Optimizing ML MPC from System & Theoretical Perspective

Yongqin Wang

Ming Hsieh Department of Electrical and Computer Engineering

University of Southern California

Presented at NIST WPEC 2024 on September 26th

Talk overview

- **Proprietary weights & sensitive data**
- MPC can share weights and data securely
- MPC induces significant overheads
	- Added computation
	- Added communication

• This talk describes and calls for **system** & **theoretical** optimizations to MPC ML

Outline

01

Secure MPC Background

- General terms
- Secret sharing
- Multiplication
- Online/offline phase

02

System

• MPC-Pipe: an efficient pipeline for n-party MPC

03

Theory

• CompactTag: minimized tag computation for actively secure MPC

04

Summary of the talk

Secure MPC: General terms

Multi-Party Computing is a *secure protocol* **to address privacy issue in the cloud.**

Secure MPC: General terms

- MPC protocols allows secure computations among *n* parties
- No assumptions about underlying hardware
- Adversaries can corrupt up to n-1 parties

Privacy can still be guaranteed even if a subset of parties is corrupted.

…….

Secure MPC: General terms

- Secure cloud computing protocol
	- *n* servers to compute to perform computations
- Step #1: Distribute shares
- Step #2: MPC servers compute
- Step #3: Retrieve Results

Secure MPC: Secret Sharing

- Additive Secrete Sharing
	- Additions: adding local shares
	- Beaver Triple Multiplications
	- MPC server Communication is required
- Binary Secrete Sharing
	- Bit extractions
	- Bitwise manipulations
- Usually implemented as Fixed point
	- In a numerical field

Additive Share(x, 2) $[x_1] = x - r, [x_2] = r$

Binary Share(x, 2) $\langle x_1 \rangle = x \, XOR \, r, \langle x_2 \rangle = r$

Additive is efficient for adding/multiplying, Binary is efficient for bitwise ops

Algorithm 1 Beaver Triple Assisted MPC Multiplication **Input:** $[x_i]$, $[y_i]$, $[a_i]$, $[b_i]$ and $[c_i]$ s.t. $C = A \cdot B$ Computes $[x_i] - [a_i]$ and $[y_i] - [b_i]$ Broadcast local $[x_i] - [a_i]$ and $[y_i] - [b_i]$ Wait until other $[x_i] - [a_i]$ and $[y_i] - [b_i]$ has been received
Computes $X - A = \sum_{i=1}^{N-1} [x_i] - [a_i]$
Computes $Y - B = \sum_{i=1}^{N-1} [y_i] - [b_i]$ Party # 1 computes $[z_1] = [c_1] + (X - A)[b_1] + (Y - B)[a_1] +$ $(X-A)(Y-B)$ Other parties compute $[z_i] = [c_i] + (X - A)[b_i] + (Y - B)[a_i]$ **Return:** $[z_i]$

A special algorithm to compute multiplications using additive shares.

Initial state

Beaver triple: $c = a * b$ a and b are completely random

Compute local operands

Broadcasting

Compute global Δ and ϵ Note that Δ and ϵ do not leak information about x and y and **appear from Uniform distribution**

Compute the resulting shares.

• Verify the results

$$
\sum_{i=0}^{N-1} [z_i] = \sum_{i=0}^{N-1} \{ [c_i] + (x - a)[b_i] + (y - b)[a_i] \} + (x - a)(y - b)
$$

= c + xb - ab + ya - ba + xy - xb - ya + ab
= c + xb - c + ya - c + xy - xb - ya + c
= c - c + c - c + xb - xb + ya - ya + xy
= xy

$$
z = \sum_{i=0}^{n-1} [z_i] = xy
$$

Hence, the result is correct.

Challenges for ML Workloads

- Induced more computation
	- $[z] = [c] + \Delta[b] + \epsilon[a] + \Delta\epsilon$ instead of just xy
- Induced communications between parties
	- Broadcasting of Δ and ϵ

Address those challenges require optimization from joint systematic & theoretical efforts.

Outline

01

Secure MPC Background

- General terms
- Secret sharing
- Multiplication
- Online/offline phase

02

System

• **MPC-Pipe: an efficient pipeline for n-party MPC**

03

Theory

• CompactTag: minimized tag computation for actively secure MPC

04

Summary of the talk

MPC-Pipe: an efficient pipeline for ML Better resource utilization & throughput

Time

an MPC server workflow

Computation and communication are *blocking* **in MPC Resulting in poor resource utilization -> poor throughput**

an MPC-Pipe server workflow

MPC-Pipe breaks data dependencies & overlaps computation and communication

an MPC-Pipe server workflow

MPC-Pipe breaks data dependencies & overlaps computation and communication

an MPC-Pipe server workflow

MPC-Pipe breaks data dependencies & overlaps computation and communication

MPC-Pipe Pipeline Schemes

- **Inter-linear pipeline**
	- **Optimizations with linear layers**
		- **Conv2d**
		- **Fully connected layers**
- Inner-layer pipeline
- Inter-batch pipeline

Three pipeline schemes for n-party MPC

- Two metadata to transmit
	- $\Delta = x a$
	- $\epsilon = y b$
- What are x and y for linear layers
	- Forward pass: x is the input, y is the **weight**
	- Backward pass: x is output gradients, y is **weight or activation feature**
- Both weight and activations are available right before forward & backward pass.

Inter-linear pipeline hides all communication with computation

• Epsilon can be available before the other input arrives

• Why in the critical path?

Inter-linear pipeline hides all communication with computation

Transmission of delta can also be overlapped with Conv2d(epsilon, a)

Time

Transmission of delta can also be overlapped with Conv2d(epsilon, a)

Time

Transmission of delta can also be overlapped with Conv2d(epsilon, a)

MPC-Pipe Pipeline Schemes

- Inter-linear pipeline
- Inner-layer pipeline
	- Optimizes within non-linear layers
		- ReLU, Maxpooling, Softmax (comparisons)
- Inter-batch pipeline
	- Overlap computation and communication across different batches

Three pipeline schemes for n-party MPC

MPC-Pipe implementation

- CrypTen library from Meta AI
- No hardware modification
- Free of additional overheads

MPC-Pipe Results: Throughput

MPC-Pipe on other frameworks

- We incorporated MPC-Pipe on PIGEON
	- The fastest 3PC/4PC MPC inference framework
- ~50% speedups due to the techniques in MPC-Pipe

Outline

01

Secure MPC Background

- General terms
- Secret sharing
- Multiplication
- Online/offline phase

02

System

• MPC-Pipe: an efficient pipeline for n-party MPC

03

Theory

• **CompactTag: minimized tag computation for actively secure MPC**

04

Summary of the talk

CompactTag: minimized tag computation for actively secure MPC

CompactTag overview

- Some protocols computes a tag for integrity
	- For Matmul, requiring cubic complexity
- CompactTag asymptotically reduces tag computation complexity
	- Using characteristics of matrix multiplicaiton

When The Parties are Malicious

Let Us Look Back: In Malicious Setting

The broadcasted value is no longer correct because one of the party introduce errors.

Check if the result is still correct

$$
\sum_{i=1}^{n} [z] = \sum_{i=1}^{n} [c_i] + (x - a + e_1) b_i] + (y - b + e_2) [a_i] + (x - a + e_1) (y - b + e_2)
$$

=
$$
\{c + (x - a)b + (y - b)a + (x - a)(y - b)\} + e_1(y + e_2) + e_2(x + e_1)
$$

=
$$
xy + e_1(y + e_2) + e_2(x + e_1) + xy
$$

This result is no longer correct.

So What is the Solution?

Information theoretical MACs

- Each operand is attached with an IT MAC
- A global key k is secretly shared to MPC parties
	- Each party will have $\left[k_i\right]$
- For any operand x , there is a tag
	- $T_x = k \cdot x$
	- k , x , and T_x are secretly shared

Initial state

Compute local delta and epsilon Compute local tags

Broadcasting

Compute global Δ and ϵ **No** broadcasting of T_{Δ_1} and T_{ϵ_1} Note: T_{Δ_i} and T_{ϵ_i} are used to check correctness of the **broadcast**

Compute $[z]$ and $[T_z]$ **Note :** [] **is computed entirely with "local tags" and the globally computed** Δ **and** ϵ

Check what is $[T_z]$

$$
\sum_{i=1}^{n} [T_z] = \sum_{i=1}^{n} [T_c] + \Delta \cdot [T_b] + \epsilon \cdot [T_a] + [k] \cdot \Delta \cdot \epsilon
$$

= $T_c + \Delta \cdot T_b + \epsilon \cdot T_a + k \cdot \Delta \cdot \epsilon$
= $k \cdot c + (x - a) \cdot T_b + (y - b) \cdot T_a + k \cdot (x - a) \cdot (y - b)$
= $k \cdot c + (x - a) \cdot (k \cdot b) + (y - b) \cdot (k \cdot a) + k \cdot (x - a) \cdot (y - b)$
= $k \cdot \{c + (x - a) \cdot b + (y - b) \cdot a + (x - a) \cdot (y - b)\}$
= $k \cdot \{x \cdot y\}$
= $k \cdot z$

After computing $\bm{\left[\mathnormal{z}\right]}$, $\bm{\left[T_{z}\right]}$

- We need to verify z, Δ , ϵ
	- $[T_z]$
	- $[T_{\Delta}]$
	- $[T_{\epsilon}]$
- There is a standard way to compute a single-element checksum
	- To pass verification, the checksum needs to be zero

But This is Not Free Lunch

Added Computation Costs of Tagged MPC

- In ML $[a]$, $[b]$ and $[c]$ are matrix
	- Δ and $[a]$ is MxN, size of intermediate value
	- [b] and ϵ is NxO, size of the weight

Tag computation for matrices has cubic complexity $\bm{O}(M \times N \times \bm{O})$. **Takes 10% to 30% of total runtime.**

CompactTag

- CompactTag computes a small tag for matrix multiplication
- Reduce tag computation from cubic to
	- $O(M \times N + M \times O + N \times O)$
	- Asymptotic reduction

CompactTag requires less computation has the same security level.

How $[z]$ and $[T_z]$ are used as matrices

- $[z]$ becomes inputs to next layer
- [z] computes next layer's $\theta = r z$
	- r is an random matrix size of $M \times Q$
- We need to verify correctness of θ using $[T_z]$

Now r , z are $M \times O$ matrices.

Compute and reconstruct matrix θ

Compute and reconstruct matrix θ . We use computed T_{θ} to verify θ .

CompactTag a small tag for matrix products

CompactTag: 3 steps

- 1. Sample random numbers χ_i
- 2. Compact operands
- 3. Compute the small tag

Skip $T_{\rm z}$ for now.

Compute a Compact T_z after broadcasting $r - z$. This is a key requirement for security

1. Sample another public matrix χ , whose dimension is 0×1

Equivalent to linearly combine all columns using χ .

Equivalent to linearly combine all columns using χ .

3. Compute a CompactTag $T_{Z}^{'}$

Significant computation complexity reduction. $O(M \times N + M \times O + N \times O)$ Have the same security level as the state-of-the-art with a modified checksum computation.

CompactTag: 3 steps

- 1. Sample random numbers χ_i
- 2. Compact operands
- 3. Compute the small tag
- 4. The same security level
	- 1. modified checksum compuation

Results with CompactTag

- Significant tag computation reduction
	- 3.44x for ResNet50
	- 18.83x for xFormer
	- 4.16x for VGG16
- Significant performance improvement on LAN/WAN

Outline

01

Secure MPC Background

- General terms
- Secret sharing
- Multiplication
- Online/offline phase

02

System

• MPC-Pipe: an efficient pipeline for n-party MPC

03

Theory

• CompactTag: minimized tag computation for actively secure MPC

04

Summary of the talk

Summary of the talk

- MPC for needs optimization from both system & theory
- System:
	- MPC-Pipe: an efficient pipeline for n-party MPC
	- Faster computation engine: PIGEON
	- Faster communication links: quantum teleportation
- Theory:
	- CompactTag: minimized tag computation for actively secure MPC
	- Modified protocol to accommodate heterogenous networks
		- Modified integrity check
		- Modified 3PC/4PC algorithm
- More contributions needed

Thank you!