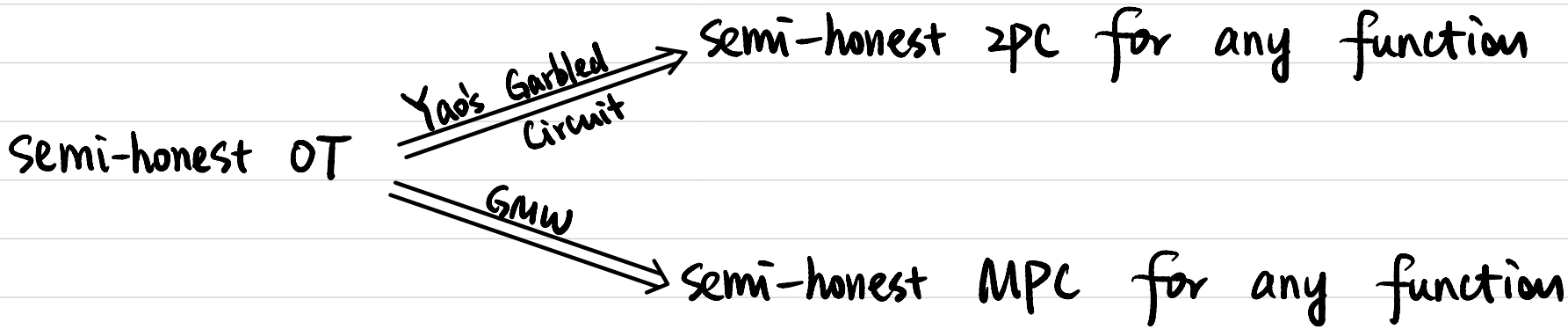
CSCI 1515 Applied Cryptography

This Lecture:

- Oblivious Transfer (OT)
- Putting it Together: Semi-Honest 2PC for Any Funtion
- GMW: Semi-Honest MPC for Any Funtion


semi-honest OT
Yao's Garbled Circuit
semi-honest 2PC for any function
GMW
semi-honest MPC for any function

Oblivious Transfer (OT)

Sender

Input: $m_0, m_1 \in \{0,1\}^\ell$

Receiver

Input: $c \in \{0,1\}$

Output: $\perp$

Output: m_c

Oblivious Transfer (OT)

Sender

Input: $m_0, m_1 \in \{0,1\}^\ell$

$a \xleftarrow{\$} \mathbb{Z}_q$

$A = g^a$ →

← $B = g^b \cdot A^c$

$k_0 := H(B^a)$

$k_1 := H((\frac{B}{A})^a)$

$Ct_0 \leftarrow Enc_{k_0}(m_0)$

$Ct_1 \leftarrow Enc_{k_1}(m_1)$ →

Receiver

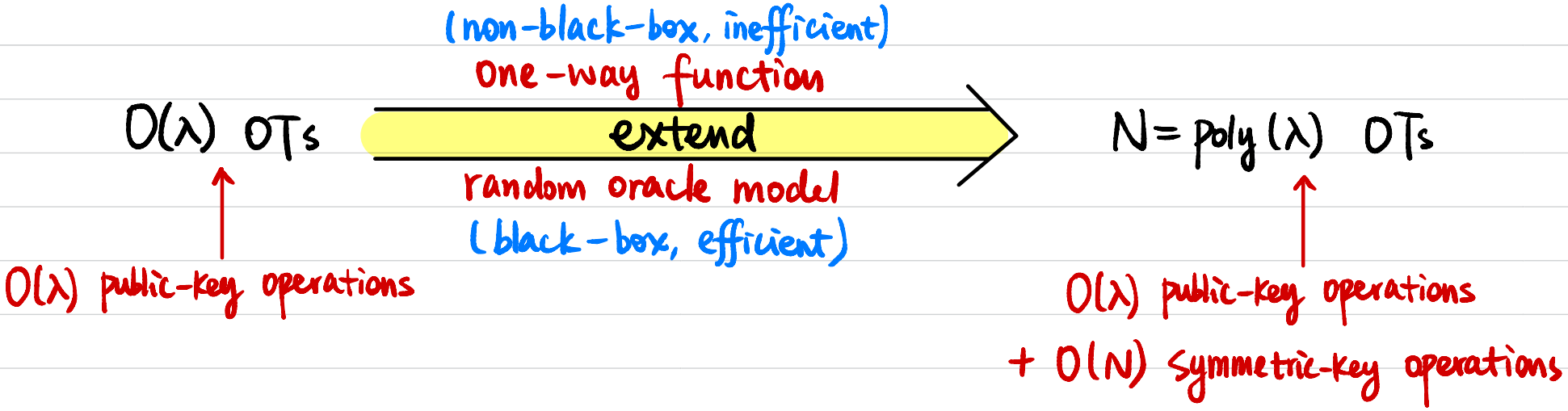
Input: $c \in \{0,1\}$

$b \xleftarrow{\$} \mathbb{Z}_q$

Output: ?

OT Extension

Can we construct OT from symmetric-key primitives only?



Putting it Together: Semi-Honest 2PC

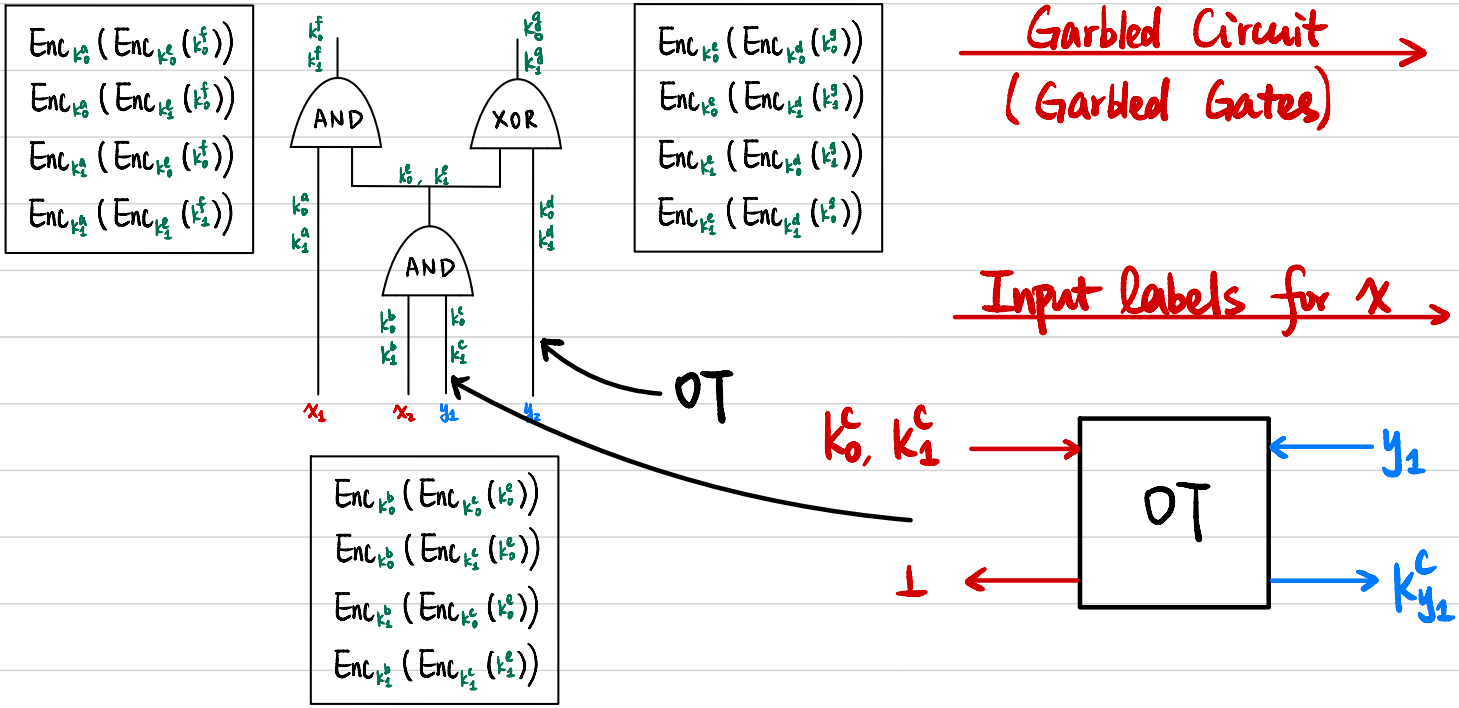
What could go wrong against malicious adversaries?

Alice (Garbler)

$x \in \{0,1\}^2$

Bob (Evaluator)

$y \in \{0,1\}^2$



Garbled Circuit (Garbled Gates)

Input labels for x

Input labels for y?

- Output labels?

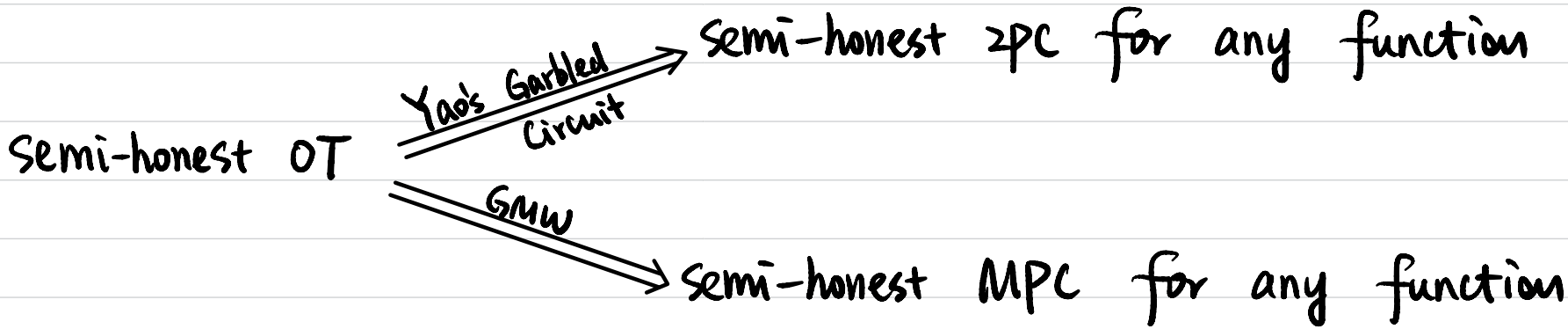
  Evaluator sends k^f, k^g back to garbler

- How to decide which ciphertext to decrypt?

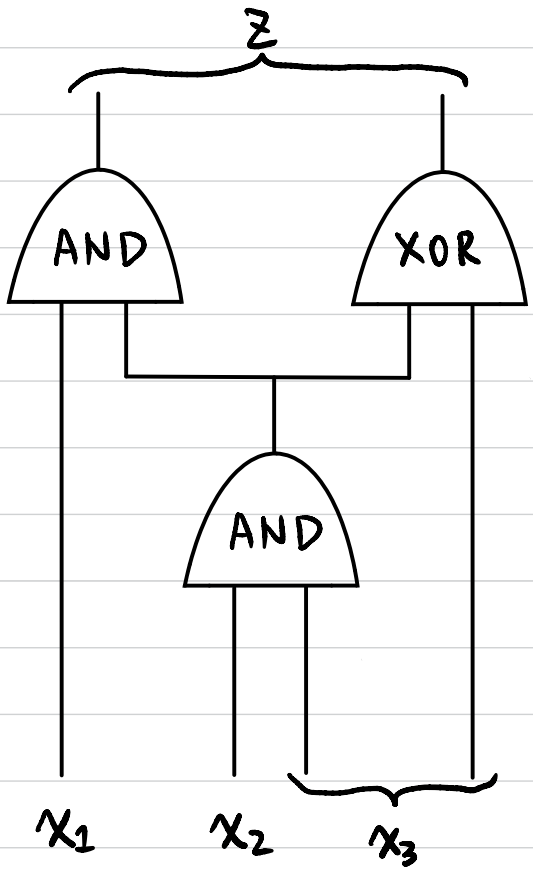
Shuffle

- $\mathrm{Enc}_{k_0^b}(\mathrm{Enc}_{k_0^c}(k_0^e \| \overbrace{0 \cdots 0}^{\lambda}))$ → garbage
- $\mathrm{Enc}_{k_0^b}(\mathrm{Enc}_{k_1^c}(k_0^e \| 0 \cdots 0))$ → $k_0^e \| 0 \cdots 0$
- $\mathrm{Enc}_{k_1^b}(\mathrm{Enc}_{k_0^c}(k_0^e \| 0 \cdots 0))$ → garbage
- $\mathrm{Enc}_{k_1^b}(\mathrm{Enc}_{k_1^c}(k_1^e \| 0 \cdots 0))$ → garbage

2λ


semi-honest OT
Yao's Garbled Circuit
semi-honest 2PC for any function
GMW
semi-honest MPC for any function

MPC for any function with $t \leq n-1$ (GMW)



Throughout the protocol, we keep the invariant:

For each wire w:

If the value of the wire is $v^w \in \{0,1\}$,

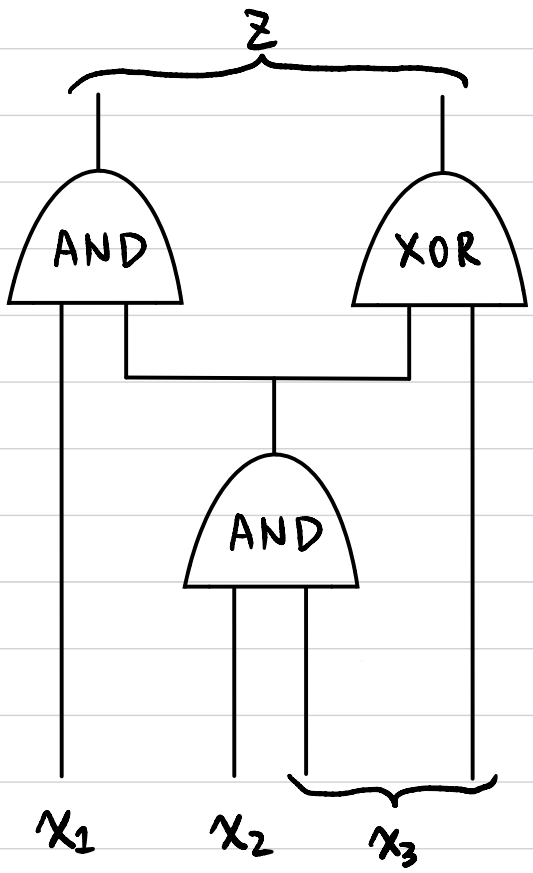
then the n parties hold an additive secret share of v^w

Each party P_i holds a random share $v_i^w \in \{0,1\}$ s.t.

$$\bigoplus_{i=1}^{n} v_i^w = v^w$$

Any $(n-1)$ shares information theoretically hide v^w.

MPC for any function with $t \leq n-1$ (GMW)



Each party P_i holds a random share $v_i^w \in \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

Inputs:

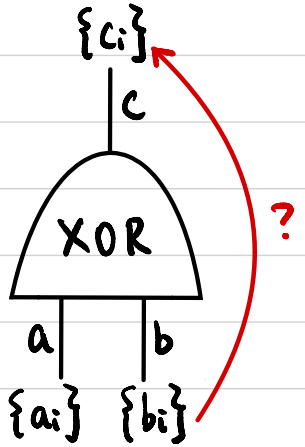
For each input wire w:

If it's from party P_k with input value $v^w \in \{0,1\}$.

P_k randomly samples $v_i^w \xleftarrow{\$} \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

↳ Sends v_i^w to party P_i.

XOR gates:

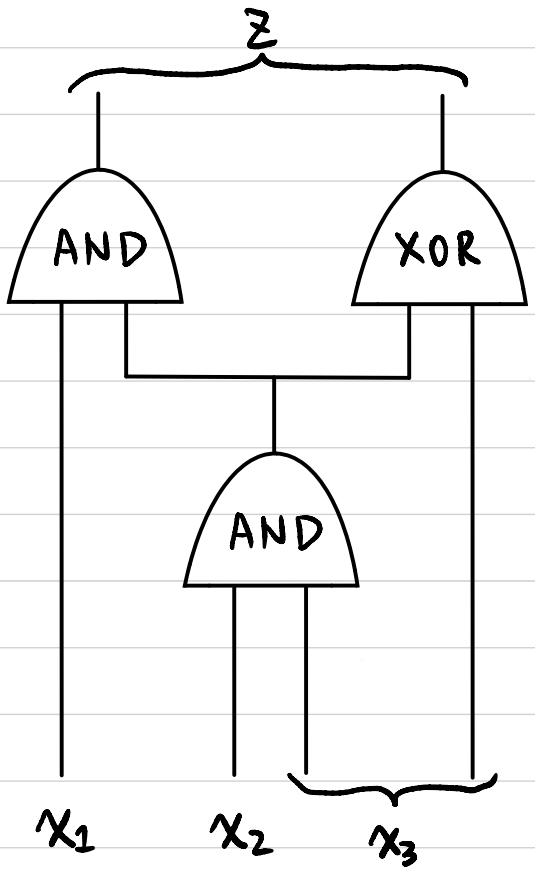


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \oplus b$

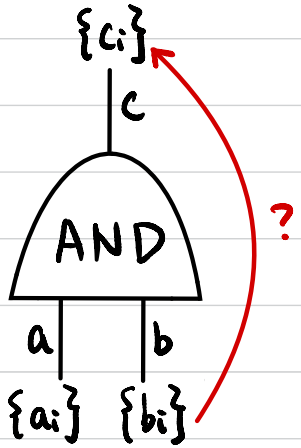
$c_i = ?$

MPC for any function with $t \leq n-1$ (GMW)



Each party P_i holds a random share $v_i^w \in \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

AND gates:



GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \cdot b$

$c_i = ?$

Outputs:

For each output wire w:

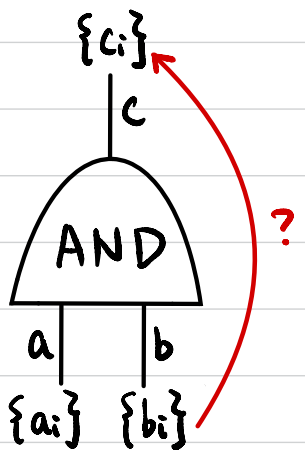
Each party P_i holds a random share $v_i^w \in \{0,1\}$

→ Sends v_i^w to all parties

Each party computes the value $v^w = \bigoplus_{i=1}^{n} v_i^w$

MPC for any function with $t \leq n-1$ (GMW)

AND gates:



GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\quad \bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \cdot b$

$c_i = ?$

$$a \cdot b = \left(\sum_{i=1}^{n} a_i\right) \cdot \left(\sum_{i=1}^{n} b_i\right) \mod 2$$

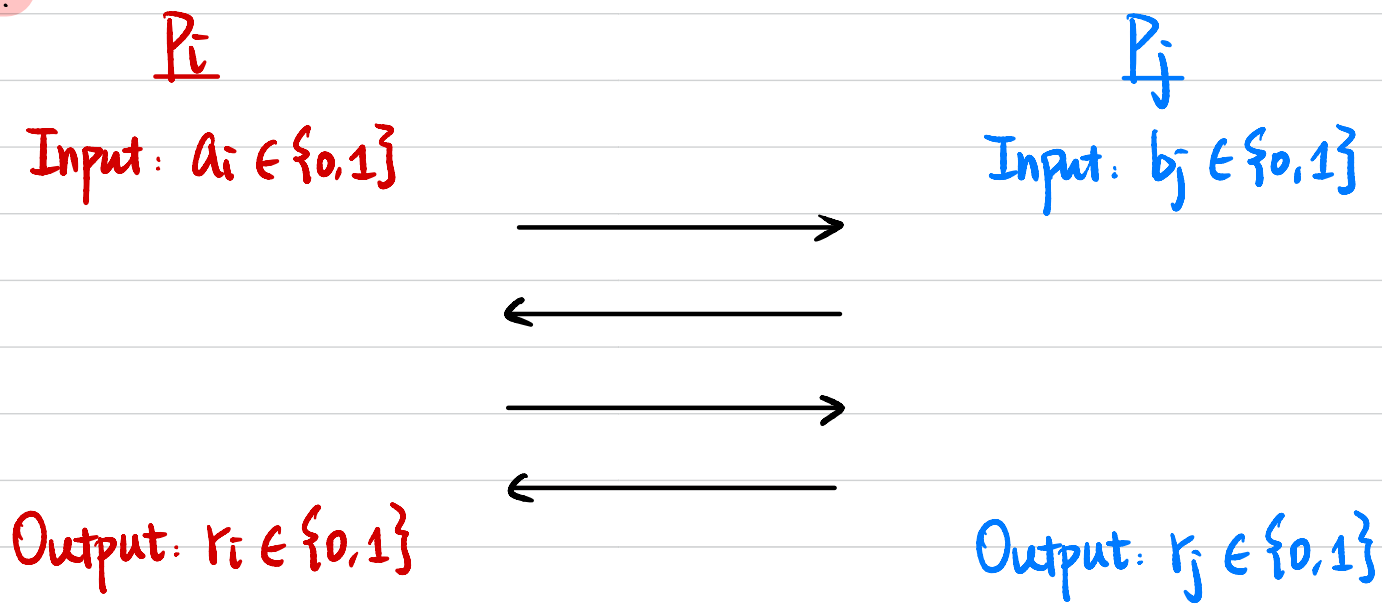
$$= \left(\sum_{i=1}^{n} a_i \cdot b_i\right) + \left(\sum_{i \neq j} a_i \cdot b_j\right) \mod 2$$

$a_i \cdot b_i$: P_i locally

$a_i \cdot b_j$: ?

MPC for any function with $t \leq n-1$ (GMW)

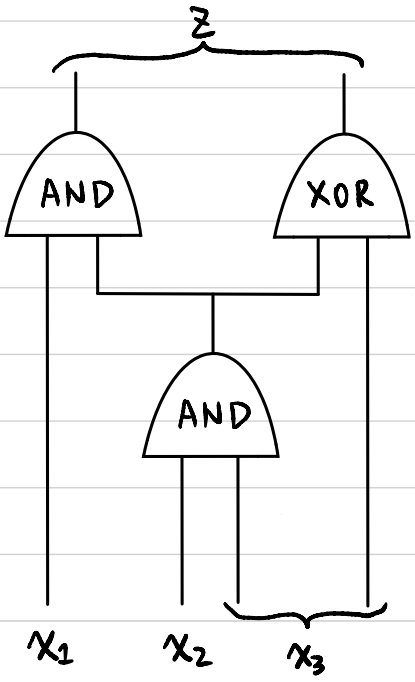
Reshare:



WANT: Random $r_i, r_j \in \{0,1\}$ s.t. $r_i + r_j = a_i \cdot b_j \pmod 2$

1) P_i randomly samples $r_i \xleftarrow{\$} \{0,1\}$
2) How to let P_j learn r_j s.t. $r_i + r_j = a_i \cdot b_j \pmod 2$?

MPC for any function with $t \leq n-1$ (GMW)



Each party P_i holds a random share $v_i^w \in \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

Inputs:

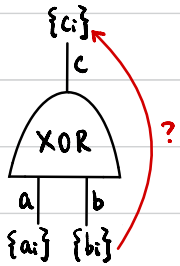
For each input wire w:

If it's from party P_k with input value $v^w \in \{0,1\}$.

P_k randomly samples $v_i^w \xleftarrow{\$} \{0,1\}$ s.t. $\bigoplus_{i=1}^{n} v_i^w = v^w$

↳ Sends v_i^w to party P_i.

XOR gates:

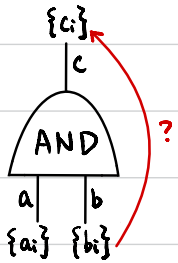


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \oplus b$

$c_i = a_i \oplus b_i$

AND gates:



GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^{n} c_i = c = a \cdot b$

$c_i = ?$

$$a \cdot b = \left(\sum_{i=1}^{n} a_i\right) \cdot \left(\sum_{i=1}^{n} b_i\right) \mod 2$$

$$= \left(\sum_{i=1}^{n} a_i \cdot b_i\right) + \left(\sum_{i \neq j} a_i \cdot b_j\right) \mod 2$$

P_i locally — Reshare

Outputs:

For each output wire w:

Each party P_i holds a random share $v_i^w \in \{0,1\}$

↳ Sends v_i^w to all parties

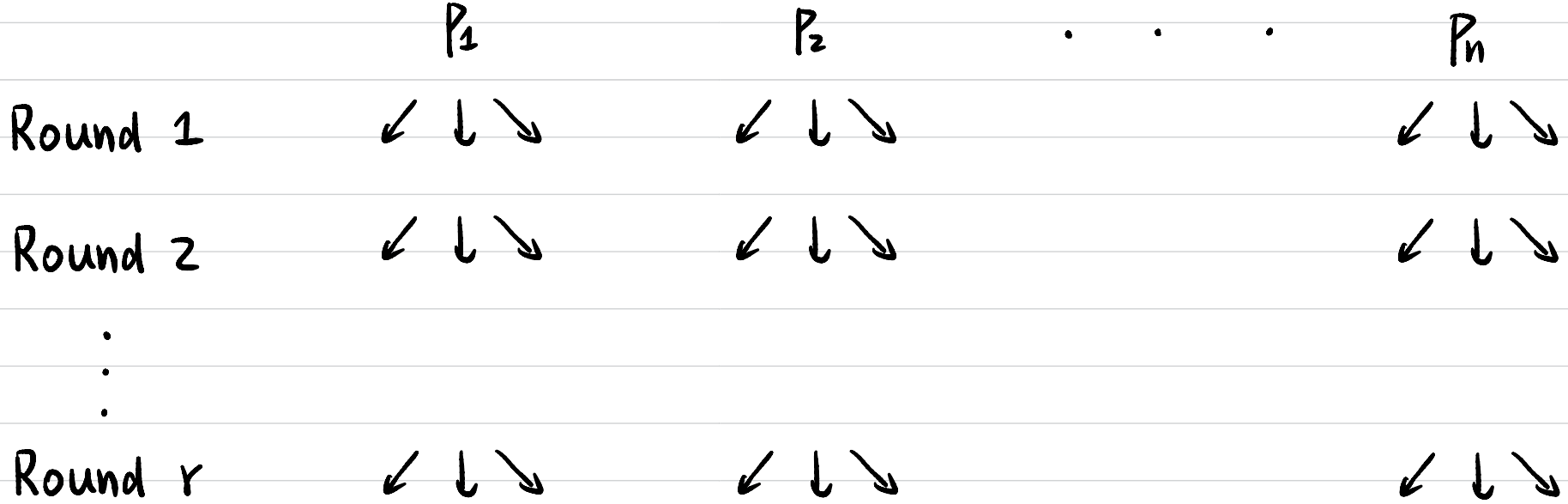
Each party computes the value $v^w = \bigoplus_{i=1}^{n} v_i^w$

MPC for any function with $t \leq n-1$ (GMW)

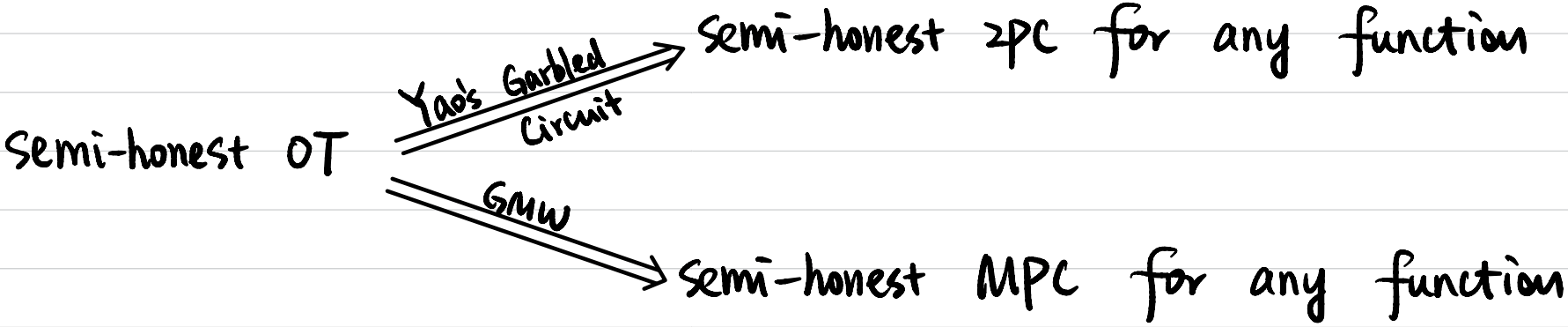
Computational Complexity?

Communication Complexity?

Round Complexity?



What could go wrong against malicious adversaries?



How to compare ?

Yao's Garbled Circuit

Goldreich-Micali-Wigderson