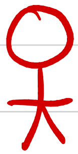
CSCI 1515 Applied Cryptography

This Lecture:

- Introduction to Secure Multi-Party Computation
- Feasibility Results of MPC
- Construction of Oblivious Transfer

Secure Multi-Party Computation

Alice



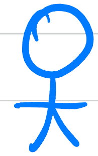
$x \in \{0,1\}$

Second date?

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

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

Bob



Who is richer?

x

$$f(x,y) = \begin{cases} 0 & \text{if } x < y \\ 1 & \text{Otherwise} \end{cases}$$

y

Mutual friends?

X

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

Y

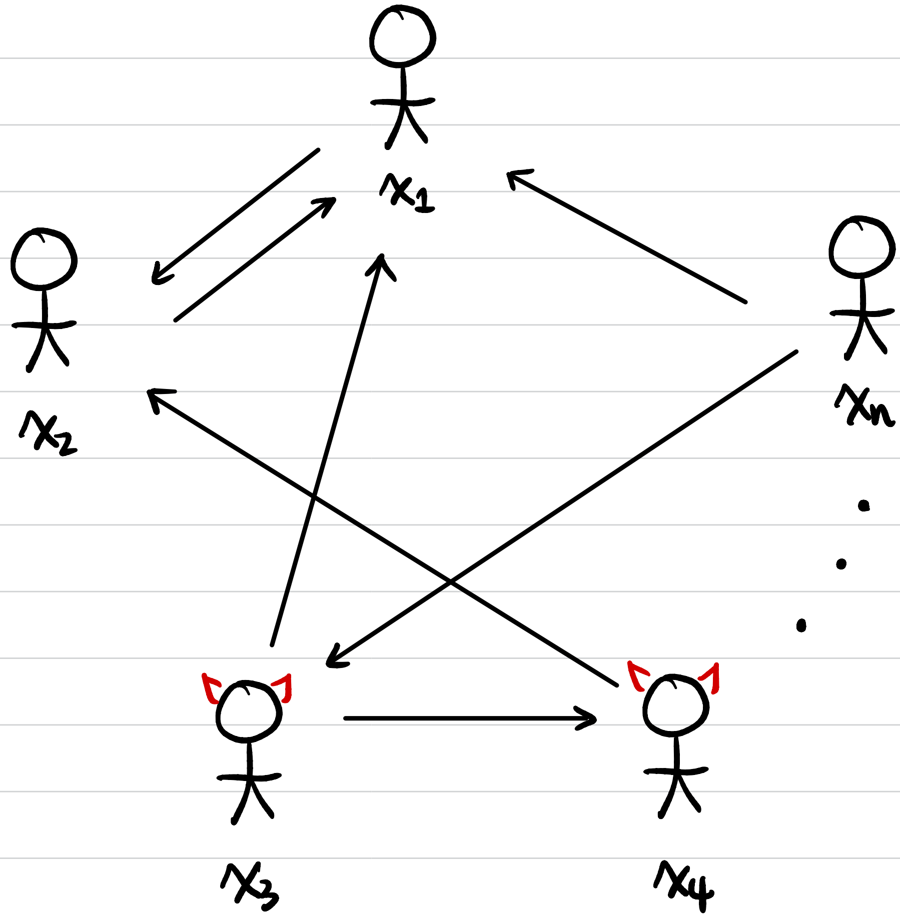
Secure Two-Party Computation (2PC)

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

Applications:

- Password Breach Alert (Chrome / Firefox / Azure / iOS Keychain)
- Privacy-Preserving Contact Tracing for COVID-19 (Apple & Google)
- Ads Conversion Measurements / Personalized Advertising (Google / Meta)

Secure Multi-Party Computation (MPC)



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

Secure Multi-Party Computation (MPC)

Applications:

- Privacy-Preserving Inventory Matching (J.P. Morgan)
- Setup Ceremony to securely generate CRS (Zcash)
- Distributed Key Management (Unbound / Coinbase)
- Federated Learning (Google Keyboard Search Suggestion)
- Auctions (Danish sugar beet auction)
- Boston gender wage gap (Boston Women's Workforce Council)

- Study / Analysis on Medical Data
- Fraud / Money Laundering Detection (banks)

Setting

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

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

- 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)

What properties do we want?

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.

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)

$\Downarrow$

\# corrupted parties ↙

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

$\Downarrow$

malicious MPC for any function with $t < n$

Information-Theoretic (IT) Security:

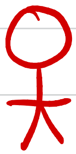
(honest majority)

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

↑ necessary

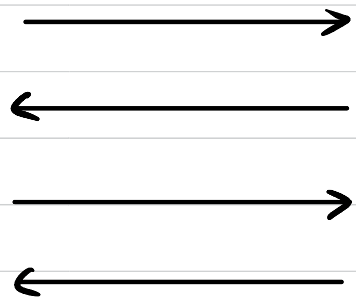
Oblivious Transfer (OT)

Sender



Receiver

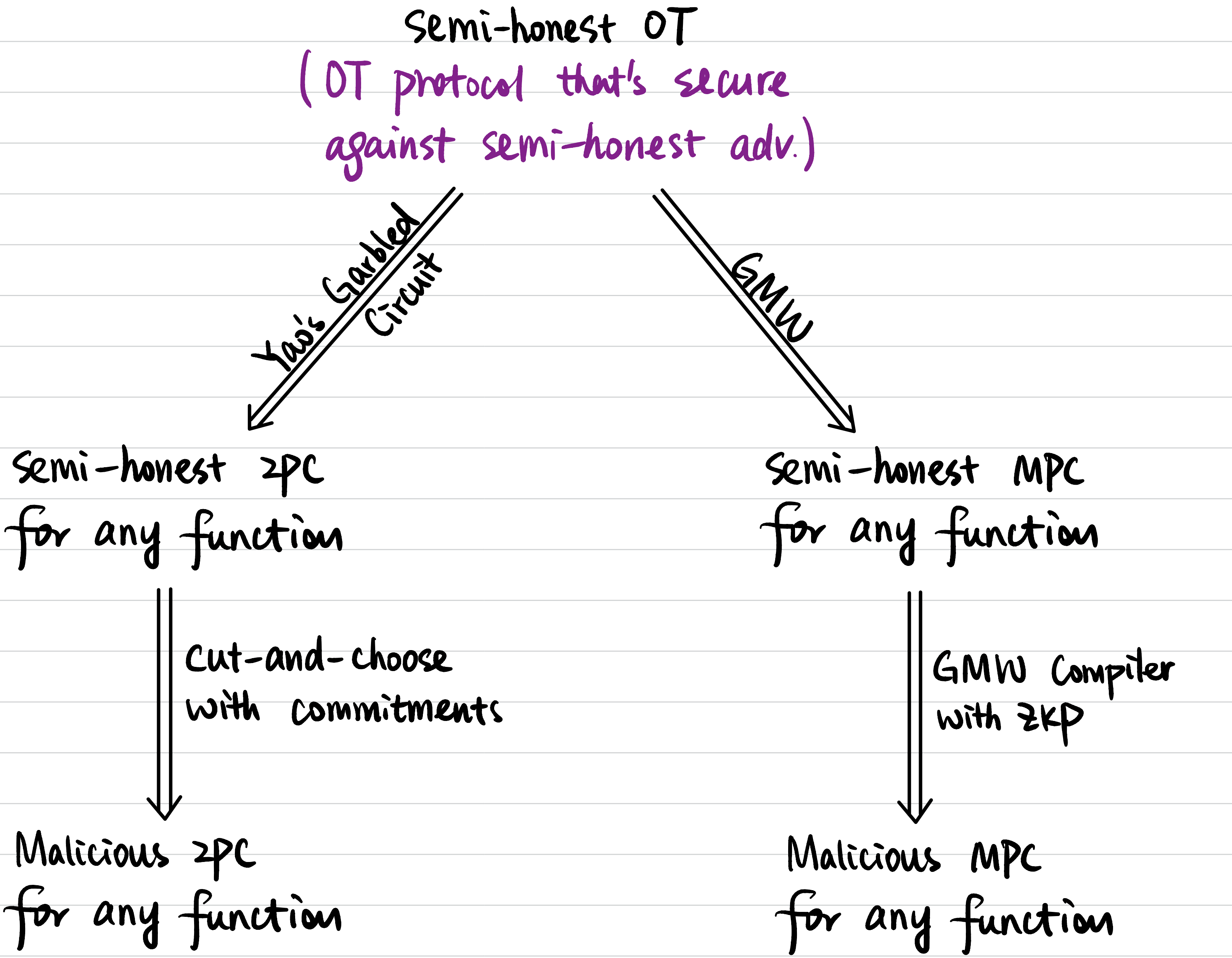
Input: $m_0, m_1 \in \{0,1\}^\ell$



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

Output: $\perp$

Output: m_b


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

Oblivious Transfer (OT)

Sender

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

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

$\xrightarrow{A = g^a}$

Receiver

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

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

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

$k_0 := H(B^a)$

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

$ct_0 \leftarrow Enc_{k_0}(m_0)$

$ct_1 \leftarrow Enc_{k_1}(m_1)$

$\longrightarrow$

Output:

$k_c := H(A^b) = H(g^{ab})$

$m_c := Dec_{k_c}(ct_c)$

Why is it correct?

Why is it secure against semi-honest Sender?

Why is it secure against semi-honest Receiver?

OT Extension

Can we construct OT from symmetric-key primitives only?

Unlikely! (theoretical impossibility)

$O(\lambda)$ Base OTs —— extend ——> N OTs

$O(\lambda)$ Base OTs ↑ $O(\lambda)$ public-key operations

N OTs ↑ $O(\lambda)$ public-key operations + $O(N)$ symmetric-key operations