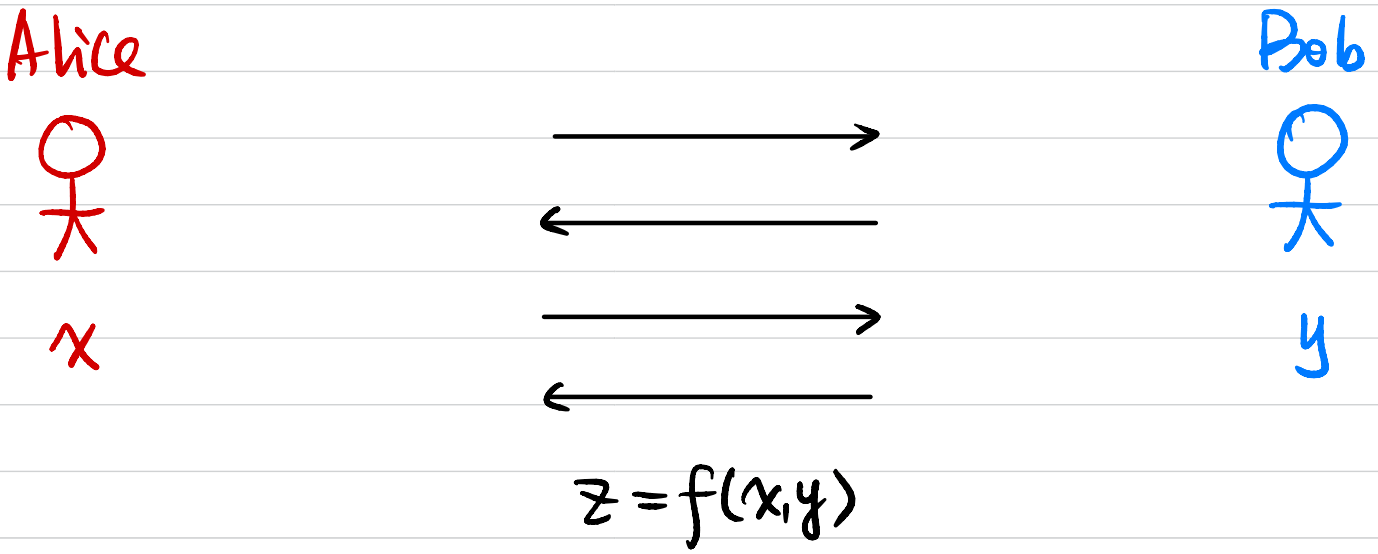
CSCI 1515 Applied Cryptography

This Lecture:

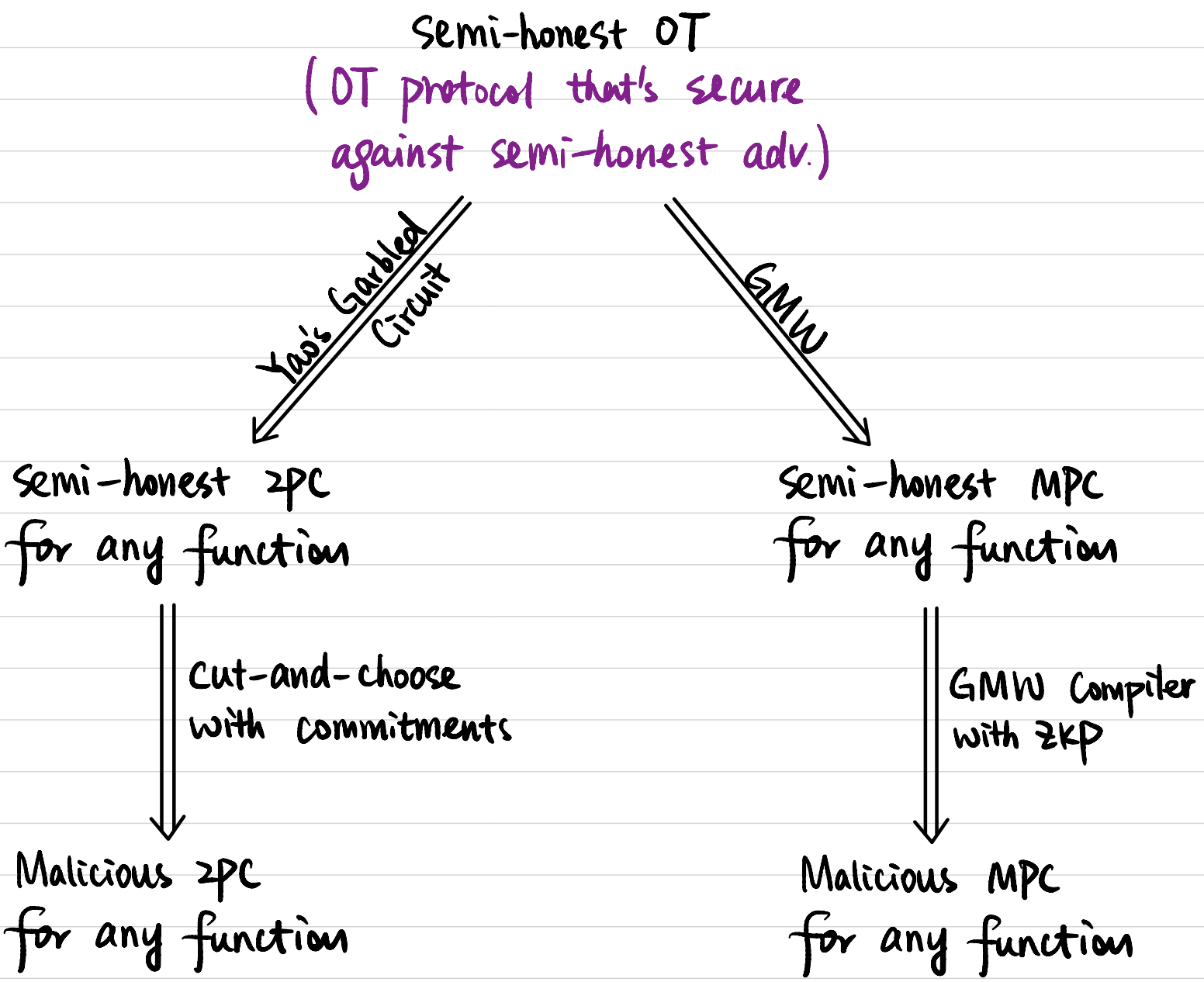
- Optimizations of Garbled Circuits (continued)
- Cut-and-Choose for Garbled Circuits
- GMW: Semi-Honest MPC for Any Function

Secure Two-Party Computation (2PC)



Allowed adversarial behavior:

- **Semi-honest:** Follow the protocol description honestly, but try to extract more information by inspecting transcript.
- **Malicious:** Can deviate arbitrarily from the protocol description.


Semi-honest OT
(OT protocol that's secure against semi-honest adv.)
Yao's Garbled Circuit
GMW
Semi-honest 2PC for any function
Semi-honest MPC for any function
Cut-and-choose with commitments
GMW Compiler with ZKP
Malicious 2PC for any function
Malicious MPC for any function

Putting it All Together: Semi-Honest 2PC

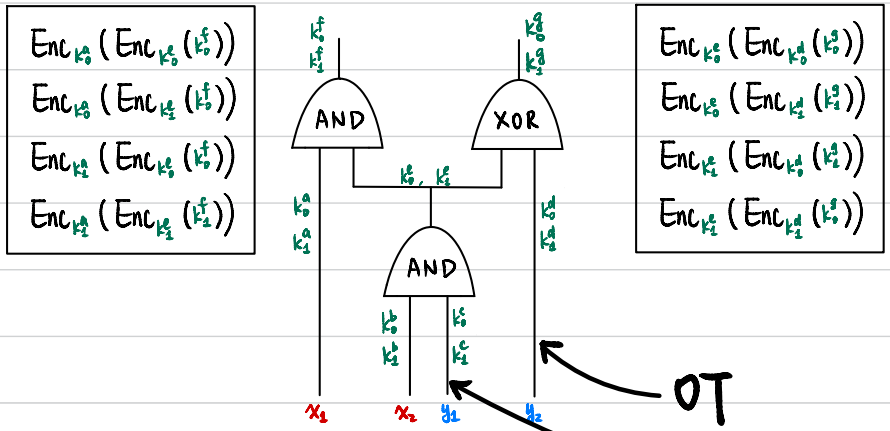
What could go wrong against malicious adversaries?

Alice (Garbler)

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

Bob (Evaluator)

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



$\mathrm{Enc}_{k_0^b}(\mathrm{Enc}_{k_0^c}(k_0^e))$
$\mathrm{Enc}_{k_0^b}(\mathrm{Enc}_{k_1^c}(k_0^e))$
$\mathrm{Enc}_{k_1^b}(\mathrm{Enc}_{k_0^c}(k_0^e))$
$\mathrm{Enc}_{k_1^b}(\mathrm{Enc}_{k_1^c}(k_1^e))$

Shuffle:
- $H(k_0^b \| k_0^c) \oplus (k_0^e \| 0\cdots0)$ → garbage
- $H(k_0^b \| k_1^c) \oplus (k_0^e \| 0\cdots0)$ → garbage
- $H(k_1^b \| k_0^c) \oplus (k_0^e \| 0\cdots0)$ → $k_0^e \| 0\cdots0$
- $H(k_1^b \| k_1^c) \oplus (k_1^e \| 0\cdots0)$ → garbage

(256, 128, 128)

$\mathrm{Enc}_{k_0^a}(\mathrm{Enc}_{k_0^e}(k_0^g))$
$\mathrm{Enc}_{k_0^a}(\mathrm{Enc}_{k_1^e}(k_1^g))$
$\mathrm{Enc}_{k_1^a}(\mathrm{Enc}_{k_0^e}(k_1^g))$
$\mathrm{Enc}_{k_1^a}(\mathrm{Enc}_{k_1^e}(k_0^g))$

Garbled Circuit (Garbled Gates) →

Input labels for x →

k_0^c, k_1^c → OT ← y_1

OT → $k_{y_1}^c$

$\perp$ ←

Input labels for y: OT

← Output labels k^f, k^g

Output z →

Communication cost?

$256 \cdot 4 \cdot |C|$

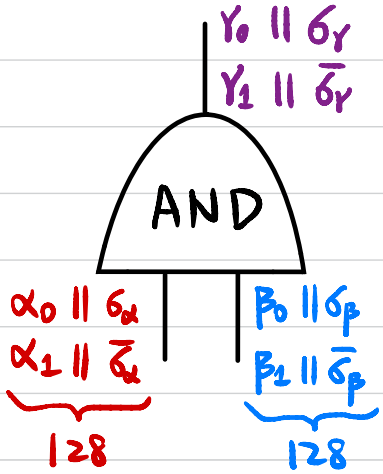
Computation cost?

G&E: $4 \cdot |C|$

Optimization 1: Point-and-Permute

4 · 128 bits / gate

$\sigma_\alpha, \sigma_\beta, \sigma_\gamma \xleftarrow{\$} \{0,1\}$
(signal bits)



$$\sigma_\alpha \;\; \sigma_\beta : \mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_0}(\overbrace{Y_0}^{127} \| \overbrace{\sigma_\gamma}^{1}))$$

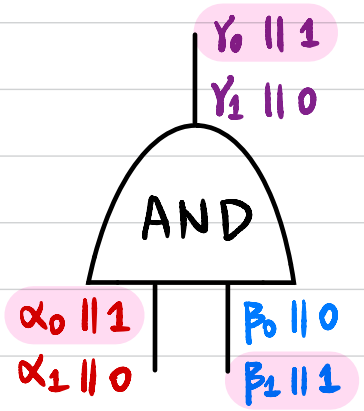
$$\sigma_\alpha \;\; \bar{\sigma}_\beta : \mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_1}(Y_0 \| \sigma_\gamma))$$

$$\bar{\sigma}_\alpha \;\; \sigma_\beta : \mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_0}(Y_0 \| \sigma_\gamma))$$

$$\bar{\sigma}_\alpha \;\; \bar{\sigma}_\beta : \mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_1}(Y_1 \| \bar{\sigma}_\gamma))$$

Example:

$\sigma_\alpha = 1 \quad \sigma_\beta = 0 \quad \sigma_\gamma = 1$



$$0 \;\; 0 : \mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_0}(Y_0 \| 1))$$

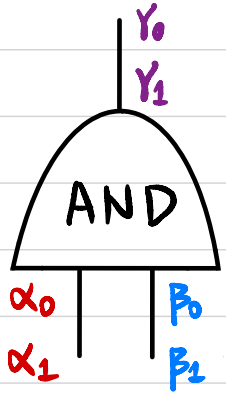
$$0 \;\; 1 : \mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_1}(Y_1 \| 0))$$

$$1 \;\; 0 : \mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_0}(Y_0 \| 1))$$

$$1 \;\; 1 : \mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_1}(Y_0 \| 1))$$

Optimization 2: Row Reduction

3 · 128 bits/gate



$\cancel{H(\alpha_0 \| \beta_0) \oplus Y_0} := 0$

$H(\alpha_0 \| \beta_1) \oplus Y_0$

$H(\alpha_1 \| \beta_0) \oplus Y_0$

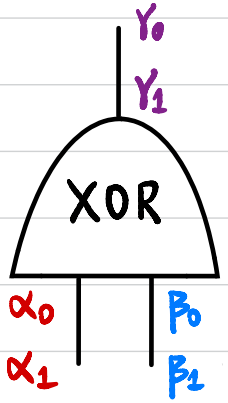
$H(\alpha_1 \| \beta_1) \oplus Y_1$

↑

Random Oracle

$Y_0 := H(\alpha_0 \| \beta_0)$

Optimization 3: Free XOR



hidden to evaluator

Sample a global $\Delta \xleftarrow{\$} \{0,1\}^\lambda$

$\alpha_1 := \alpha_0 \oplus \Delta$

$\beta_1 := \beta_0 \oplus \Delta$

$Y_1 := Y_0 \oplus \Delta$

$Y_0 := \alpha_0 \oplus \beta_0$

Other Optimizations:

- Half-gates: 2λ bits per AND gate + free XOR
- Three halves: $\sim 1.5\lambda$ bits per AND gate + free XOR

Cut-and-Choose for Garbled Circuits

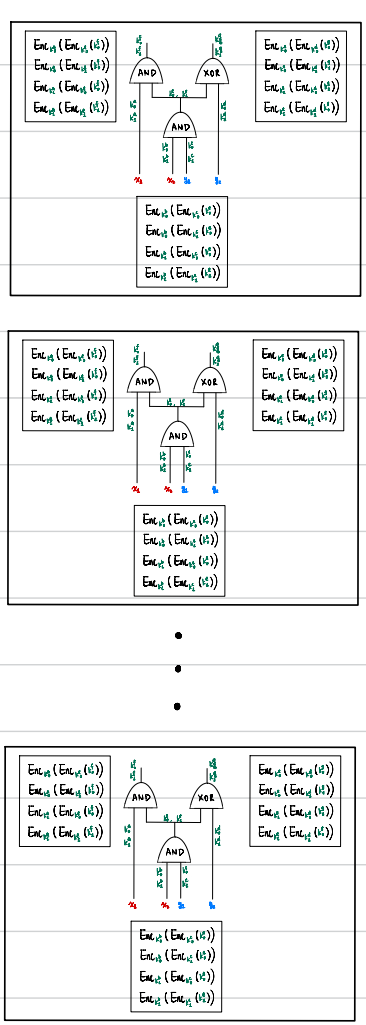
Alice (Garbler)

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

Bob (Evaluator)

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

λ



λ Garbled Circuits (Garbled Gates) →

← Randomly pick $(\lambda-1)$ of them

Reveal all the labels for the $(\lambda-1)$ GCs →

Verify correctness

↓

Continue with the remaining GC

If Alice generated 1 GC incorrectly,

Pr[Alice passes verification] = ?

Cut-and-Choose for Garbled Circuits

Alice (Garbler)

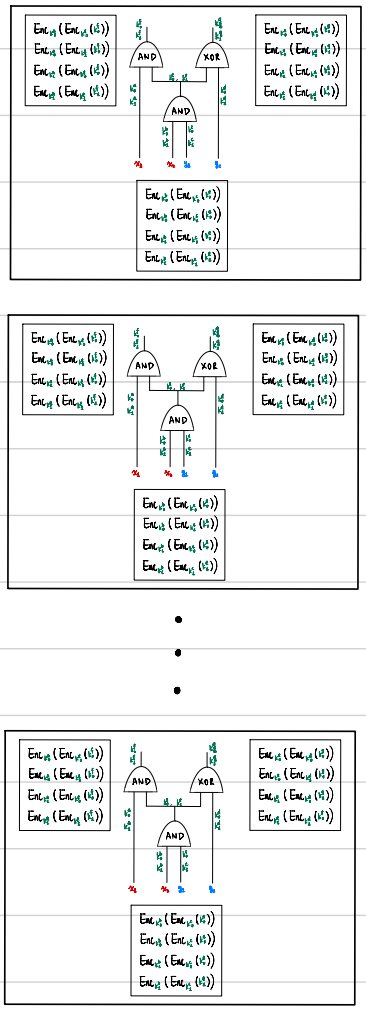
$x \in \{0,1\}^2$

Bob (Evaluator)

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

2λ Garbled Circuits (Garbled Gates) →

2λ



← Randomly pick λ of them

Reveal all the labels for the λ GCs →

Verify correctness

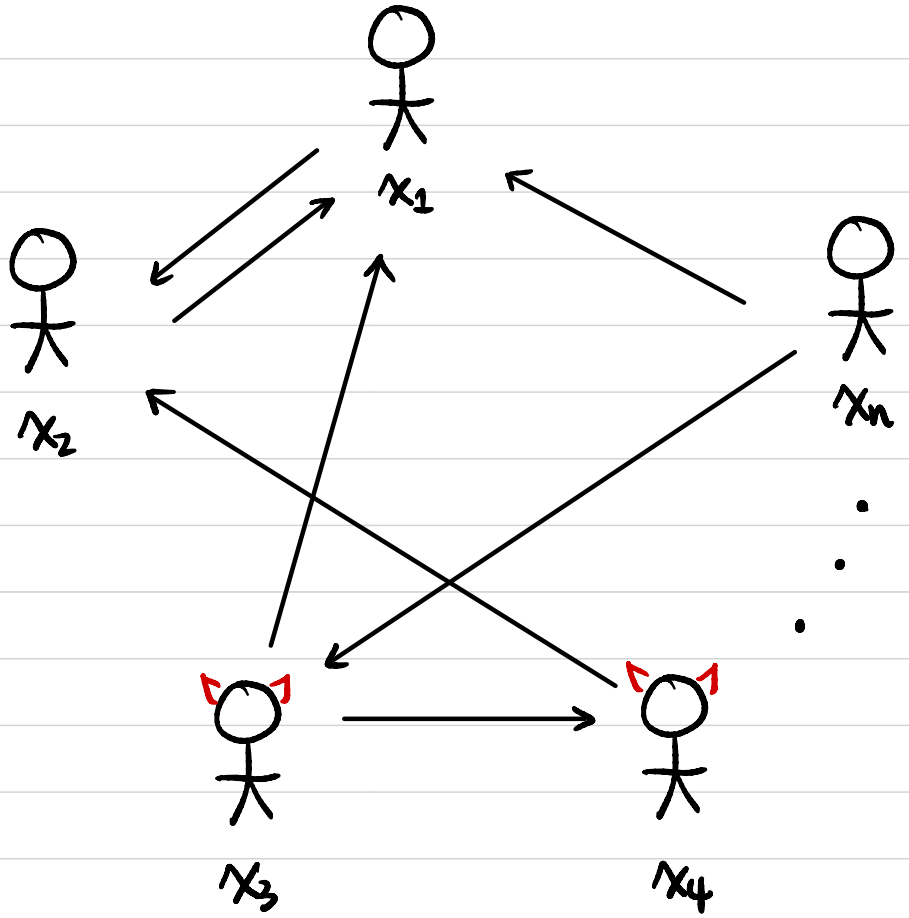
↓

Continue with the remaining λ GCs ⇒ Take majority

If Alice generated $\geq \frac{\lambda}{2}$ GCs incorrectly,

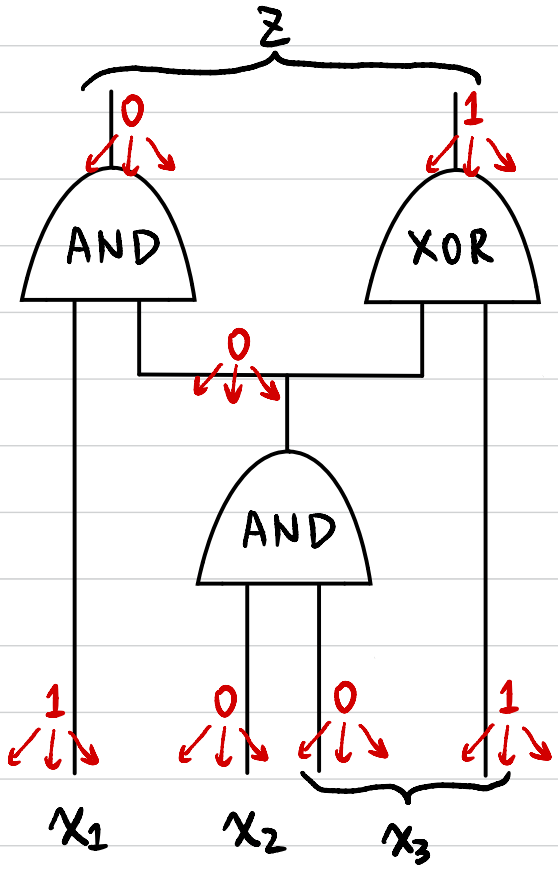
Pr[Alice passes all verification] ≤ ?

Secure Multi-Party Computation (MPC)



$$z = f(x_1, \cdots, x_n)$$

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 $v^w \in \{0,1\}$,

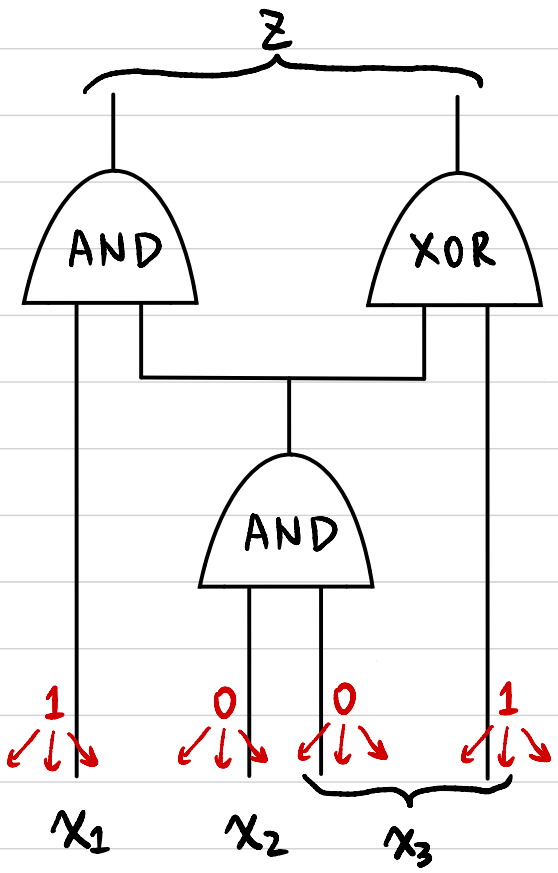
then the n parties hold an additive secret share of v^w

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:

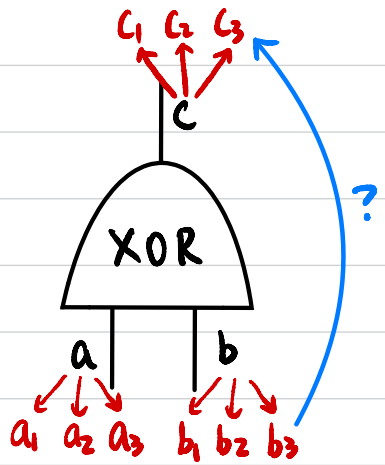
For each input wire w:

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:

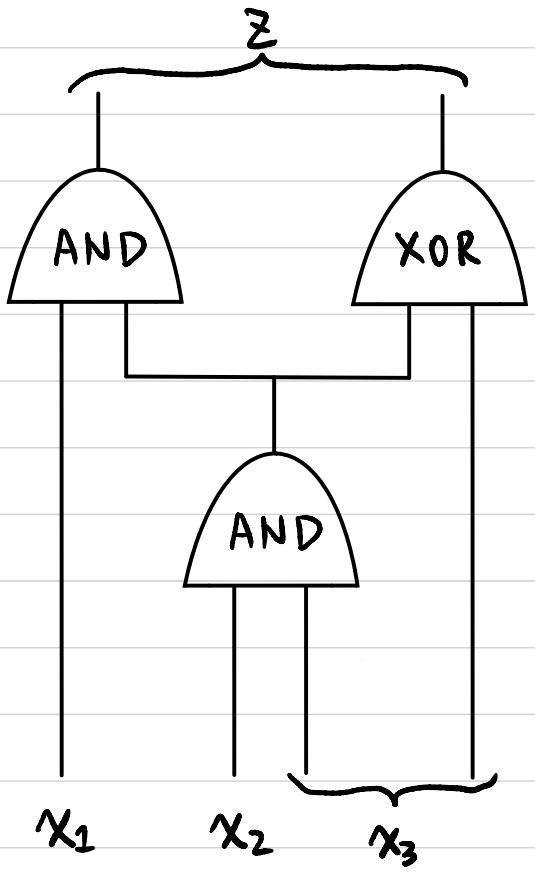


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\quad \bigoplus_{i=1}^{n} b_i = b$

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

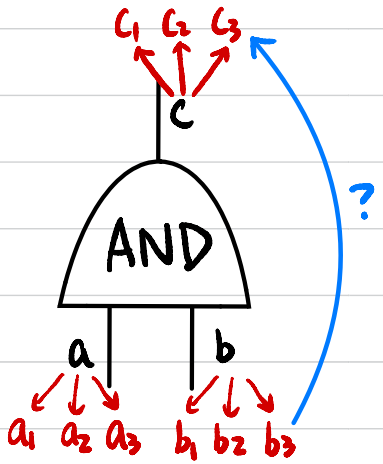
$c_i = ?$

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

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

$c_i = ?$

Outputs:

For each output wire w:

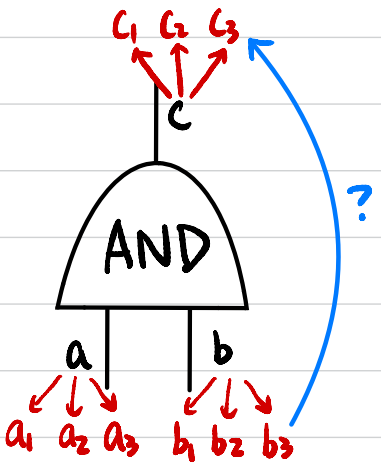
Each party P_i holds a random share $v_i^w \in \{0,1\}$

→ 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 $t \le n-1$ (GMW)

AND gates:



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

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

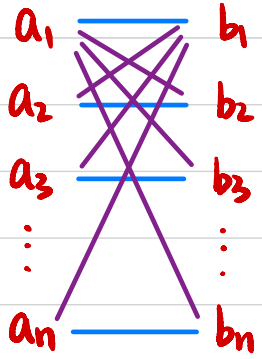
$c_i = ?$

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

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

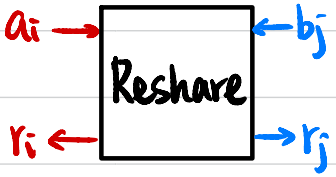
$a_i \cdot b_i$: P_i locally

$a_i \cdot b_j$: ?



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

Reshare:



P_i

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

P_j

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

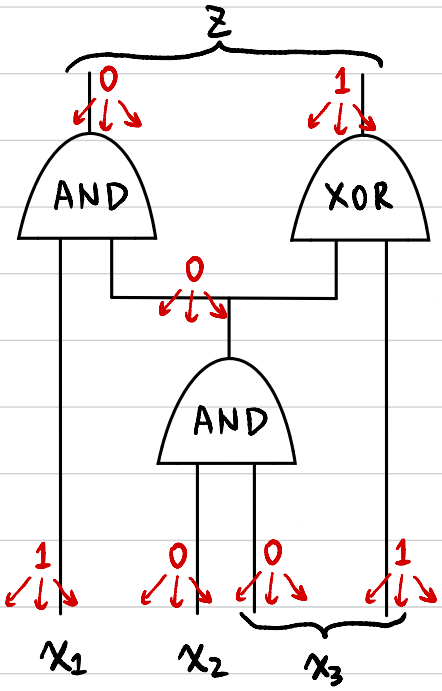
Output: $r_i \in \{0,1\}$

Output: $r_j \in \{0,1\}$

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

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

MPC for any function with $t \le n-1$ (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:

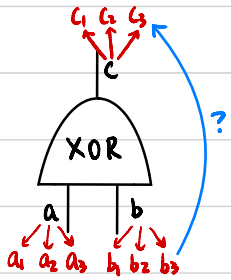
For each input wire w:

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:

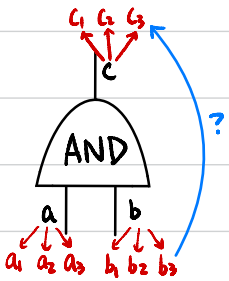


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

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

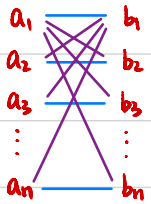
$c_i = a_i \cdot b_i \oplus \sum_{j \neq i} r_{i,j}^{(1)} \oplus \sum_{j \neq i} r_{j,i}^{(2)}$

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

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

P_i locally

Reshare $r_{i,j}^{(1)}$ $r_{i,j}^{(2)}$



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

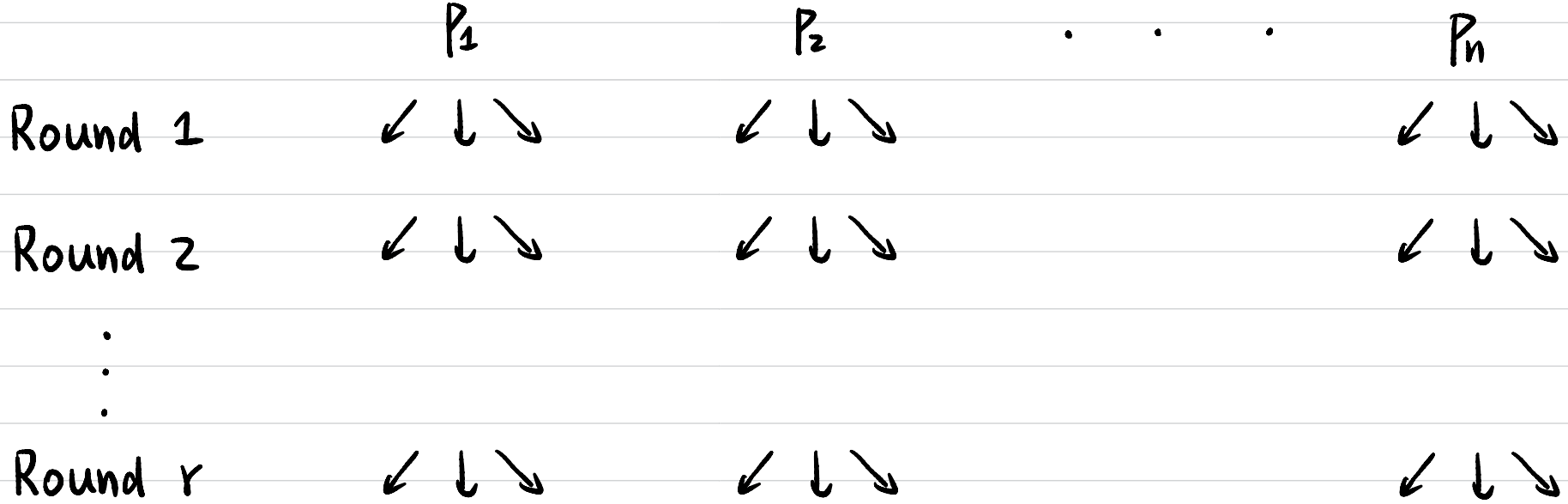
Each party computes the value $v^w = \bigoplus_{i=1}^{n} v_i^w$

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

Computational Complexity?

Communication Complexity?

Round Complexity?



What could go wrong against malicious adversaries?