

CSCI 1515 Applied Cryptography

This Lecture:

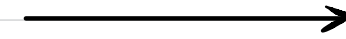
- Optimizations of Garbled Circuits (continued)
- Cut-and-Choose for Garbled Circuits
- GMW: Semi-Honest MPC for Any Function

Secure Two-Party Computation (2PC)

Alice



x



$$z = f(x, y)$$

Bob



y

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 ZPC
for any function

Semi-honest MPC
for any function

cut-and-choose
with commitments

GMW Compiler
with ZKP

Malicious ZPC
for any function

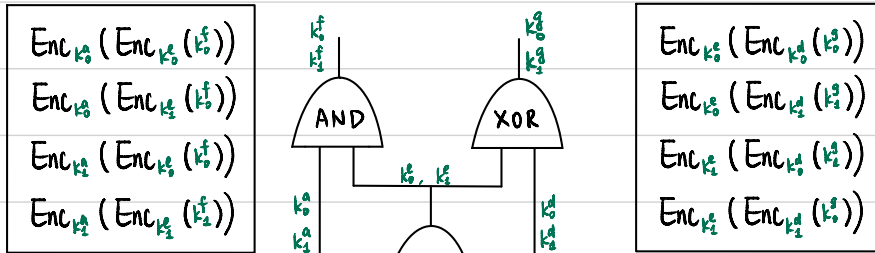
Malicious MPC
for any function

Putting it All Together: Semi-Honest ZPC

What could go wrong against malicious adversaries?

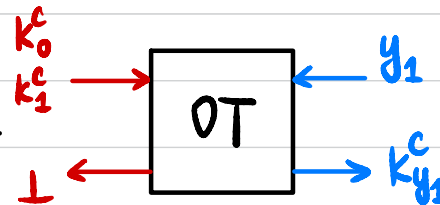
Alice (Garbler)

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



Garbled Circuit
(Garbled Gates)

Input labels for x



Input labels for y : OT

- $Enc_{k_0^c}(Enc_{k_0^c}(k_0^c))$
- $Enc_{k_0^c}(Enc_{k_1^c}(k_0^c))$
- $Enc_{k_1^c}(Enc_{k_0^c}(k_1^c))$
- $Enc_{k_1^c}(Enc_{k_1^c}(k_1^c))$

- Shuffle
- $H(k_0^b || k_0^c) \oplus (k_0^e || 0 \dots 0) \rightarrow$ garbage
 - $H(k_0^b || k_1^c) \oplus (k_0^e || 0 \dots 0) \rightarrow$ garbage
 - $H(k_1^b || k_0^c) \oplus (k_0^e || 0 \dots 0) \rightarrow k_0^e || 0 \dots 0$
 - $H(k_1^b || k_1^c) \oplus (k_1^e || 0 \dots 0) \rightarrow$ garbage
- 256 128 128

Output labels k^f, k^g

Output z

Communication cost?
 $256 \cdot 4 \cdot |C|$

Computation cost?
G & E: $4 \cdot |C|$

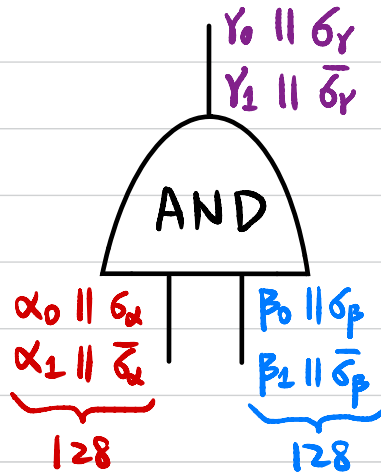
Bob (Evaluator)

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

Optimization 1: Point-and-Permute

4 · 128 bits / gate

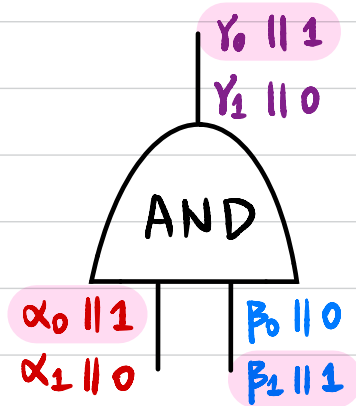
$\sigma_\alpha, \sigma_\beta, \sigma_\gamma \in \{0, 1\}$
(signal bits)



$$\begin{aligned} \sigma_\alpha \quad \sigma_\beta & : \text{Enc}_{\alpha_0} (\text{Enc}_{\beta_0} (\overset{127}{Y_0} \parallel \overset{1}{\sigma_\gamma})) \\ \sigma_\alpha \quad \bar{\sigma}_\beta & : \text{Enc}_{\alpha_0} (\text{Enc}_{\beta_1} (Y_0 \parallel \sigma_\gamma)) \\ \bar{\sigma}_\alpha \quad \sigma_\beta & : \text{Enc}_{\alpha_1} (\text{Enc}_{\beta_0} (Y_0 \parallel \sigma_\gamma)) \\ \bar{\sigma}_\alpha \quad \bar{\sigma}_\beta & : \text{Enc}_{\alpha_1} (\text{Enc}_{\beta_1} (Y_1 \parallel \bar{\sigma}_\gamma)) \end{aligned}$$

Example:

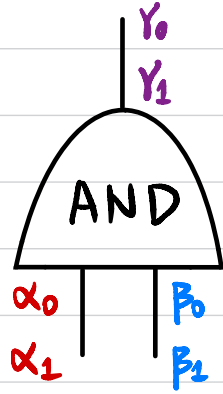
$\sigma_\alpha = 1 \quad \sigma_\beta = 0 \quad \sigma_\gamma = 1$



$$\begin{aligned} 0 \quad 0 & : \text{Enc}_{\alpha_1} (\text{Enc}_{\beta_0} (Y_0 \parallel 1)) \\ 0 \quad 1 & : \text{Enc}_{\alpha_1} (\text{Enc}_{\beta_1} (Y_1 \parallel 0)) \\ 1 \quad 0 & : \text{Enc}_{\alpha_0} (\text{Enc}_{\beta_0} (Y_0 \parallel 1)) \\ 1 \quad 1 & : \text{Enc}_{\alpha_0} (\text{Enc}_{\beta_1} (Y_0 \parallel 1)) \end{aligned}$$

Optimization 2: Row Reduction

3 · 128 bits/gate



$$\cancel{H(\alpha_0 \parallel \beta_0) \oplus \gamma_0} := 0 \quad \gamma_0 := H(\alpha_0 \parallel \beta_0)$$

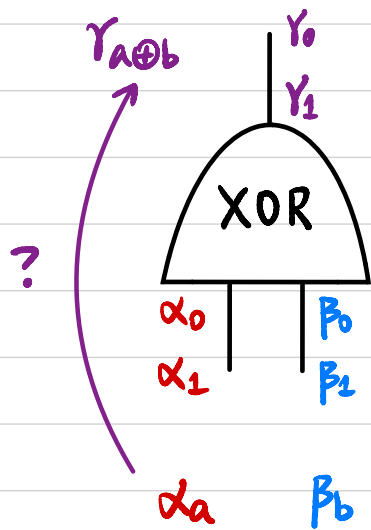
$$H(\alpha_0 \parallel \beta_1) \oplus \gamma_0$$

$$H(\alpha_1 \parallel \beta_0) \oplus \gamma_0$$

$$H(\alpha_1 \parallel \beta_1) \oplus \gamma_1$$

↑
Random Oracle

Optimization 3: Free XOR



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

hidden to evaluator

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

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

$$\gamma_1 := \gamma_0 \oplus \Delta$$

$$\gamma_0 := \alpha_0 \oplus \beta_0$$

$$\gamma_{a+b} = \alpha_a \oplus \beta_b$$

$$\alpha_0 \oplus \beta_0 = \gamma_0$$

$$\alpha_0 \oplus \beta_1 = \alpha_0 \oplus (\beta_0 \oplus \Delta) = \gamma_0 \oplus \Delta = \gamma_1$$

$$\alpha_1 \oplus \beta_0 = (\alpha_0 \oplus \Delta) \oplus \beta_0 = \gamma_1$$

$$\alpha_1 \oplus \beta_1 = (\alpha_0 \oplus \Delta) \oplus (\beta_0 \oplus \Delta) = \gamma_0$$

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 Garbled Circuits
(Garbled Gates)

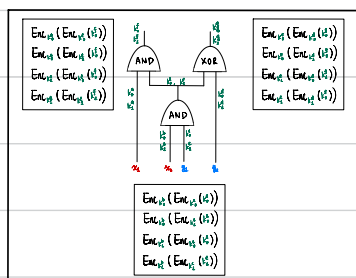
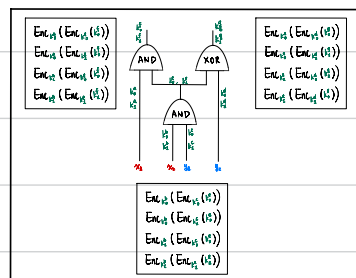
Randomly pick $(\lambda-1)$ of them

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

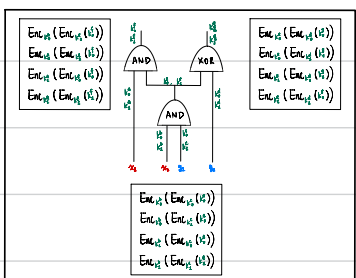
Verify correctness

Continue with the remaining GC

If Alice generated 1 GC incorrectly,
 $\Pr[\text{Alice passes verification}] = \frac{1}{\lambda}$



⋮



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)

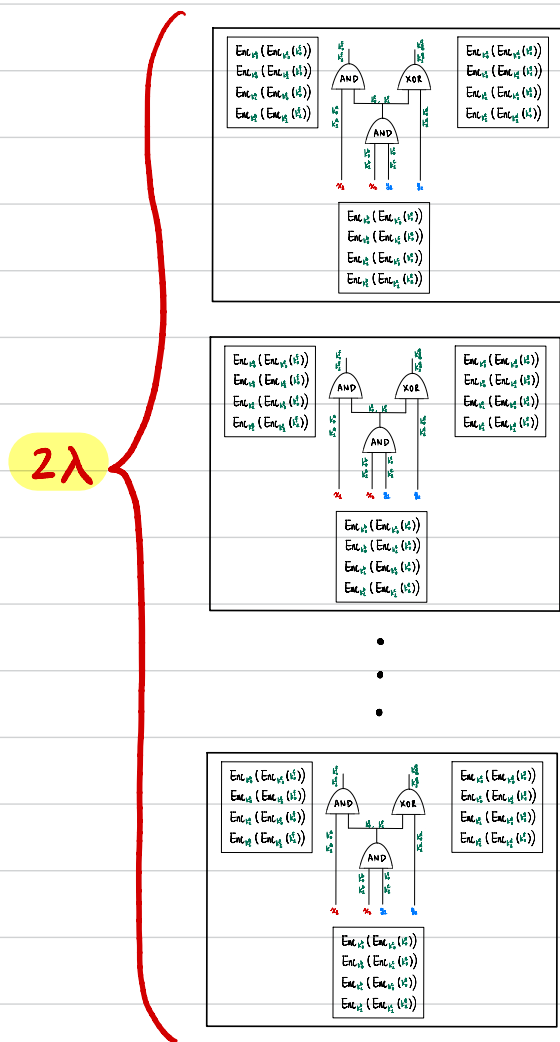
← Randomly pick λ of them

Reveal all the labels
for the λ GCs

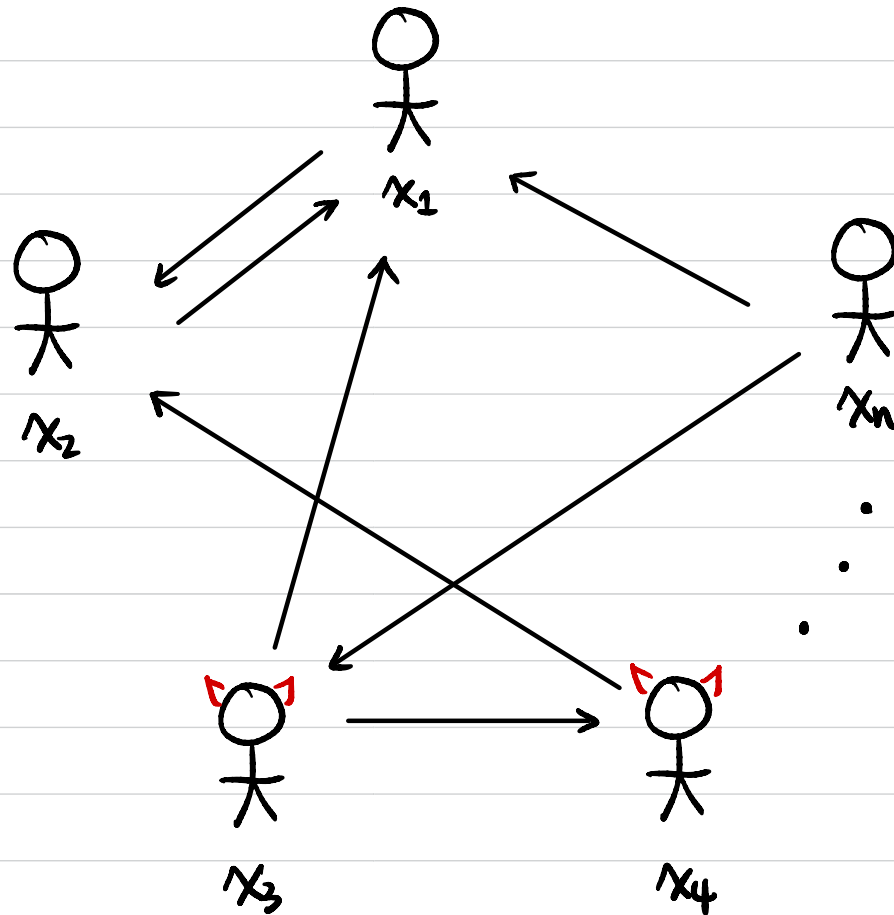
Verify correctness

Continue with the remaining λ GCs \Rightarrow Take majority

If Alice generated $\geq \frac{\lambda}{2}$ GCs incorrectly,
 $\Pr[\text{Alice passes all verification}] \leq 2^{-\lambda/2}$

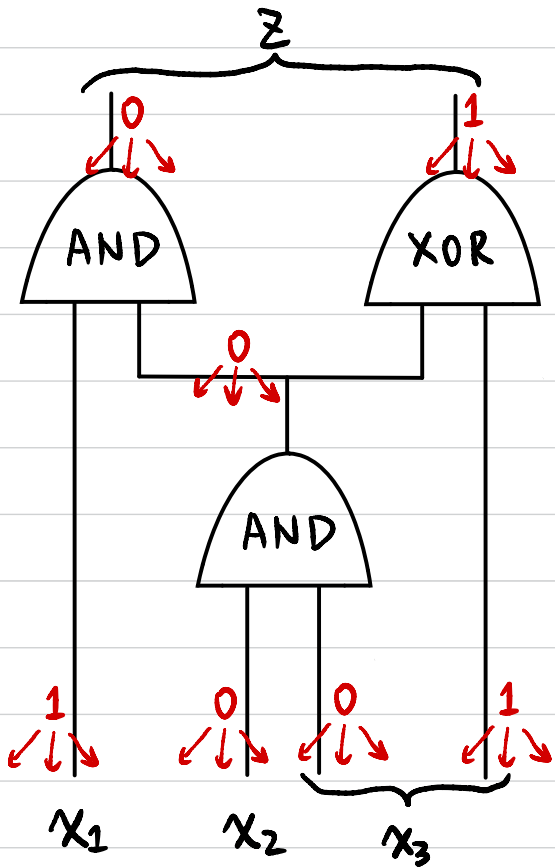


Secure Multi-Party Computation (MPC)



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

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

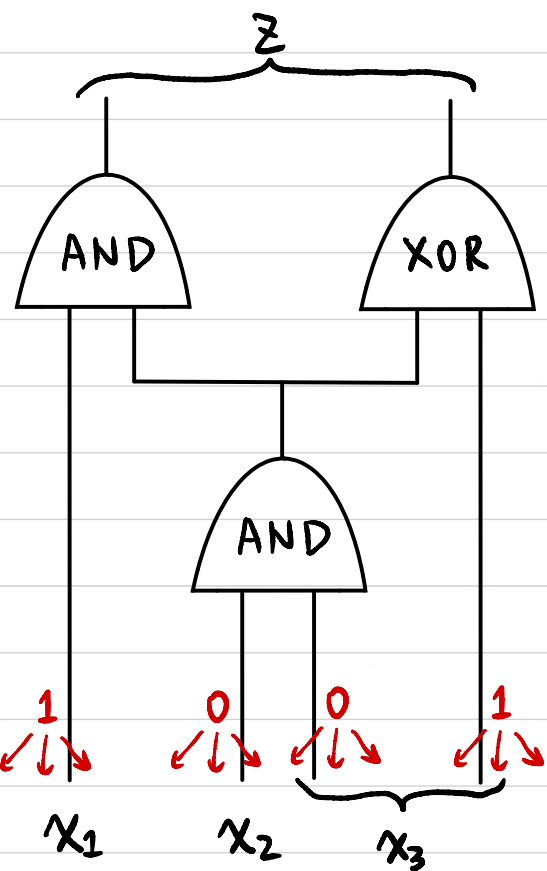
Inputs:

For each input wire w :

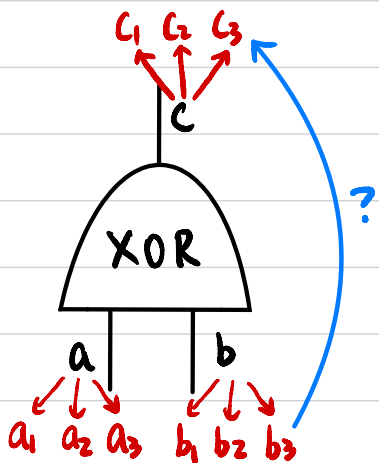
If it's from party P_k with input value $v^w \in \{0,1\}$,

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

Sends v_i^w to party P_i .



XOR gates:



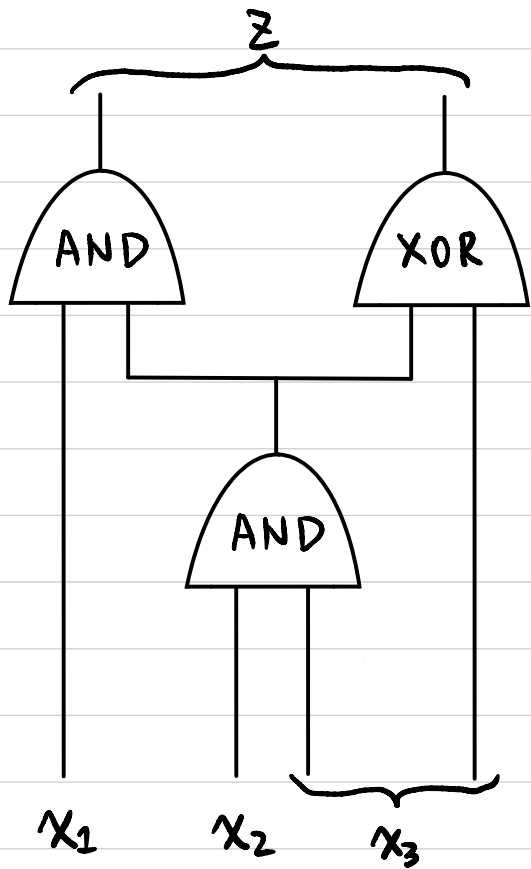
GIVEN: $\bigoplus_{i=1}^n a_i = a$ $\bigoplus_{i=1}^n b_i = b$

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

$$c_i = a_i \oplus b_i$$

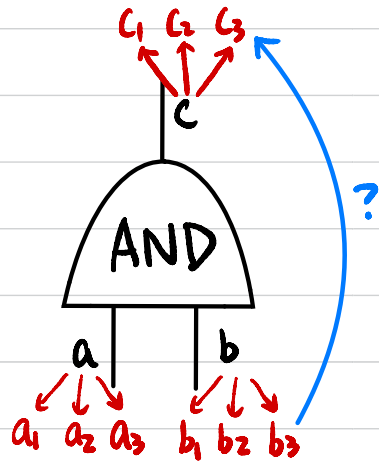
$$\bigoplus_{i=1}^n c_i = \bigoplus_{i=1}^n (a_i \oplus b_i) = \left(\bigoplus_{i=1}^n a_i\right) \oplus \left(\bigoplus_{i=1}^n b_i\right) = a \oplus b$$

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 :

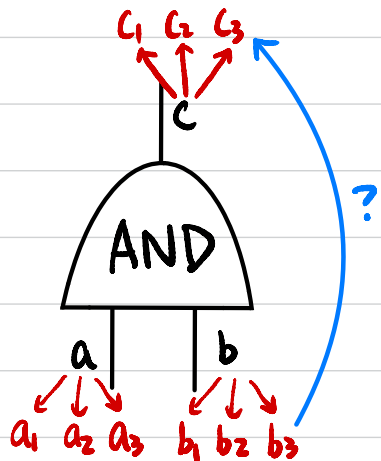
Each party P_i holds a random share $v_i^w \in \{0,1\}$

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

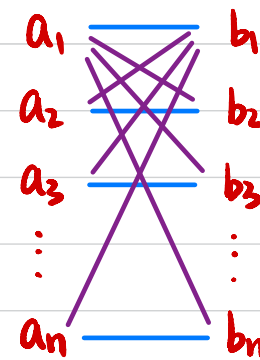
WANT: $\{c_i\}$ s.t. $\bigoplus_{i=1}^n c_i = c = a \cdot b$

$c_i = ?$

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

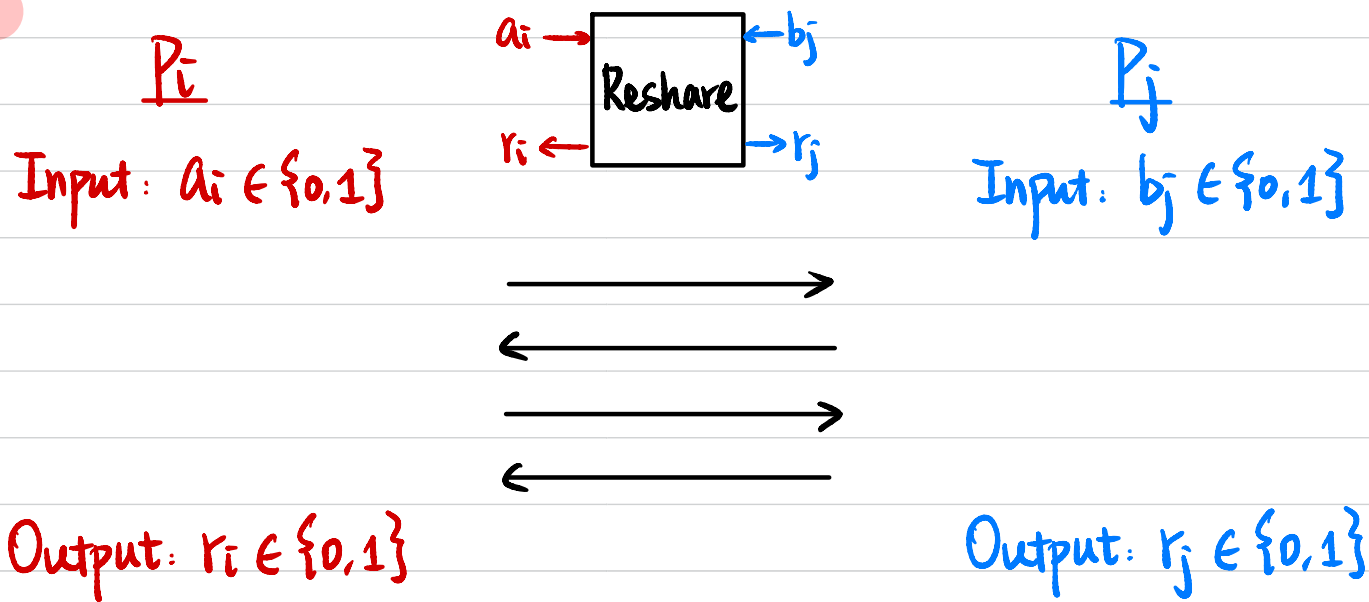
$$= \left(\sum_{i=1}^n a_i \cdot b_i \right) + \left(\sum_{i \neq j} a_i \cdot b_j \right) \pmod{2}$$

\uparrow
P_i locally
 \uparrow
?



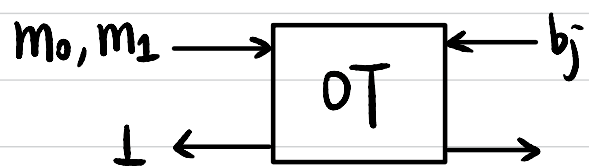
MPC for any function with $t \leq n-1$ (GMW)

Reshare:



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

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



If $b_j = 0 \Rightarrow r_j = ? \rightarrow m_0$

If $b_j = 1 \Rightarrow r_j = ? \rightarrow m_1$