Two-Party Computation (2PC) from Garbled Circuits and Oblivious Transfer
Multi-Party Computation (MPC)
- also Secure Computation (SC)
- today: 2PC
  - Alice knows \( x \in \{0,1\}^k \)
  - Bob knows \( y \in \{0,1\}^k \)
  - both know \( f(x,y) = (0,1)^k \)
  - the "functionality"
- goal: both learn \( f(x,y) \), and nothing else
- Note: can also support Alice learning \( f_A(x,y) \) and Bob learning \( f_B(x,y) \) - not covered today.

Applications of 2PC
1. Yao's millionaire problem: Alice has \( x \) dollars, Bob has \( y \).
   \( f(x,y) \) output 1 if \( x \geq y \), otherwise 0.
   (Alice & Bob have... problems...)

2. Private Advertising Campaign Evaluation:
   - Google knows who saw what ad:
   - Macy's knows who bought stuff
     - Activity: Neil
     - Bill
     - Wilson
     - Output:
     - Select Count (*)
     - From Google Join Macy's
     - Where ad = "Blender"

3. Private Contact Discovery: Signal knows its users' phone #s
   - I know my contacts list
   - I want to learn the intersection; signal should learn nothing
   - "Private Set Intersection" (PSI)

4. Zero Knowledge (sort of)
   - Prover knows \( (x,w) \)
   - Verifier knows \( x \)
   - Verifier learns \( (x,w) \in R \)

Definition & Security
Let \((A,B)\) be an interactive protocol for functionality \( f: \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k \). Note: \(A,B\) are randomized, interactive algorithms.

It is correct if for all \( x,y \in \{0,1\}^k \):
\[
\Pr[A(x),B(y) = f(x,y)] = 1
\]

Two popular security types:
- Semi-honest: Adversarial parties follow the protocol but inspect their data to try to learn more (~ HVEK) (aka honest-but-curious)
- Malicious: Adversarial parties may deviate from the protocol (~ EIK)

Today: Semi-honest, until the very end

Semi-honest security: Privacy
- 3 two efficient simulators \( S_a, S_b \) s.t. \( \forall x,y \in \{0,1\}^k \)
  \[
  \begin{align*}
  &\Pr[S_a(x,f(x,y)) \in \{\text{view}([A,w],[B,y])\}] \\
  &\Pr[S_b(y,f(x,y)) \in \{\text{view}([A,w],[B,y])\}] 
  \end{align*}
  \]

Example MPC: Oblivious Transfer (OT) (Construction by Bellare & Micali)
- Sender knows \( m_0,m_1 \in \{0,1\}^k \) → learn nothing
- Receiver knows \( m \) → learn \( m_0 \)

Question: if Sender sends \( m_0 \) \& \( m_1 \), is that a secure 2PC for the OT functionality? No! Receiver's simulator can't simulate \( m_1 \)
**Instantiating the security definitions**

Protocol \( (S,R) \) is secure if:

- Correctness: \( \Pr[\text{output}\{\text{Sim}(m,m,e\{0,1\}^e)\}\neq m] = 1 \)
- Sender privacy: \( \exists \text{ all} \ Sim \ s.t. \ \forall m,m.e\{0,1\}^e \ \forall b \neq (0,1) \)
  \[ \text{Sim}(b,m) \equiv \text{View}(\{\text{Sim}(m,m,e\{0,1\}),b\}) \]
- Receiver privacy: \( \exists \text{ all} \ Sim \ s.t. \ \forall m,m.e\{0,1\}^e \ \forall b \neq (0,1) \)
  \[ \text{Sim}_m(m,m) \equiv \text{View}(\{\text{Sim}(m,m,e\{0,1\}),b\}) \]

**Bellare & Micali OT**

Based on El-Gamal encryption variant:

- \( S(m_0,m_1,e\{0,1\}^e) \)
- \( \text{Pr}[\text{output}\{\text{Sim}(m_0,m_1,e\{0,1\}^e)\}\neq m_0] = 1 \)
- Sender privacy:
  \[ \forall m_0,m_1.e\{0,1\}^e \ \forall b \neq (0,1) \]
  \[ \text{Sim}(b,m) \equiv \text{View}(\{\text{Sim}(m_0,m_1,e\{0,1\}),b\}) \]
- Receiver privacy:
  \[ \forall m_0,m_1.e\{0,1\}^e \ \forall b \neq (0,1) \]
  \[ \text{Sim}_m(m_0,m_1) \equiv \text{View}(\{\text{Sim}(m_0,m_1,e\{0,1\}),b\}) \]

**Analysis**

Correctness: \( \text{Sim}(m_0,m_1) = \text{h}_1 \cdot \text{h}_2 \), output \( (b, c/m) \) (generate rest of transcript per the protocol)

Sender Privacy:
- First, show that the EG variant is semantically secure/CPA secure (equiv. for PKE)
  - CDH implies guess \( \text{pr}_{\text{E}} \) is hard to guess. RO implies \( \text{h}(\text{pr}_{\text{E}}) \) is hard to guess
  \[ \text{Sim}(b,m) : \]
  \[ k \in \mathbb{Z}_k \]
  \[ c \in \mathbb{G}_1 \]
  \[ \text{Y} = \text{g}^k \]
  \[ \text{C} = \text{h}(\text{pr}_{\text{E}}) \]
- Output \( (C, K, K_0, K_1) \)
- Next, show \( \neq \) real view (compare)
  \[ (C, K, \text{Enc}(\text{g}^k, m_0, m_1), \text{Enc}(\text{g}^k, m_0, m_1)) \]
- If some \( A \) can distinguish, can attack CPA security of \( \text{E} \).

Yao's Garbled Circuits (2PC)

- 2PC for any boolean circuit \( f \)
  - Directed acyclic graph
  - 2-input AND, XOR gates with unlimited fan-out
  - input wire for each bit of input
  - output wire for each bit of output
  - Garbling, high-level idea: (see OT + Sym. Ciph.)
    1. Alice sends a "garbled" circuit to Bob
    2. Bob uses OT to get information to evaluate the circuit on the correct inputs only.
    3. Alice "degarbles" the output

**Warm-Up: Garbling a single AND-gate**

- Let \( (E,D) \) be a symmetric cipher with key space \( \mathbb{K} \)
  - Note: we'll end up requiring some unusual (but not unrealistic) properties of \( (E,D) \)
  
  \[ f(x,0) = 0 \]
  
  \[ f(x,1) = x \]

- Garbler (Alice):
  1. Sample two keys for each input wire \( K_{x,0}, K_{x,1}, K_{y,0}, K_{y,1} \in \mathbb{K} \)
  2. Output wire too \( K_{x,0}, K_{x,1}, K_{y,0}, K_{y,1} \in \mathbb{K} \)

- Note: ignore the red stuff till later.
2. Garble each gate:

\[
\begin{align*}
\text{Garbling } C &\text{ is } \{ E(k_{x},E(k_{y},0)), E(k_{x}, E(k_{y},1)) \}_{a_{x},b_{x},a_{y},b_{y}} \text{ in a randomized order.}
\end{align*}
\]

3. Sends the garbling and $k_{x}$ where $R$ is the value of $i$, to Receiver.

4. Evaluators use $m_{x} \leftarrow D(k_{x}, D(k_{x}, C))$ for $x \in C$.

5. Of the $m_{x}$'s will be $1$ a decryption error since we used the wrong key.

6. He non-$1$ $m_{x}$ will be $\overline{\text{AND}}(x, y)$.

7. Receiver sends $\overline{z}$ to garbler.

Yao's actual protocol (full circuit):

The previous protocol has flow inputs as "garbled" or keys, but the gate outputs are in the clear - limits composition.

Let's make a few changes... See red above.

Now, the multi-gate protocol is:

1. Garbler:
   a. Samples $k_{x_{0}}, k_{x_{1}}, \ldots, k_{x_{r}}$, for each wire $x_{i}$.
   b. Receive $m_{i} \leftarrow E(k_{x_{i}}, E(k_{y_{i}}, 0))$ for each gate $y_{i} \in C$.
   c. Sends $c_{i}$ for all gate $y_{i}$ and $k_{x_{i}}$ for wire $x_{i}$ corresponding to input $x_{i}$ of value $x_{i}$.

2. Evaluators use OTs to obtain $k_{x_{i}}, y_{i}$ for wire $y_{i}$ corresponding to input $y_{i}$ of value $y_{i}$.

3. Evaluators evaluate gate-by-gate:

4. Evaluators sends outputs keys to garbler, who replies with output bits.

Visualizing the multi-gate protocol:

- Note: Not maliciously secure. (see this problem)

- Optimizations go: one aim of research
  - Avoid trial decryption
  - "half-gates": two $c$'s per gate, instead of 4.
- "OT extension": a mechanism for doing $O(n)$ OTs with $O(1)$ group operations (instead of $O(n)$)
  - Important if there are many inputs
- "free XOR"
  - At protocol start, sample $R \leftarrow M$ (mix).
  - For all $x_{w}$, sample $k_{x_{w}} \in M$, set $k_{x_{w}} \leftarrow k_{x_{w}} \oplus R$
  - Note: For $x_{w} \leftarrow y_{w}$, fix $k_{x_{w}} = k_{x_{w}} \oplus k_{y_{w}}$
  - then $k_{x_{w}} \oplus k_{y_{w}} = k_{x_{w}} \oplus (k_{y_{w}} \oplus k_{y_{w}}) = k_{y_{w}}$
  - $k_{y_{w}} \oplus k_{y_{w}} = k_{y_{w}} \oplus k_{y_{w}} = k_{y_{w}}$
  - Thus, no additional $k_{x_{w}}$ @ gate! No garbling needed!

- "Free If": Given $\text{MUX}_{x_{w}}$ in circuit $\oplus$, and trial time roughly proportional to the longest path (not sum).
Malicious Security

Problem: A might not follow the protocol
- sample biased randomness
- send wrong messages
- attempt to A their input mid-protocol
- refuse to send some messages
- can be devastating: GC is totally insecure against a malicious adversary (HW problem)

- How to obtain malicious security?
  - Beautiful idea: [GMW’87]
    1) Commit to inputs/randomness
    2) For each m from P, P proves in zero knowledge that m is the correct message

- Somewhat inefficient

- More efficient approaches exist (perhaps next lecture).