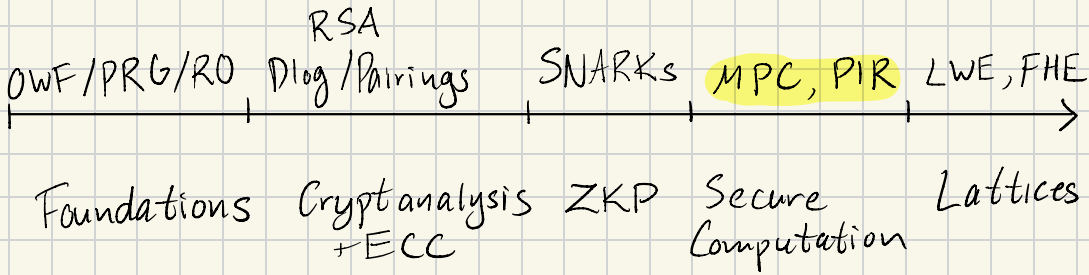
Two Party Computation (2PC) from Garbled Circuits and Oblivious Transfer



Outline

- Secure MPC (e.g. 2PC)
- Oblivious Transfer (specific 2PC)
- Yao's Garbled Circuits (2PC for circuits)
- Malicious security

Multi-Party Computation (MPC)

aka Secure Computation

Today: 2PC = 2-Party Computation

- Alice knows $x \in \{0,1\}^\ell$
- Bob knows $y \in \{0,1\}^\ell$
- Both know functionality $f: \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^m$

Goal: both learn $f(x,y)$ and nothing else

↳ generally can also support Alice + Bob only learning a part of the input

e.g. $f(x,y) = (f_A(x,y), f_B(x,y))$

↑ Alice learns this — ↑ Bob learns this

Applications of 2PC

1) Yao's millionaire problem
   - 2 millionaires want to find out who is richer w/o revealing how much money they have

2) Private Advertising Campaign Evaluation
   - Google knows $x = \{\text{users who saw ad}\}$
   - Macy's knows $y = \{\text{users who bought item}\}$

Functionality $f(x,y)$ = # of users who saw ad and bought item

3) Private Contact Discovery
- Alice knows x = her contact list
- Signal knows y = all Signal users' phone #s

Functionality $f(x,y) = f_{signal}(x,y) = \perp$
$f_A(x,y)$ = intersection of Alice's contacts w/ all Signal users

4) Zero Knowledge
- Prover knows (x,w)
- Verifier knows x

Functionality $f(x,y) = \begin{cases} 1 & \text{if } (x,w) \in R \\ 0 & \text{else} \end{cases}$

2PC Definitions and Security

*A, B are randomized interactive algs

Let (A,B) be an interactive protocol for functionality $f: \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^m$

Correctness: for all $x, y \in \{0,1\}^\ell$,
$\Pr[\langle A(x), B(y) \rangle = f(x,y)] = 1$

Security: two popular security types

~HVZK - semi-honest: adversarial parties follow the protocol but inspect their data to learn more

~ZK - malicious: adversarial parties may deviate from the protocol

↳ focus on semi-honest today

Semi-Honest Privacy

$\exists$ 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)

- Sender knows $m_0, m_1 \in \{0,1\}^\ell$ → learns nothing
- Receiver knows b → learns m_b

↳ Consider trivial protocol where sender sends m_0 and m_1

Q: Is this secure OT?

Correct? Yes!

Secure? No!

- Sender learns nothing
- Receiver learns more than just m_b

Instantiating Security Definition for OT

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

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

$$Pr[output_R(\langle S(m_0,m_1), R(b)\rangle = m_b)] = 1$$

($output_R$: receiver's output)

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

$$Sim_R(b, m_b) \approx View_R[\langle S(m_0,m_1), R(b)\rangle]$$

Receiver Privacy: $\exists$ efficient Sim_S s.t.
$\forall m_0, m_1 \in \{0,1\}^\ell$, $\forall b \in \{0,1\}$
$Sim_S(m_0, m_1) \approx View_S[\langle S(m_0, m_1), R(b)\rangle]$

Bellare + Micali OT

- based on ElGamal encryption variant
- let $\langle g \rangle = \mathbb{G}$, $|\mathbb{G}| = q$, $H: \mathbb{G} \to \{0,1\}^\ell$ (H: random oracle)
  - $sk \xleftarrow{\$} \mathbb{Z}_q$, $pk \leftarrow g^{sk}$
  - $Enc(m \in \{0,1\}^\ell, pk)$:
    - $r \xleftarrow{\$} \mathbb{Z}_q$
    - $ct \leftarrow (g^r, H(pk^r) \oplus m)$
  - $Dec(c = (u, v), sk)$
    $m \leftarrow v \oplus H(u^{sk})$
- * semantically secure assuming CDH

$S(m_0, m_1 \in \{0,1\}^\ell)$ | $R(b \in \{0,1\})$

$c \xleftarrow{\$} \mathbb{G}$

$\xrightarrow{c}$

$k \xleftarrow{\$} \mathbb{Z}_q$] EG secret key
$y_b \leftarrow g^k$] 2 EG pk's
$y_{1-b} \leftarrow c/g^k$] R knows sk for y_b

$\xleftarrow{y_0, y_1}$

$r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q$

$c_0 \leftarrow (g^{r_0}, H(y_0^{r_0}) \oplus m_0)$ ← Encrypt m_0 to y_0

$c_1 \leftarrow (g^{r_1}, H(y_1^{r_1}) \oplus m_1)$ ← Encrypt m_1 to y_1

$\xrightarrow{c_0, c_1}$

parse c_b as (u, v)
$m_b \leftarrow H(u^k) \oplus v$

Analysis

- Correctness: $H(u^k) \oplus v$
  $= H(g^{r_b \cdot k}) \oplus H(y_b^{r_b}) \oplus m_b$
  $= H(g^{r_b \cdot k}) \oplus H(g^{k \cdot r_b}) \oplus m_b$
  $= m_b$

- Receiver Privacy:
  $\underline{Sim_S(m_0, m_1)}$
  - sample $C \xleftarrow{\$} \mathbb{G}$
  - sample $y_0 \xleftarrow{\$} \mathbb{G}$, set $y_1 = C / y_0$
    ↳ easy to see $\{y_0, y_1\}$ identically distributed as in $View_S$ for honest protocol

- Sender Privacy
  $\underline{Sim_R(b, m_b)}$
  - sample $C \xleftarrow{\$} \mathbb{G}$
  - sample $k \xleftarrow{\$} \mathbb{Z}_q$, $y_b \leftarrow g^k$, $y_{1-b} \leftarrow C / g^k$
  - $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q$
  - $c_b = (g^{r_b}, m_b \oplus H(y_b^{r_b}))$ — Enc of m_b as in protocol
  - $c_{1-b} = (g^{r_{1-b}}, (m_b + 1) \oplus H(y_{1-b}^{r_{1-b}}))$ — Enc of $m_b + 1$ (Sim doesn't know m_{1-b})
  - output (C, k, c_0, c_1)

Ex: $b = 0$

Simulated View: $(C, Enc(m_0, g^k), Enc(m_0 + 1, C/g^k))$

Real View: $(C, Enc(m_0, g^k), Enc(m_1, C/g^k))$

$Enc(m_0 + 1, C/g^k)$ and $Enc(m_1, C/g^k)$ computationally indistinguishable by CPA
→ if adversary can distinguish, it breaks semantic security

Yao's Garbled Circuits (2PC)

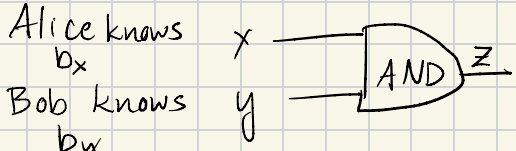
- 2PC for any boolean circuit f
  - directed acyclic graph
  - 2-input AND, XOR gates w/ unlimited fan-out
  - input wire for each bit of input
  - output wire for each bit of output

Garbling Circuits (only need symmetric cipher + OT)

↳ High Level Idea

1. Alice "garbles" the circuit f and sends it to Bob
2. Bob uses OT to get information to evaluate the circuit on Bob's input only

Warm-Up: garbling a single AND gate



Goal: Alice and Bob want to learn $f(b_x \in \{0,1\}, b_y \in \{0,1\}) = AND(b_x, b_y) = z$

- let (E, D) be a symmetric cipher w/ key space K
  ↳ we require an unusual but not unrealistic property of (E, D): decrypting with wrong key outputs ⊥

1) Garbler (Alice)

a. Samples two keys for each wire

$$k_x^0, k_x^1, k_y^0, k_y^1 \xleftarrow{\$} K$$

b. 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

c. Shuffles $\{c_{ij}\}_{i,j \in \{0,1\}}$

Sends garbling and $k_x^{b_x}$

2) Garbler (Alice) and Evaluator (Bob) run OT

↓ Sender (k_y^0, k_y^1)

↓ Receiver (b_y)

↳ Bob gets $k_y^{b_y}$

3) Evaluator (Bob) runs $m_i \leftarrow Dec(k_x^{b_x}, Dec(k_y^{b_y}, c))$ for $c \in C$

- only the decryption on $c_{b_x \wedge b_y}$ will succeed
  → evaluator gets $\hat{z} = b_x \wedge b_y$

4) Evaluator (Bob) sends $\hat{z}$ to Garbler (Alice)

Analysis

Correctness: straightforward

Privacy: (informal)

↳ Garbler's view: OT protocol msgs + output

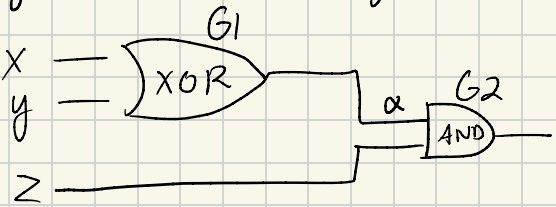
- informally, this is private if OT is private

↳ Evaluator's view:
- 4 ciphertexts
- OT protocol messages
- Random keys $k_x^{b_x}, k_y^{b_y}$
- output
  ↳ can be simulated using security of encryption scheme & OT

Why shuffle?
↳ otherwise evaluator can infer garbler's input based on which c_{ij} it can decrypt

Garbling Full Circuit (Yao '86)

Ex



e.g. Garbler has b_x, b_y
Evaluator has b_z

Idea: chain garbled gates

Instead of encrypting gate output, encrypt inputs to next gate

for G1:
$k_x^0, k_x^1, k_y^0, k_y^1, k^0, k^1 \xleftarrow{\$} K$

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

⇒

x	y	x XOR y
k_x^0	k_y^0	$Enc(k_x^0, Enc(k_y^0, k^0))$
k_x^0	k_y^1	$Enc(k_x^0, Enc(k_y^1, k^1))$
k_x^1	k_y^0	$Enc(k_x^1, Enc(k_y^0, k^1))$
k_x^1	k_y^1	$Enc(k_x^1, Enc(k_y^1, k^0))$

Encrypt key corresponding to output for each row

Same idea for G2

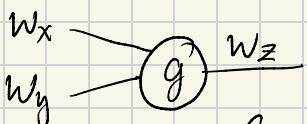
for G2: sample $k_2^0, k_2^1 \xleftarrow{\$} K$

α	z	α AND z
k^0	k_2^0	$Enc(k^0, Enc(k_2^0, 0))$
k^0	k_2^1	$Enc(k^0, Enc(k_2^1, 0))$
⋮		⋮

for last gate, encrypt outputs using row's key pair

Full Protocol

1) Garbler
   a) For each wire w, sample $k_w^0, k_w^1 \xleftarrow{\$} K$
   b) Produce 4 ciphertexts for each gate:



$$C_g = \left\{ Enc\left(k_{w_x}^{b_x}, Enc\left(k_{w_y}^{b_y}, k_{w_z}^{g(b_x, b_y)}\right)\right)\right\}_{b_x, b_y \in \{0,1\}}$$

↳ for last gate, encrypt $g(b_x, b_y)$ instead of keys

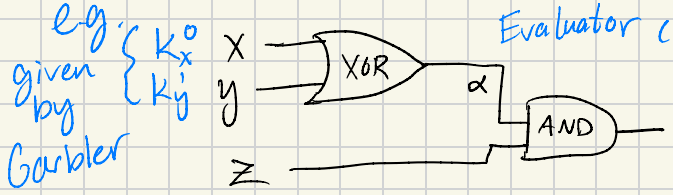
   c) For all gates, shuffle C_g and send to Evaluator. Also send $k_w^{b_w}$ for all input wires that Garbler has
      e.g., Garbler sends k_x^0, k_y^1 to Bob

2) Garbler/Evaluator run OT for each input wire that Evaluator has

Garbler (Sender)	Evaluator (Receiver)
eg. (k_z^0, k_z^1)	$(b_z = 1)$

after OT, Bob gets $k_z^{b_z} = k_z^1$

3) Evaluator evaluates the circuit gate by gate



Evaluator can only decrypt k_α^1 (0 XOR 1 = 1)

Evaluator can only decrypt 1 AND 1 ⇒ gets 1

4) Evaluator sends output to Garbler

Analysis

Correctness: ✓

Privacy: very complicated; first formal proof in 2004

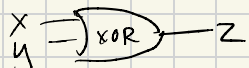
Note: this is not maliciously secure (see HW)

Optimizing Garbled Circuits
- a lot of research
- "half gates": send only 2 CTs instead of 4 per gate
- "OT extension": can do $O(n)$ OTs with $O(\lambda)$ group ops

- "free XOR": allows Alice to send NOTHING for XOR gates

Idea: pick random $R \xleftarrow{\$} K$

for all wires w, sample $k_w^0 \xleftarrow{\$} K$, but set $k_w^1 \leftarrow k_w^0 \oplus R$ (instead of sampling uniformly from K)

Observe for $z = x \oplus y$, 

let $k_x^0 \oplus k_y^0 = k_z^0$

then $k_x^0 \oplus k_y^1 = k_x^0 \oplus k_y^0 \oplus R$
$= k_z^0 \oplus R$
$= k_z^1$

similarly, $k_x^1 \oplus k_y^0 = k_x^0 \oplus R \oplus k_y^0$
$= k_z^1$

and $k_x^1 \oplus k_y^1 = k_x^0 \oplus R \oplus k_y^0 \oplus R$
$= k_x^0 \oplus k_y^0$
$= k_z^0$

$\Rightarrow$ XOR of input keys gives you output key! No garbling needed!

Malicious Security

- Garbler might not follow the protocol:
  - sample biased randomness
  - send wrong messages
  - might refuse to send messages

How to get malicious security?

- [GMW'87]
  1. Commit to randomness
  2. For each m from Garbler, Garbler proves in ZK that m is correct message

somewhat inefficient...