Pencil: Private and Extensible Collaborative Machine Learning without the Non-Colluding Assumption

Xuanqi Liu, Zhuotao Liu, Qi Li, Ke Xu, Mingwei Xu

Tsinghua University

NDSS SYMPOSIUM/2024

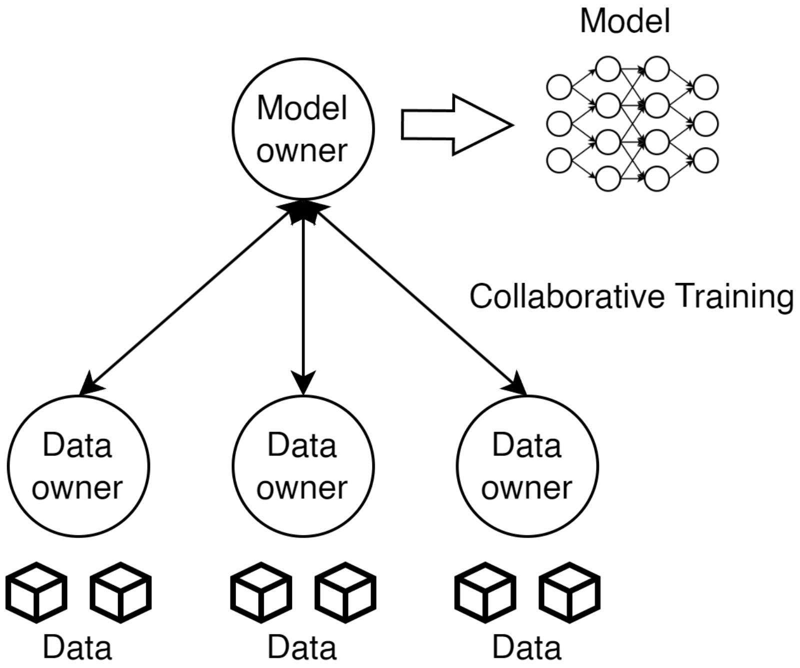
Presented by Internet Society

Overview

- Key problem: Machine learning when model and data are separated

Overview

- Key problem: Machine learning when model and data are separated
  - A Model Owner (MO) wishes to use the data of multiple Data Owners (DOes) to train a model.



Overview

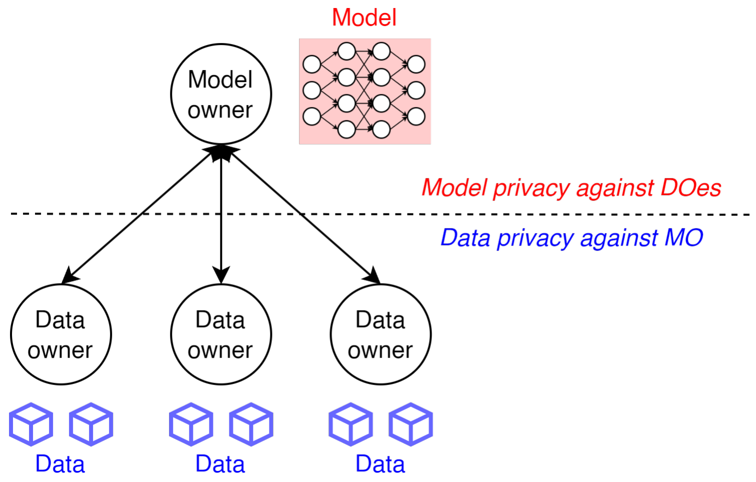
- Requirements
  1) **Data privacy** – Fundamentally, raw data of DOes should not be leaked.

Overview

- Requirements

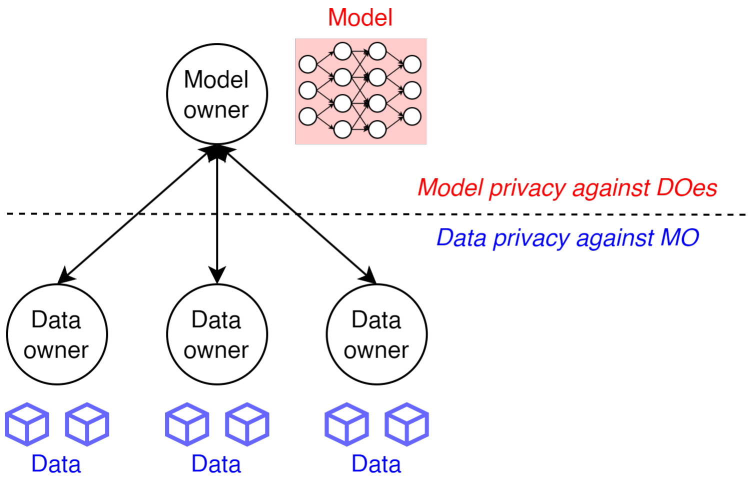
  1) **Data privacy** – Fundamentally, raw data of DOes should not be leaked.

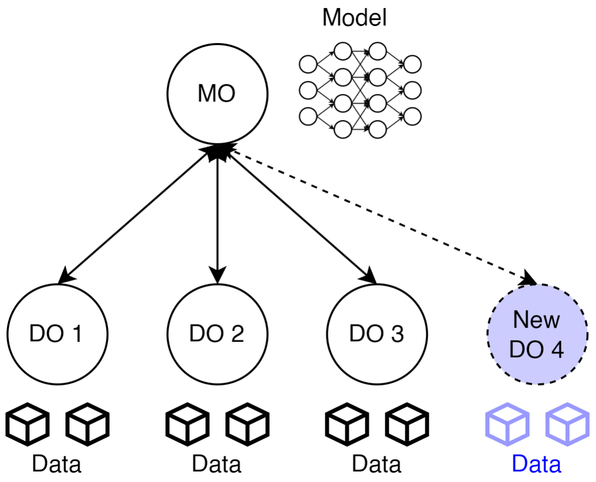
  2) **Model privacy** – Since MO may wish to finetune an existing model, the model parameters should not be leaked to other participants. Furthermore, the model should be able to be deployed independently by MO after training.



Overview

- Requirements
  1) **Data privacy** – Fundamentally, raw data of DOes should not be leaked.
  2) **Model privacy** – Since MO may wish to finetune an existing model, the model parameters should not be leaked to other participants. Furthermore, the model should be able to be deployed independently by MO after training.
  3) **Extensibility** – The framework should scale to more DOes without significant increased cost





Overview

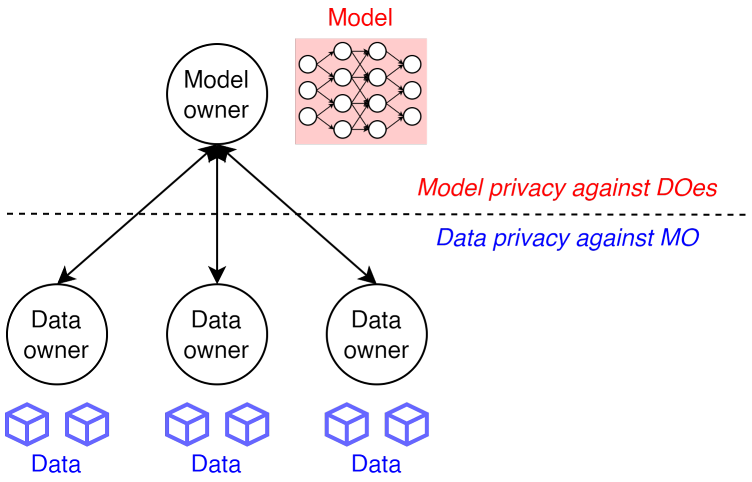
- Requirements

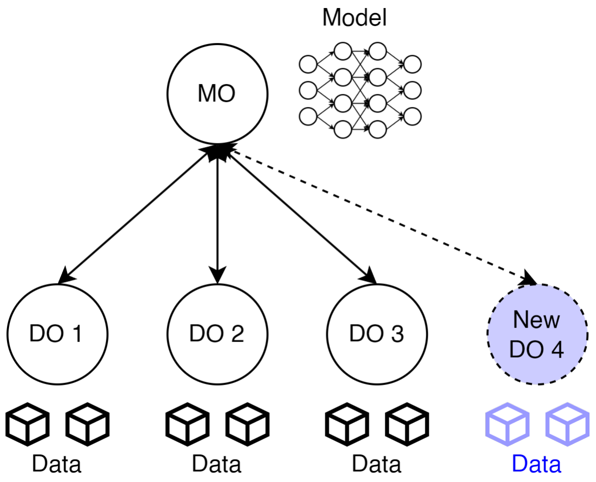
  1) **Data privacy** – Fundamentally, raw data of DOes should not be leaked.

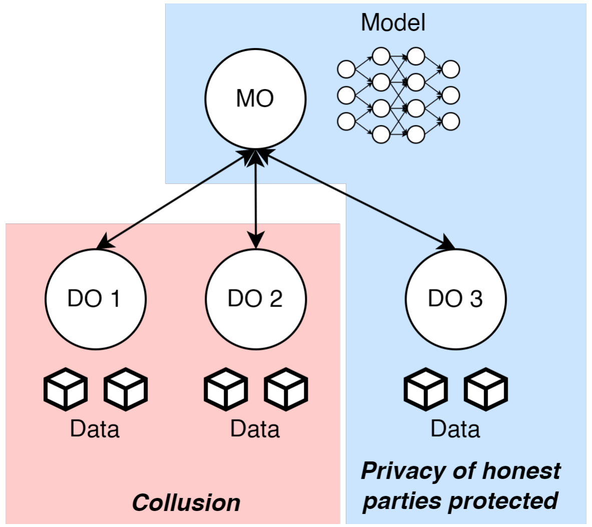
  2) **Model privacy** – Since MO may wish to finetune an existing model, the model parameters should not be leaked to other participants. Furthermore, the model should be able to be deployed independently by MO after training.

  3) **Extensibility** – The framework should scale to more DOes without significant increased cost

  4) **Against collusion** – Colluding parties should not have advantage to break privacy of other honest parties

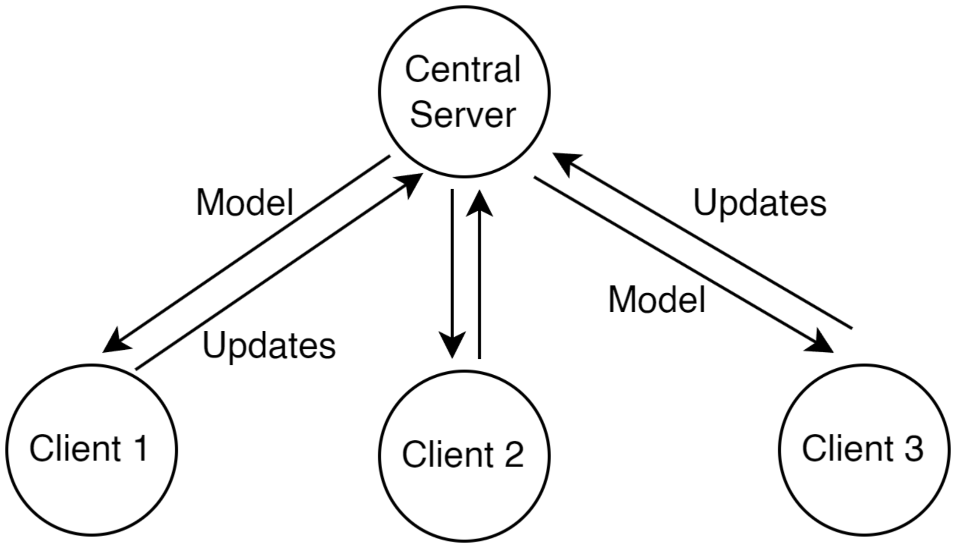






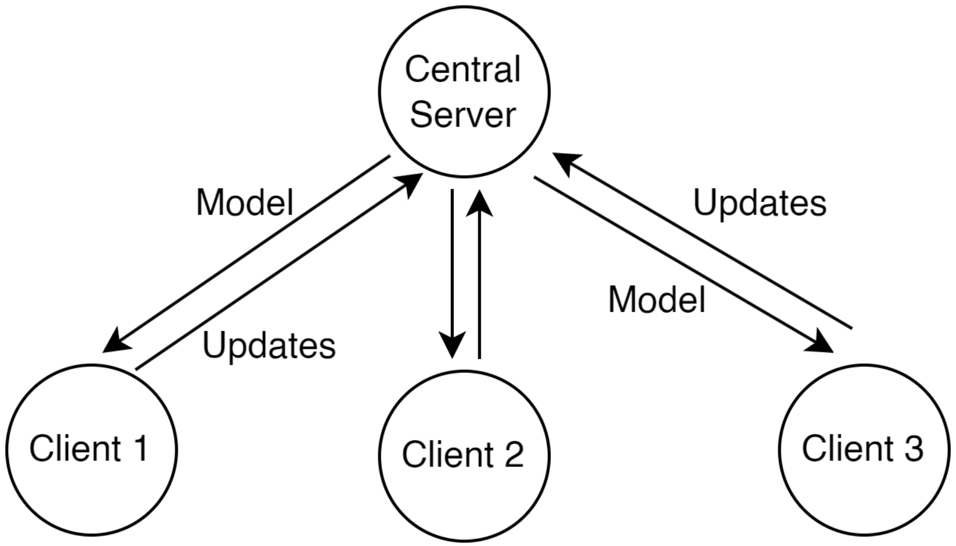
Problems of prior works

- Federated Learning
  - In every iteration, Server (MO) distributes the model to Clients (DOes).
  - Clients train with local data and upload the updates for aggregation.



Problems of prior works

- Federated Learning
  - In every iteration, Server (MO) distributes the model to Clients (DOes).
  - Clients train with local data and upload the updates for aggregation.

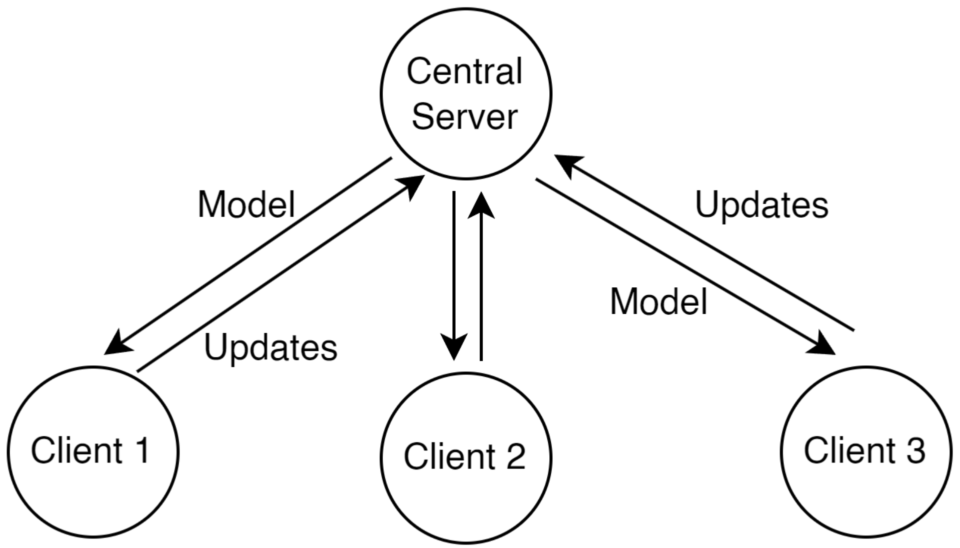


Pros

- Raw data never leave the client;
- Extensible (arbitrary joining and leaving)

Problems of prior works

- Federated Learning
  - In every iteration, Server (MO) distributes the model to Clients (DOes).
  - Clients train with local data and upload the updates for aggregation.



Pros

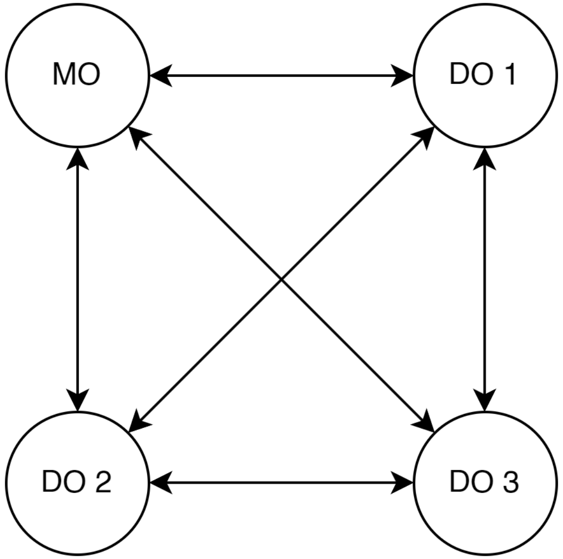
- Raw data never leave the client;
- Extensible (arbitrary joining and leaving)

Cons

- The model privacy isn't protected at all!

Problems of prior works

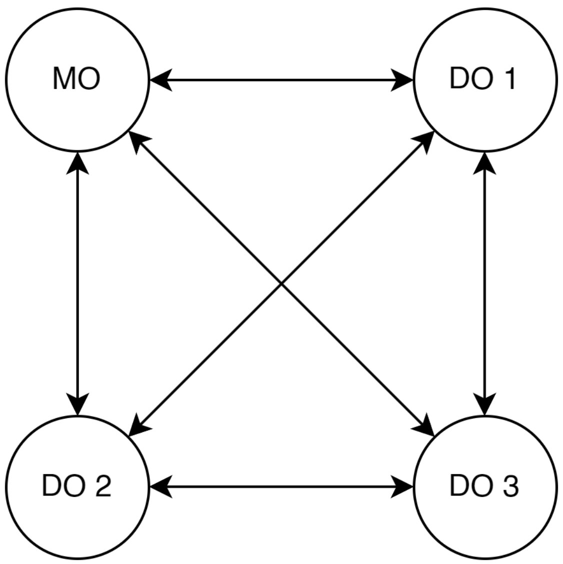
- Secure Multiparty Computation (MPC)
  - MO and DOes participate in n-party computation as servers.



Participants' roles are symmetric in MPC.
All model and data are secret-shared.

Problems of prior works

- Secure Multiparty Computation (MPC)
  - MO and DOes participate in n-party computation as servers.



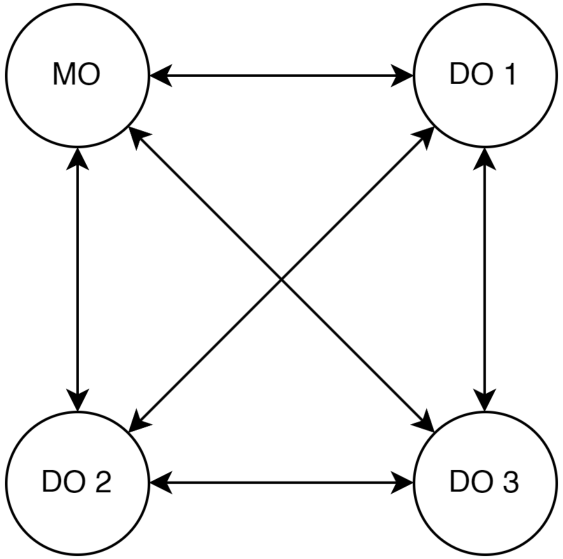
Participants' roles are symmetric in MPC.
All model and data are secret-shared.

Pros

- Model (and data) privacy guaranteed.

Problems of prior works

- Secure Multiparty Computation (MPC)
  - MO and DOes participate in n-party computation as servers.



Participants' roles are symmetric in MPC.
All model and data are secret-shared.

Pros

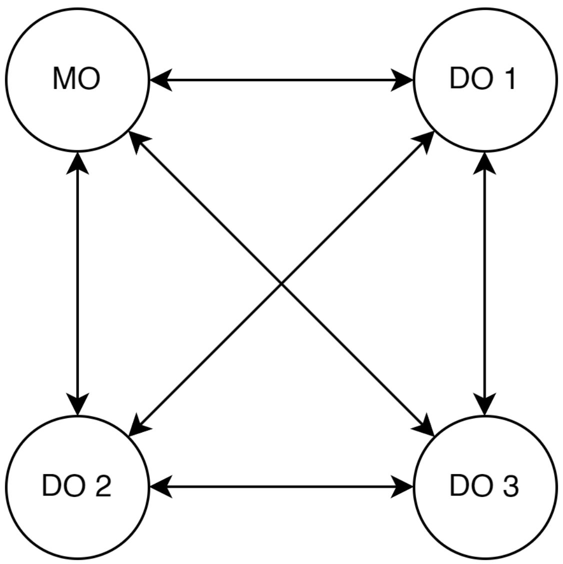
- Model (and data) privacy guaranteed.

Cons

- Not extensible: introducing new DOes requires new protocol design.

Problems of prior works

- Secure Multiparty Computation (MPC)
  - MO and DOes participate in n-party computation as servers.



Participants' roles are symmetric in MPC.
All model and data are secret-shared.

Pros

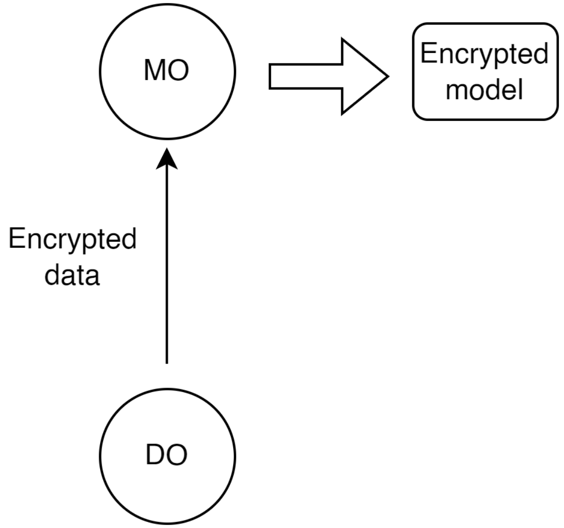
- Model (and data) privacy guaranteed.

Cons

- Not extensible: introducing new DOes requires new protocol design.
- Not secure against collusion
- Huge communication overhead.

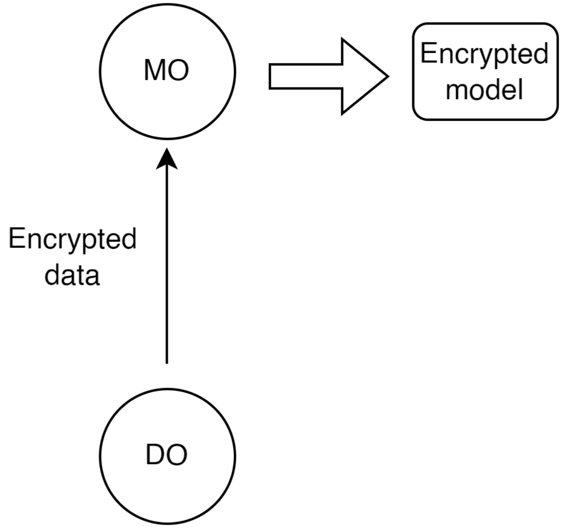
Problems of prior works

- Pure Homomorphic Encryption (HE) Approaches
  - DO uploads encrypted data for training; MO obtains an encrypted model.



Problems of prior works

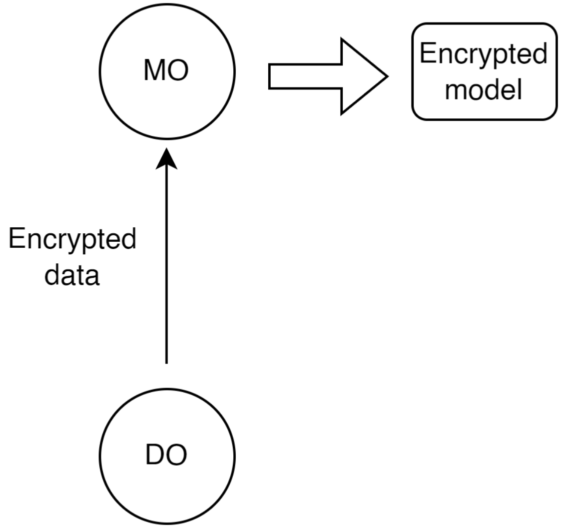
- Pure Homomorphic Encryption (HE) Approaches
  - DO uploads encrypted data for training; MO obtains an encrypted model.



The encrypted model could only be used by the DO assisting training.

Problems of prior works

- Pure Homomorphic Encryption (HE) Approaches
  - DO uploads encrypted data for training; MO obtains an encrypted model.

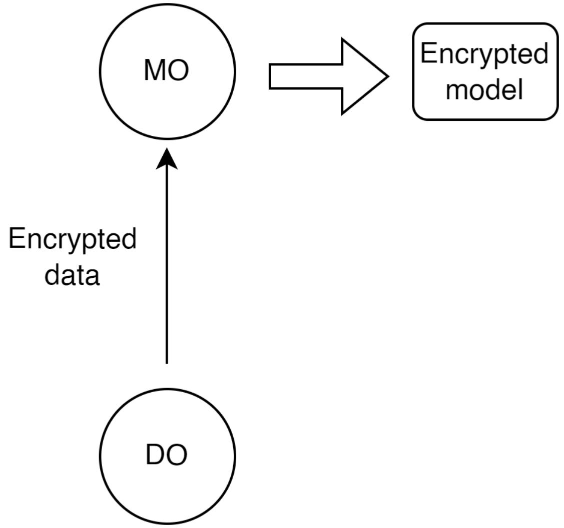


The encrypted model could only be used by the DO assisting training.

Pros
- Low communication overhead.

Problems of prior works

- Pure Homomorphic Encryption (HE) Approaches
  - DO uploads encrypted data for training; MO obtains an encrypted model.



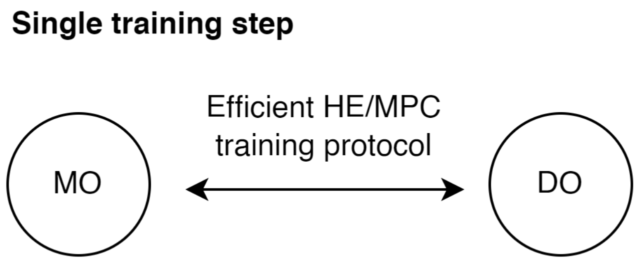
The encrypted model could only be used by the DO assisting training.

Pros
- Low communication overhead.

Cons
- Not extensible: only one DO!
- Heavy computation

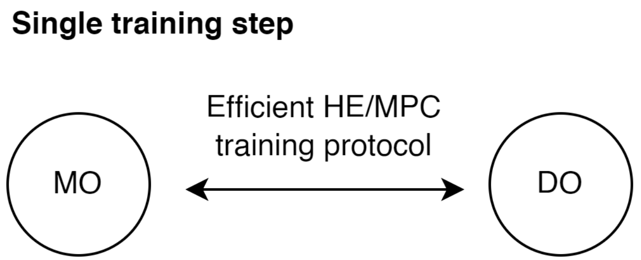
Our solution

- 2-party training: HE + MPC
  - Data and model privacy guaranteed.



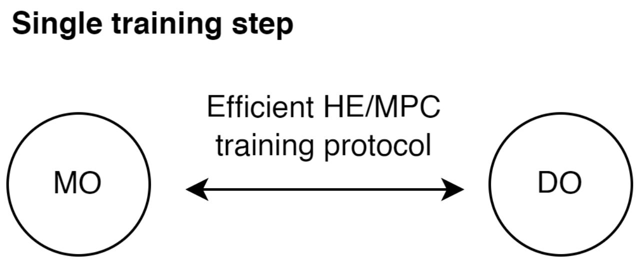
Our solution

- 2-party training: HE + MPC
  - Data and model privacy guaranteed.
  - Model updates are given only to MO.



Our solution

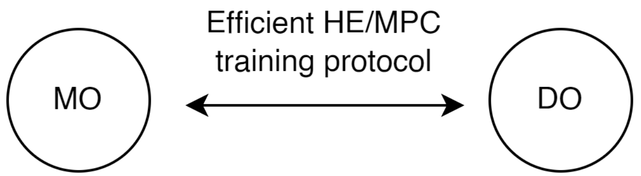
- 2-party training: HE + MPC
  - Data and model privacy guaranteed.
  - Model updates are given only to MO.
    - i.e. the model is not shared nor encrypted w.r.t specific DO(es)



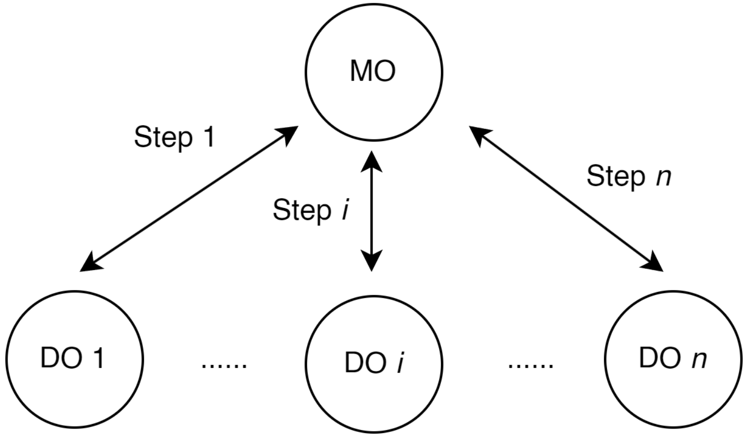
Our solution

- 2-party training: HE + MPC
  - Data and model privacy guaranteed.
  - Model updates are given only to MO.
    - i.e. the model is not shared nor encrypted w.r.t specific DO(es)
- Extensibility and collusion defence
  - MO trains with a different DO in each step.

Single training step



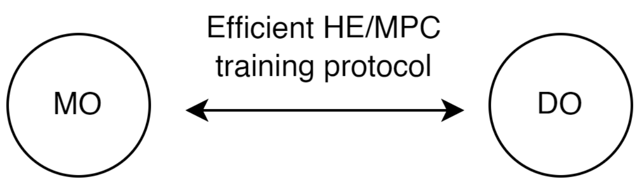
Multiple steps across DOes



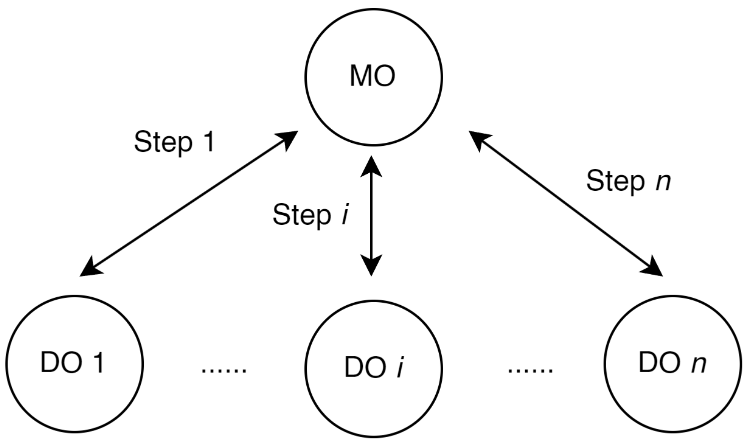
Our solution

- 2-party training: HE + MPC
  - Data and model privacy guaranteed.
  - Model updates are given only to MO.
    - i.e. the model is not shared nor encrypted w.r.t specific DO(es)
- Extensibility and collusion defence
  - MO trains with a different DO in each step.
  - Since no privacy leaks in 2-party, collusion could not break the privacy of any party.

Single training step



Multiple steps across DOes



Pencil training overview

- Secret shares throughout FP and BP

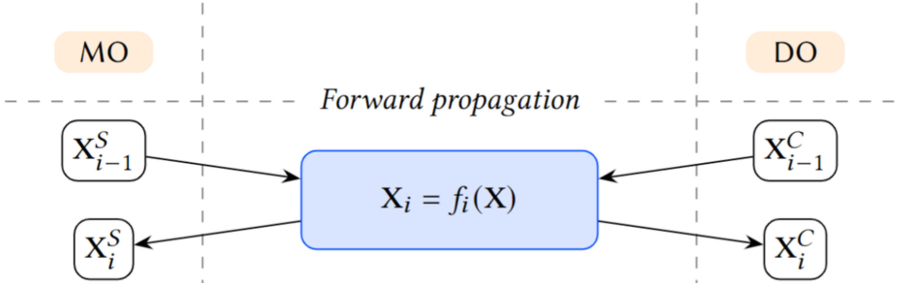
Pencil training overview

- Secret shares throughout FP and BP
  - l-layer sequential model
  - $X_i = f_i(X_{i-1})$

Pencil training overview

- Secret shares throughout FP and BP
  - l-layer sequential model
  - $X_i = f_i(X_{i-1})$
  - In FP, the two parties keep $\langle X_i \rangle$ in secret shares

$$\langle \mathbf{X}_i \rangle_0 + \langle \mathbf{X}_i \rangle_1 = f_i(\langle \mathbf{X}_{i-1} \rangle_0 + \langle \mathbf{X}_{i-1} \rangle_1).$$



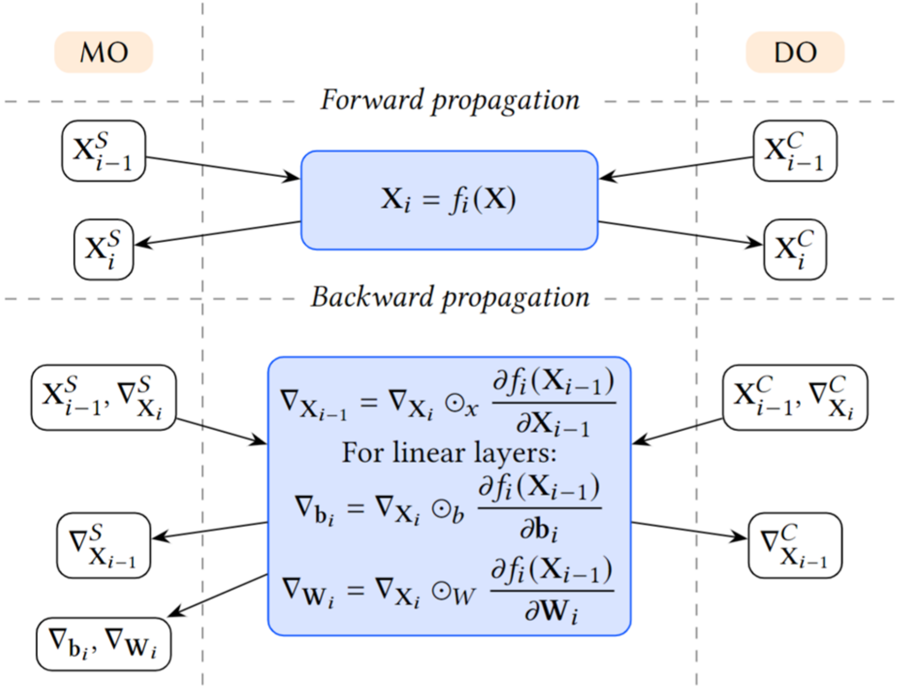
Pencil training overview

- Secret shares throughout FP and BP
  - l-layer sequential model
  - $X_i = f_i(X_{i-1})$
  - In FP, the two parties keep $\langle X_i \rangle$ in secret shares

$$\langle \mathbf{X}_i \rangle_0 + \langle \mathbf{X}_i \rangle_1 = f_i(\langle \mathbf{X}_{i-1} \rangle_0 + \langle \mathbf{X}_{i-1} \rangle_1).$$

  - In BP, similarly $\langle \nabla_{X_i} \rangle$ is shared;

$$\langle \nabla_{\mathbf{X}_{i-1}} \rangle_0 + \langle \nabla_{\mathbf{X}_{i-1}} \rangle_1 = (\langle \nabla_{\mathbf{X}_i} \rangle_0 + \langle \nabla_{\mathbf{X}_i} \rangle_1) \odot_x \frac{\partial f_i(\mathbf{X}_{i-1})}{\partial \mathbf{X}_{i-1}}$$



Pencil training overview

- Secret shares throughout FP and BP
  - l-layer sequential model
  - $X_i = f_i(X_{i-1})$
  - In FP, the two parties keep $\langle X_i \rangle$ in secret shares

$$\langle \mathbf{X}_i \rangle_0 + \langle \mathbf{X}_i \rangle_1 = f_i(\langle \mathbf{X}_{i-1} \rangle_0 + \langle \mathbf{X}_{i-1} \rangle_1).$$

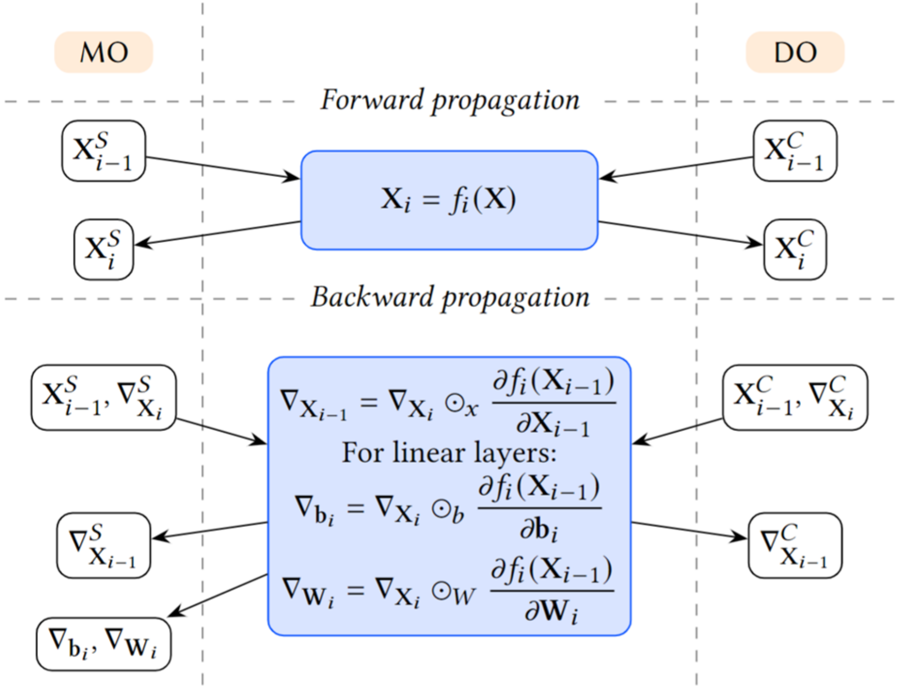
  - In BP, similarly $\langle \nabla_{X_i} \rangle$ is shared;

$$\langle \nabla_{\mathbf{X}_{i-1}} \rangle_0 + \langle \nabla_{\mathbf{X}_{i-1}} \rangle_1 = (\langle \nabla_{\mathbf{X}_i} \rangle_0 + \langle \nabla_{\mathbf{X}_i} \rangle_1) \odot_x \frac{\partial f_i(\mathbf{X}_{i-1})}{\partial \mathbf{X}_{i-1}}$$

  - … but the weight gradients are given to the MO.

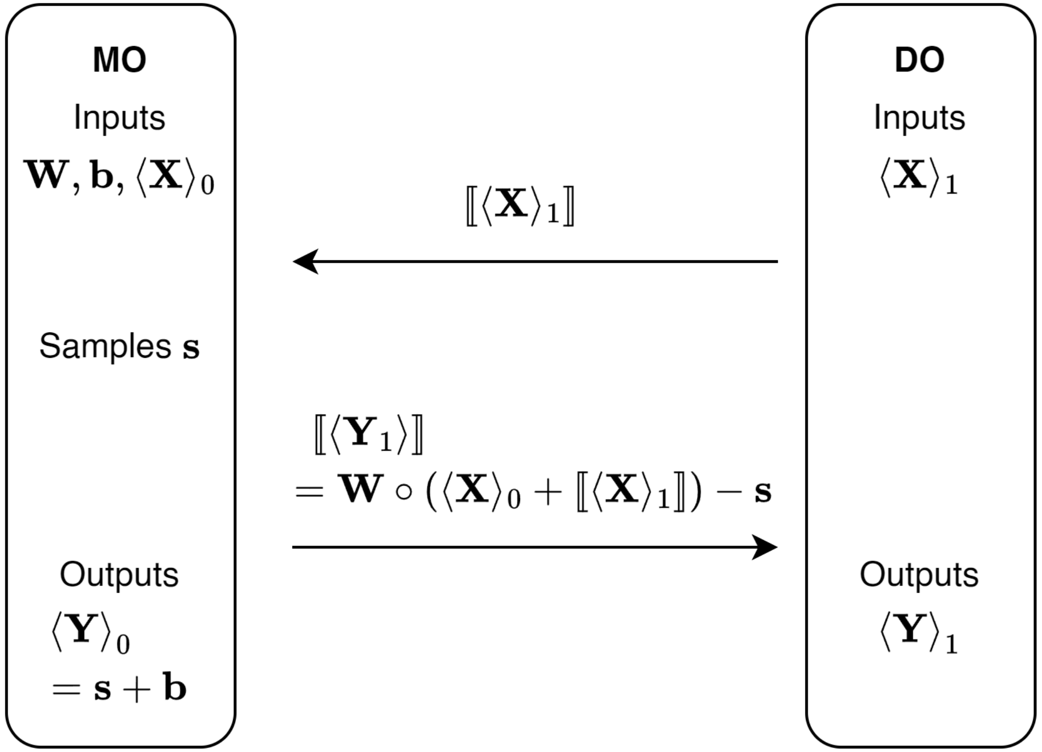
$$\nabla_{\mathbf{b}_i} = \nabla_{\mathbf{X}_i} \odot_b \frac{\partial f_i(\mathbf{X}_{i-1}; \mathbf{W}_i, \mathbf{b}_i)}{\partial \mathbf{b}_i}$$

$$\nabla_{\mathbf{W}_i} = \nabla_{\mathbf{X}_i} \odot_W \frac{\partial f_i(\mathbf{X}_{i-1}; \mathbf{W}_i, \mathbf{b}_i)}{\partial \mathbf{W}_i}$$



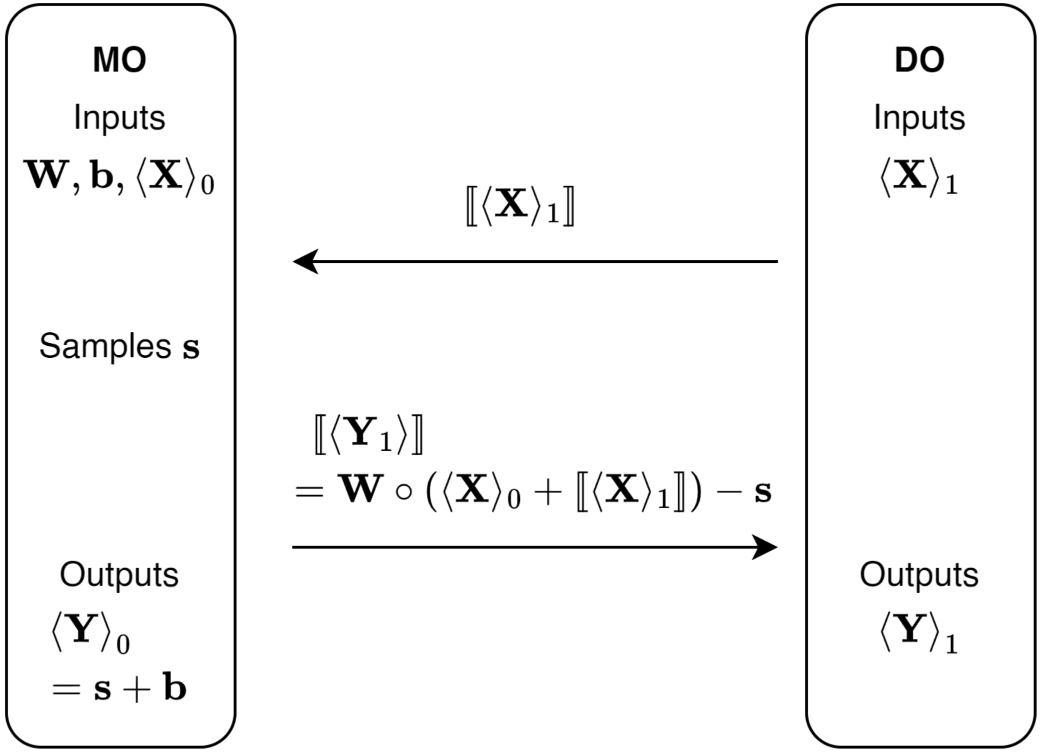
Training: Linear layers = HE

- Forward propagation
  - 2-round protocol with HE



Training: Linear layers = HE

- Forward propagation
  - 2-round protocol with HE
  - As a general solution, this algorithm does not specify how $\mathbf{W} \circ [\![\mathbf{X}]\!]$ is evaluated.
  - Our implementation uses batched polynomial encoding, but other methods (e.g. Gazelle's encoding) could be used.



Training: Linear layers = HE

- Backpropagation
  - Gradient of x, shared: $\nabla_x = W \odot_W \nabla_\mathrm{y}$
    - E.g. For FC, $\langle Y \rangle = W\langle X \rangle, \langle \nabla X \rangle = \langle \nabla Y \rangle \cdot W^T$

Training: Linear layers = HE

- Backpropagation
  - Gradient of x, shared: $\nabla_x = W \odot_W \nabla_\mathrm{y}$
    - E.g. For FC, $\langle Y \rangle = W \langle X \rangle, \langle \nabla X \rangle = \langle \nabla Y \rangle \cdot W^T$
  - Gradient of b, given to MO: $\nabla_b = \nabla_y \odot_b \frac{\partial f}{\partial b}$.
    - For FC and Conv, simply sum up ∇y and reveal to MO.

Training: Linear layers = HE

- Backpropagation
  - Gradient of x, shared: $\nabla_x = W \odot_W \nabla_{\mathrm{y}}$
    - E.g. For FC, $\langle Y \rangle = W \langle X \rangle, \langle \nabla X \rangle = \langle \nabla Y \rangle \cdot W^T$
  - Gradient of b, given to MO: $\nabla_b = \nabla_y \odot_b \frac{\partial f}{\partial b}$.
    - For FC and Conv, simply sum up ∇y and reveal to MO.
  - Gradient of W, given to MO

Training: Linear layers = HE

- For gradient of W, it is the product of two secret-shared values

$$\nabla_{\mathbf{W}} = \nabla_{\mathbf{Y}} \odot \frac{\partial f(\mathbf{X}; \mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} = \nabla_{\mathbf{Y}} \odot \mathbf{X}$$

Training: Linear layers = HE

- For gradient of W, it is the product of two secret-shared values

$$\nabla_{\mathbf{W}} = \nabla_{\mathbf{Y}} \odot \frac{\partial f(\mathbf{X}; \mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} = \nabla_{\mathbf{Y}} \odot \mathbf{X}$$

- Solution: HE for cross terms

(no need for cipher-cipher multiplication)

$$[\![\nabla_{\mathbf{W}}]\!] = (\langle \nabla_{\mathbf{Y}} \rangle_0 + [\![\langle \nabla_{\mathbf{Y}} \rangle_1]\!]) \odot (\langle \mathbf{X} \rangle_0 + [\![\langle \mathbf{X} \rangle_1]\!]).$$

Training: Linear layers = HE

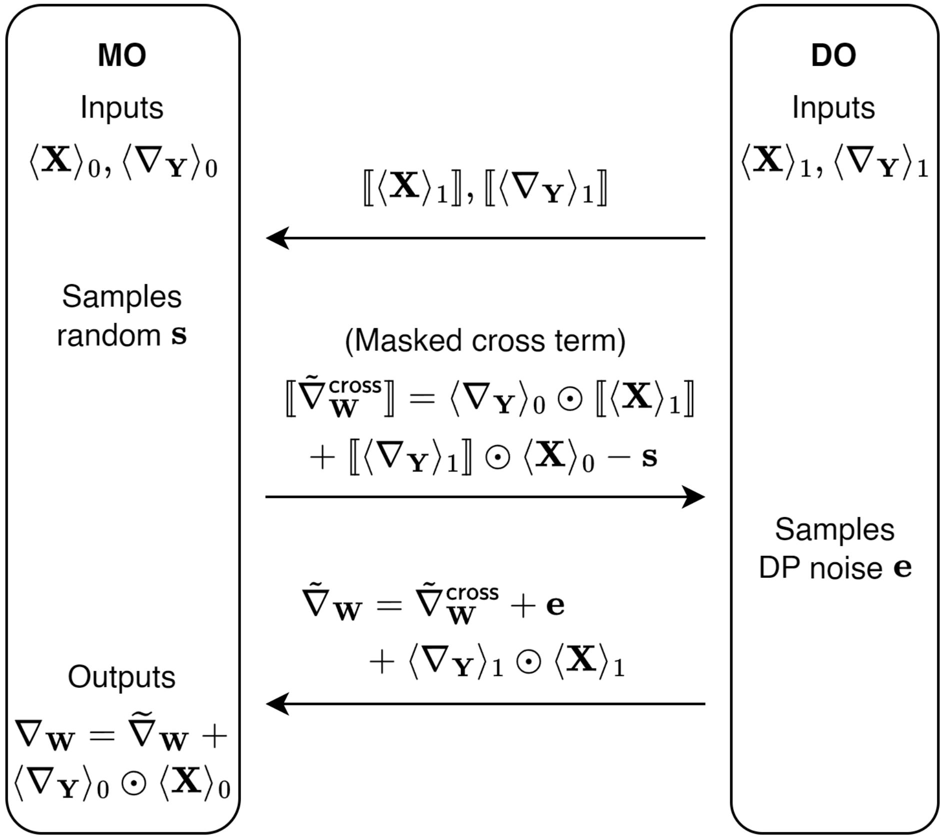
- For gradient of W, it is the product of two secret-shared values

$$\nabla_{\mathbf{W}} = \nabla_{\mathbf{Y}} \odot \frac{\partial f(\mathbf{X}; \mathbf{W}, \mathbf{b})}{\partial \mathbf{W}} = \nabla_{\mathbf{Y}} \odot \mathbf{X}$$

- Solution: HE for cross terms

(no need for cipher-cipher multiplication)

$$[\![\nabla_{\mathbf{W}}]\!] = (\langle \nabla_{\mathbf{Y}} \rangle_0 + [\![\langle \nabla_{\mathbf{Y}} \rangle_1]\!]) \odot (\langle \mathbf{X} \rangle_0 + [\![\langle \mathbf{X} \rangle_1]\!]).$$



Training: Non-linear layers = MPC

- Build with Two-party MPC

Training: Non-linear layers = MPC

- Build with Two-party MPC
- Example: $\mathrm{ReLU}(x) = \mathrm{DReLU}(x) \cdot x$

Training: Non-linear layers = MPC

- Build with Two-party MPC
- Example: $\mathrm{ReLU}(x) = \mathrm{DReLU}(x) \cdot x$
  - DReLU => secure comparison protocol
  - Boolean-arithmetic multiplication => OT-based multiplexing

Optimzing

- Substantial part of computation lies in HE linear operation evaluation

Optimzing

- Substantial part of computation lies in HE linear operation evaluation
  - $W \circ [\![X]\!]$ in FP, results shared
  - $W \odot_x [\![\nabla_Y]\!]$ in BP, results shared
  - $\langle \nabla_Y \rangle_0 \odot [\![\langle X \rangle_1]\!] + [\![\langle \nabla_Y \rangle_1]\!] \odot \langle X \rangle_0$ in BP, results shared

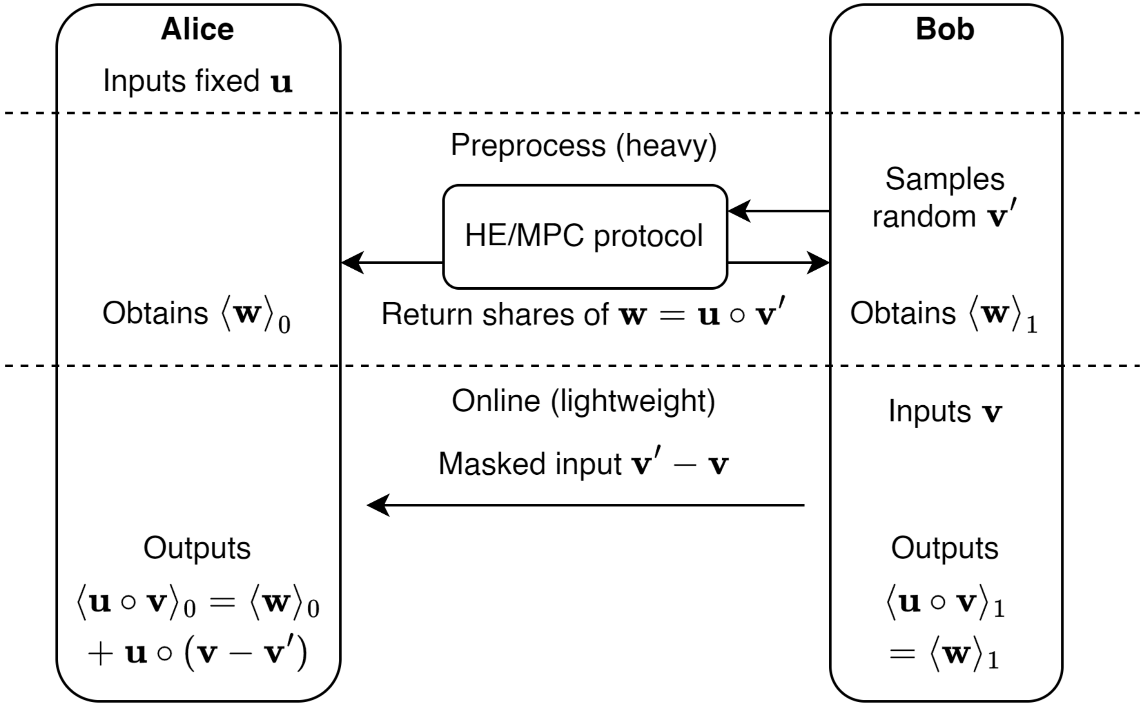
Optimzing

- Substantial part of computation lies in HE linear operation evaluation
  - $W \circ [\![X]\!]$ in FP, results shared
  - $W \odot_x [\![\nabla_Y]\!]$ in BP, results shared
  - $\langle\nabla_Y\rangle_0 \odot [\![\langle X\rangle_1]\!] + [\![\langle\nabla_Y\rangle_1]\!] \odot \langle X\rangle_0$ in BP, results shared
- **Generalization**: Is there a way to accelerate online evaluation of general operator $u \circ v$, each party holding one operand?

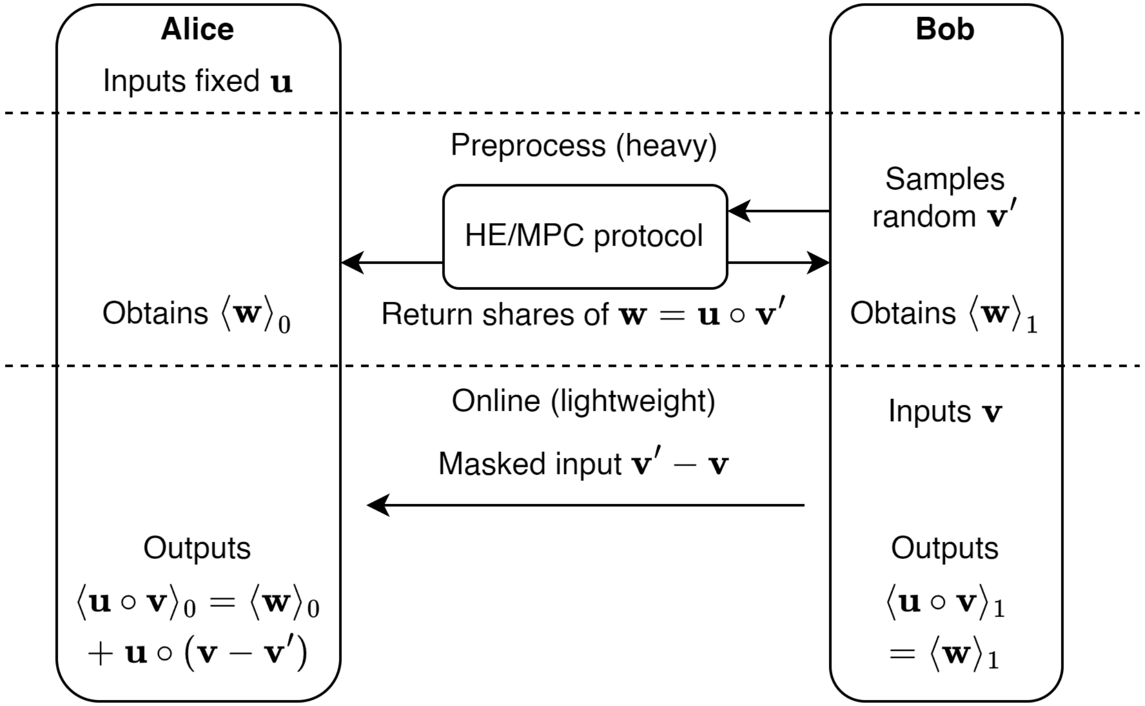
Optimzing

- Substantial part of computation lies in HE linear operation evaluation
  - $W \circ [\![X]\!]$ in FP, results shared
  - $W \odot_x [\![\nabla_Y]\!]$ in BP, results shared
  - $\langle \nabla_Y \rangle_0 \odot [\![\langle X \rangle_1]\!] + [\![\langle \nabla_Y \rangle_1]\!] \odot \langle X \rangle_0$ in BP, results shared
- **Generalization**: Is there a way to accelerate online evaluation of general operator $u \circ v$, each party holding one operand?
- First, let's consider a fixed u and variable v's.

Traditional preprocessing: Beaver triples

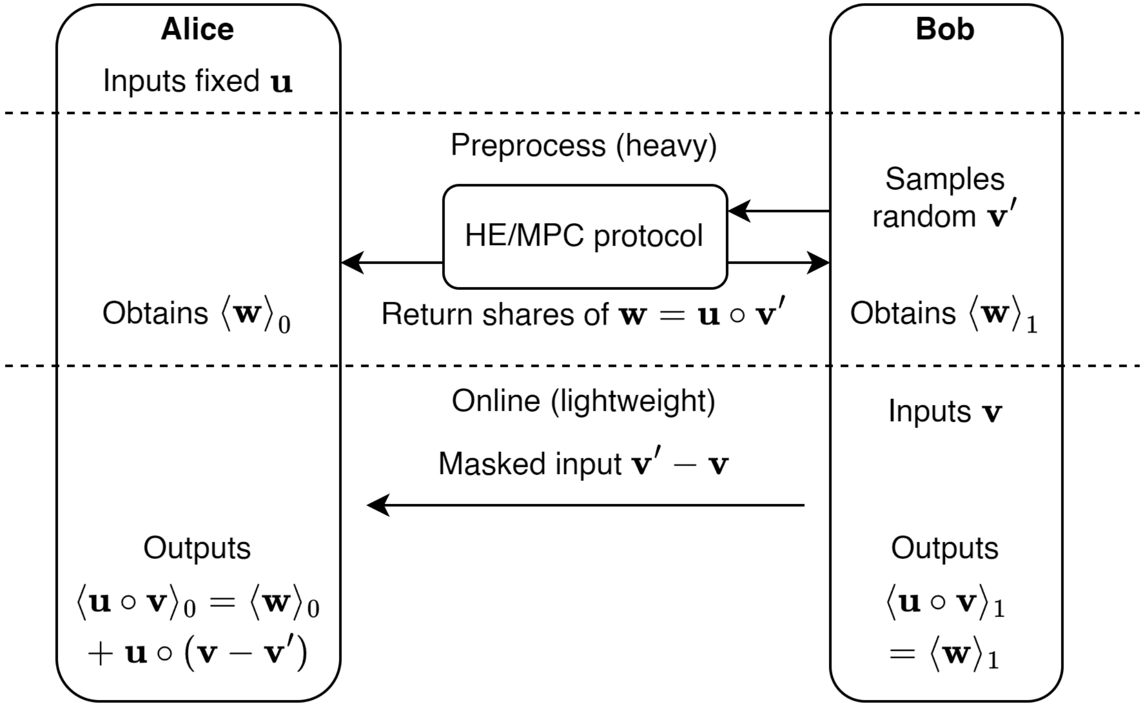


Traditional preprocessing: Beaver triples



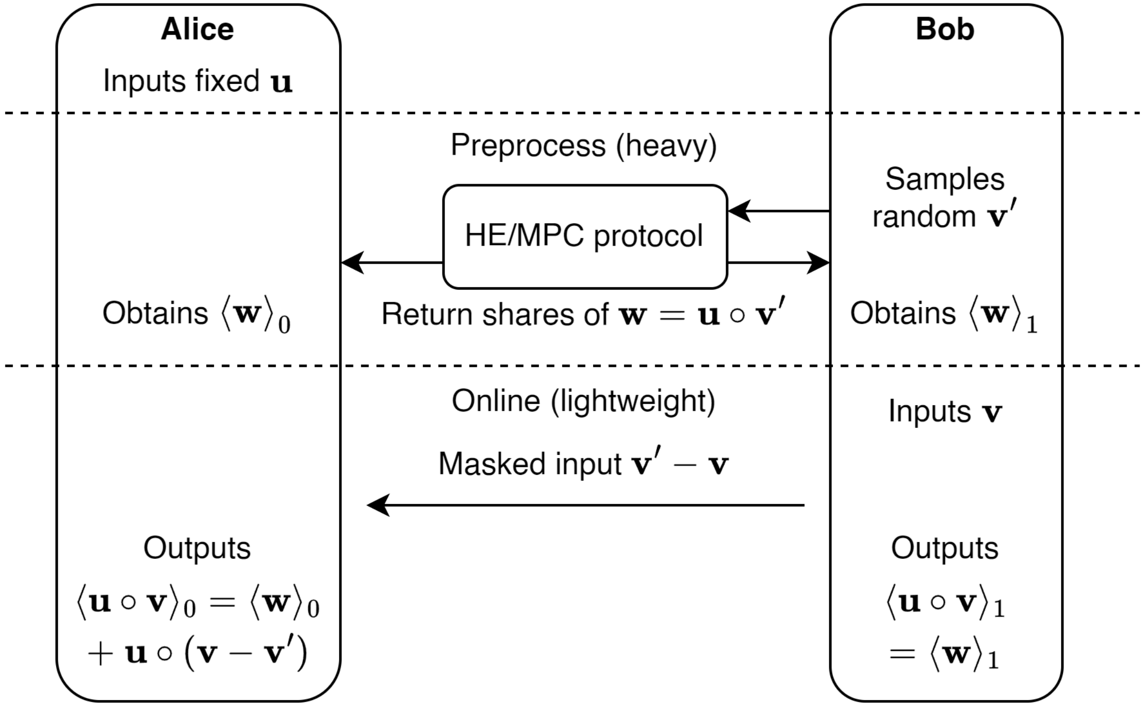
- Preprocessed shares cannot be reused, or info is leaked.

Traditional preprocessing: Beaver triples



- Preprocessed shares cannot be reused, or info is leaked.
  - If, for v_0 and v_1, the same sharing of $w = u' \circ v'$ was used, Bob would send $v' - v_0$ and $v' - v_1$ to Alice, so Alice would obtain the difference $v_0 - v_1$ (a direct linear combination of 2 input values)

Traditional preprocessing: Beaver triples



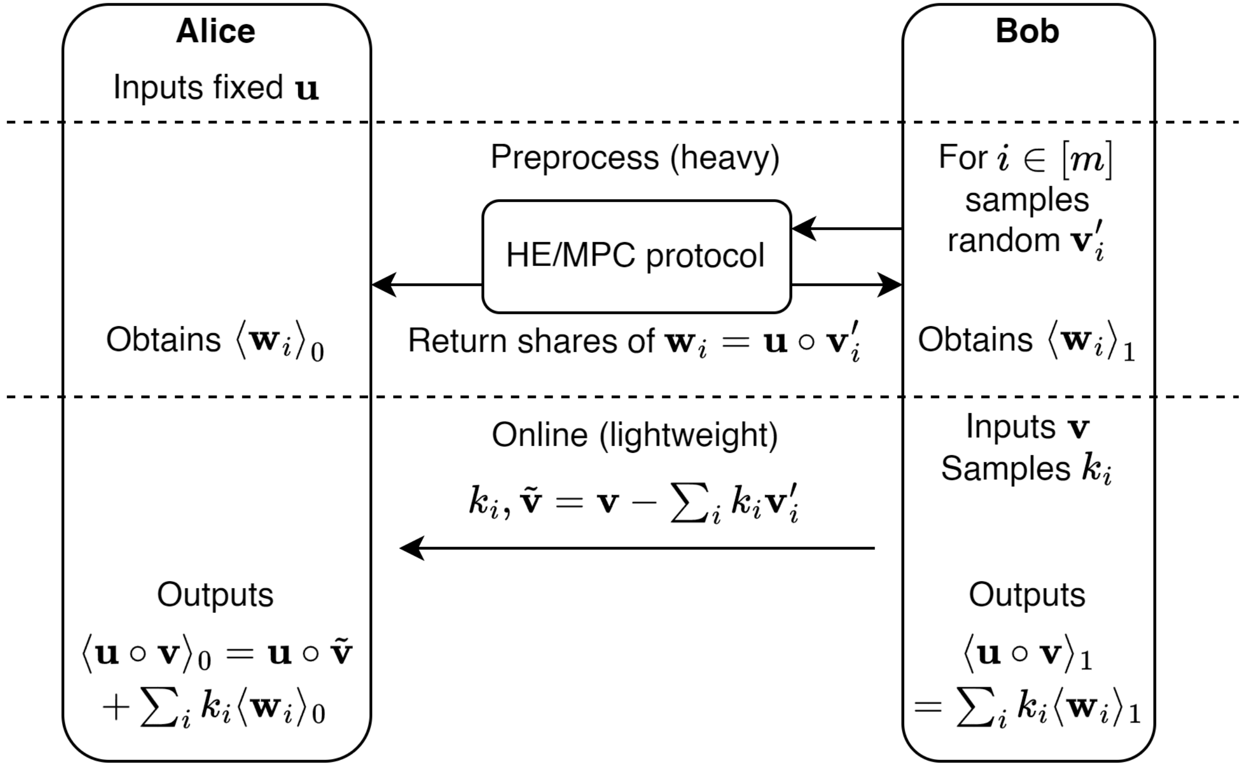
- Preprocessed shares cannot be reused, or info is leaked.
  - If, for v_0 and v_1, the same sharing of $w = u' \circ v'$ was used, Bob would send $v' - v_0$ and $v' - v_1$ to Alice, so Alice would obtain the difference $v_0 - v_1$ (a direct linear combination of 2 input values)
- Total communication is not reduced, while total computation is even increased.

Novel approach: Multiple masks

- Solution: use multiple masks to increase the revealed linear combination complexity.

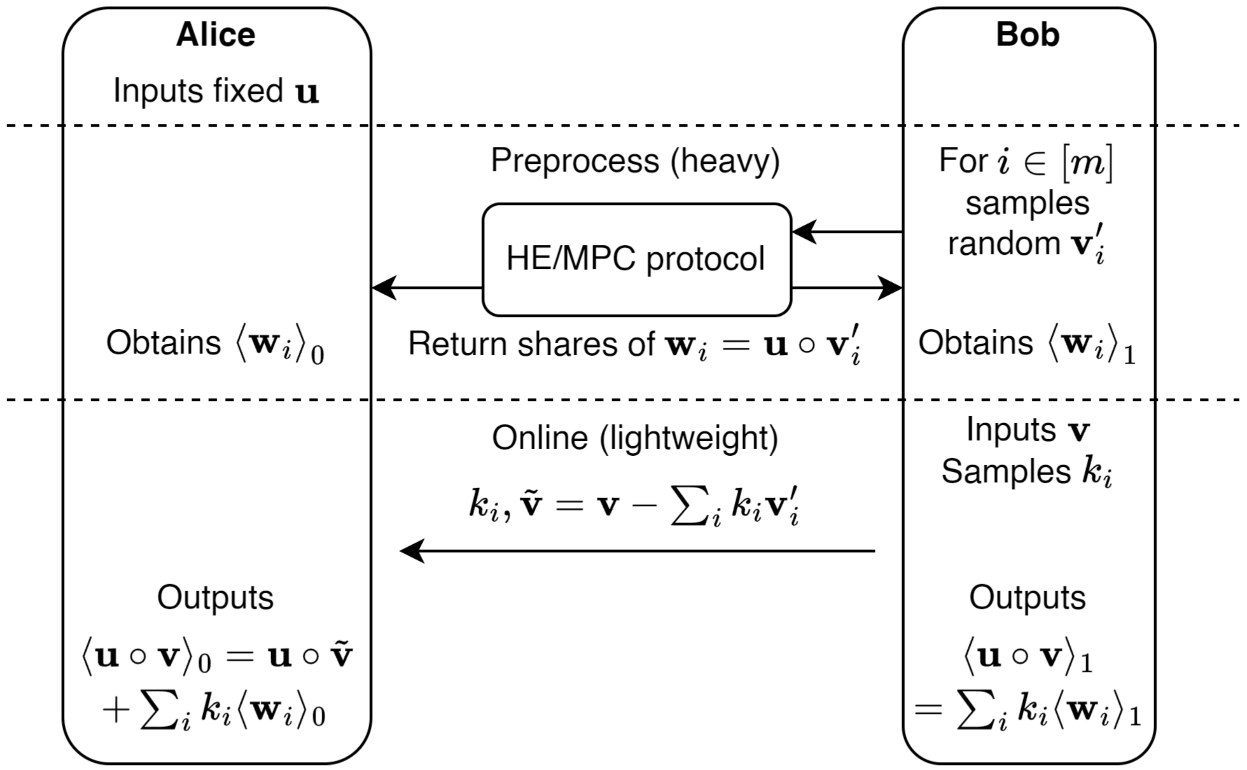
Novel approach: Multiple masks

- Solution: use multiple masks to increase the revealed linear combination complexity.
  - Bob samples m random v_i''s to conduct preprocessing.



Novel approach: Multiple masks

- Solution: use multiple masks to increase the revealed linear combination complexity.
  - Bob samples m random v_i''s to conduct preprocessing.
  - Reuse the shares $\langle w_i \rangle$ for multiple online executions

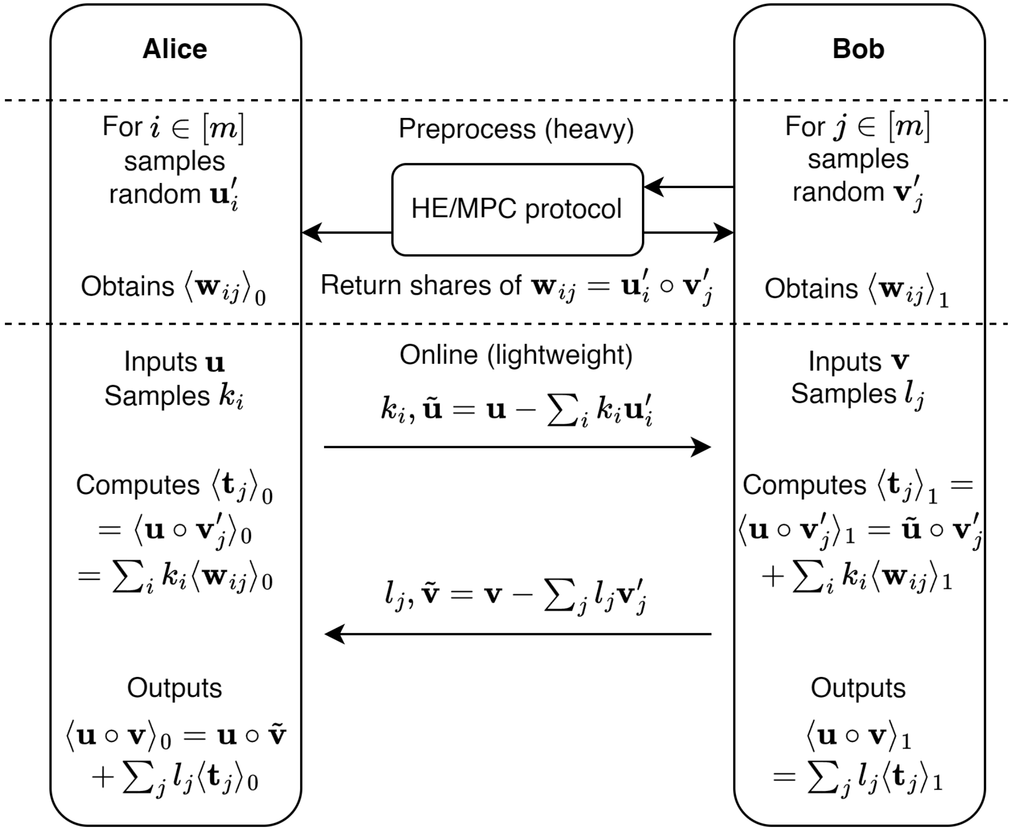


Multiple masks

- Extending to variable u's
  - Similarly, Alice samples multiple masks u_i'

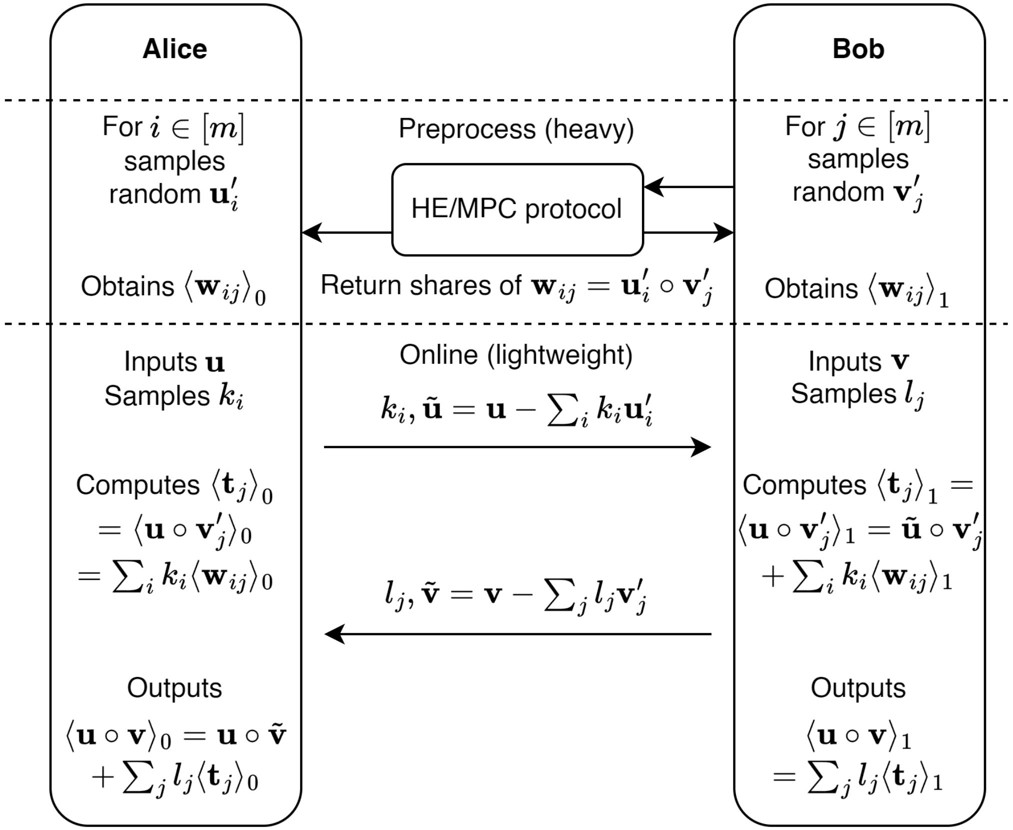
Multiple masks

- Extending to variable u's
  - Similarly, Alice samples multiple masks u_i'



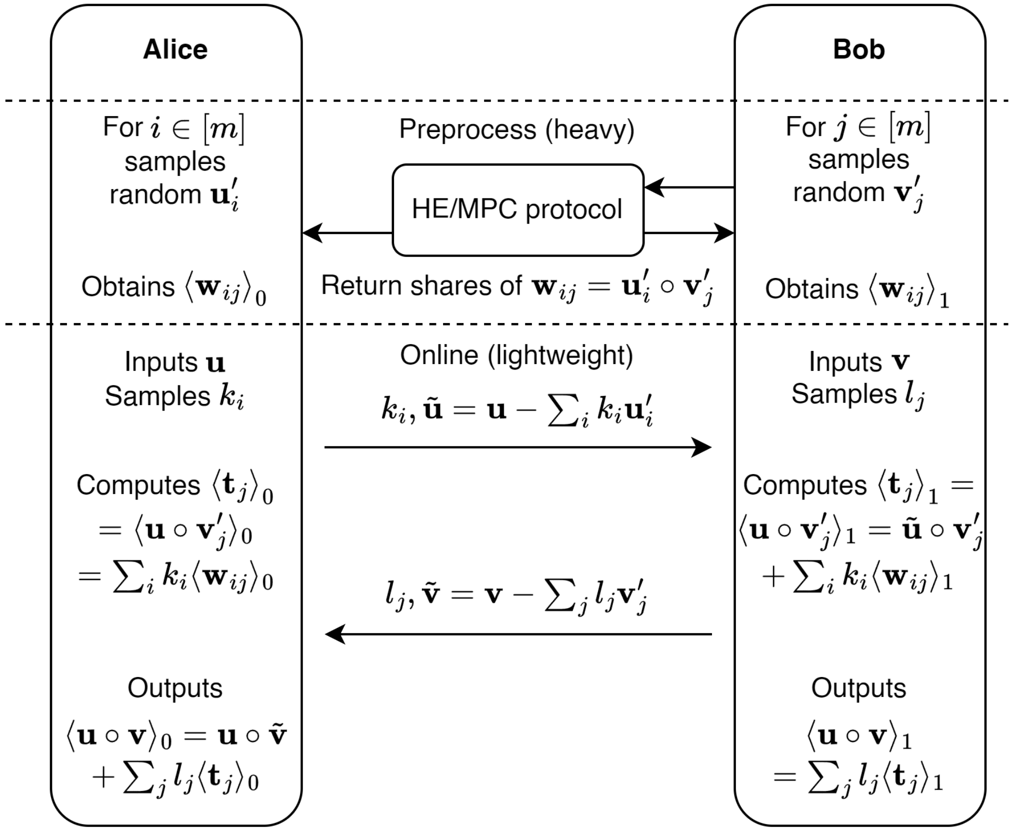
Multiple masks

- Extending to variable u's
  - Similarly, Alice samples multiple masks u_i'
- If the online phase is executed T times:
  - Traditional: T times HE/MPC evaluation of $\circ$
  - Ours: m^2 times of HE/MPC evaluation, regardless of T



Multiple masks

- Extending to variable u's
  - Similarly, Alice samples multiple masks u_i'
- If the online phase is executed T times:
  - Traditional: T times HE/MPC evaluation of $\circ$
  - Ours: m^2 times of HE/MPC evaluation, regardless of T
- Security analysis shows
  - To eliminate masks, an attacker would require at least $m + 1$ equations
  - Complexity of breaking one u or v is $O(2^{fm})$, f being the fixed-point precision



Evaluation: Training costs

Since the preprocessing technique introduces different security properties, we denote Pencil with or without it as Pencil and Pencil+ respectively.

Evaluation: Training costs

Since the preprocessing technique introduces different security properties, we denote Pencil with or without it as Pencil and Pencil+ respectively.

Scenario	Task	Model	Pencil			Pencil$^+$				
			Online			Preprocessing		Online		
			TP_{LAN}	TP_{WAN}	C	T_{prep}	C_{prep}	TP_{LAN}	TP_{WAN}	C
Train from scratch	MNIST	NN1	9.73×10^4	5.12×10^4	1.66	0.02	3.35	26.52×10^4	19.87×10^4	0.23
	MNIST	NN2	7.70×10^4	4.43×10^4	1.71	0.02	4.13	13.72×10^4	10.75×10^4	0.36
	CIFAR10	NN3	2.58×10^4	1.62×10^4	4.11	0.05	10.26	2.90×10^4	1.98×10^4	2.86
	CIFAR10	NN4	0.18×10^4	0.12×10^4	44.89	0.70	83.12	0.22×10^4	0.15×10^4	34.90
Transfer learning	CIFAR10	NN5	0.52×10^4	0.39×10^4	11.33	0.91	46.00	1.55×10^4	1.24×10^4	2.90
	CIFAR10	NN6	1.83×10^4	1.17×10^4	5.48	0.30	15.96	8.05×10^4	5.89×10^4	0.82

TABLE III: Training costs for different ML tasks. For the online phase, TP stands for the throughput (images/hour) of the training system, and subscript LAN, WAN indicate the network settings; C stands for the online communication (MB) per image. For Pencil$^+$, we also report the time (T_{prep}, hours) and communication (C_{prep}, GB) of preprocessing. Note that the preprocessing overhead is one-time overhead.

With Pencil+ and transfer learning, a model for CIFAR10 classification could be trained within 6.5 hours (10 epochs)

Evaluation: Training costs

Model	Throughput (10^4 img/h)				Comm. (MB/img)		
	[2]	[12]	P	P+	[12]	P	P+
2×128FC	0.7	0.11	9.7	29.3	552	1.7	0.2
3×128FC	0.6	0.10	8.1	18.9	658	2.2	0.3
2×512FC	0.2	0.03	2.6	13.2	3470	5.2	0.8

TABLE VII: Performance comparison with QUOTIENT [2] and Semi2k [12] in the 2 party setting. The models are represented as $n \times m$FC, as used by [2]. P represents Pencil and P^+ represents $Pencil^+$.

- Comparison with previous 2PC works shows improvements of up to 2 orders of magnitude.

Evaluation: Training costs

Model	Throughput (10^4 img/h)				Comm. (MB/img)		
	[2]	[12]	P	P+	[12]	P	P+
2×128FC	0.7	0.11	9.7	29.3	552	1.7	0.2
3×128FC	0.6	0.10	8.1	18.9	658	2.2	0.3
2×512FC	0.2	0.03	2.6	13.2	3470	5.2	0.8

TABLE VII: Performance comparison with QUOTIENT [2] and Semi2k [12] in the 2 party setting. The models are represented as $n \times m$FC, as used by [2]. P represents Pencil and P+ represents Pencil+.

- Comparison with previous 2PC works shows improvements of up to 2 orders of magnitude.

Model	Throughput (10^3 img/h)			Comm. (per img)		
	[12]	Pencil	Pencil+	[12]	Pencil	Pencil+
2 parties	1.11	97	293	0.55GB	1.7MB	0.2MB
3 parties	0.61	97	293	2.58GB	1.7MB	0.2MB
4 parties	0.41	97	293	6.06GB	1.7MB	0.2MB
5 parties	0.07	97	293	57.69GB	1.7MB	0.2MB

TABLE VIII: Performance comparison with Semi2k [12] in multiple party setting.

- Unlike previous general n-PC frameworks, extending to multiple DOes does not introduce extra overhead for Pencil.

Thank you for listening!