVER 0		SERVER 1
16280432298212562843		14052653260407326067
7233090451320594075		9299030725943297803
8394278896146359934		11079105942734303167
12605804215631305748		18497853776721253696
14468922559315646082	$\xleftarrow{14468922559315646082}$ $\xrightarrow[9902646149782746607]{}$	9902646149782746607
Reconstruction Ans $= 2.5$	Intermediate value Reconstruction?	Reconstruction Ans $= 2.5$
11932412653977310625		11656944753379615164
15966898911639313689		4299638653364405072
5644686034858105469		10747697829954897842
16214254855426078338		15152134660921080468
9299032496203297992	$\xleftarrow{3218511257450256594}$ $\xrightarrow[9299032496203297992]{}$	3218511257450256594
Reconstruction Ans $= 3$		Reconstruction Ans $= 3$

INDIA URBAN DATA EXCHANGE

How to tackle

SERVER 0	SERVER 1
16280432298212562843	14052653260407326067
7233090451320594075	9299030725943297803
8394278896146359934	11079105942734303167
12605804215631305748	18497853776721253696
14468922559315646082	9902646149782746607

~~14468922559315646082~~

~~9902646149782746607~~

Send final shares

Final shares sent to output owner

Send final shares

11932412653977310625	11656944753379615164
15966898911639313689	4299638653364405072
5644686034858105469	10747697829954897842
16214254855426078338	15152134660921080468
9299032496203297992	3218511257450256594

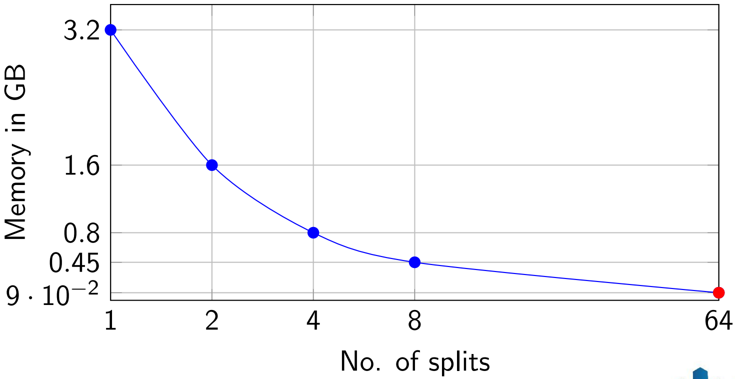
~~3218511257450256594~~

~~9299032496203297992~~

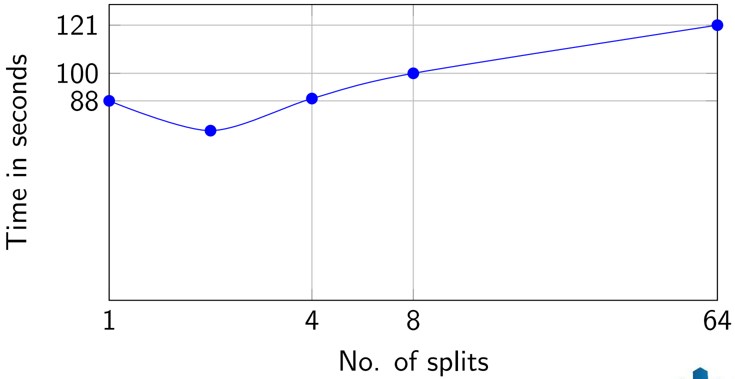
Send final shares

Send final shares

Decrease in Memory consumption



Increase in time



Optimizing time requirement - The OT overhead

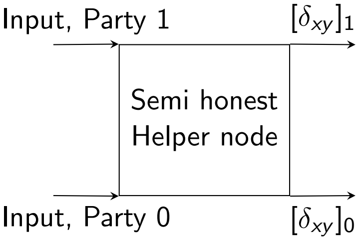
OTs take the maximum time required for the SMPC operation. Eg: 64% of the execution time for a comparison operation.

```
===============================================================================
millionaires_problem
===============================================================================
MOTION version: 8c2b56f-dirty @ temp_test
invocation: ./bin/millionaires_problem --my-id 1 --party 0,::1,7000 --party 1
by srishti04@srishti04, PID 22296
===============================================================================
print_human_readable() run time
Run time statistics over 1 iterations
-------------------------------------------------------------------------------
                          mean        median        stddev
-------------------------------------------------------------------------------
MT Presetup              0.006 ms      0.000 ms      0.000 ms
MT Setup                 0.231 ms      0.000 ms      0.000 ms
SP Presetup              0.000 ms      0.000 ms      0.000 ms
SP Setup                 0.000 ms      0.000 ms      0.000 ms
SB Presetup              0.005 ms      0.000 ms      0.000 ms
SB Setup                 0.003 ms      0.000 ms      0.000 ms
Base OTs               759.617 ms      0.000 ms      0.000 ms
OT Extension Setup       4.645 ms      0.000 ms      0.000 ms
-------------------------------------------------------------------------------
Preprocessing Total    813.413 ms      0.000 ms      0.000 ms
Gates Setup            322.550 ms      0.000 ms      0.000 ms
Gates Online            52.959 ms      0.000 ms      0.000 ms
-------------------------------------------------------------------------------
Circuit Evaluation    1191.570 ms      0.000 ms      0.000 ms
===============================================================================
Communication with each other party:
Sent: 0.037 MiB in 389 messages
Received: 0.474 MiB in 4741 messages
===============================================================================
```

Figure: Run time statistics of server 1 while running Millionaires problem in MOTION2NX

Solution - The Helper node

The cryptographic overhead of OTs is responsible for high execution time. To get away with that we introduce a semi-honest third party, the HELPER NODE.



Solution - The Helper node

- Helper node cannot reconstruct the actual input
- It has no knowledge of the output or its shares
- It does not need to know the operation the compute servers are aiming to perform
- Does not introduce a single point of failure for privacy breach

Summary

Memory requirement: 0.1GB, Time of execution: 12 seconds.
Time of execution has come down ten times comparing with the split version of similar memory requirement.

Increase in time

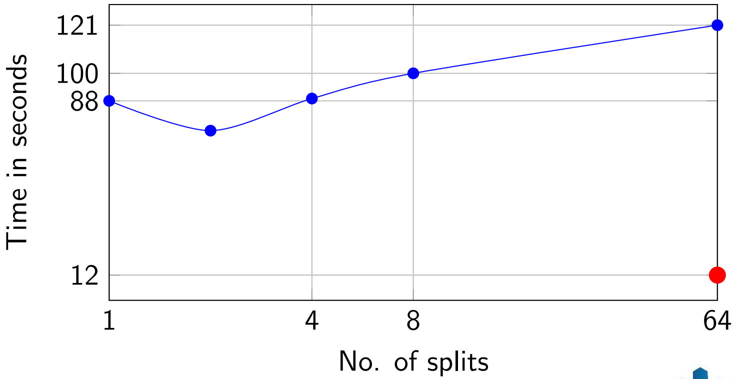


Table of Contents

Realistic setup - Servers on cloud

The following is the setup

Server 0	Azure: b1s 1vcpu, 1GB, 30GB SSD
Server 1	AWS: t2.micro 1vcpu, 1GB, 30GB SSD
Helper node	AWS: t2.nano 1vcpu, 0.5GB, 30GB SSD
Image Provider	Laptop
Model Provider	Laptop

Summary

Splits	Memory requirement	Time of execution
16 splits	0.253GB	200 seconds
64 splits	0.0988GB	280 seconds
Helper node	0.1GB	12 seconds

INDIA URBAN DATA EXCHANGE

Table of Contents

Our Contribution

- Implement any n layer neural net on cloud
- Can implement Data provider and non data provider models
- No output reconstruction at the compute servers
- A framework agnostic solution to bring down memory requirement
- Total code with relevant examples is made available on GitHub for the benefit of the community