Two-Party Computation (2PC) from Garbled Circuits and Oblivious Transfer

Today:

- ☐ Multi-Party Computation (& 2PC)
- ☐ Oblivious Transfer (specific 2PC)
- ☐ Yao's Garbled Circuits (2PC for boolean circuits)
- ☐ Malicious Security

Multi-Party Computation (MPC)

- a.k.a. Secure Computation (SC)
- today: 2 PC
  - Alice knows $x \in \{0,1\}^\ell$
  - Bob knows $y \in \{0,1\}^\ell$
  - both know $f: \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^m$
    ↑ 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 $x>y$, otherwise 0.
     (Alice & Bob have... problems...)
  2. Private Advertising Campaign Evaluation:
     Google knows who saw what ad:

user	ad
Alex	Shoes
Neil	Blender
Wilson	Blender
⋮	

     Macy's knows who bought stuff

user
Neil
Jane
⋮

     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 <u>the intersection</u>; signal should learn nothing
     "Private Set Intersection" (PSI)
  4. Zero Knowledge (sort of)
     Prover knows (x,w)
     Verifier knows x
     Verifier learns $(x,w) \overset{?}{\in} R$

Definitions & Security

Let (A,B) be an interactive protocol (not proof) for functionality $f: \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^m$. Note: A,B are randomized, interactive algorithms.

It is correct if for all $x,y \in \{0,1\}^\ell$:

$$\Pr[\langle A(x), B(y)\rangle = f(x,y)] = 1$$

Two popular security types:

- Semi-honest: Adversarial parties follow the protocol but inspect their data to try to learn more (~ HVZK) (aka honest-but-curious)
- Malicious: Adversarial parties may deviate from the protocol (~ ZK)

Today: Semi-honest, until the very end

Semi-honest security: Privacy

∃ two efficient simulators S_A, S_B s.t. $\forall x,y \in \{0,1\}^\ell$

$$\{S_A(x, f(x,y))\} \approx_c \{View_A[\langle A(x), B(y)\rangle]\}$$
$$\{S_B(y, f(x,y))\} \approx_c \{View_B[\langle A(x), B(y)\rangle]\}$$

Example MPC: Oblivious Transfer (OT) (Contribution by Bellare & Micali)

- Sender knows $m_0, m_1 \in \{0,1\}^\ell$ → learns nothing
- Receiver knows b. → learns m_b
- Question: if Sender sends m_0 & m_1, is that a secure 2PC for the OT functionality? No! Reciever's simulator can't simulate m_{1-b}

Instantiating the security definitions:

Protocol (S,R) is a secure OT if:

Correctness: $\forall m_0, m_1 \in \{0,1\}^\ell$: $\Pr[\text{output}_R(\langle S(m_0,m_1), R(b)\rangle) = m_b] = 1$

Sender privacy: $\exists$ eff. Sim_R s.t. $\forall m_0, m_1 \in \{0,1\}^\ell$ $\forall b \in \{0,1\}$

$$\text{Sim}_R(b, m_b) \approx \text{View}_R[\langle S(m_0,m_1), R(b)\rangle]$$

Reciever privacy: $\exists$ eff Sim_S s.t. $\forall m_0, m_1 \in \{0,1\}^\ell$ $\forall b \in \{0,1\}$

$$\text{Sim}_S(m_0, m_1) \approx \text{View}_S[\langle S(m_0,m_1), R(b)\rangle]$$

Bellare & Micali OT

Based on El-Gamal encryption variant:

Let $\langle g\rangle = G$, $|G| = q$, $H: G \to \{0,1\}^\ell$ (random oracle)

- $sk \xleftarrow{\$} \mathbb{Z}_q$, $pk \leftarrow g^{sk}$
- $\text{Enc}(m \in \{0,1\}^\ell, pk)$: $r \xleftarrow{\$} \mathbb{Z}_q$, $c \leftarrow (g^r, H(pk^r) \oplus m)$

$S(m_0, m_1 \in \{0,1\}^\ell)$		$R(b \in \{0,1\})$
$c \xleftarrow{\$} G$	$\xrightarrow{c}$	$k \xleftarrow{\$} \mathbb{Z}_q$] EG secret key
		$y_b \leftarrow g^k$] 2 EG public keys
* $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q$	$\xleftarrow{y_0, y_1}$	$y_{1-b} \leftarrow c/g^k$] sk is for y_b!
$c_0 \leftarrow (g^{r_0}, H(y_0^{r_0}) \oplus m_0)$		
$c_1 \leftarrow (g^{r_1}, H(y_1^{r_1}) \oplus m_1)$	$\xrightarrow{c_0, c_1}$	parse c_b as (u,v)
		$m_b \leftarrow H(u^k) \oplus v$

* check $y_0 y_1 = c$ for malicious security

Analysis

Correctness: $H(u^k) \oplus v = H(g^{rk}) \oplus H(pk^r) \oplus m = m$.

Recv Privacy: $\text{Sim}_S(m_0, m_1) = h \xleftarrow{\$} G$, output $(h, c/h)$ (generate rest of transcript per the protocol)

Sender Privacy:

First, show that the EG variant is semantically secure / CPA secure (equiv. for PKE)

$\to$ CDH implies guessing pk^r is hard to guess; RO implies $H(pk^r)$ is hard to guess

$\text{Sim}(b, m_b)$:

$k \xleftarrow{\$} \mathbb{Z}_q$, $c \xleftarrow{\$} G$, $y_b \leftarrow g^k$, $y_{1-b} \leftarrow c/g$, $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q$, $c_i \leftarrow (g^{r_i}, H(y^{r_i}, m_i))$ $m_{1-b} \leftarrow m_{b+1}$

output (c, k, c_0, c_1)

Need to show $\approx$ real view (case $b=0$)

$$\left.\begin{array}{l} (c, k, \text{Enc}(g^k, m_0), \text{Enc}(c/g^k, m_{0+1})) \\ \quad \text{vs.} \\ (c, k, \text{Enc}(g^k, m_0), \text{Enc}(c/g^k, m)) \end{array}\right\}$$

if some A can distinguish, can attack CPA security of Enc.

real view.

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: (uses OT + Sym. Cipher)
  1. Alice sends a "garbled" circuit to Bob
  2. Bob uses OT to get information to evaluate the circuit on the correct inputs <u>only</u>.
  3. Alice "ungarbles" the outputs
- Warm-Up: garbling a single AND-gate
  - let (E,D) be a symmetric cipher with key space $\mathcal{K}$ Note: we'll end up requiring some unusual (but not unrealistic) properties of (E,D). essentially, decrypting w/ the wrong key should give $\perp$: a noticable error.
  - $f(x \in \{0,1\}, y \in \{0,1\}) = \text{AND}(x,y) = z$.
  - Garbler (Alice):
    1. Samples two keys for each input wire $k_{x,0}, k_{x,1}, k_{y,0}, k_{y,1} \xleftarrow{\$} \mathcal{K}$
    2.

Note: ignore the red stuff till later

output wire too $k_{z,0}, k_{z,1} \xleftarrow{\$} \mathcal{K}$

2. Garbles each gate

x	y	z	garbling:
0	0	0	$E(k_{x,0}, E(k_{y,0}, 0)) = c_{00}$
0	1	0	$E(k_{x,0}, E(k_{y,1}, 0)) = c_{01}$
1	0	0	$E(k_{x,1}, E(k_{y,0}, 0)) = c_{10}$
1	1	1	$E(k_{x,1}, E(k_{y,1}, 1)) = c_{11}$

shuffle

x	y	z	garbling:
0	0	0	$E(k_{x,0}, E(k_{y,0}, k_{z,0})) = c_{00}$
0	1	0	$E(k_{x,0}, E(k_{y,1}, k_{z,0})) = c_{01}$
1	0	0	$E(k_{x,1}, E(k_{y,0}, k_{z,0})) = c_{10}$
1	1	1	$E(k_{x,1}, E(k_{y,1}, k_{z,1})) = c_{11}$

shuffle

Garbling C is the set $\{E(k_{x,a}, E(k_{y,b}, k_{z,AND(a,b)}))\}_{a,b\in\{0,1\}}$ in a randomized order

3. Sends the garbling and $k_{x,\hat{x}}$ where $\hat{x}$ is the value of x, to Evaluator

- Evaluator & Garbler run OT w/ Evaluator as Reciever($\hat{y}$) and Garbler as Sender($k_{y,0}, k_{y,1}$)
- Evaluator runs $m_i \leftarrow D(k_{x,\hat{y}}, D(k_{x,\hat{x}}, c))$ for $c_i \in C$
  - 3/4 of the m_i's will be $\perp$ – a decryption error, since we used the wrong keys.
  - the non-$\perp$ m_i will be $\hat{z} = AND(\hat{x}, \hat{y})$!
- Evaluator sends $\hat{z}$ to garbler.

$K_{z,AND(\hat{x},\hat{y})}$

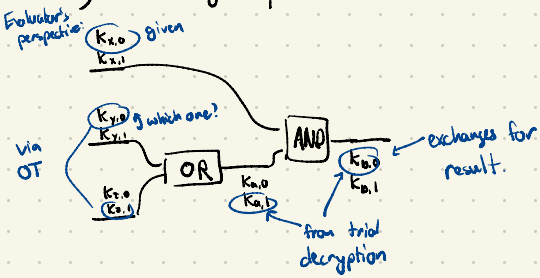
- Evaluator sends $k_{z,AND(\hat{x},\hat{y})}$ to Garbler, who replies with $\hat{z} = AND(\hat{x},\hat{y})$

- Yao's actual protocol (full circuit)
  - The previous protocol has a flaw – inputs are "garbled" as 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 $k_{w,0}, k_{w,1} \xleftarrow{\$} \mathcal{K}$ for each wire w.
   b. Produces sets $C_g = \{E(k_{w_x,a}, E(k_{w_y,b}, k_{w_z,g(a,b)}))\}_{a,b\in\{0,1\}}$ for each gate $w_x, w_y \to g \to w_z$
   c. Sends C_g for all gates g and $k_{w_x,\hat{x}}$ for wire w_x corresponding to input x of value $\hat{x}$.
2. Evaluator uses OTs to obtain $k_{w_y,\hat{y}}$ for wire w_y corresponding to input y of value $\hat{y}$.
3. Evaluator 'evaluates' gate-by-gate
4. Evaluator sends output keys to Garbler, who replies w/ output bits!

- Visualizing the multi-gate protocol:



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

- Optimizations galore: active area of research
  - "point and permute": avoid trial decryptions
  - "half-gates": two CT's per gate instead of 4.
  - "OT extension": a mechanism for doing $O(\lambda)$ OTs with $O(\lambda)$ group operations (instead of $O(n)$)
    - important if there are many inputs
  - "free XOR"
    - At protocol start, sample $R \xleftarrow{\$} \mathcal{K}$ (once).
    - For all wires w sample $k_{w,0} \xleftarrow{\$} \mathcal{K}$, set $k_{w,1} \leftarrow k_{w,0} \oplus R$
    - Observe: For $z = x \oplus y$, fix $k_{z,0} = k_{x,0} \oplus k_{y,0}$

      then $k_{x,0} \oplus k_{y,1} = k_{x,0} \oplus k_{y,0} \oplus R = k_{z,1}$

      $k_{x,1} \oplus k_{y,0} = k_{x,0} \oplus R \oplus k_{y,0} = k_{z,1}$

      $k_{x,1} \oplus k_{y,1} = k_{x,0} \oplus R \oplus k_{y,0} \oplus R = k_{z,0}$

      thus, $\oplus$ evaluates the $\oplus$ gate! No garbling needed!

- "Free If": Gives MUXes in circuit w/ Garble and Eval 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 Δ their input mid-protocol
- refuse to send some messages
- can be devastating: GC is totally unsecure 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 effecient approaches exist (perhaps next lecture).