CSCI 1515 Applied Cryptography

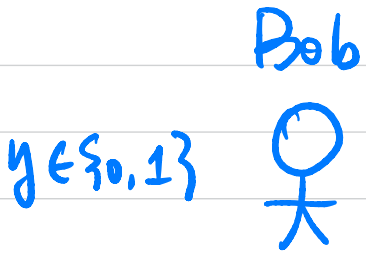
This Lecture:

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

Secure Multi-Party Computation

Second date?

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



Who is richer?

x

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

y

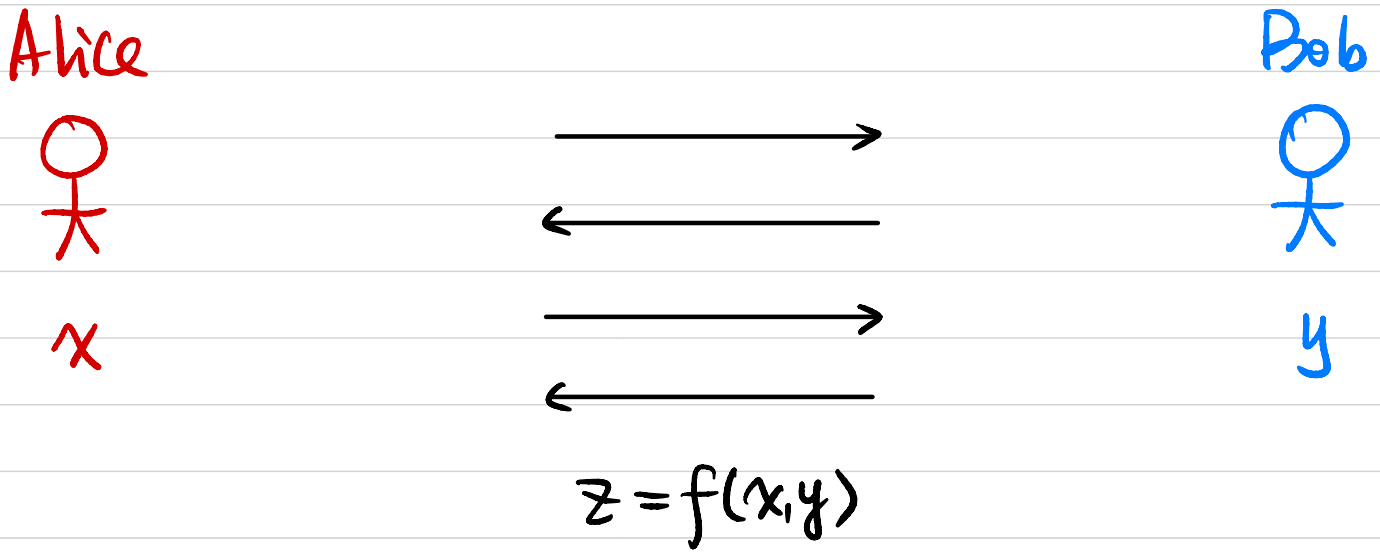
Mutual friends?

X

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

Y

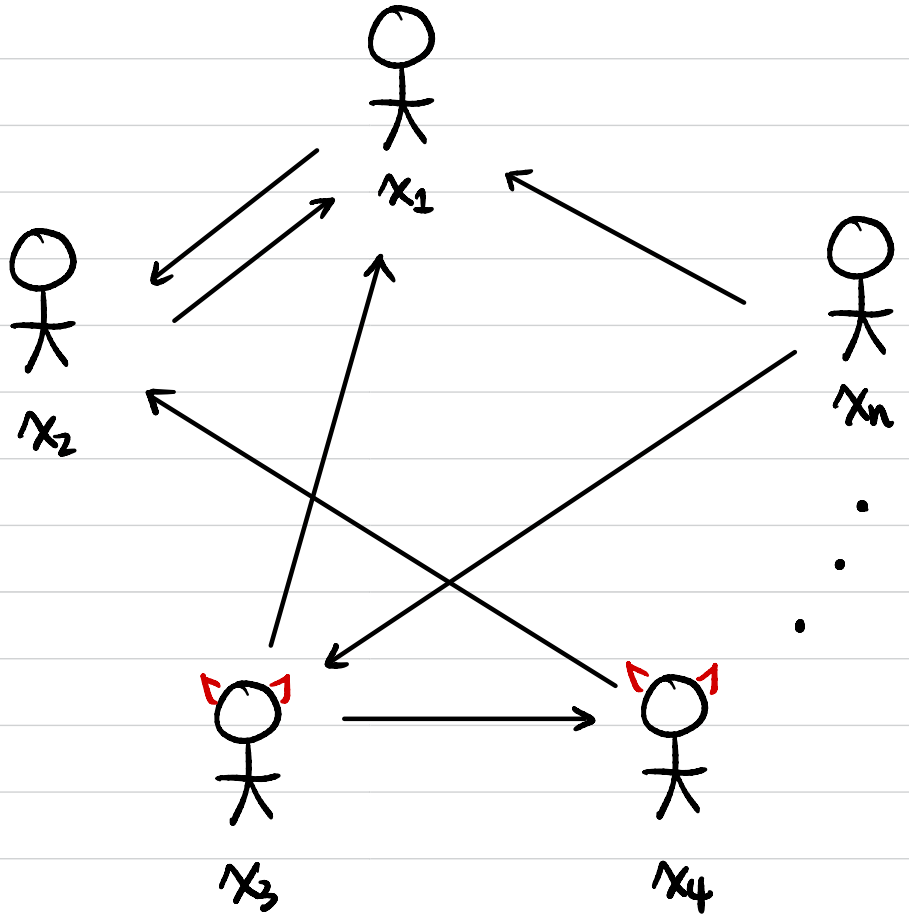
Secure Two-Party Computation (2PC)



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)

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

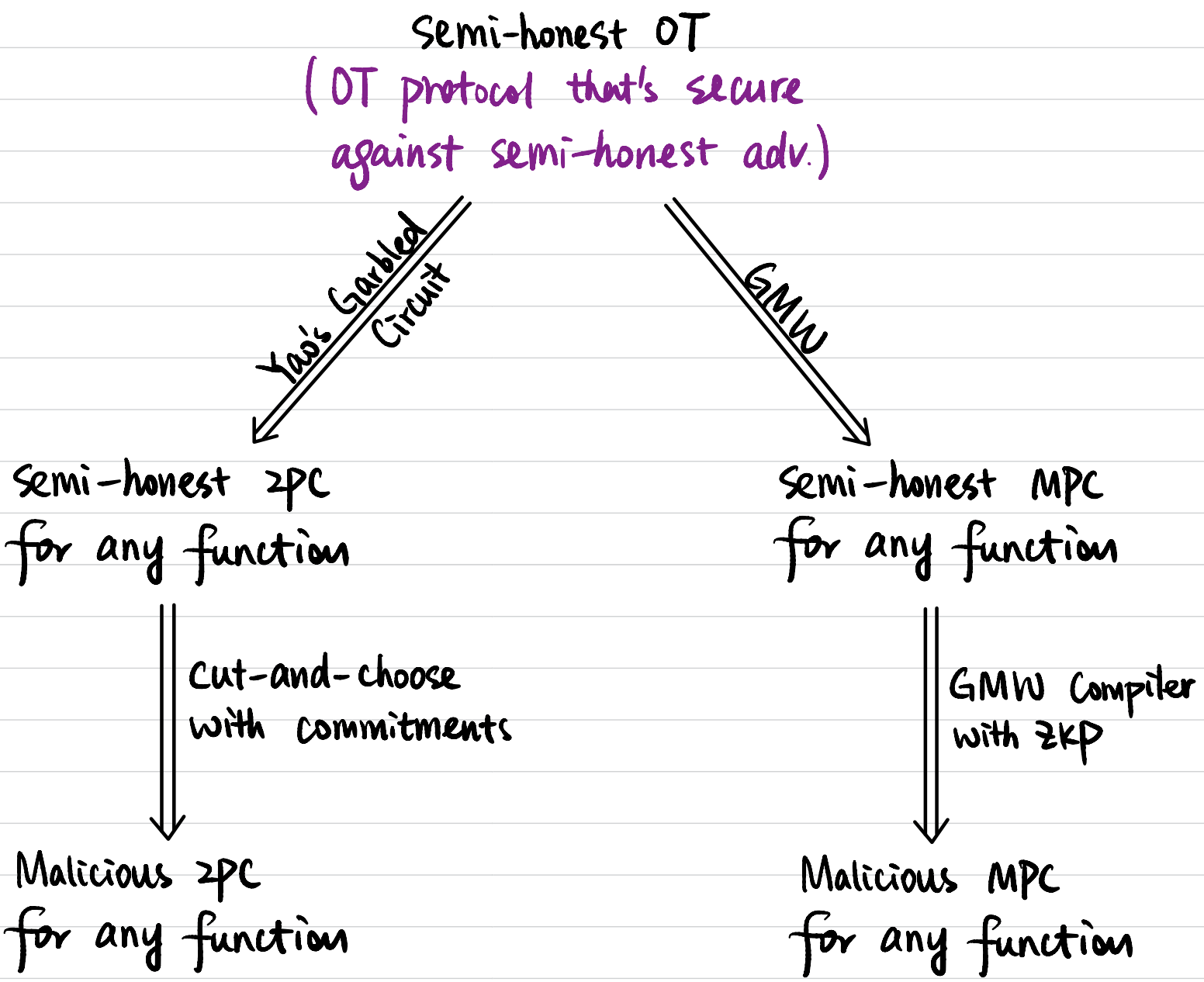
Receiver

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

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

Output: $\perp$

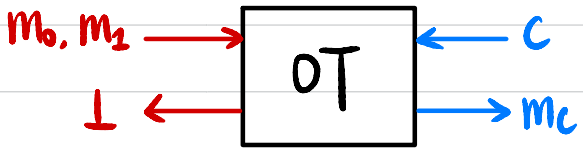
Output: m_b


Semi-honest OT
(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)

DLOG: $g^a \not\to a$

CDH: $(g^a, g^b) \not\to g^{ab}$



Sender

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

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

$A = g^a$ →

← $B = g^b \cdot A^c$

$c=0$: $B = g^b$

$k_0 := H(B^a)$ $k_0 = H(g^{ab})$

$k_1 := H((\frac{B}{A})^a)$ $k_1 = H(g^{ab})$

$k_1 = H(g^{ab-a^2})$

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

$ct_1 \leftarrow Enc_{k_1}(m_1)$ →

Receiver

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

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

If $g^a \xrightarrow{A} g^{a^2}$

Then $(g^a, g^b) \xrightarrow{B} g^{ab}$

B: $A(g^a) \to g^{a^2}$

$A(g^b) \to g^{b^2}$

$A(g^{a+b}) \to g^{(a+b)^2} = g^{a^2+b^2+2ab}$ $\Big\} \to (g^{2ab})^{2^{-1}} = g^{ab}$

$c=0$: $B = g^b$

$c=1$: $B = g^b \cdot A$

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)