Scalable Multi-Party Computation Protocols for Machine Learning in the Honest-Majority Setting

Fengrun Liu, Xiang Xie, Yu Yu

Machine Learning and Privacy

Product Recommendation

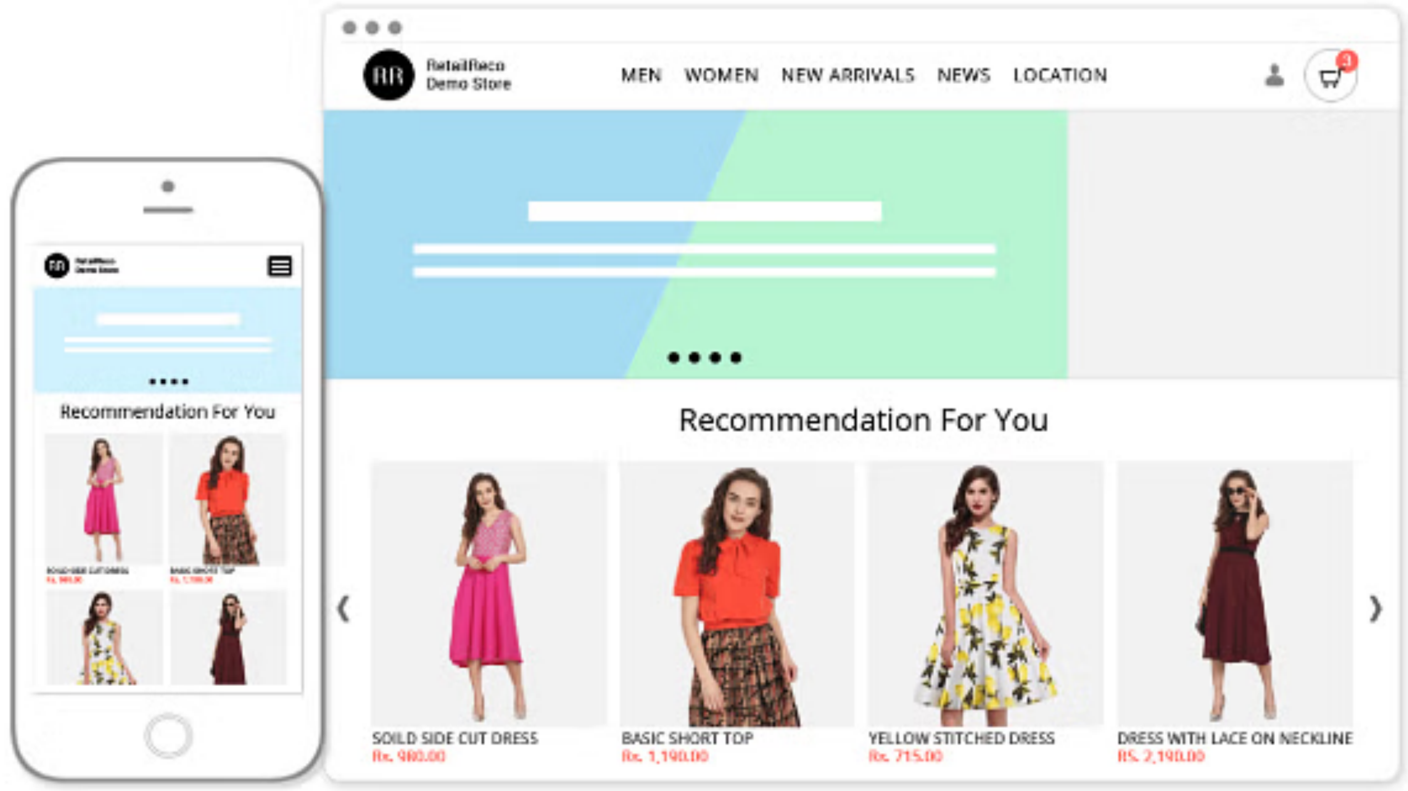
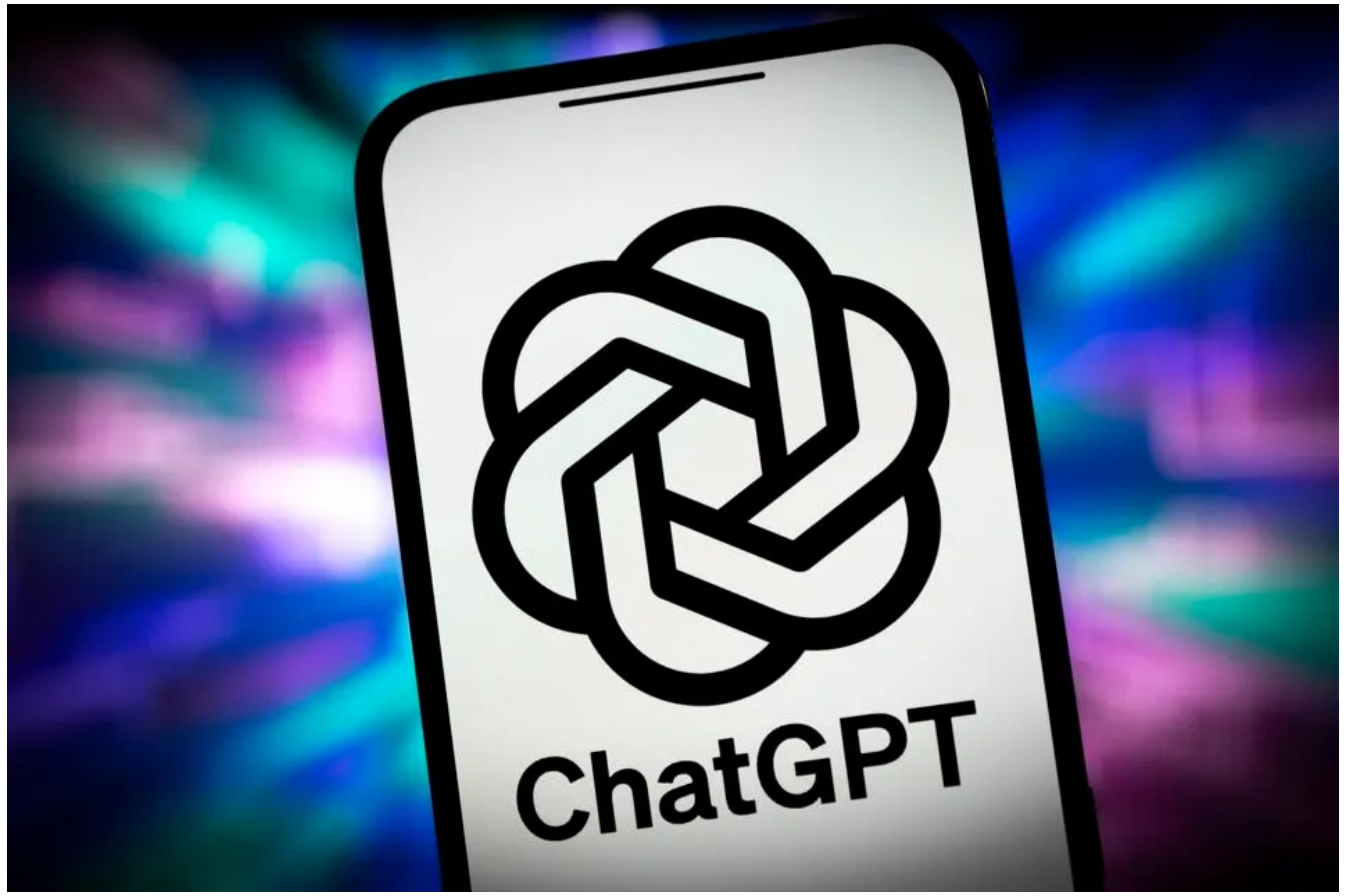


Image Processing

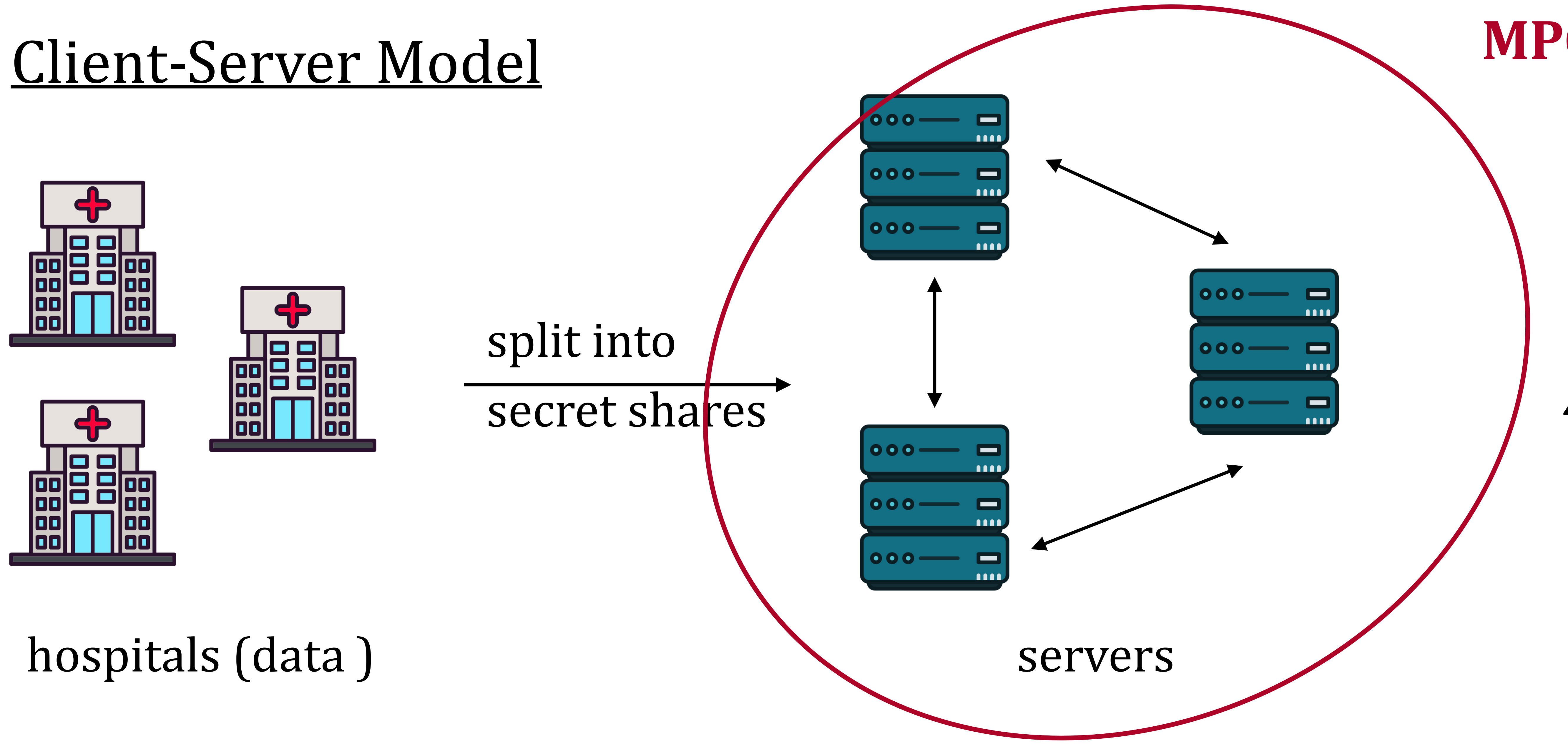
Language Model: generate human-like responses



More data —> Better models

Privacy-preserving Machine Learning (PPML)

Client-Server Model



MPC Protocols for PPML

2PC: [SecureML] [Ezpc] [Chameleon] [Delphi] ...

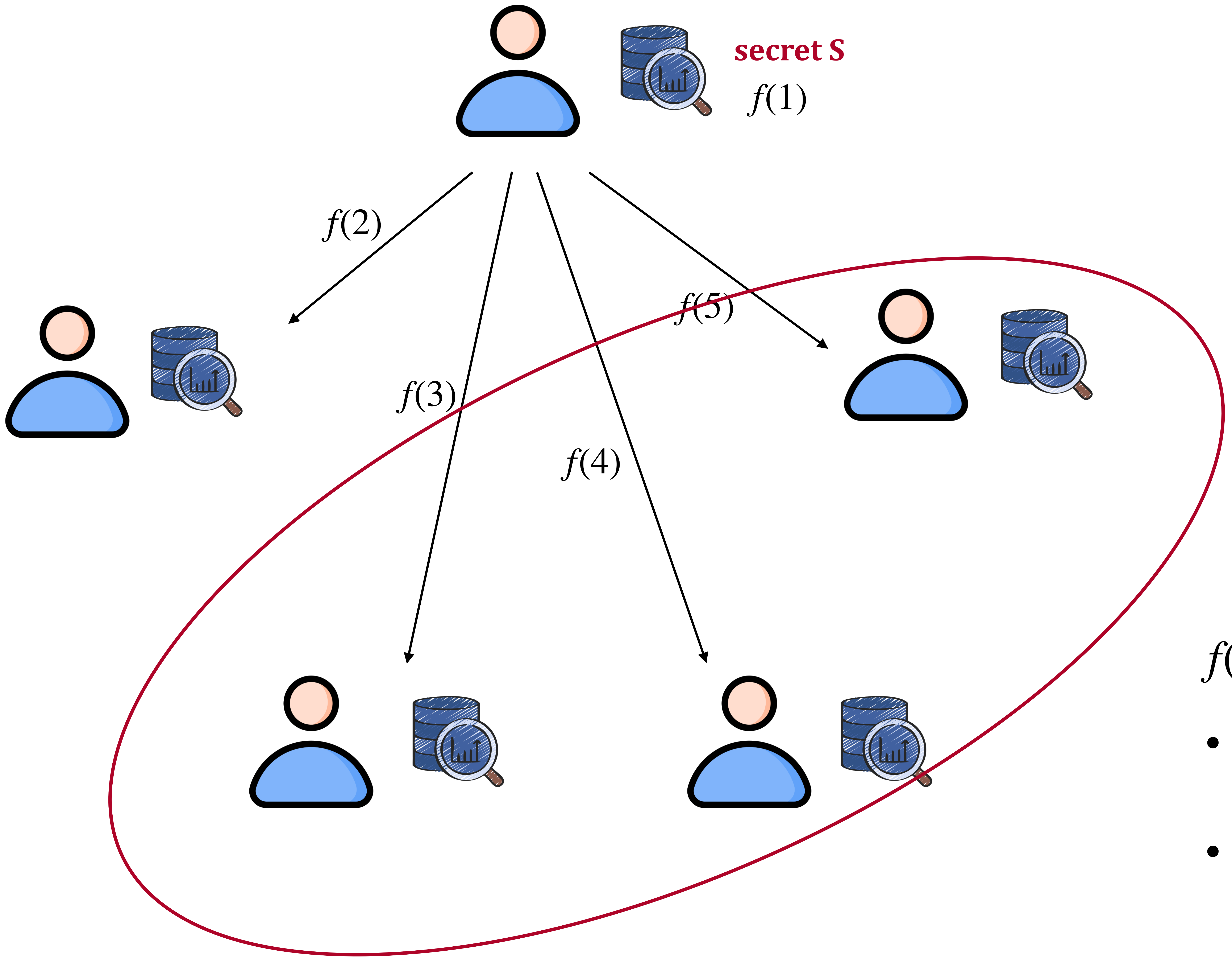
3PC: [ABY3] [SecureNN] [Falcon] [DEK20] ...

4PC: [Flash] [Trident] [Fantastic Four] [Swift] ...

How about MPC Protocols for arbitrary n parties?

General-purpose MPC:
[DN07] [DPSZ12] [SPDZ] [Mascot] ...

Scalability from Shamir's Secret Sharing



=> secret S

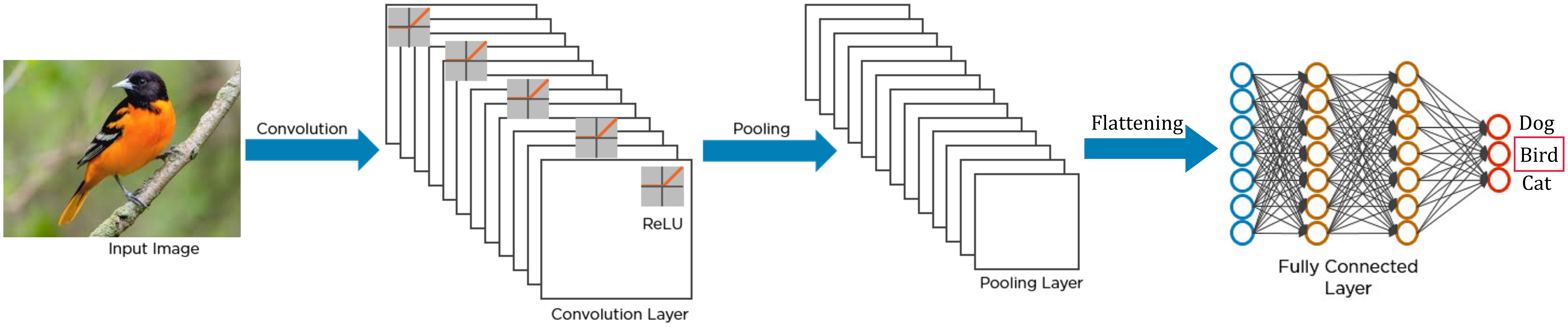
$$f(x) = s + a_1x + a_2x^2$$

$$f(x) = s + a_1x + a_2x^2 + \ldots + a_tx^t$$

- distributed to **n** parties as shares
  (denoted by $[s]$)
- can be reconstructed from $\geq$ **t + 1** parties

t-privacy

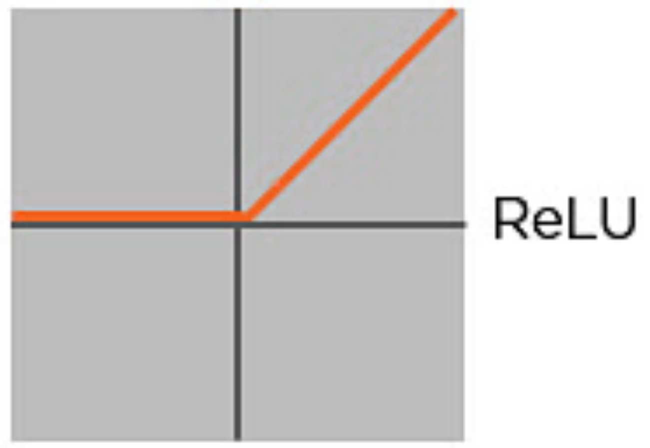
Two Obstacles in Oblivious Inference on a Neural Network



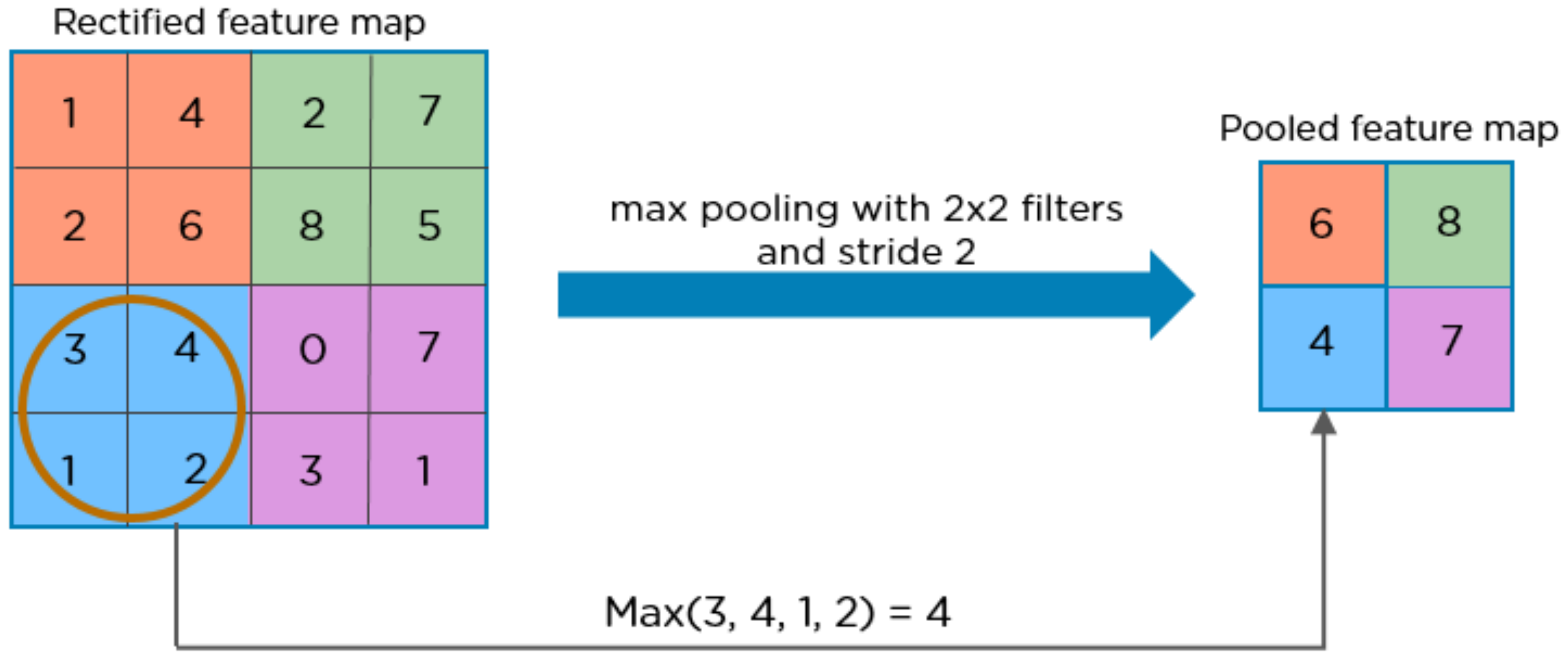
- Decimal number
- Non-linear function

$$\mathrm{DReLU}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$\mathrm{ReLU}(x) = \mathrm{DReLU}(x) \cdot x$$



$$\mathrm{Max}(a, b) = \mathrm{ReLU}(a - b) + b$$



Decimal Multiplications in Integer Ring $\mathbb{Z}_{2^\ell}$

- Represent an integer $\bar{x} \in [-2^{\ell-1}, 2^{\ell-1})$

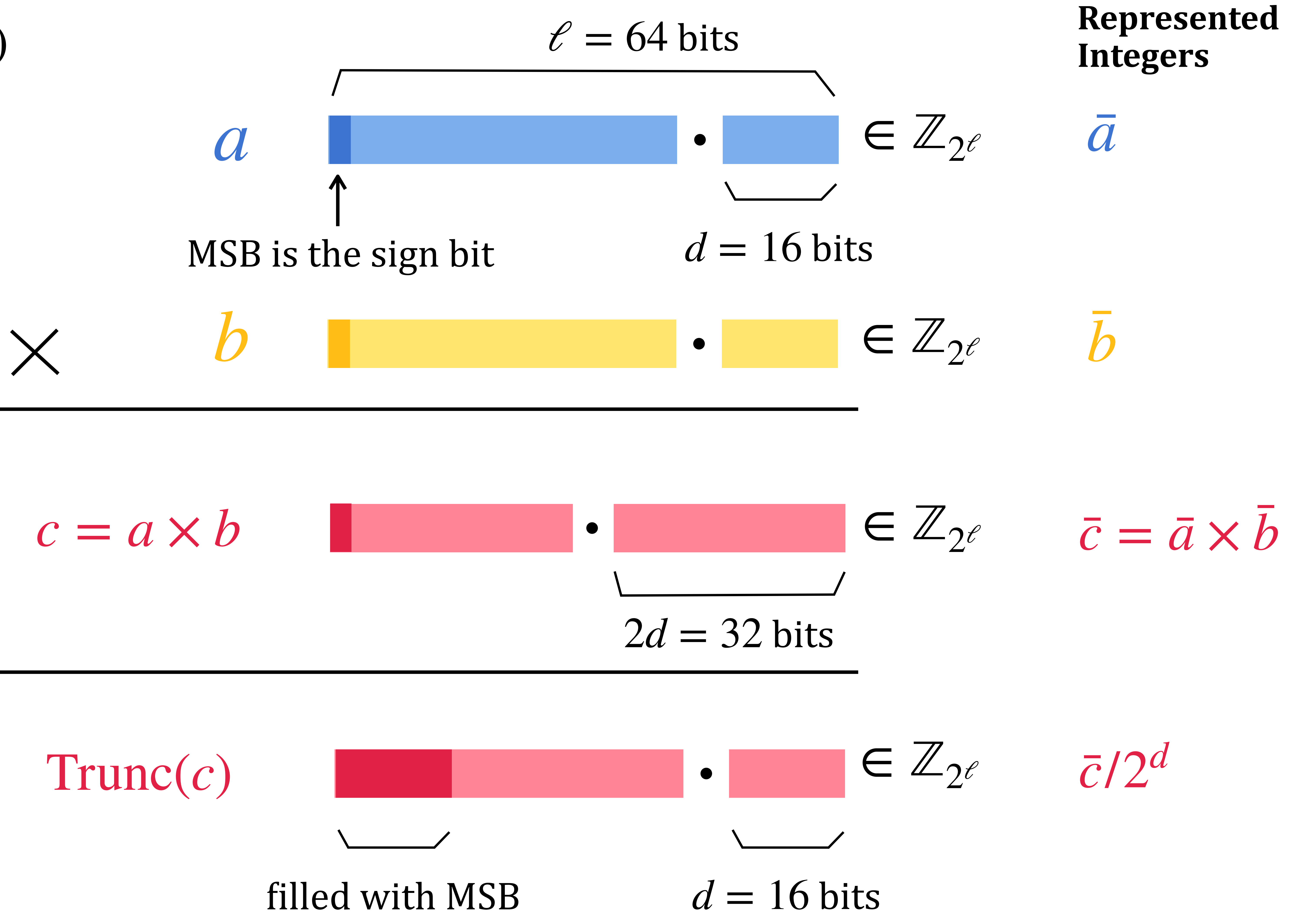
$$x = \bar{x} \pmod{2^\ell} = \begin{cases} \bar{x}, & \bar{x} \geq 0 \\ 2^\ell - \bar{x}, & \bar{x} < 0 \end{cases}$$

(2's complement)

- Truncation on c : performing $\bar{c}/2^d$

shift the bits down by d positions

and fill the top d bits with MSB of c



Decimal Multiplications in Mersenne Field $\mathbb{F}_p$ $(p = 2^\ell - 1)$

- Represent an integer $\bar{x} \in (-2^{\ell-1}, 2^{\ell-1})$

$$x = \bar{x} \pmod{2^\ell - 1} = \begin{cases} \bar{x}, & \bar{x} \geq 0 \\ 2^\ell - 1 - \bar{x}, & \bar{x} < 0 \end{cases}$$

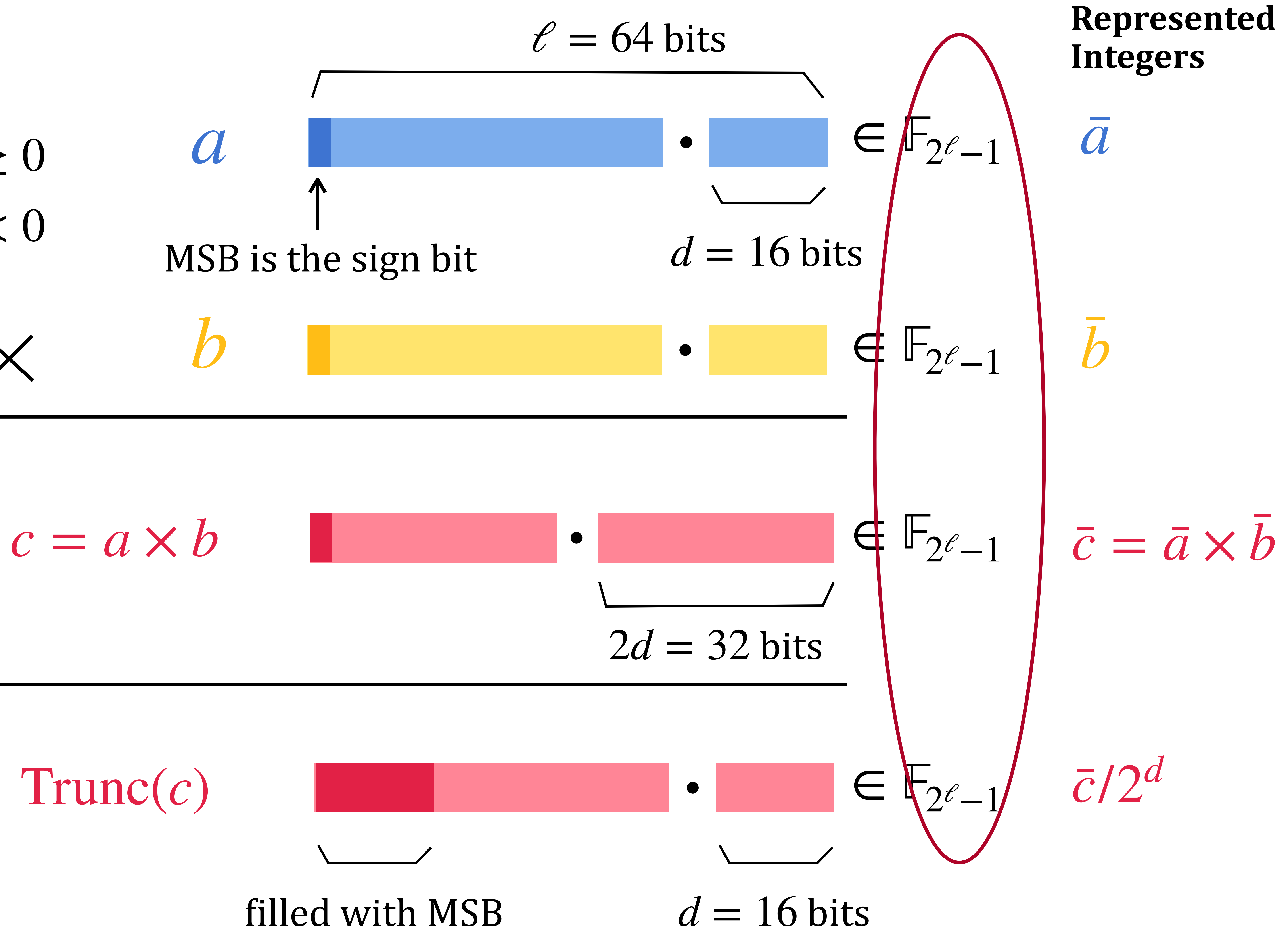
$\times$

- Truncation on c : performing $\bar{c}/2^d$

Truncation in $\mathbb{F}_{2^\ell - 1}$ = Truncation in $\mathbb{Z}_{2^\ell}$

shift the bits down by d positions

and fill the top d bits with MSB of c



Previous Truncation Protocol with A Large Gap

Preprocess: a pair $([r], [\mathrm{Trunc}(r)])$ where $r \leftarrow \mathbb{F}_{2^\ell - 1}$

Online:

1. $[a] = [x] + [r]$
2. Reveal a
3. $[\mathrm{Trunc}(x)] = \mathrm{Trunc}(a) - [\mathrm{Trunc}(r)]$

A Large Gap !!

only holds w.h.p. for small $x < < 2^\ell - 1$

Our Truncation Protocol with Only 1-bit Gap

For example, we have $a = x + r$ in $\mathbb{F}_{2^9-1}$.

x=162	0 1 0 1 0 0 0 1 0	$\bar{x} > 0$
+ r=161	0 1 0 1 0 0 0 0 1	+ $\bar{r} > 0$
= a=323	1 0 1 0 0 0 0 1 1	$\neq$ $\bar{a} < 0$

$$\xrightarrow[\text{d=3}]{\text{truncation}}$$

Trunc(x)=20	0 0 0 0 1 0 1 0 0
+ Trunc(r)=20	0 0 0 0 1 0 1 0 0
$\neq$ Trunc(a)=488	1 1 1 1 0 1 0 0 0

$a = x + r$

$\bar{a} \neq \bar{x} + \bar{r}$

(positive overflow)

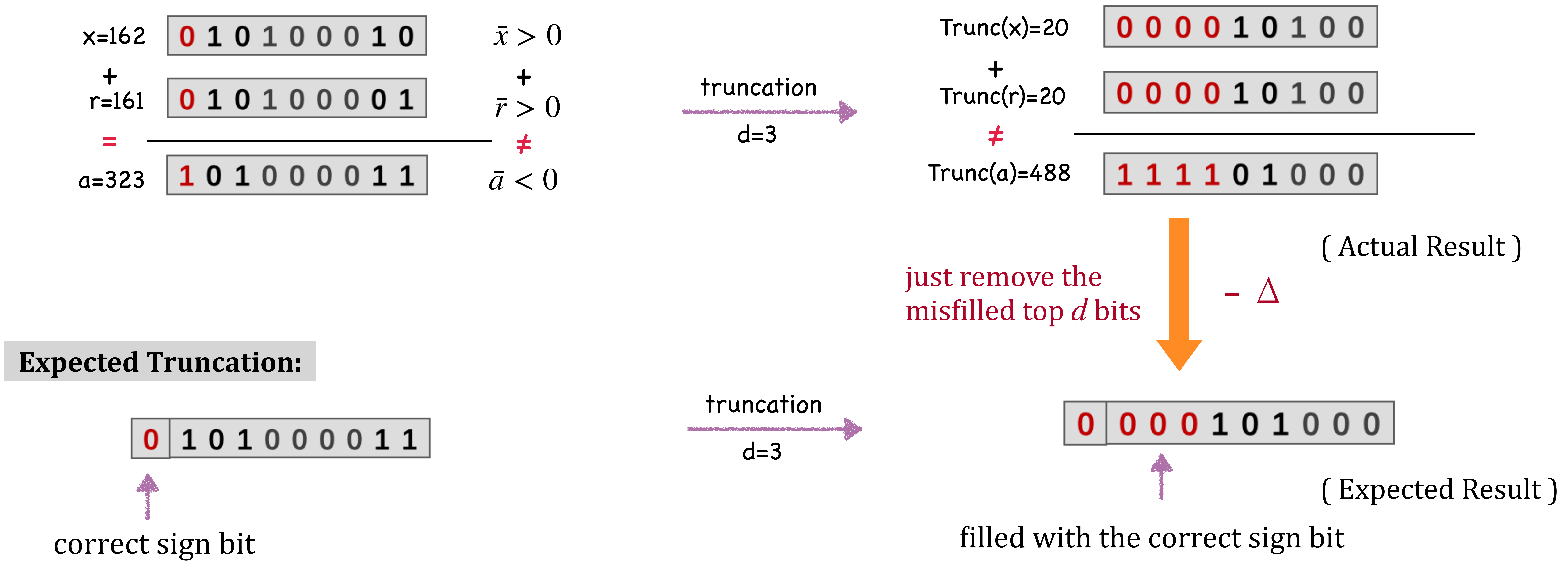
$\text{Trunc}(a) \neq \text{Trunc}(x) + \text{Trunc}(r)$

incorrect sign bit:
falsely indicates the result is negative

filled with the incorrect sign bit

Our Truncation Protocol with Only 1-bit Gap

For example, we have $a = x + r$ in $\mathbb{F}_{2^9-1}$.



Our Truncation Protocol with Only 1-bit Gap

For example, we have $a = x + r$ in $\mathbb{F}_{2^9-1}$.

x is always positive

x=162 `0 1 0 1 0 0 0 1 0` $\quad \bar{x} > 0$

+

r=161 `0 1 0 1 0 0 0 0 1` $\quad \boxed{\bar{r} > 0}$

=

a=323 `1 0 1 0 0 0 0 1 1` $\quad \boxed{\bar{a} < 0}$

positive overflow happen

$\xrightarrow[\text{d=3}]{\text{truncation}}$

Trunc(x)=20 `0 0 0 0 1 0 1 0 0`

+

Trunc(r)=20 `0 0 0 0 1 0 1 0 0`

≠

Trunc(a)=488 `1 1 1 1 0 1 0 0 0`

Our Truncation Protocol with Only 1-bit Gap

Preprocess: a triple $([r], [\mathrm{Trunc}(r)], [\mathrm{MSB}(r)])$ where $r \leftarrow \mathbb{F}_{2^{\ell}-1}$

Online: for positive input $x \in [0, 2^{\ell-1})$

1-bit Gap

1. $[a] = [x] + [r]$
2. Reveal a
3. $[e] = (1 - [\mathrm{MSB}(r)]) \cdot \mathrm{MSB}(a)$
4. $[\mathrm{Trunc}(x)] = \mathrm{Trunc}(a) - [\mathrm{Trunc}(r)] + [e] \cdot (2^{\ell-d} - 1)$

$e = 1$ indicating positive overflow happens

holds for any $x \in [0, 2^{\ell-1})$
representing positive numbers

$+$ [DN07] $(<r>, [\mathrm{Trunc}(r)], [\mathrm{MSB}(r)])$ $=$ 1-round Fixed-point Multiplication Protocol with Only 1-bit Gap

Non-linear Function via Bitwise Comparison

arithmetic comparison ($x < 0$)

$$\mathrm{DReLU}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Fact: MSB(x) = LSB (2x)   holds in odd rings

1. y = 2x + r
2. Reveal y
3. LSB(2a) = LSB(y) ⊕ LSB(r) ⊕ (y_B < r_B)

public   secret

bitwise comparison

bitwise comparison (y_B < r_B)

look for the first different bit

(public) y_B   0 0 1 0 1 0 0 0 1 0 1 0 1

(secret) r_B   0 0 1 0 1 0 1 0 0 0 1 0 0

XOR

(secret)   0 0 0 0 0 0 1 0 1 0 0 0 1

* Prefix-OR

(secret)   0 0 0 0 0 0 1 1 1 1 1 1 1

↓

(secret) e_B   0 0 0 0 0 0 1 0 0 0 0 0 0

(y_B < r_B) = <e_B, r_B>
= 1

* **Prefix-OR** involves ℓ multiplications: $b_j = \bigvee_{i=1}^{j} a_i$ for $j = 1, \ldots, \ell$

Round-Efficient Prefix-OR Protocol via Prefix-AND

Online Complexity of [NO07]: 5 rounds

Online Complexity of Prefix-Mult[BB89]: 1 round

* Prefix-OR: compute $b_j = \bigvee_{i=1}^{j} a_i$ for $j = 1,\ldots,\ell$

* Prefix-AND: compute $b_j = \bigwedge_{i=1}^{j} a_i$ for $j = 1,\ldots,\ell$

Prefix-OR

(secret) 0 0 0 0 0 0 1 0 1 0 0 0 1

(secret) 0 0 0 0 0 0 1 1 1 1 1 1 1

Prefix-AND

(secret) 1 1 1 1 1 1 0 1 0 1 1 1 0

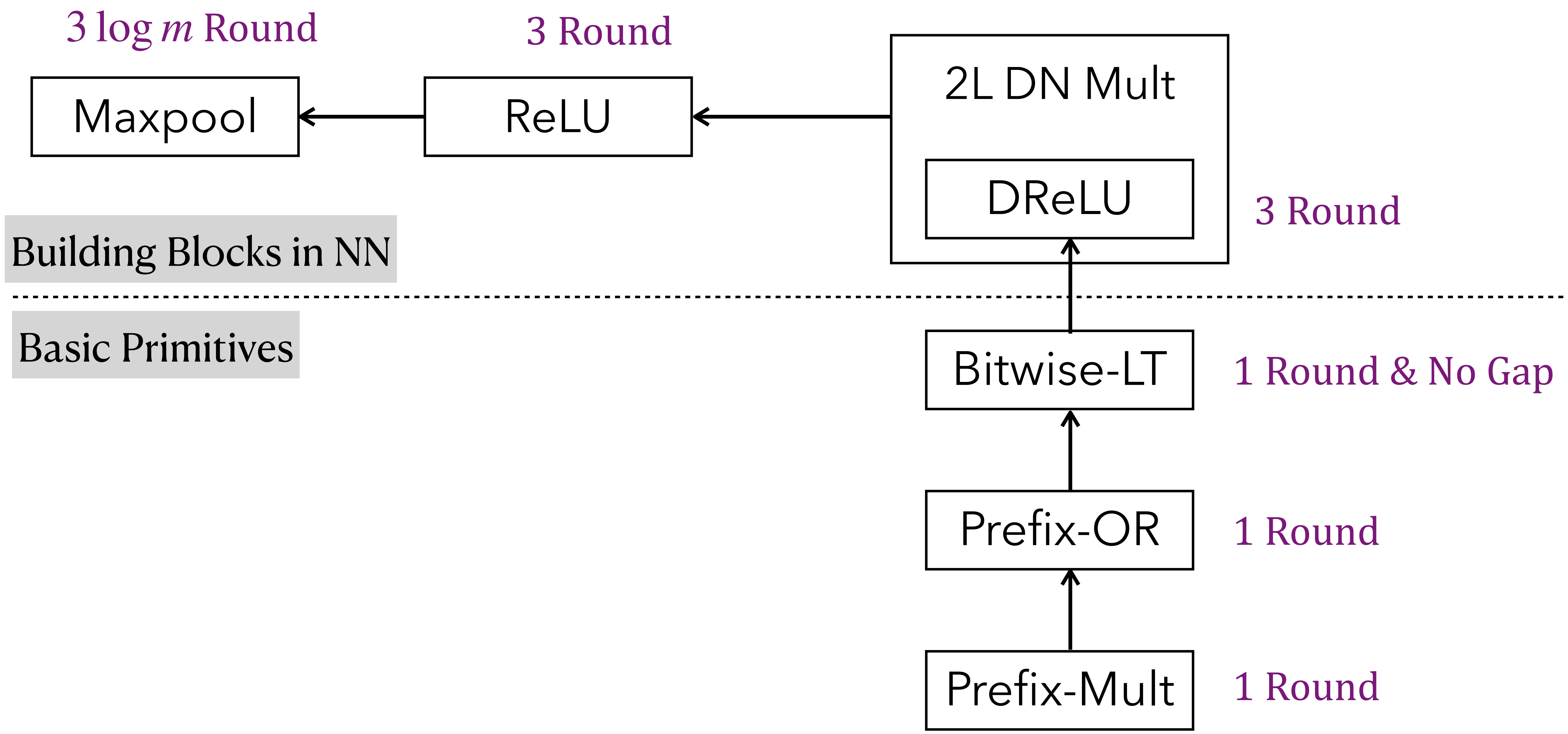
(secret) 1 1 1 1 1 1 0 0 0 0 0 0 0

1. locate the first 1-bit's position
2. set all the following bits to 1 by OR operation

Dual Problem

1. locate the first 0-bit's position
2. set all the following bits to 0 by AND
   (zero out all the following bits)

Other Building Blocks

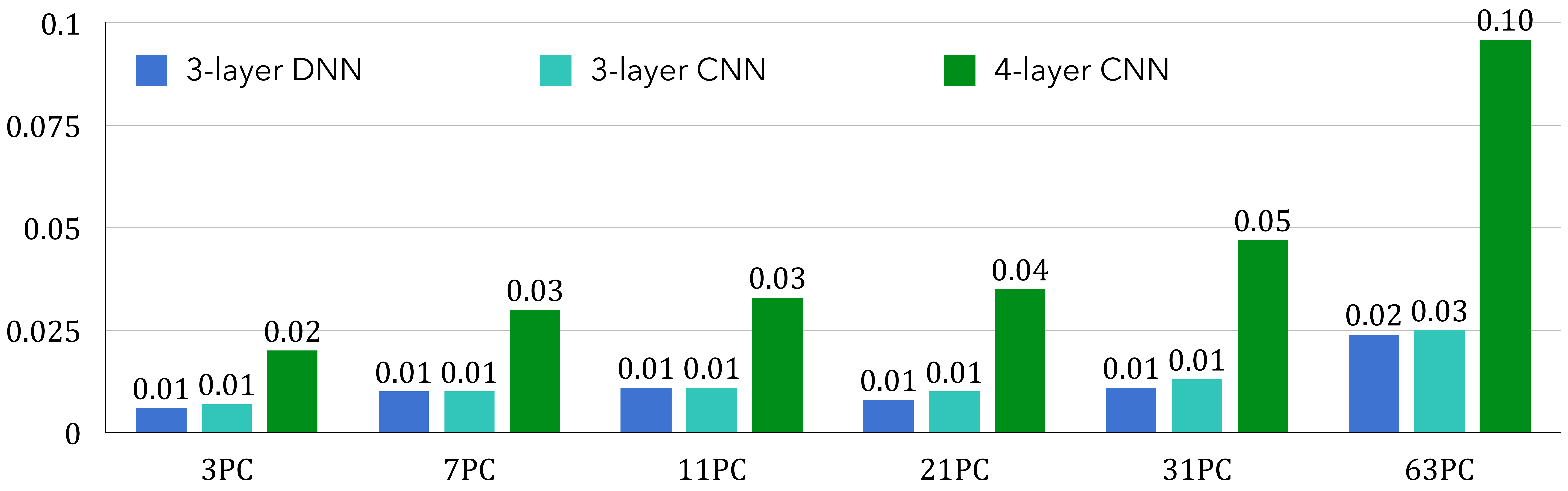


Round Complexity in Online Phase

Performance: Oblivious Inference

Stimulate 3-63 parties on 11 servers

LAN: 15Gb/s, delay 0.3ms
WAN: 100Mb/s, delay 40ms



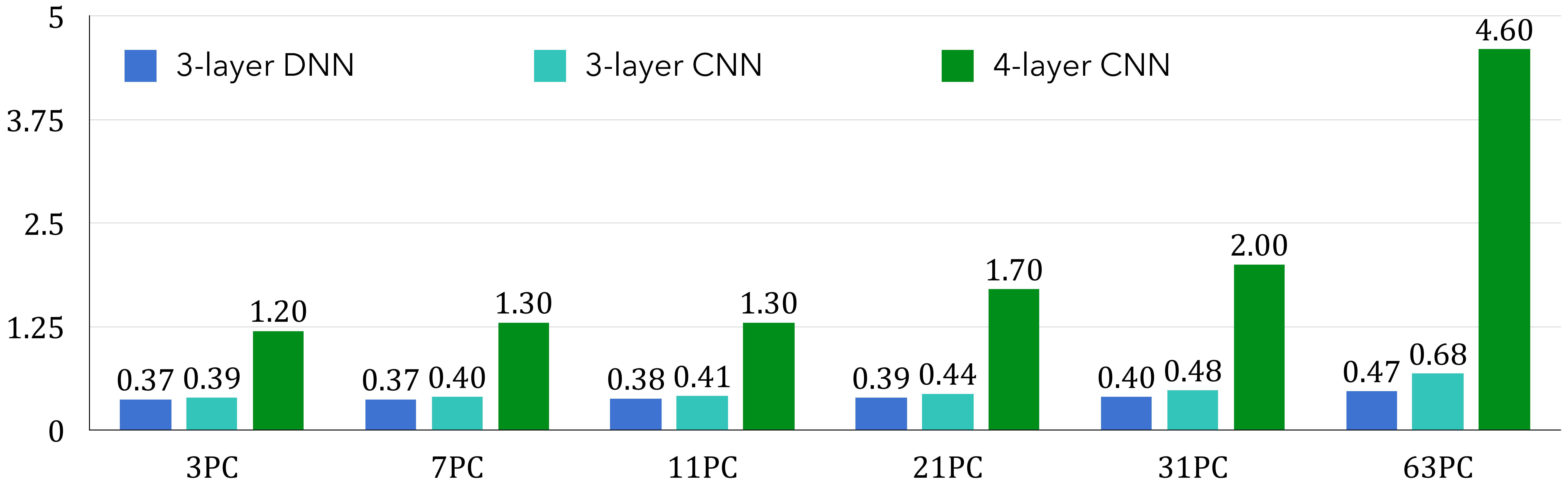
online runtime (s) from 3PC to 63PC in the LAN setting

Performance: Oblivious Inference

Stimulate 3-63 parties on 11 servers

LAN: 15Gb/s, delay 0.3ms

WAN: 100Mb/s, delay 40ms



online runtime (s) from 3PC to 63PC in the WAN setting

The End, Questions?