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

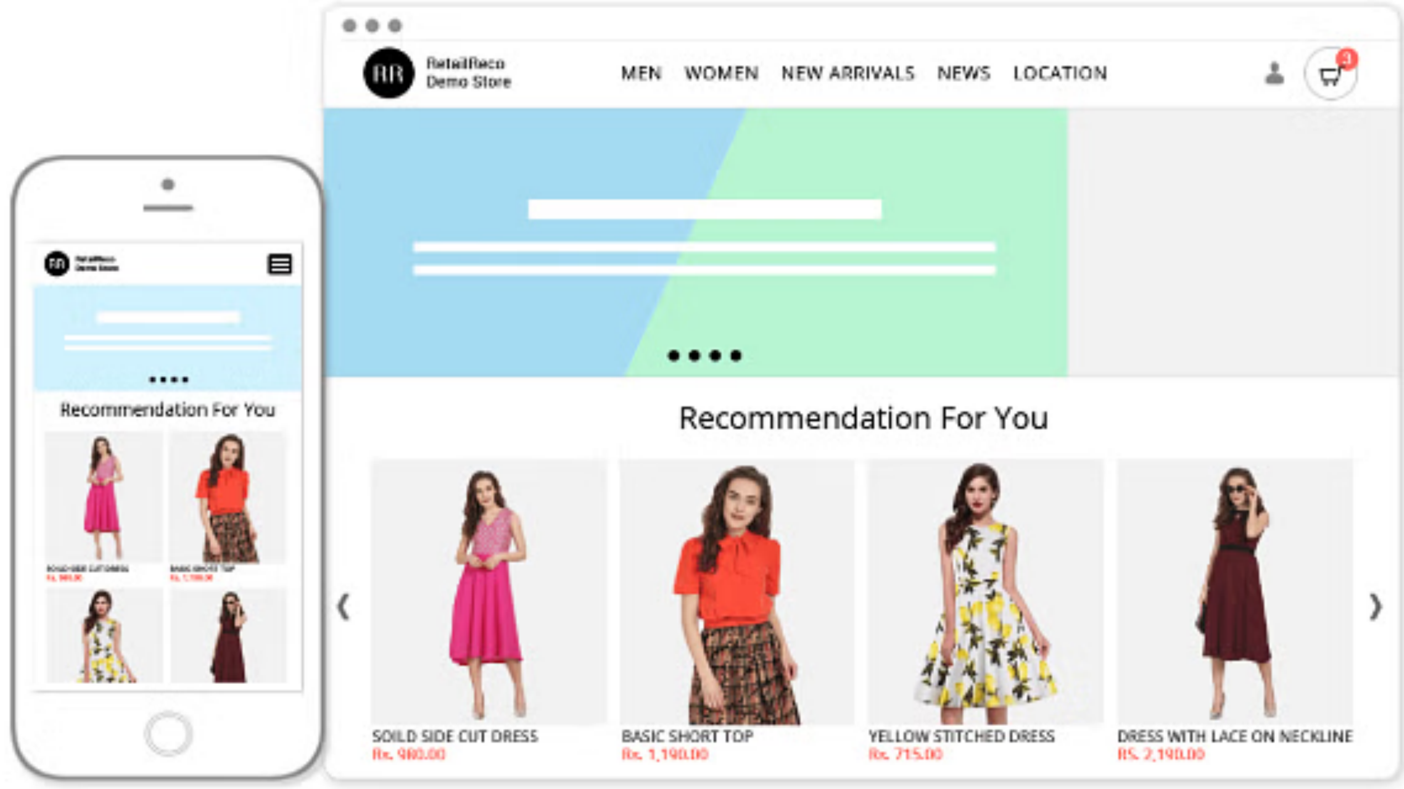
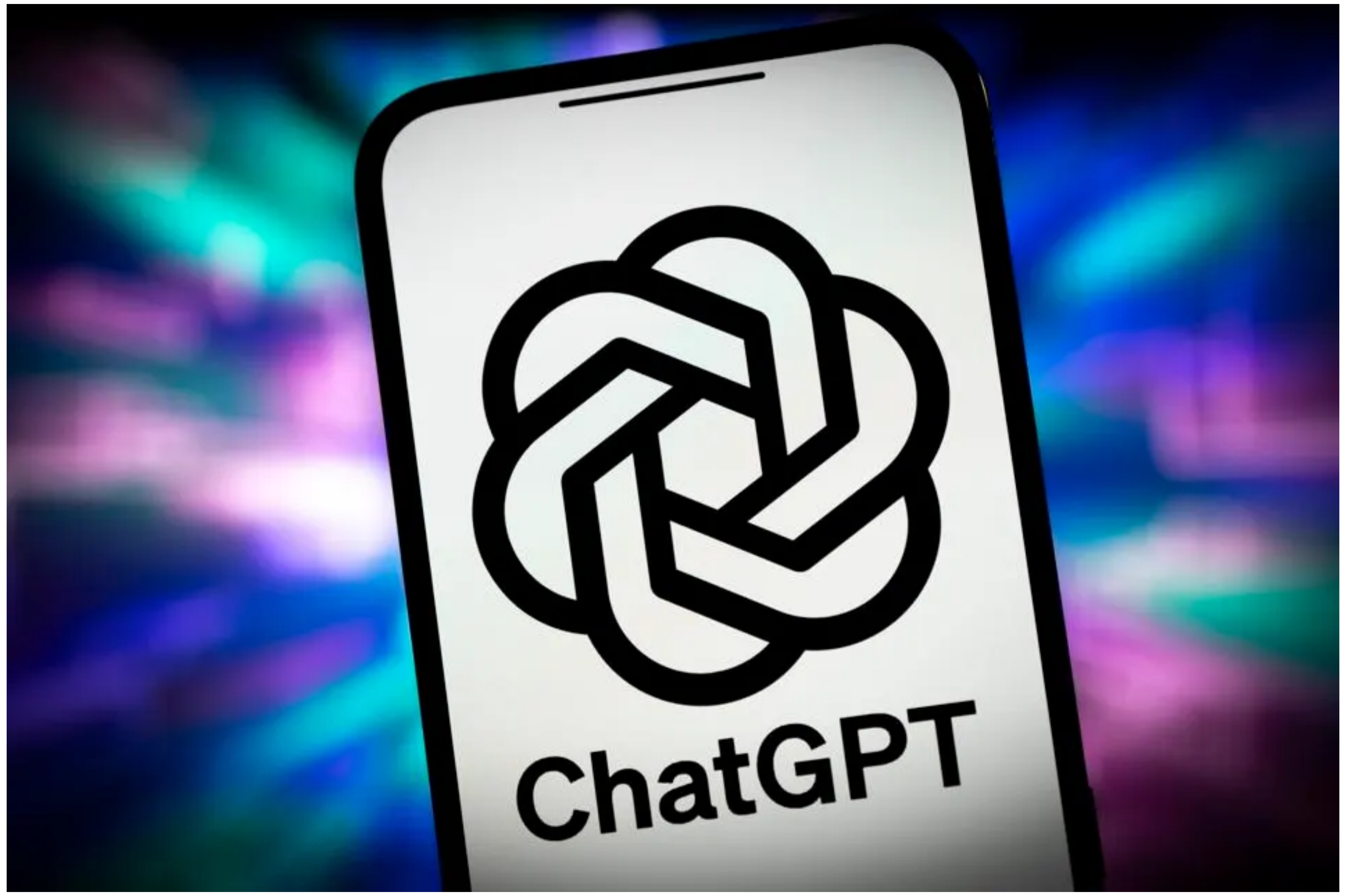


Image Processing

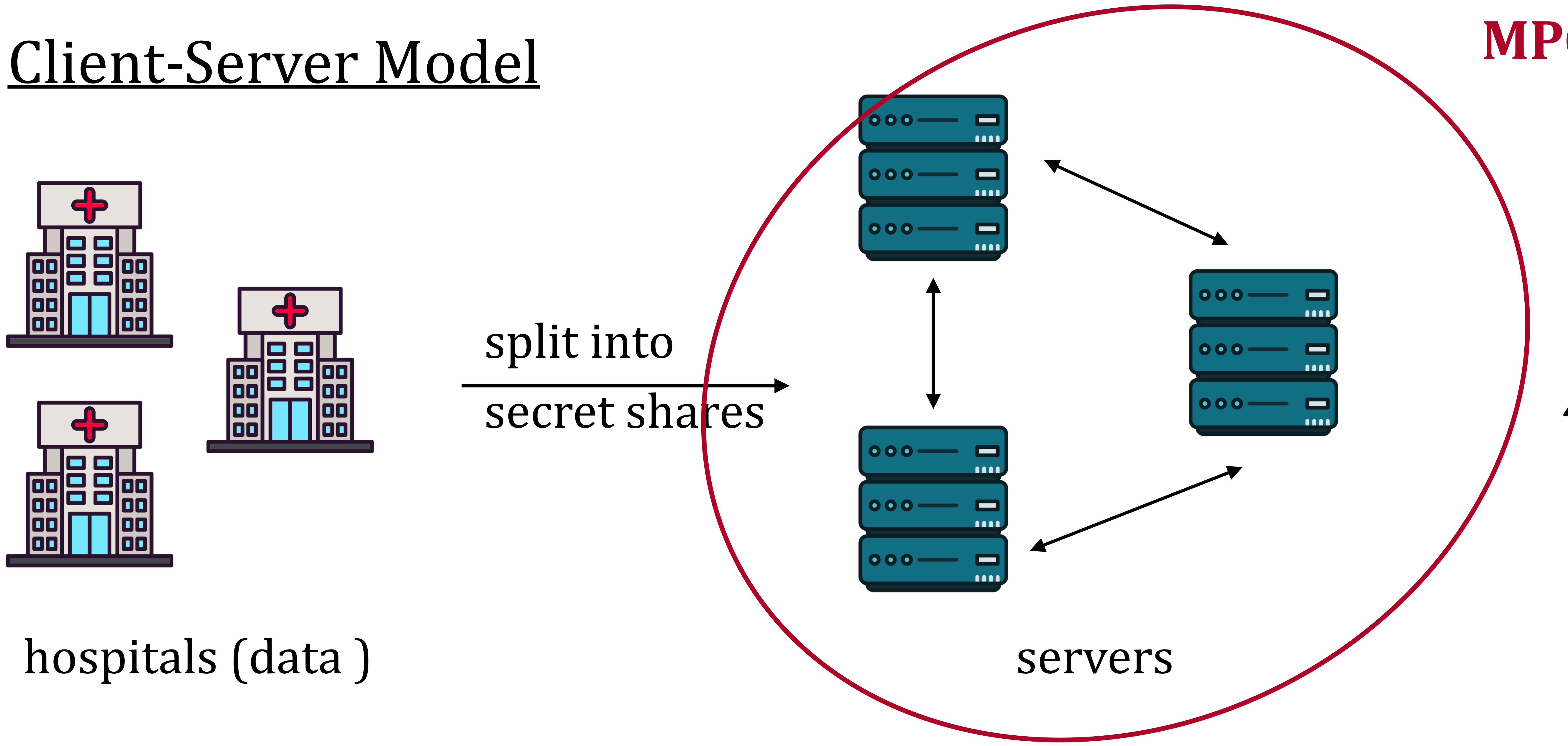
Virtual Assistants: generate human-like responses



More data —> Better models

Privacy-preserving Machine Learning (PPML)

Client-Server Model



MPC Protocols for PPML

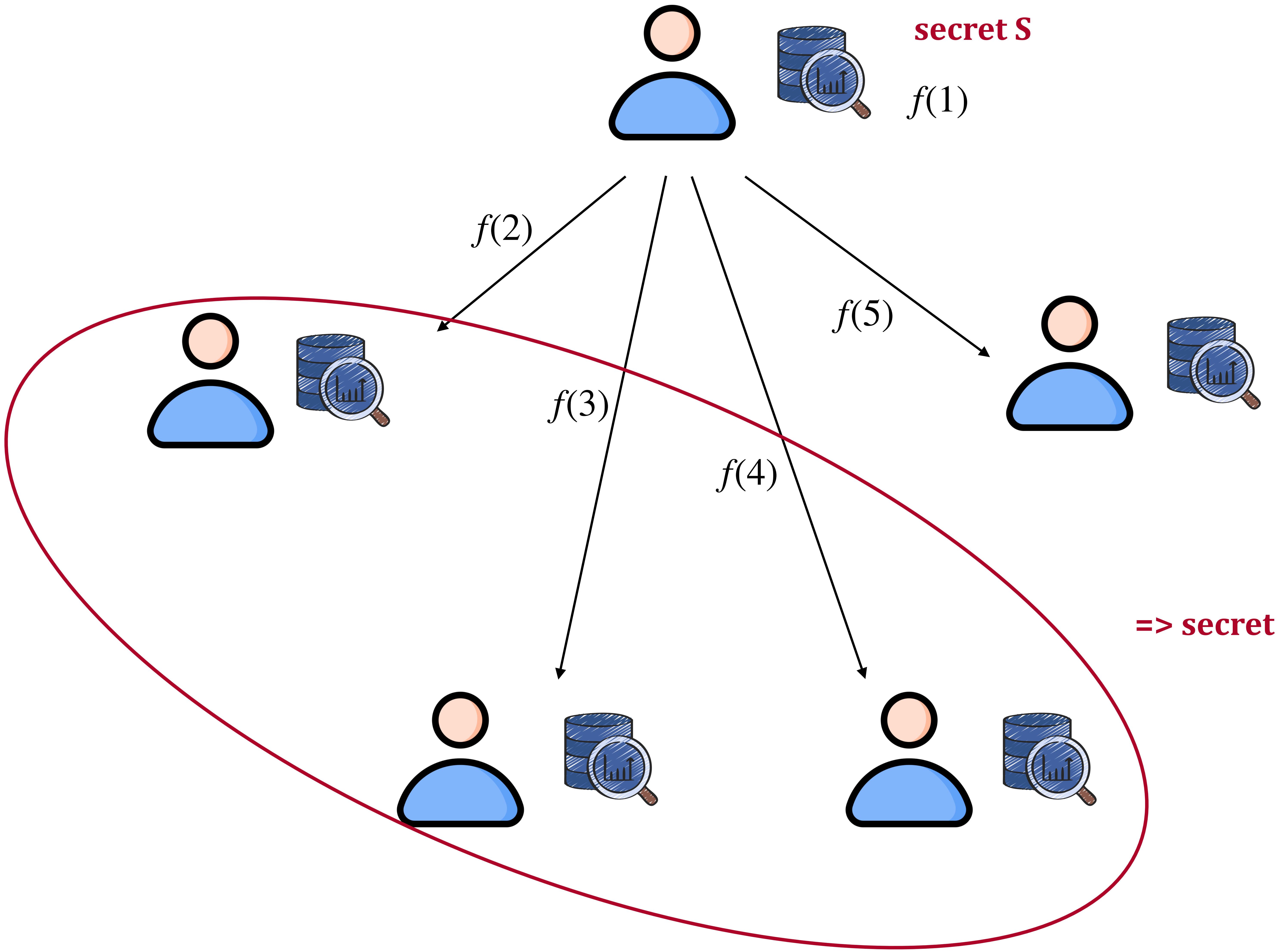
2PC: [SecureML] [Ezpc] [Chameleon] [Delphi] ...

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

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

Our work: scalable and efficient PPML protocols for any number of parties.

Scalability from Shamir's Secret Sharing



$$f(x) = s + a_1 x + a_2 x^2$$

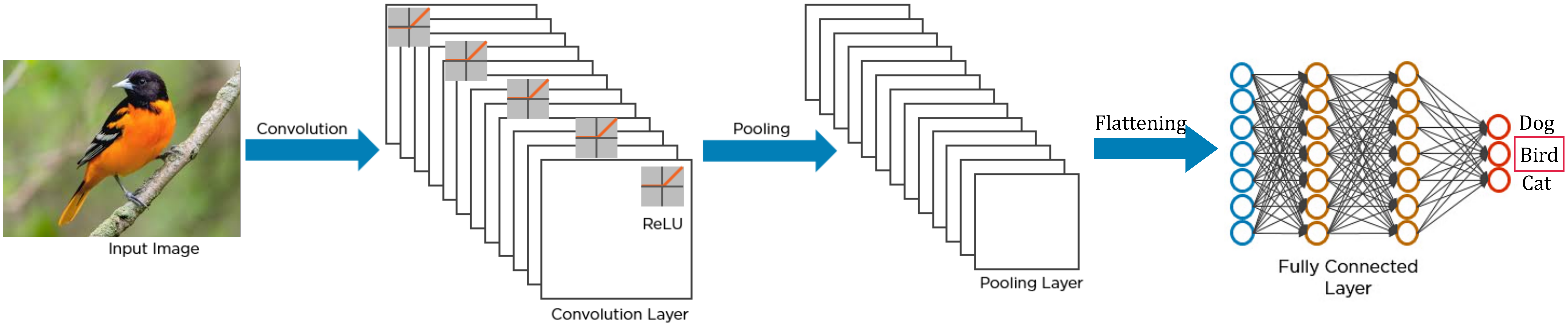
=> secret S

$$f(x) = s + a_1 x + a_2 x^2 + \ldots + a_t x^t$$

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

t-privacy

Two Challenges in Privacy-preserving Neural Networks

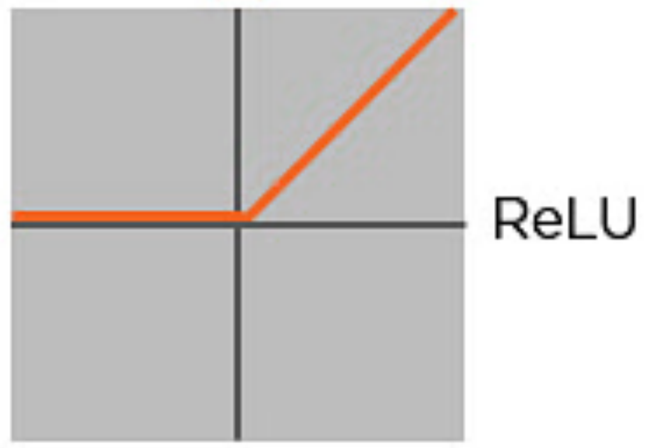


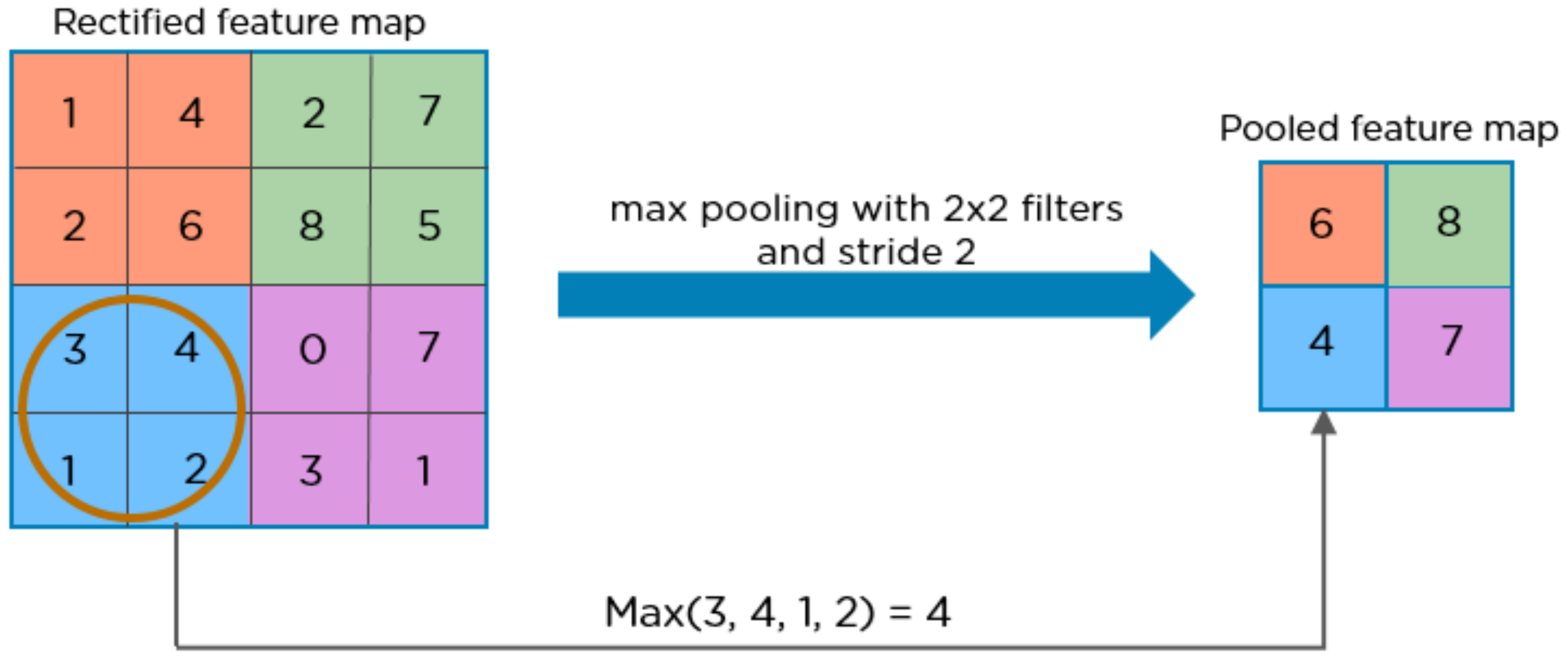
- Decimal number
- Non-linear function

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

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

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



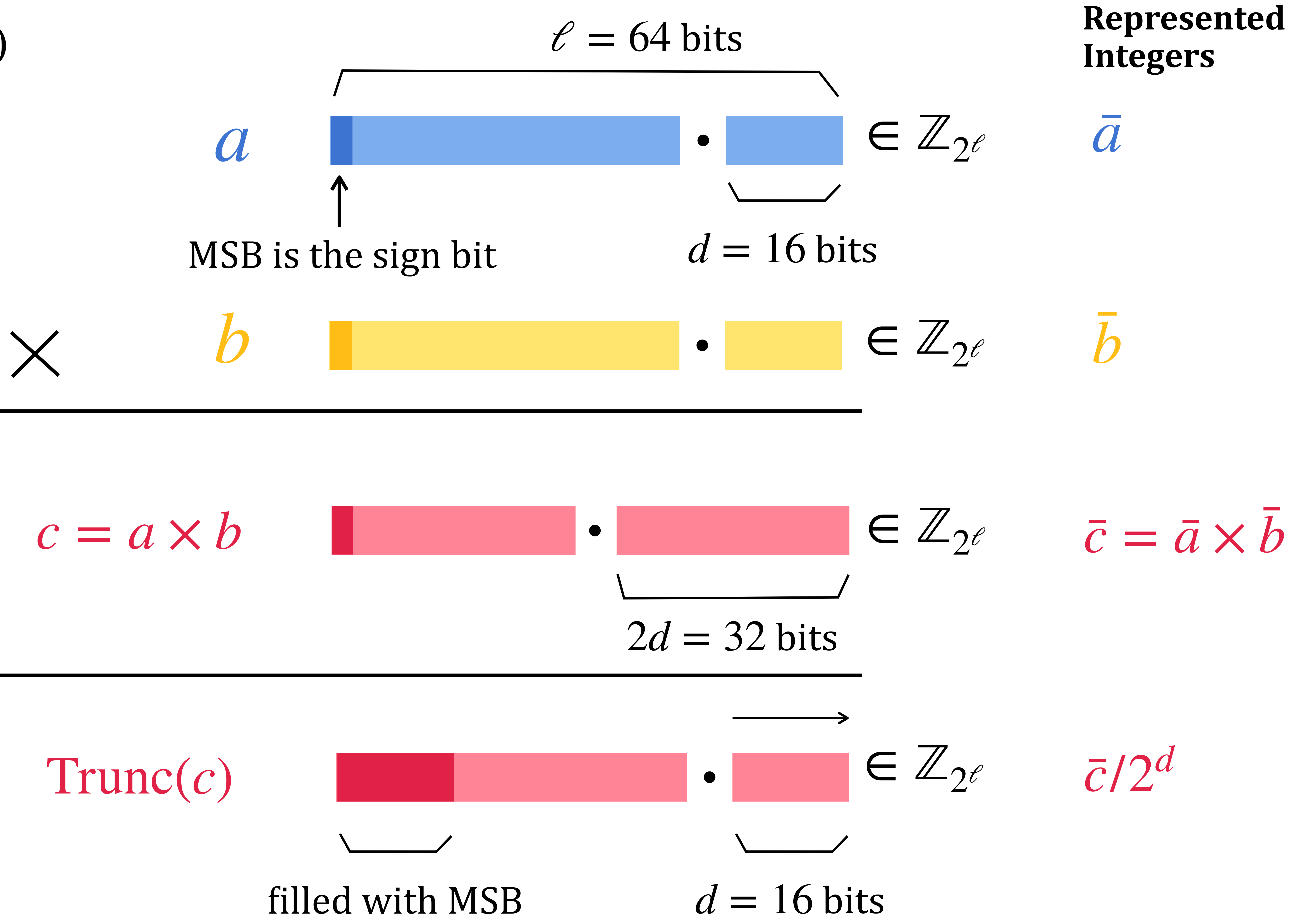


Decimal Multiplications in Integer Ring $\mathbb{Z}_{2^\ell}$

(2's complement)

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

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



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

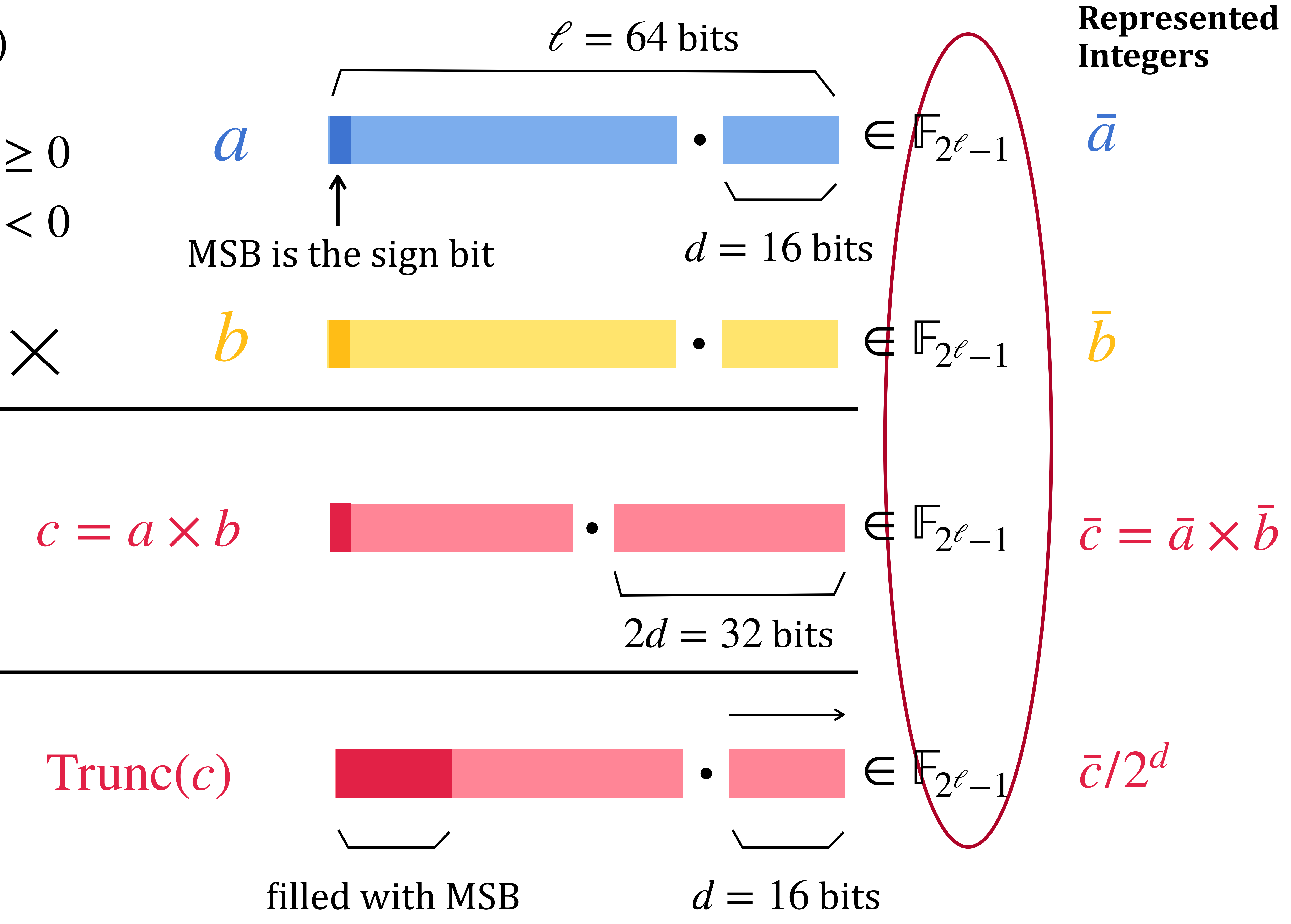
$$x = \bar{x} \pmod{2^\ell - 1} = \begin{cases} \bar{x}, & \bar{x} \geq 0 \\ 2^\ell - 1 - \bar{x}, & \bar{x} < 0 \end{cases}$$

- 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 ($[r]$, $[\text{Trunc}(r)]$) where $r \leftarrow \mathbb{F}_{2^\ell-1}$

Online: input $[x]$

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

A Large Gap !!

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

Previous Truncation Protocol with A Large 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 $\quad \bar{x} > 0$

+ $\quad$ +

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

= $\quad \neq$

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

$a = x + r$

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

(positive overflow)

truncation

d=3

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

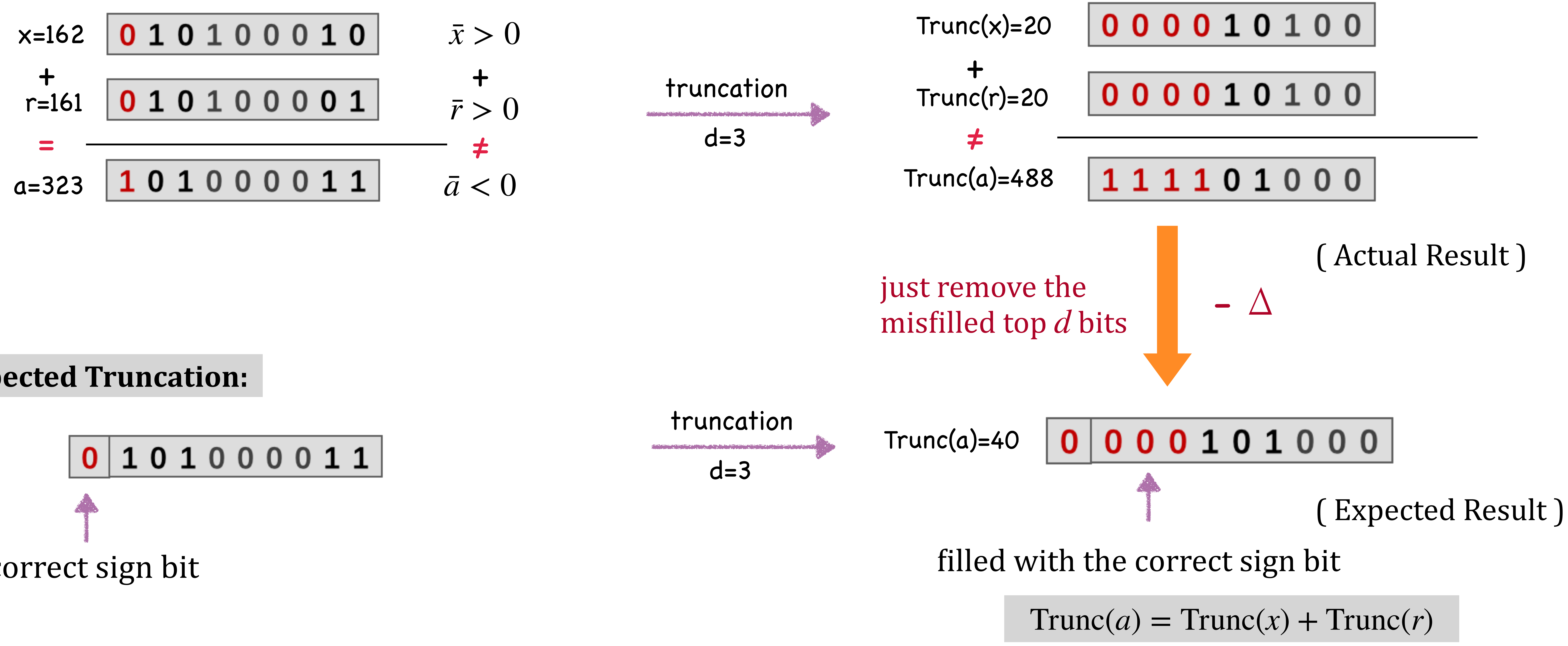
$\text{Trunc}(a) \neq \text{Trunc}(x) + \text{Trunc}(r)$

incorrect sign bit:
falsely indicates the result is negative

filled with the incorrect sign bit

Our Truncation Protocol with Only 1-bit Gap

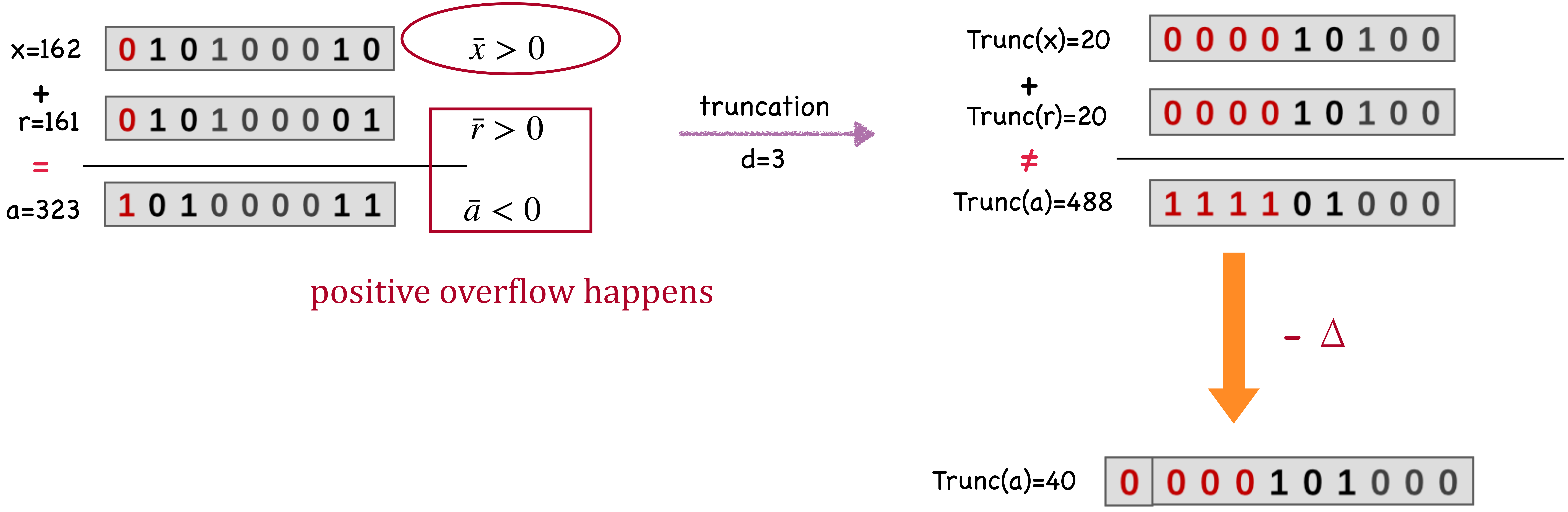
For example, we have $a = x + r$ in $\mathbb{F}_{2^9-1}$.



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 = 1-bit gap



(Truncation Result)

+ [DN07] = 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(2x) = 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 $j = 1, \ldots, \ell$

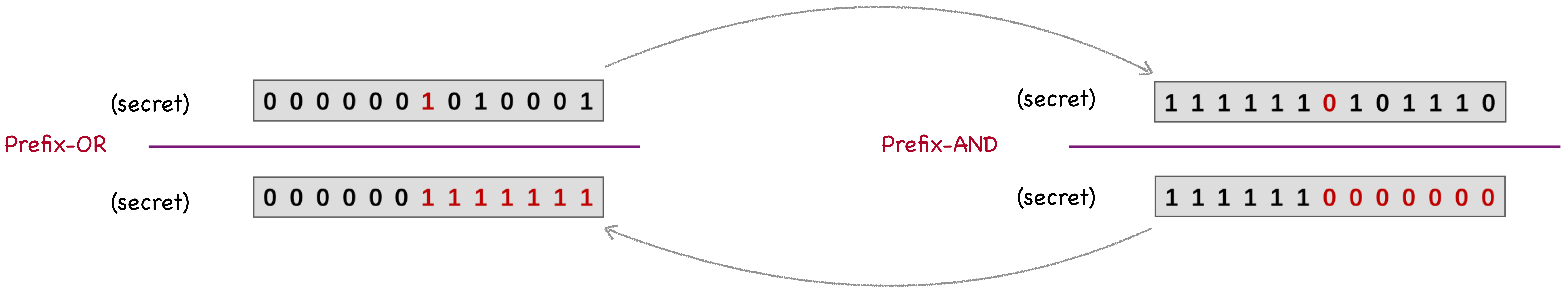
Round-Efficient Prefix-OR Protocol via Prefix-AND

Online Complexity of [NO07]: 5 rounds

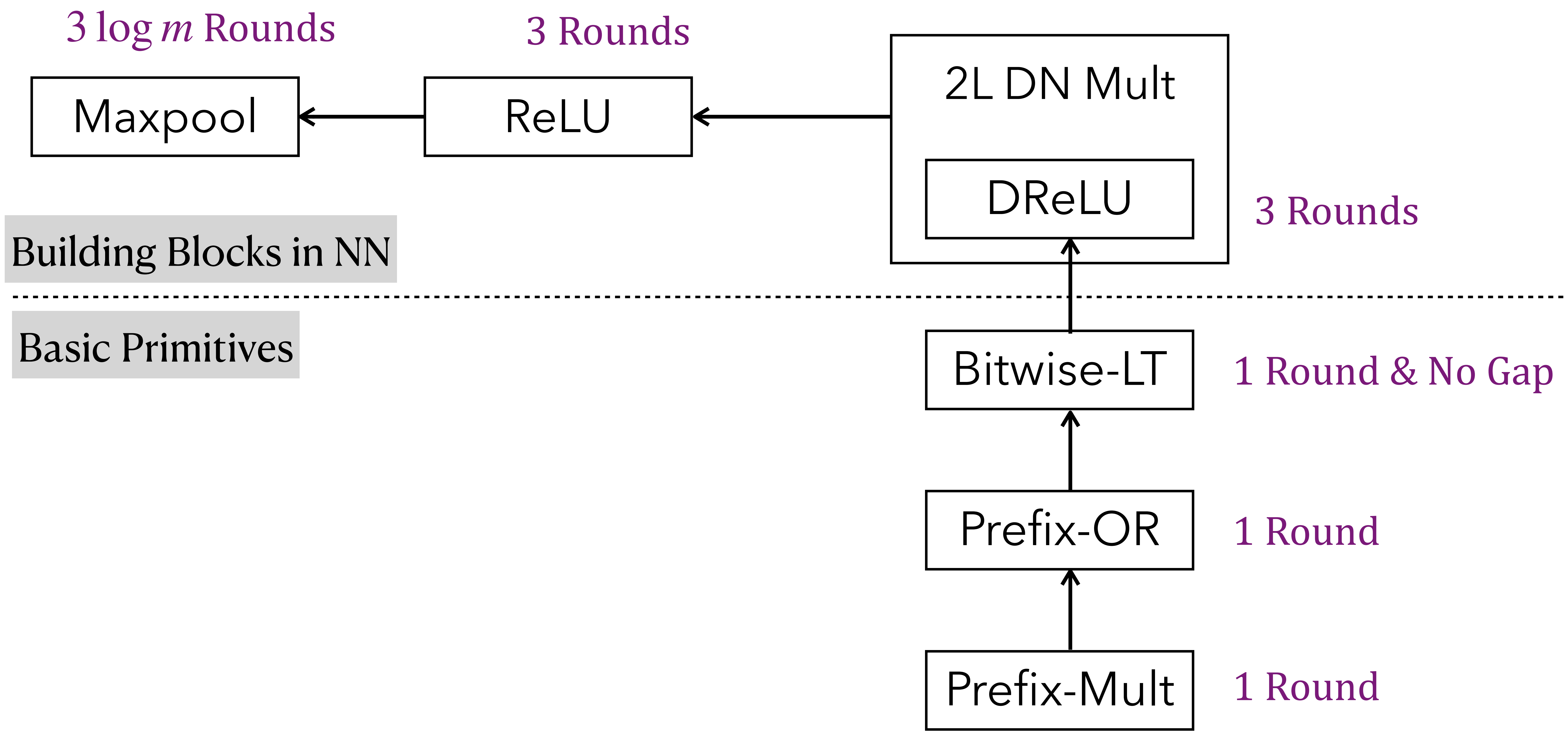
Online Complexity of Prefix-Mult[BB89]: 1 round

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

* Prefix-AND: compute $\bar{b}_j = \bigwedge_{i=1}^{j} \bar{a}_i$ for $j = 1, \ldots, \ell$



Other Building Blocks



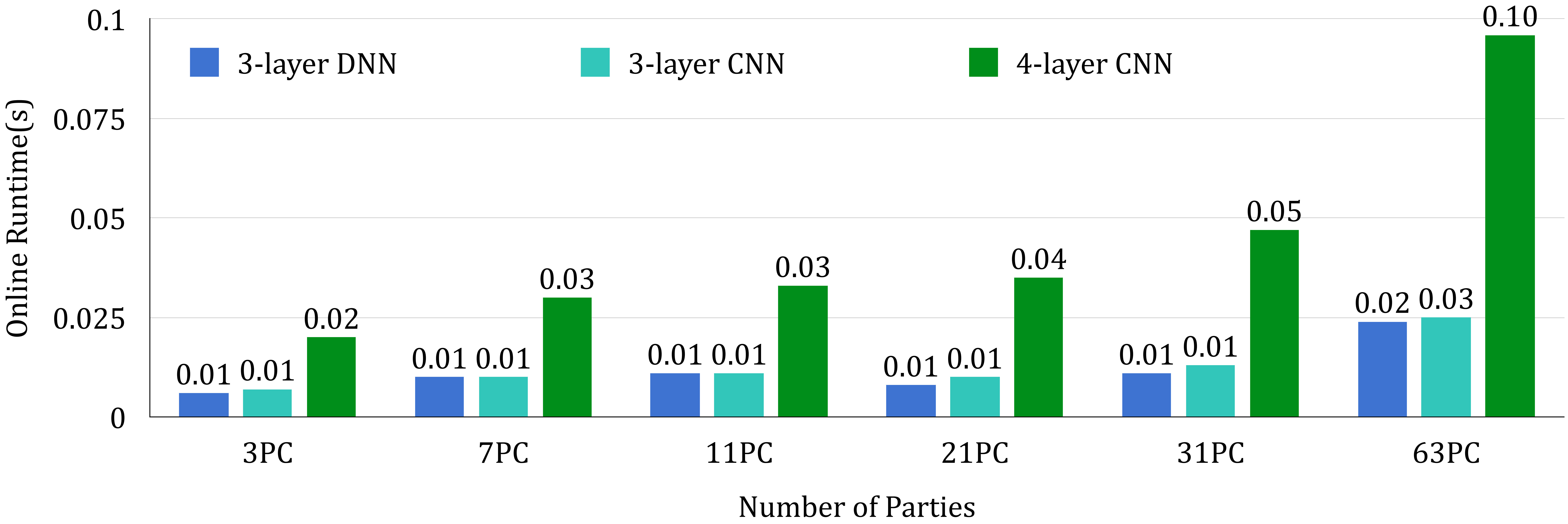
Round Complexity in Online Phase

Performance: Private Inference

Simulate 3-63 parties on 11 servers

LAN: 15Gb/s, delay 0.3ms

WAN: 100Mb/s, delay 40ms



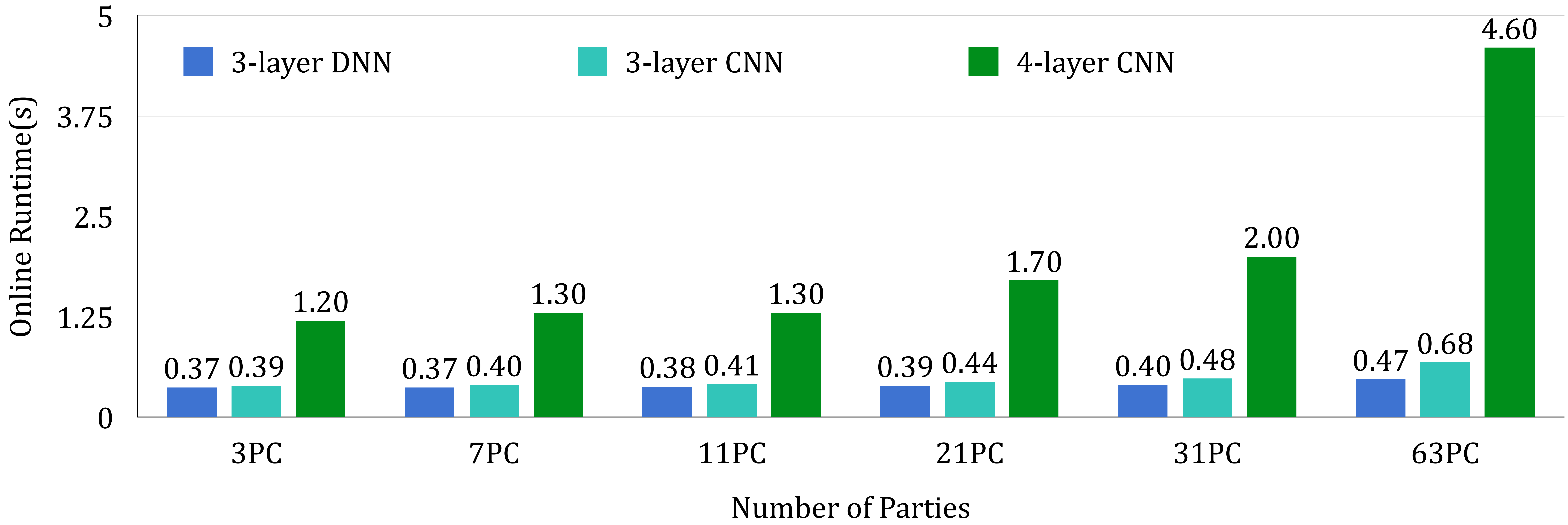
online runtime (s) from 3PC to 63PC in the LAN setting

Performance: Private Inference

Simulate 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 f7ed.com/liu

The End, Questions?