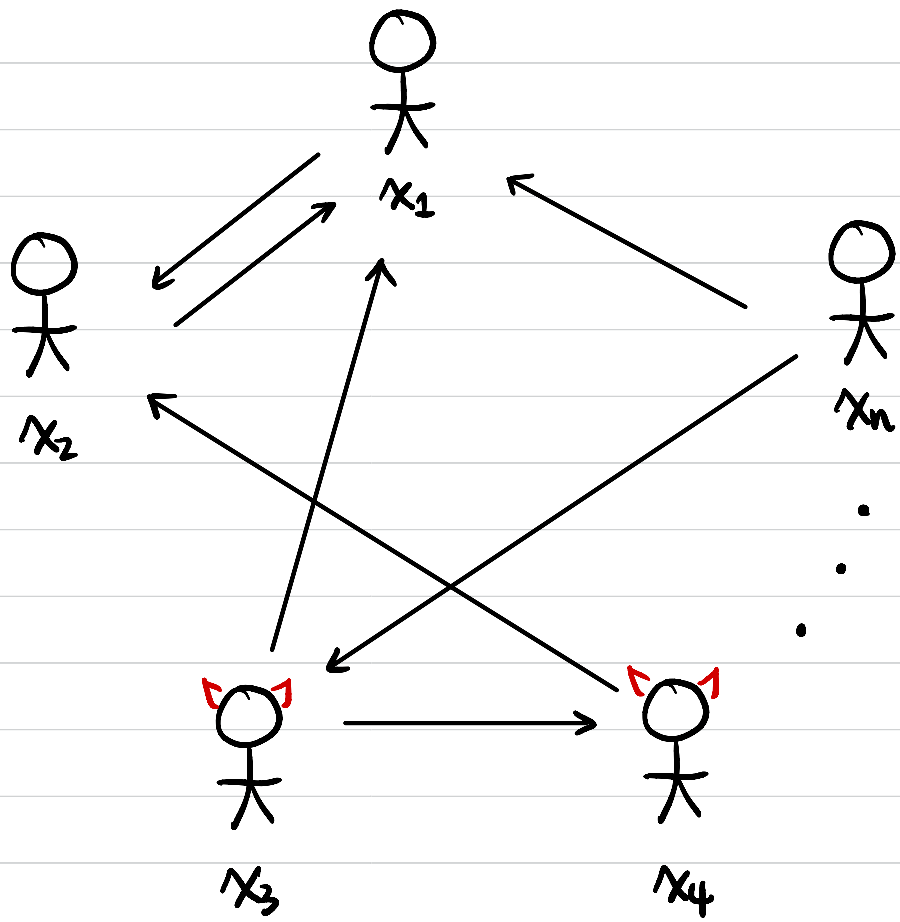
CSCI 1515 Applied Cryptography

This Lecture:

- GMW: Semi-Honest MPC for Any Function
- GMW Compiler: Malicious MPC for Any Funtion

Secure Multi-Party Computation (MPC)



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

Adversary's Power

Allowed adversarial behavior:

- Semi-honest / passive / honest-but-curious:

  Follow the protocol description honestly,

  but try to extract more information by inspecting transcript.

- Malicious /active:

  Can deviate arbitrarily from the protocol description.

Semi-honest OT
(OT protocol that's secure against semi-honest adv.)

$\Downarrow$ Yao's Garbled Circuit

Semi-honest 2PC for any function

$\Downarrow$ Cut-and-choose with commitments

Malicious 2PC for any function

$\Downarrow$ GMW

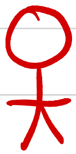
Semi-honest MPC for any function $t \leq n-1$

$\Downarrow$ GMW Compiler with ZKP

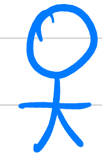
Malicious MPC for any function $t \leq n-1$

Oblivious Transfer (OT)

Sender



Receiver

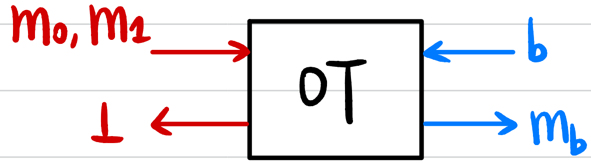


Input: $m_0, m_1 \in \{0,1\}^\ell$

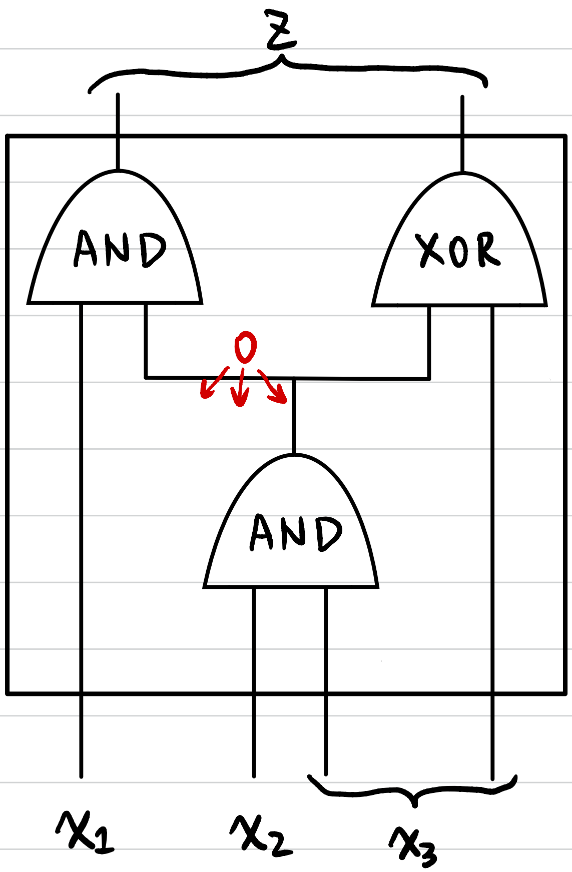
Input: $b \in \{0,1\}$

Output: $\perp$

Output: m_b



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



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

Throughout the protocol, we keep the invariant:

For each wire w:

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

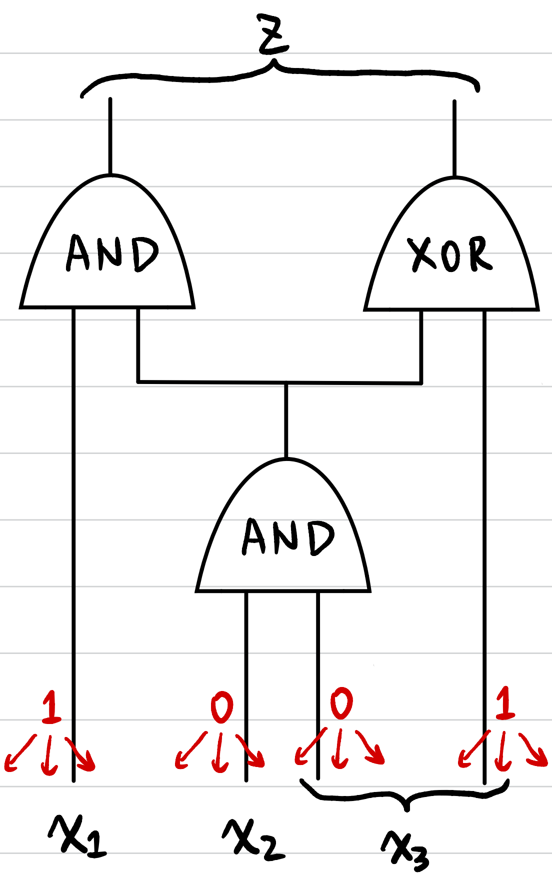
then the n parties hold an additive secret share of v^w

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:

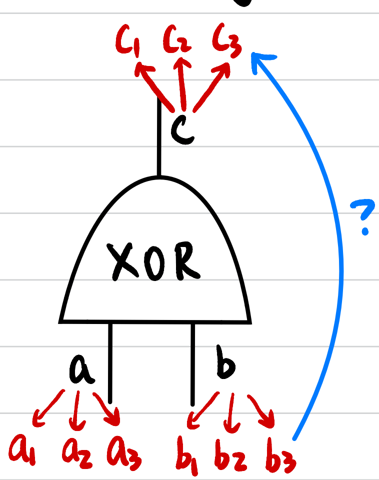
For each input wire w:

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:



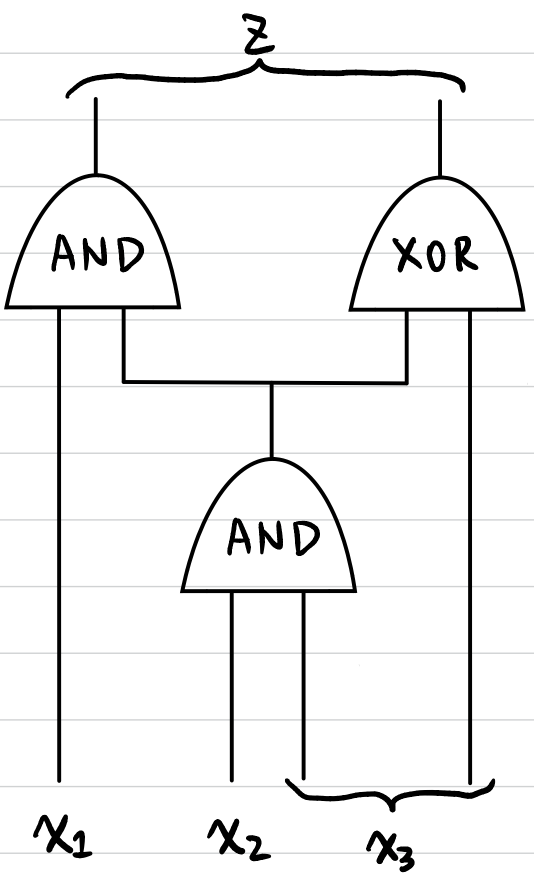
GIVEN: $\bigoplus_{i=1}^{n} a_i = a \qquad \bigoplus_{i=1}^{n} b_i = b$

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

$c_i = a_i \oplus b_i$

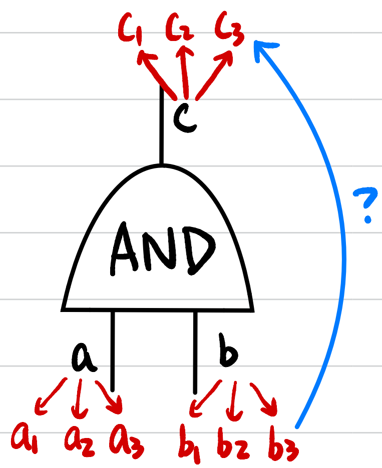
$\bigoplus_{i=1}^{n} c_i = \bigoplus_{i=1}^{n} (a_i \oplus b_i) = \left(\bigoplus_{i=1}^{n} a_i\right) \oplus \left(\bigoplus_{i=1}^{n} b_i\right) = a \oplus b$

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



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

AND gates:



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

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

$c_i = ?$

Outputs:

For each output wire w:

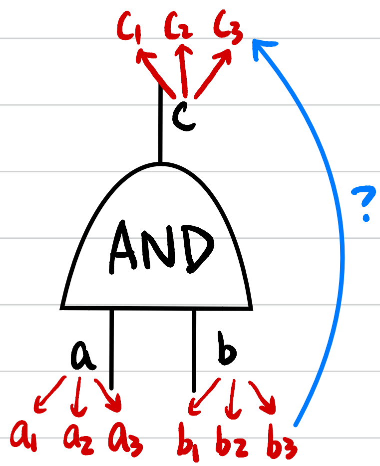
Each party P_i holds a random share $v_i^w \in \{0,1\}$

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

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

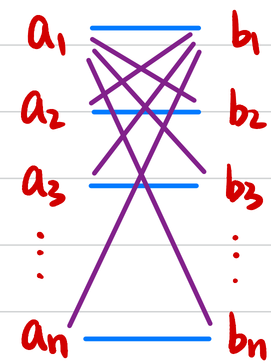
$c_i = ?$

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

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

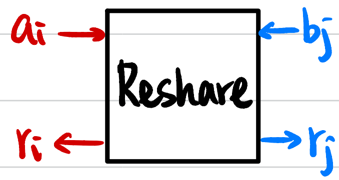
P_i locally

r_i ? r_j



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

Reshare:



P_i

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

P_j

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

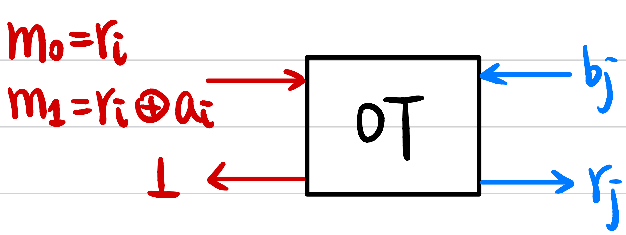
Output: $r_i \in \{0,1\}$

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 $r_i \xleftarrow{\$} \{0,1\}$

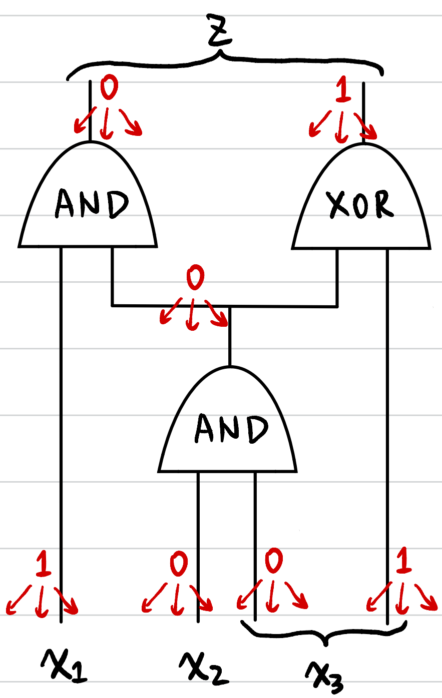
2) How to let P_j learn r_j s.t. $r_i \oplus r_j = a_i \cdot b_j$?



If $b_j = 0 \Rightarrow r_j = r_i$

If $b_j = 1 \Rightarrow r_j = r_i \oplus a_i$

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:

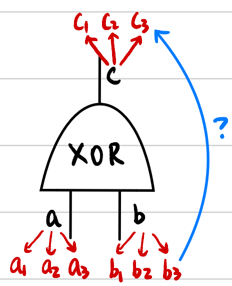
For each input wire w:

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:

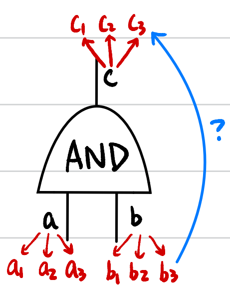


GIVEN: $\bigoplus_{i=1}^{n} a_i = a \qquad \bigoplus_{i=1}^{n} b_i = b$

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

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

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

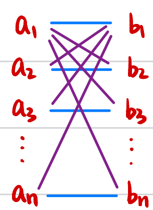
$$a \cdot b = \left(\sum_{i=1}^{n} a_i\right) \cdot \left(\sum_{i=1}^{n} b_i\right) \mod 2$$

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

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 $t \le n-1$ (GMW)

Computational Complexity?

$O(\#XOR + n \cdot \#AND)$ per party

Communication Complexity?

$O(n^2 \cdot \#AND)$ total $\quad O(n \cdot \#AND)$ per party

Round Complexity? $O(\text{depth of AND gates})$

	P_1	P_2	$\cdots$	P_n
Round 1	↙ ↓ ↘	↙ ↓ ↘		↙ ↓ ↘
Round 2	↙ ↓ ↘	↙ ↓ ↘		↙ ↓ ↘
⋮				
Round r	↙ ↓ ↘	↙ ↓ ↘		↙ ↓ ↘

What could go wrong against malicious adversaries?

GMW Compiler

Given a semi-honest protocol:

Once inputs & randomness are fixed, protocol is deterministic.

Step 1: Each party P_i commits to its input x_i & randomness r_i to be used in the semi-honest protocol.

Step 2: Run semi-honest protocol.

Along with every message, prove in ZK that the message is computed correctly (based on its input, randomness, transcript so far)