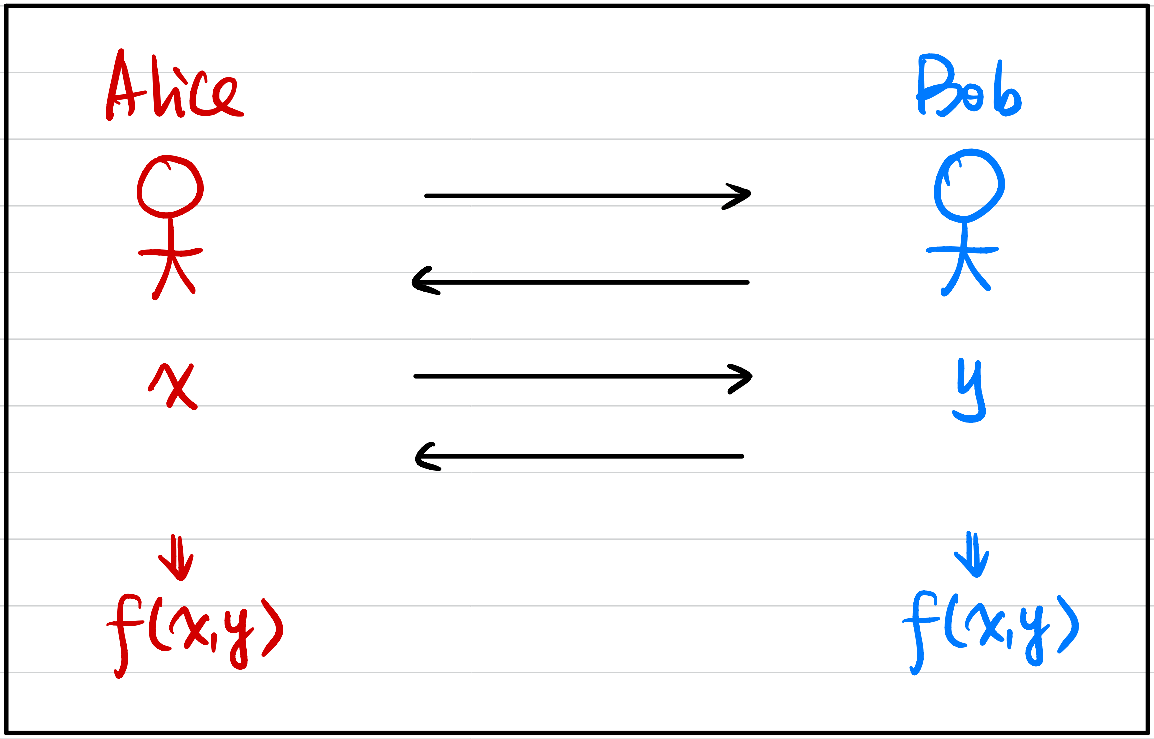
CSCI 1510

- Definitions of MPC (continued)
- Private Set Intersection
- Oblivious Transfer
- Semi-Honest MPC for Any Function (GMW)
- Malicious MPC (GMW Compiler)

Security Against Semi-Honest Adversaries



Alice's view:

$$\mathrm{View}_A^{\pi}(x, y, n) := (x, \text{ internal random tape } r, \text{ messages from Bob})$$

Given x, $f(x,y)$, Alice's view can be "simulated".

Security Against Semi-Honest Adversaries

Def (Semi-honest security for 2PC)

Let f be a functionality. We say a protocol Π securely computes f against semi-honest adversaries if $\exists$ PPT algorithms S_A, S_B s.t. $\forall x, y$,

$$\left\{\begin{pmatrix} S_A(1^n, x, f(x,y)) \\ f(x,y) \end{pmatrix}\right\}_{n\in\mathbb{N}} \simeq \left\{\begin{pmatrix} \mathrm{View}_A^{\Pi}(x,y,n) \\ \mathrm{Output}^{\Pi}(x,y,n) \end{pmatrix}\right\}_{n\in\mathbb{N}}$$

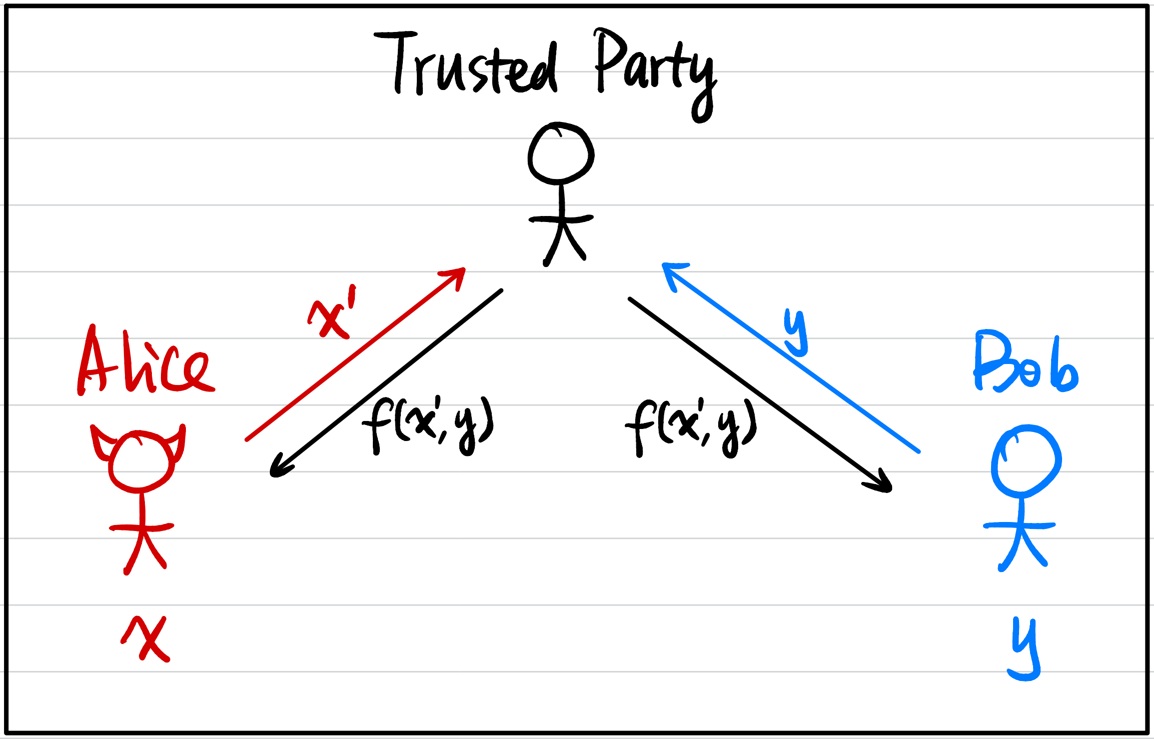
$$\left\{\begin{pmatrix} S_B(1^n, y, f(x,y)) \\ f(x,y) \end{pmatrix}\right\}_{n\in\mathbb{N}} \simeq \left\{\begin{pmatrix} \mathrm{View}_B^{\Pi}(x,y,n) \\ \mathrm{Output}^{\Pi}(x,y,n) \end{pmatrix}\right\}_{n\in\mathbb{N}}$$

perfect / statistical / computational

$\equiv$ $\quad$ $\overset{s}{\simeq}$ $\quad$ $\overset{c}{\simeq}$

Security Against Malicious Adversaries

What's the best we can hope for? (Ideal World)



Security Against Malicious Adversaries (Real / Ideal Paradigm)

Execution in the Real World:

(PPT) adversary $\mathcal{A}$ corrupting party $i \in \{\text{Alice}, \text{Bob}\}$

$$\text{REAL}^{\Pi}_{\mathcal{A},i} := \begin{pmatrix} \mathcal{A}\text{'s output} \\ \text{Honest party's output in Real World} \end{pmatrix}$$

Execution in the Ideal World:

PPT adversary S corrupting party $i \in \{\text{Alice}, \text{Bob}\}$

$$\text{IDEAL}^{f}_{S,i} := \begin{pmatrix} S\text{'s output} \\ \text{Honest party's output in Ideal World} \end{pmatrix}$$

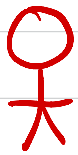
Def (malicious security for 2PC)

Let f be a functionality. We say a protocol Π securely computes f against malicious adversaries if $\forall$ (PPT) $\mathcal{A}$ in the real word, $\exists$ PPT S in the ideal world s.t. $\forall i \in \{\text{Alice}, \text{Bob}\}$, $\forall x, y$,

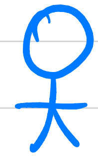
$$\left\{\text{REAL}^{\Pi}_{\mathcal{A},i}(x,y,n)\right\}_{n\in\mathbb{N}} \simeq \left\{\text{IDEAL}^{f}_{S,i}(x,y,n)\right\}_{n\in\mathbb{N}}$$

Private Set Intersection (PSI)

Alice



Bob



Input: $X = \{x_1, x_2, \cdots, x_n\}$

$V = \{v_1, v_2, \cdots, v_n\}$

Input: $Y = \{y_1, y_2, \cdots, y_n\}$

PSI: $f(X, Y) = X \cap Y$

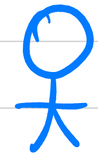
PSI-CA: $f(X, Y) = |X \cap Y|$

PSI-SUM: $f((X, V), Y) = |X \cap Y|, \sum_{i: x_i \in Y} v_i$

Private Set Intersection (PSI)

Alice

Bob



Input: $X = \{x_1, x_2, \cdots, x_n\}$

Input: $Y = \{y_1, y_2, \cdots, y_n\}$

$\xrightarrow{H(x_1), \cdots, H(x_n)}$

$H(y_1), \cdots, H(y_n)$

$\Downarrow$

$X \cap Y$

Is it (semi-honest) secure?

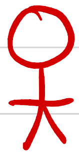
Is it possible to achieve 2PC / MPC with 1 round of communication?

DDH-based PSI

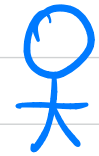
Cyclic group $\mathbb{G}$ of order q with generator g

$H: \{0,1\}^* \to \mathbb{G}$

Alice



Bob



Alice Input: $X = \{x_1, x_2, \cdots, x_n\}$

Bob Input: $Y = \{y_1, y_2, \cdots, y_n\}$

Alice: $k_A \xleftarrow{\$} \mathbb{Z}_q$

Bob: $k_B \xleftarrow{\$} \mathbb{Z}_q$

$$\xleftarrow{\quad H(Y)^{k_B} := \{H(y_1)^{k_B}, \cdots, H(y_n)^{k_B}\} \quad}$$

$$\xrightarrow{\quad H(X)^{k_A},\ H(Y)^{k_A \cdot k_B} \quad}$$

Bob: $H(X)^{k_A \cdot k_B} \cap H(Y)^{k_A \cdot k_B}$

$$\Downarrow$$

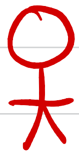
$$X \cap Y$$

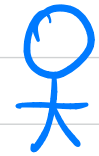
$$\xleftarrow{\quad X \cap Y \quad}$$

Thm If DDH is hard in $\mathbb{G}$ and H is modeled as a random oracle, then this protocol is semi-honest secure.

PSI-CA ?

PSI-CA: $f(X,Y) = |X \cap Y|$

Alice 

Bob 

Alice — Input: $X = \{x_1, x_2, \cdots, x_n\}$

Bob — Input: $Y = \{y_1, y_2, \cdots, y_n\}$

Alice: $k_A \xleftarrow{\$} \mathbb{Z}_q$

Bob: $k_B \xleftarrow{\$} \mathbb{Z}_q$

$$\xleftarrow{\quad H(Y)^{k_B} := \{H(y_1)^{k_B}, \cdots, H(y_n)^{k_B}\} \quad}$$

$$\xrightarrow{\quad H(X)^{k_A},\ H(Y)^{k_A \cdot k_B} \quad}$$

Bob: $H(X)^{k_A \cdot k_B} \cap H(Y)^{k_A \cdot k_B}$

$$\Downarrow$$

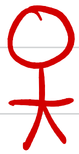
$X \cap Y$

$$\xleftarrow{\quad X \cap Y \quad}$$

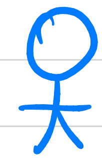
PSI-SUM?

PSI-SUM: $f((X, V), Y) = |X \cap Y|, \sum_{i: x_i \in Y} v_i$

Alice



Bob



Alice — Input: $X = \{x_1, x_2, \cdots, x_n\}$

$V = \{v_1, v_2, \cdots, v_n\}$

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

Bob — Input: $Y = \{y_1, y_2, \cdots, y_n\}$

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

$$\xleftarrow{\quad H(Y)^{k_B} := \{H(y_1)^{k_B}, \cdots, H(y_n)^{k_B}\} \quad}$$

$$\xrightarrow{\quad H(X)^{k_A},\ H(Y)^{k_A \cdot k_B} \quad}$$

$$H(X)^{k_A \cdot k_B} \cap H(Y)^{k_A \cdot k_B}$$

$$\Downarrow$$

$$X \cap Y$$

$$\xleftarrow{\quad X \cap Y \quad}$$

Feasibility Results

Computational Security:

Semi-honest Oblivious Transfer (OT)

$\Downarrow$

\# corrupted parties

semi-honest MPC for any function with $t < n$

$\Downarrow$

malicious MPC for any function with $t < n$

Information-Theoretic (IT) Security:

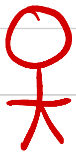
(honest majority)

semi-honest/malicious MPC for any function with $t < n/2$

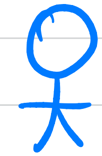
necessary

Oblivious Transfer (OT)

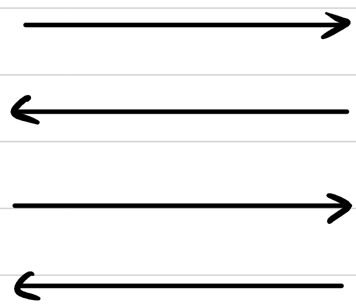
Sender



Receiver



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



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

Output: $\perp$

Output: m_c

Oblivious Transfer (OT)

Cyclic group $\mathbb{G}$ of order q with generator g

$H: \mathbb{G} \to \{0,1\}^{\ell}$

Sender

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

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

$\xrightarrow{A = g^a}$

Receiver

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

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

$\xleftarrow{B = g^b \cdot A^c}$

$k_0 := H(B^a)$

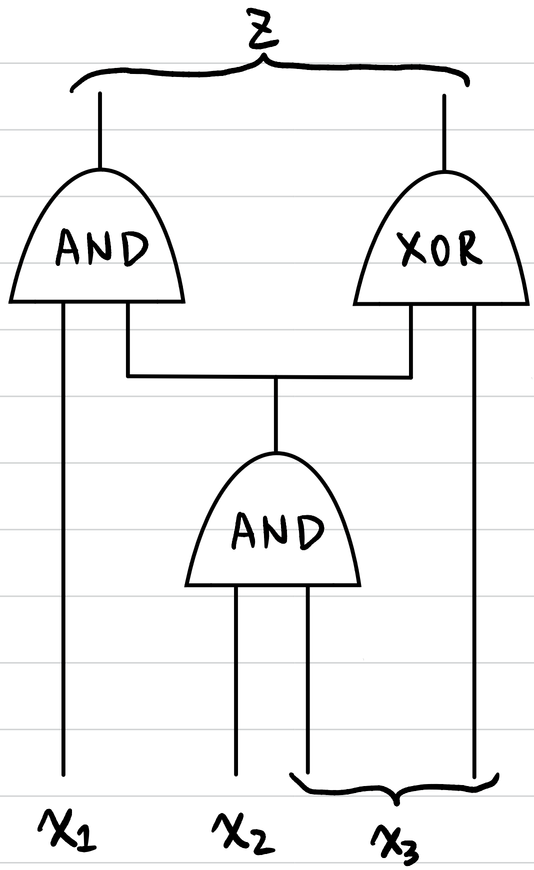
$k_1 := H\left(\left(\frac{B}{A}\right)^a\right)$

$$\xrightarrow[ct_1 := k_1 \oplus m_1]{ct_0 := k_0 \oplus m_0}$$

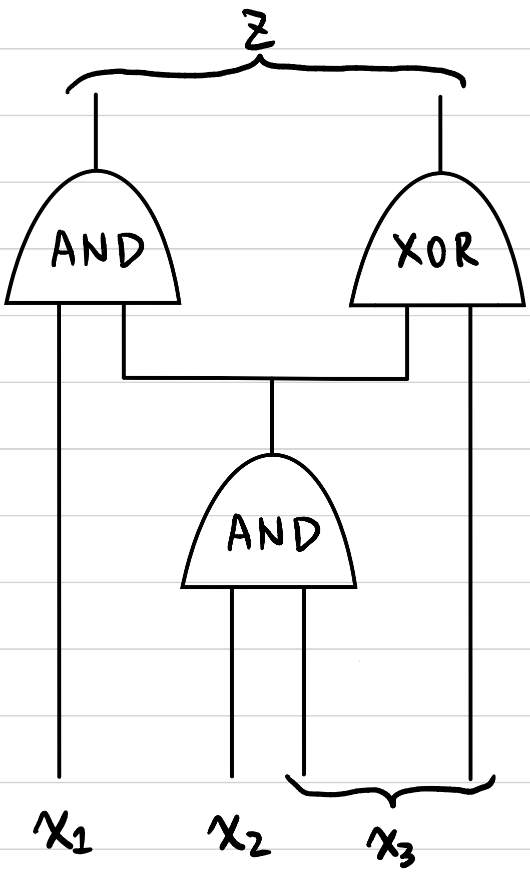
Output: ?

Thm If CDH is hard in $\mathbb{G}$ and H is modeled as a random oracle, then this protocol is semi-honest secure.

Arbitrary Function → Represent it as a Boolean circuit



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

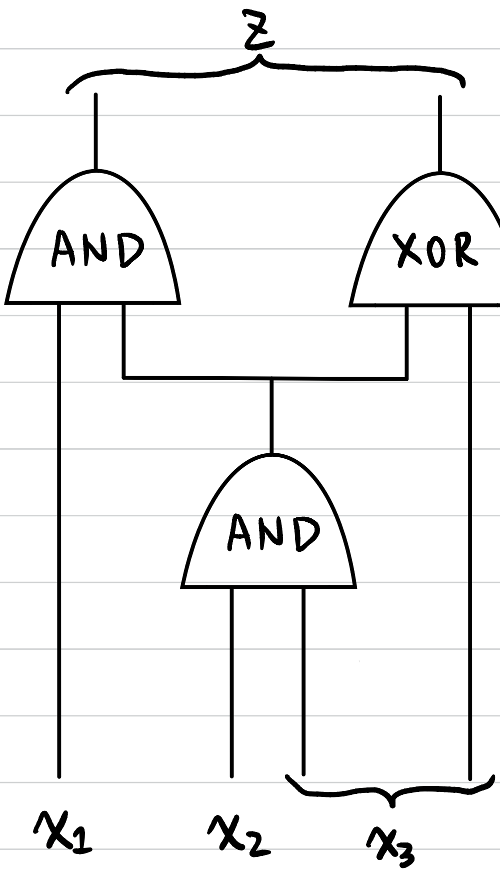
then the n parties hold an additive secret share of v^w

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 $t \le n-1$ (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:

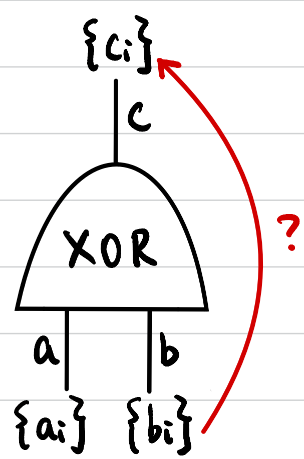
For each input wire w:

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

$\hookrightarrow$ Sends v_i^w to party P_i.

XOR gates:

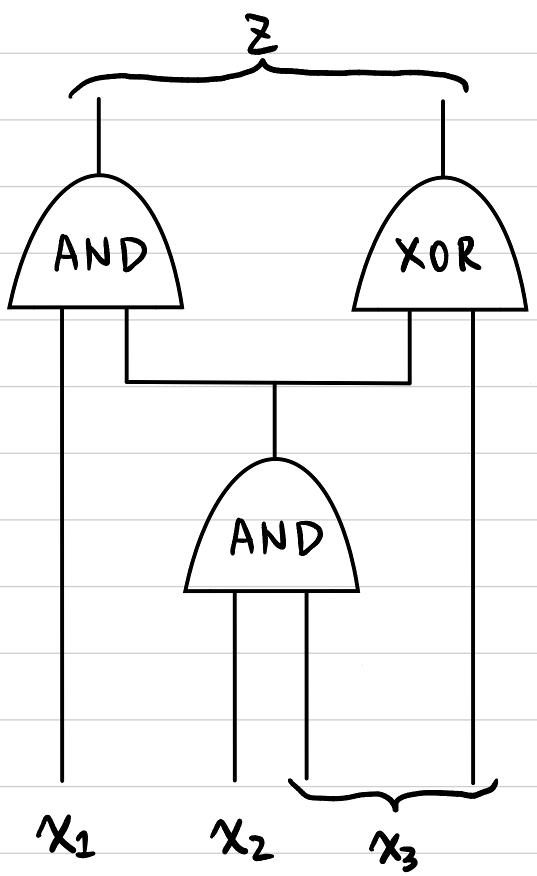


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\qquad \bigoplus_{i=1}^{n} b_i = b$

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

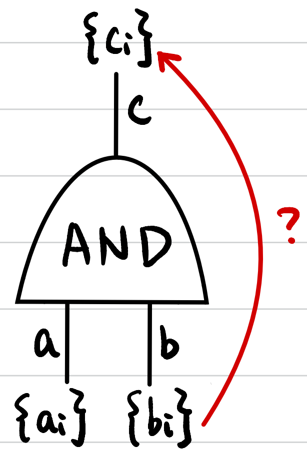
$c_i = ?$

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

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

$c_i = ?$

Outputs:

For each output wire w:

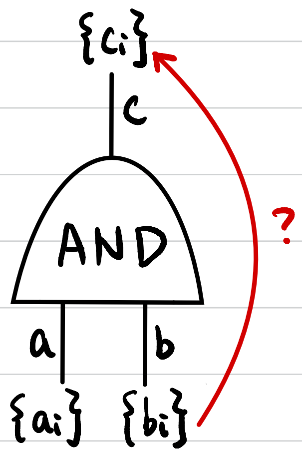
Each party P_i holds a random share $v_i^w \in \{0,1\}$

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

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

$c_i = ?$

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

$$= \Big(\underbrace{\sum_{i=1}^{n} a_i \cdot b_i}_{P_i \text{ locally}}\Big) + \Big(\underbrace{\sum_{i \neq j} a_i \cdot b_j}_{?}\Big) \mod 2$$

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

Reshare:

P_i | P_j

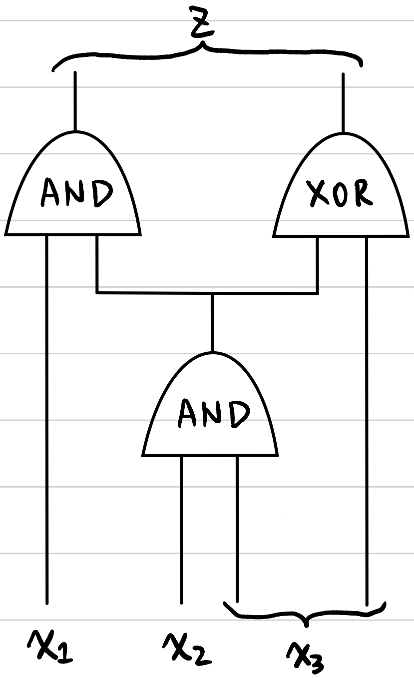
Input: $a_i \in \{0,1\}$ | Input: $b_j \in \{0,1\}$

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

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

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 $t \le n-1$ (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:

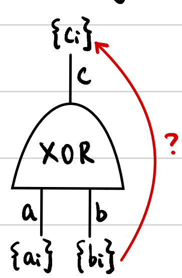
For each input wire w:

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:

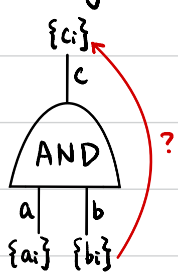


GIVEN: $\bigoplus_{i=1}^{n} a_i = a$ $\bigoplus_{i=1}^{n} b_i = b$

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

$c_i = ?$

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

$$= \Big(\underbrace{\sum_{i=1}^{n} a_i \cdot b_i}_{P_i \text{ locally}}\Big) + \Big(\underbrace{\sum_{i \ne j} a_i \cdot b_j}_{\text{Reshare}}\Big) \mod 2$$

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?

	P_1	P_2	$\cdots$	P_n
Round 1	$\swarrow \downarrow \searrow$	$\swarrow \downarrow \searrow$		$\swarrow \downarrow \searrow$
Round 2	$\swarrow \downarrow \searrow$	$\swarrow \downarrow \searrow$		$\swarrow \downarrow \searrow$
$\vdots$				
Round r	$\swarrow \downarrow \searrow$	$\swarrow \downarrow \searrow$		$\swarrow \downarrow \searrow$

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)