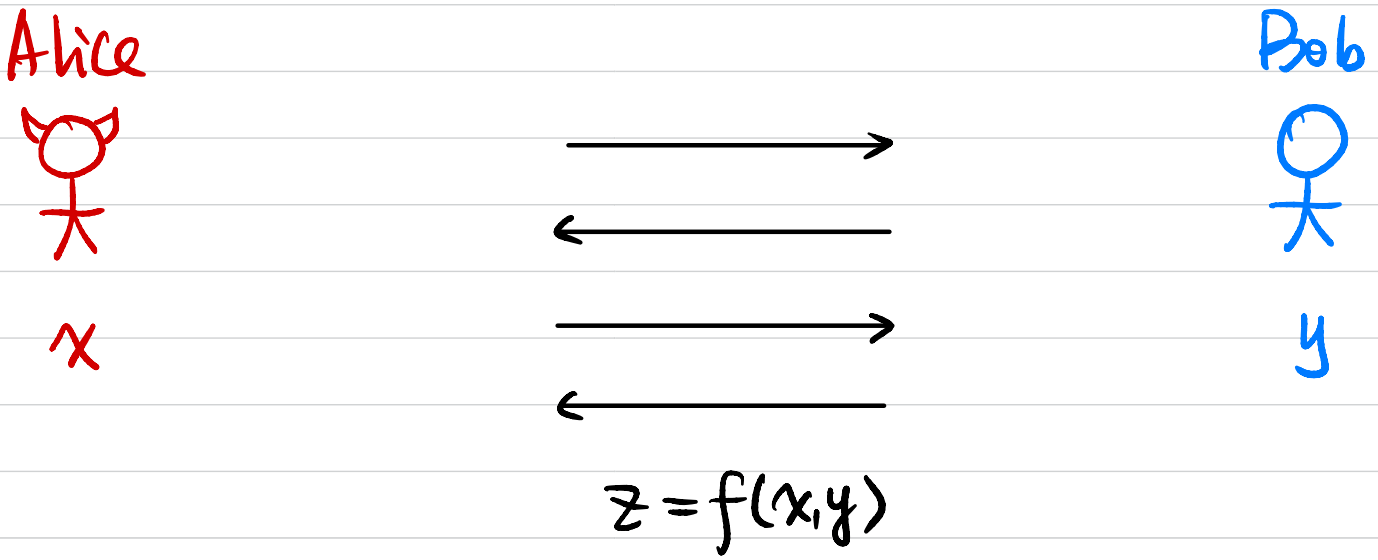
CSCI 1515 Applied Cryptography

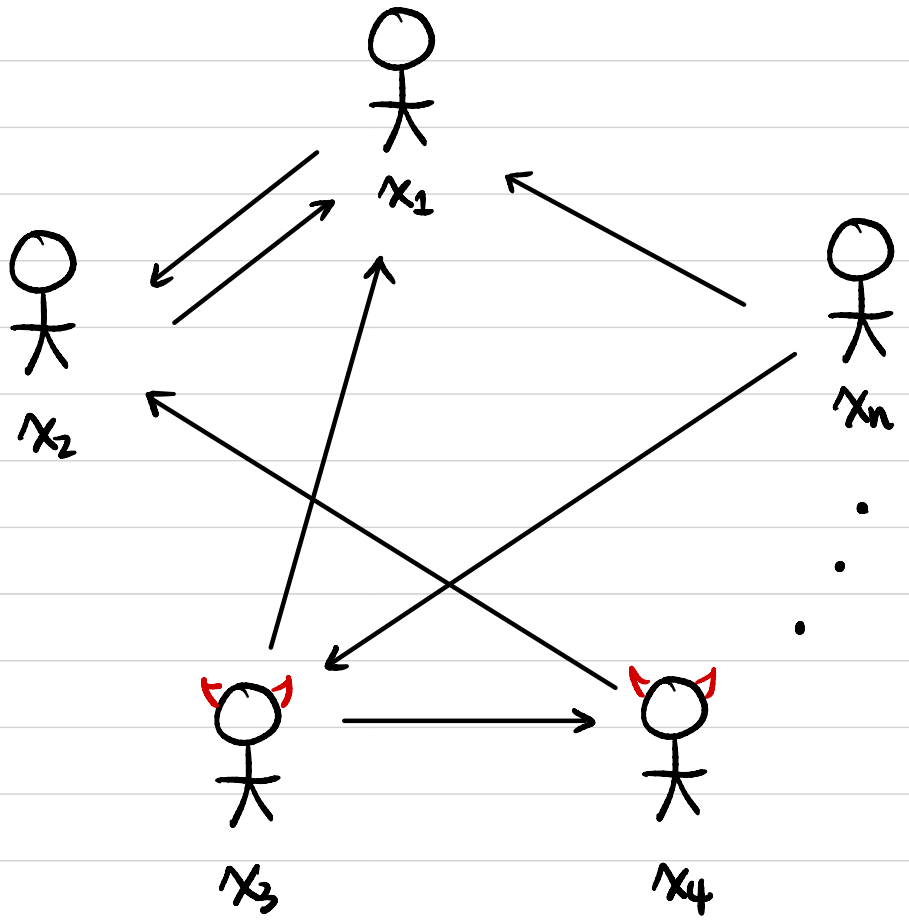
This Lecture:

- Feasibility Results of MPC
- Yao's Garbled Circuits and Optimizations

Secure Two-Party Computation (2PC)



Secure Multi-Party Computation (MPC)



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

Setting

- n parties $P_1, P_2, \cdots, P_n$
  with private inputs $x_1, x_2, \cdots, x_n$

- Jointly compute $f(x_1, x_2, \cdots, x_n)$

- Communication:
  Authenticated secure point-to-point channels between each pair (P_i, P_j)
  (sometimes also assume broadcast channel)

- The adversary can "corrupt" a subset of the parties
  (e.g. at most t parties)

General Security Properties

- **Correctness:** The function is computed correctly.
- **Privacy:** Only the output is revealed.
- **Independence of Inputs:** Parties cannot choose inputs depending on others' inputs.

- **Security with Abort:** Adversary may "abort" the protocol.
  (preventing honest parties from receiving the output)
- **Fairness:** If one party receives output, then all receive output.
- **Guaranteed Output Delivery (GOD):** Honest parties always receive output.

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.

Adversary's Computing Power:

- **Unbounded computing power** ⇒ Information-Theoretic (IT) Security
- **PPT bounded** ⇒ Computational Security

Feasibility Results

Computational Security:

semi-honest Oblivious Transfer (OT)

semi-honest MPC for any function with $t < n$ (# corrupted parties)

malicious MPC for any function with $t < n$

Information-Theoretic (IT) Security:

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

necessary

Oblivious Transfer (OT)

Sender

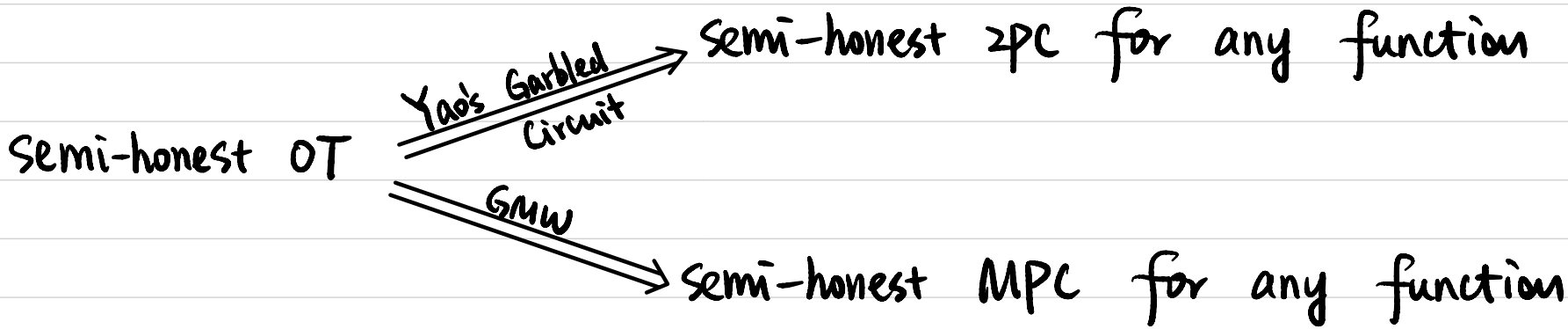
Receiver

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

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

Output: $\perp$

Output: m_b

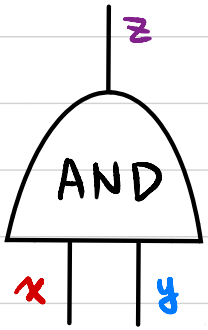

semi-honest OT
Yao's Garbled Circuit
semi-honest 2PC for any function
GMW
semi-honest MPC for any function

Example: Private Dating

Alice

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

$f(x,y) = x \wedge y$



Bob

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

Truth Table:

$x=0$	$y=0$	$\Rightarrow$	$z=0$
$x=0$	$y=1$	$\Rightarrow$	$z=0$
$x=1$	$y=0$	$\Rightarrow$	$z=0$
$x=1$	$y=1$	$\Rightarrow$	$z=1$

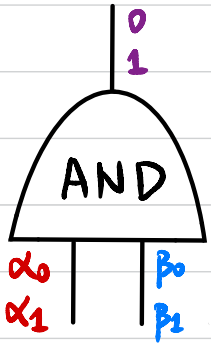
Example: Private Dating

Alice

Bob

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

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



$\alpha_0, \alpha_1, \beta_0, \beta_1 \xleftarrow{\$} \{0,1\}^\lambda$

(labels)

Garbled Truth Table:

α_0	β_0	$\Rightarrow$	0
α_0	β_1	$\Rightarrow$	0
α_1	β_0	$\Rightarrow$	0
α_1	β_1	$\Rightarrow$	1

$\text{Enc}_{\alpha_0}(\text{Enc}_{\beta_0}(0))$ ←

$\text{Enc}_{\alpha_0}(\text{Enc}_{\beta_1}(0))$

$\text{Enc}_{\alpha_1}(\text{Enc}_{\beta_0}(0))$ ←

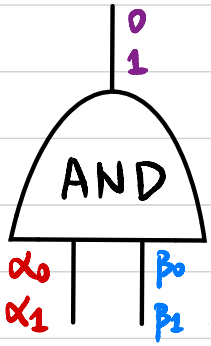
$\text{Enc}_{\alpha_1}(\text{Enc}_{\beta_1}(1))$ ←

Garbled Gate

Example: Private Dating

Alice

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



$\alpha_0, \alpha_1, \beta_0, \beta_1 \xleftarrow{\$} \{0,1\}^\lambda$

(labels)

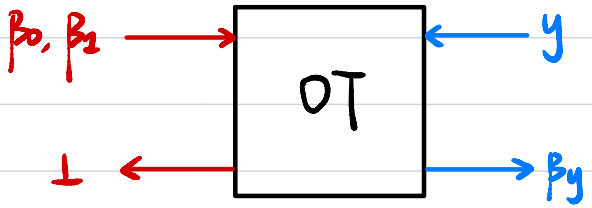
Bob

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

$\mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_0}(0))$
$\mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_1}(0))$
$\mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_0}(0))$
$\mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_1}(1))$

Input label for x: α_x

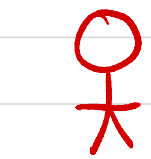
Input label for y?



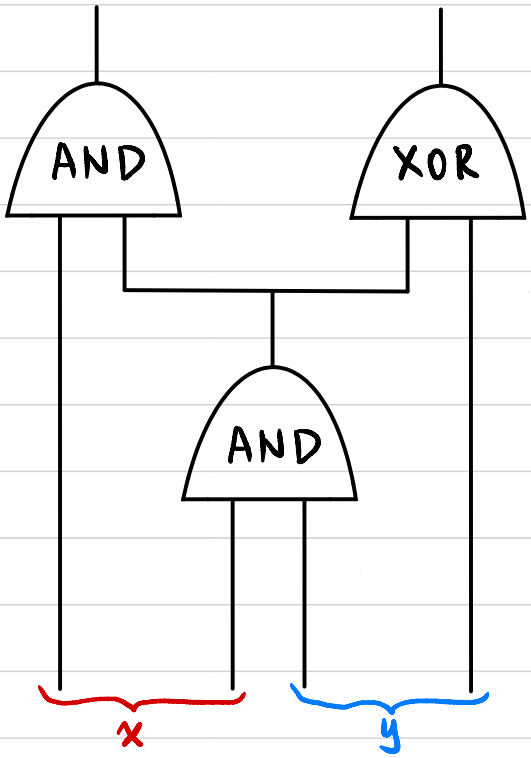
z

Arbitrary Function → Represent it as a Boolean circuit

Alice



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

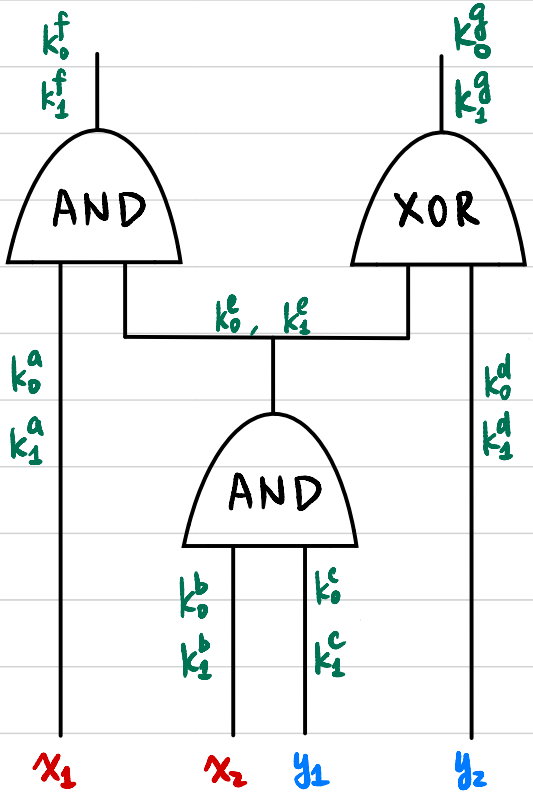


Bob

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

Yao's Garbled Circuit

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



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

Each label $\xleftarrow{\$} \{0,1\}^\lambda$

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

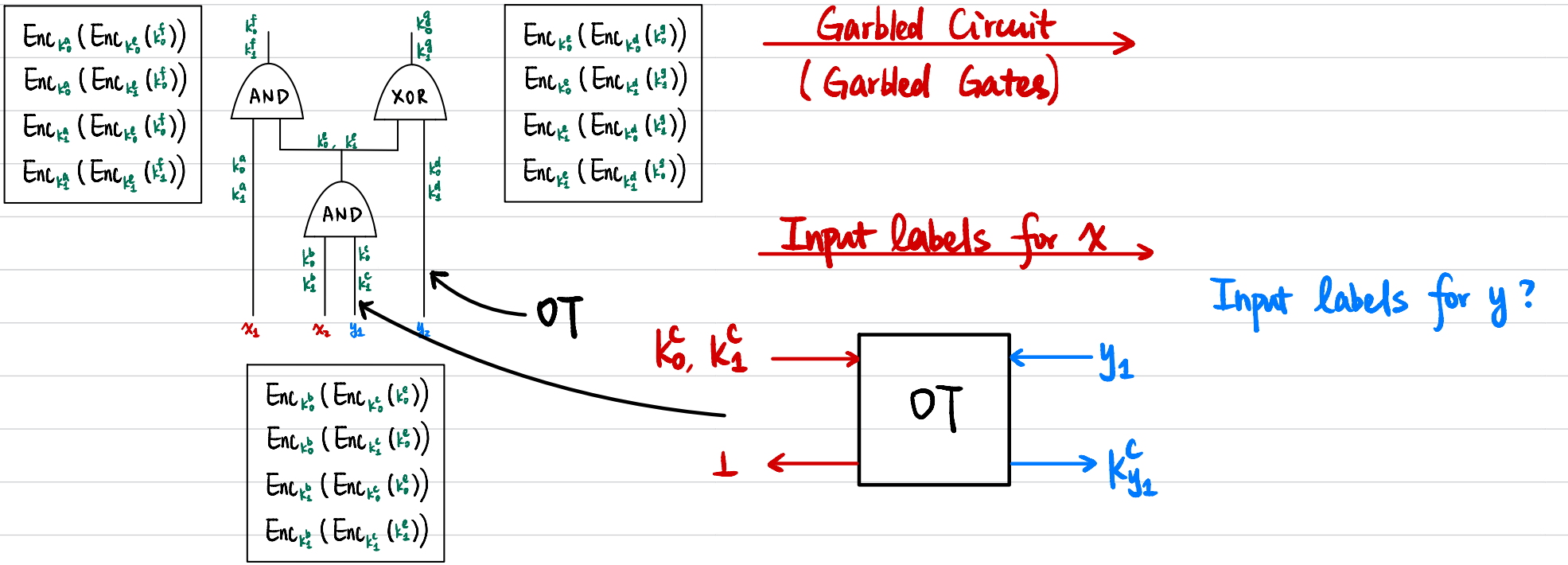
Secure 2PC

Alice (Garbler)

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

Bob (Evaluator)

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



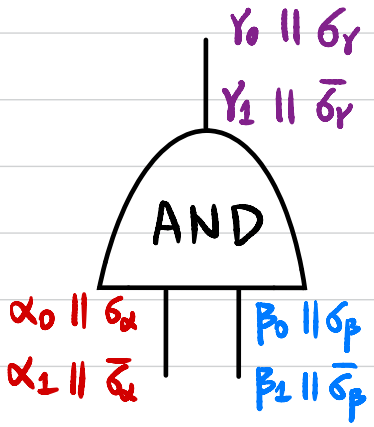
- Output labels?
  - Set $k_0^f = k_0^g = 0$; $k_1^f = k_1^g = 1$
  - Garbler sends (k_0^f, k_1^f), (k_0^g, k_1^g) to evaluator
  - Evaluator sends k^f, k^g back to garbler
- How to decide which ciphertext to decrypt?

Shuffle:

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

2λ

Optimization 1: Point-and-Permute



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

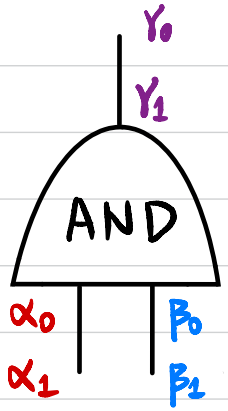
$\sigma_\alpha \quad \sigma_\beta : \mathrm{Enc}_{\alpha_0}(\mathrm{Enc}_{\beta_0}(\overbrace{Y_0}^{\lambda} \| \overbrace{\sigma_\gamma}^{1}))$

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

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

$\bar{\sigma}_\alpha \quad \bar{\sigma}_\beta : \underbrace{\mathrm{Enc}_{\alpha_1}(\mathrm{Enc}_{\beta_1}(Y_1 \| \bar{\sigma}_\gamma))}_{\lambda+1}$

Optimization 2: Row Reduction



~~$H(\alpha_0 \| \beta_0) \oplus Y_0$~~

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

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

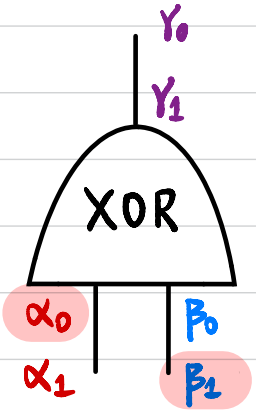
$H(\alpha_1 \| \beta_1) \oplus Y_1$

↑

Random Oracle

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

Optimization 3: Free XOR



hidden to evaluator

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

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

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

$Y_1 := Y_0 \oplus \Delta$

$Y_0 := \alpha_0 \oplus \beta_0$

$$\alpha_0 \oplus \beta_1$$
$$= \alpha_0 \oplus (\beta_0 \oplus \Delta)$$
$$= (\alpha_0 \oplus \beta_0) \oplus \Delta$$
$$= Y_0 \oplus \Delta$$
$$= Y_1$$

Other Optimizations:

- Half-gates: 2λ bits per AND gate + free XOR
- Slicing-and-dicing: $\sim 1.5\lambda$ bits per AND gate + free XOR