MIT 6.875 & Berkeley CS276

Secure two-party computation and Yao Garbled Circuits

Lecture 24

Credit: Some slides adapted from Gil Segev

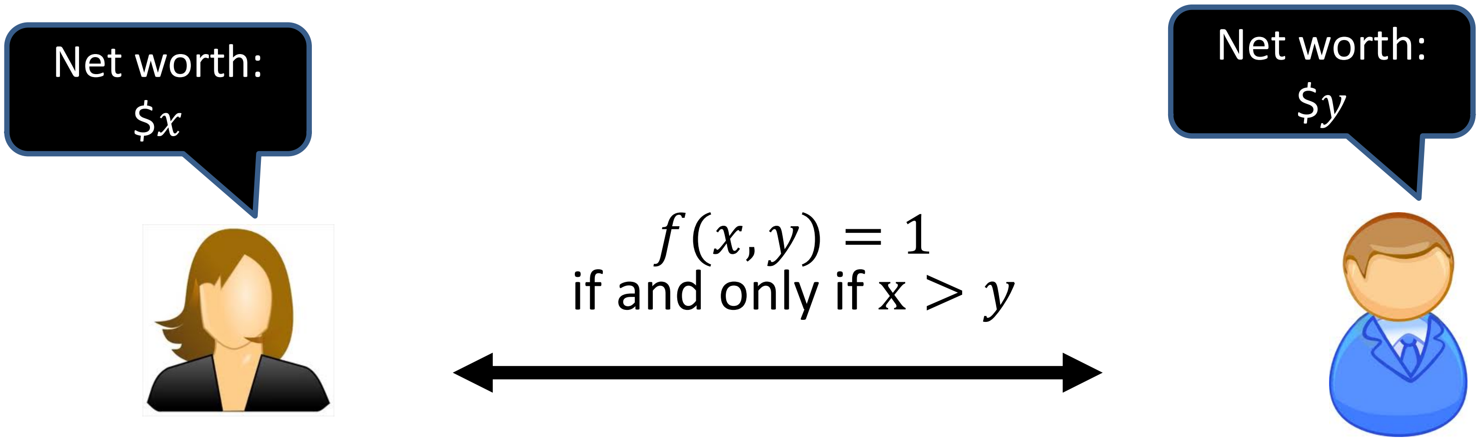
In this lecture…

Recording…

Secure two-party computation:

- Paradigm
- Security definition for semi-honest adversaries
- Construction via Yao garbled circuits

The Millionaires' Problem

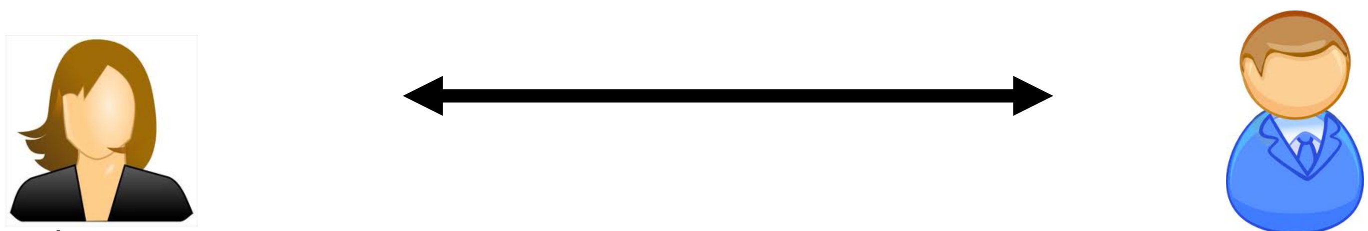


Alice and Bob want to know who is richer without revealing their inputs to each other. How can they compute $f(x, y)$?

The paradigm of secure computation

Alice and Bob hold inputs x and y and wish to compute $f(x, y)$

Goal: no one learns anything about x or y other than $f(x, y)$



Alice: x

Bob: y

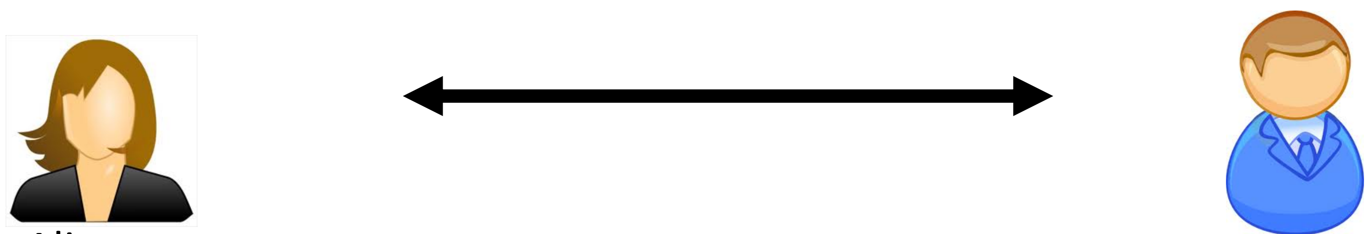
Adversarial models:

- **Semi-honest/honest-but-curious:** Each party follows the protocol, but tries to learn additional information from the transcript
- **Malicious:** Parties can behave arbitrarily, even deviate from the protocol in order to learn additional information

The paradigm of secure computation

Alice and Bob hold inputs x and y and wish to compute $f(x, y)$

Goal: no one learns anything about x or y other than $f(x, y)$



Alice: x

Bob: y

How would you define this?

simulation paradigm

Notation

Alice: x
random tape: r_A

Protocol (A, B)

Bob: y
random tape: r_B

- $\langle A(x), B(y)\rangle(1^n)$ is the distribution of the transcript on inputs x and y
- $out_A[\langle A(x), B(y)\rangle(1^n)]$ is the distribution of A's output
- $out_B[\langle A(x), B(y)\rangle(1^n)]$ is the distribution of B's output
- $view_A[\langle A(x), B(y)\rangle(1^n)]$ is the distribution of A's view: random tape r_A and transcript
- $view_B[\langle A(x), B(y)\rangle(1^n)]$ is the distribution of B's view: random tape r_B and transcript

Security in the semi-honest model

Protocol (A, B)

Alice: x
random tape: r_A

Bob: y
random tape: r_B

Definition: An efficient protocol $\langle A, B\rangle$ securely computes a deterministic function $f = (f_1, f_2)$ in the semi-honest model if there exist PPT simulators S_A and S_B such that for every $\{x, y\} \in \{0,1\}^*$, the following hold:

Correctness:

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

Security against semi-honest Alice:

$$S_A(x, f_1(x,y)) \approx_c view_A(\langle A(x), B(y)\rangle)$$

Security against semi-honest Bob:

Symmetric

> Note that f is known to the simulators since f is set before the exists quantifier on the simulators in the definition statement. This means that the definition does not guarantee privacy of f. Note that privacy of f can be achieved by setting f to be a universal circuit of a max size, and providing the function specific information as another input to this universal circuit.

Security in the Semi-Honest Model

Theorem (Yao '86):
Assuming the existence of a secure Oblivious Transfer protocol in the semi-honest model, any efficiently-computable deterministic two-output function can be securely computed in the semi-honest model.

$$f(x, y) = \left(f_1(x, y), f_2(x, y)\right)$$

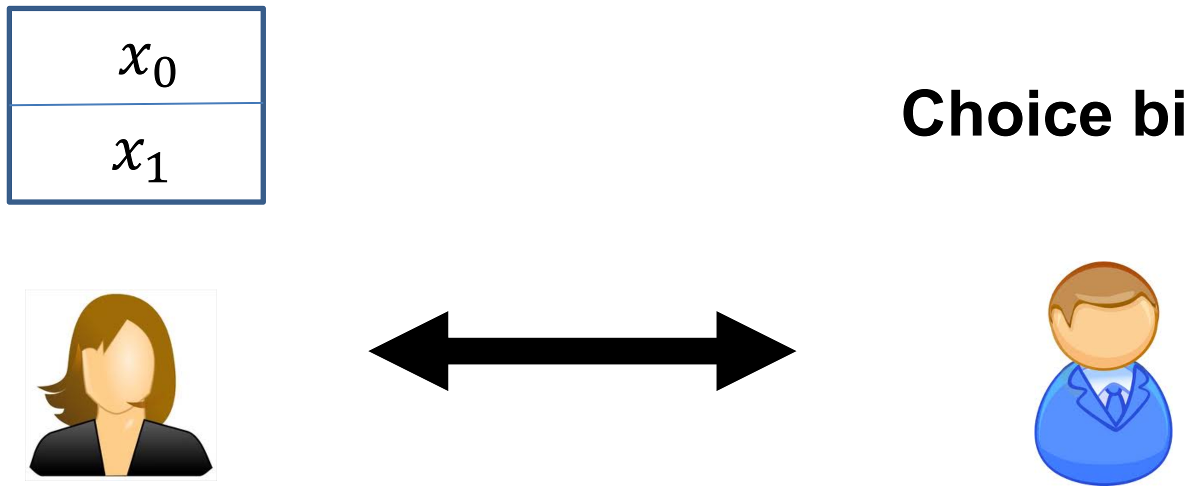
- Groundbreaking result initiating research on secure computation
- Inspired fundamental protocols for the multi-party & malicious models
- Various applications beyond secure computation

Tools to recall

- Oblivious Transfer (OT)
- CPA-secure privacy-key encryption scheme

Recall: Oblivious Transfer (OT)

Choice bit: *b*



Sender

Receiver

- Sender holds two bits x_0 and x_1.
- Receiver holds a choice bit b.
- Receiver should learn x_b, sender should learn nothing.

"Special" CPA Encryption

- We will use a CPA-secure private-key encryption scheme $(\mathrm{G}, \mathrm{E}, \mathrm{D})$ with two additional properties
- Notation: $\mathrm{Range}_n(k) \stackrel{\text{def}}{=} \{\mathrm{E}_k(x) : x \in \{0,1\}^n\}$

Property 1: Elusive range

For every PPT algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\Pr_{k \leftarrow \mathrm{G}(1^n)}[A(1^n) \in \mathrm{Range}_n(k)] \leq \nu(n)$$

Property 2: Efficiently verifiable range

There exists a PPT algorithm M such that $M(1^n, k, c) = 1$ if and only if $c \in \mathrm{Range}_n(k)$

Ideas how to construct?

Construction

Property 1: Elusive range

For every PPT algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\Pr_{k \leftarrow G(1^n)}[A(1^n) \in \mathrm{Range}_n(k)] \leq \nu(n)$$

Property 2: Efficiently verifiable range

There exists a PPT algorithm M such that $M(1^n, k, c) = 1$ if and only if $c \in \mathrm{Range}_n(k)$

Construction:

- Let F be a PRF where $F_k: \{0,1\}^n \rightarrow \{0,1\}^{2n}$ for $k \in \{0,1\}^n$

$$\mathrm{E}_k(x; r) = (r, F_k(r) \oplus x0^n)$$

Why does it satisfy the two properties?

Boolean circuits

Gates are Boolean gates (AND, XOR, OR) taking as input two bits and outputting one bit

- How would you express the millionaire's $f(x, y) = x > y$ as a Boolean circuit C?

The Millionaires' Function as a Circuit

$$f(x, y) = 1$$
if and only if $x > y$

x y

…	0	1	0	0	…

…	0	1	1	1	1	1	1

Unit Vector u_x = 1 in the x^{th} location and 0 elsewhere

Vector v_y = 1 from the $(y+1)^{th}$ location onwards

$$\boldsymbol{f(x,y) = \langle u_x, v_y \rangle = \sum_{i=1}^{U} u_x[i] \wedge v_y[i]}$$

An AND for each u_i, v_i, then OR between all results in a tree-like fashion

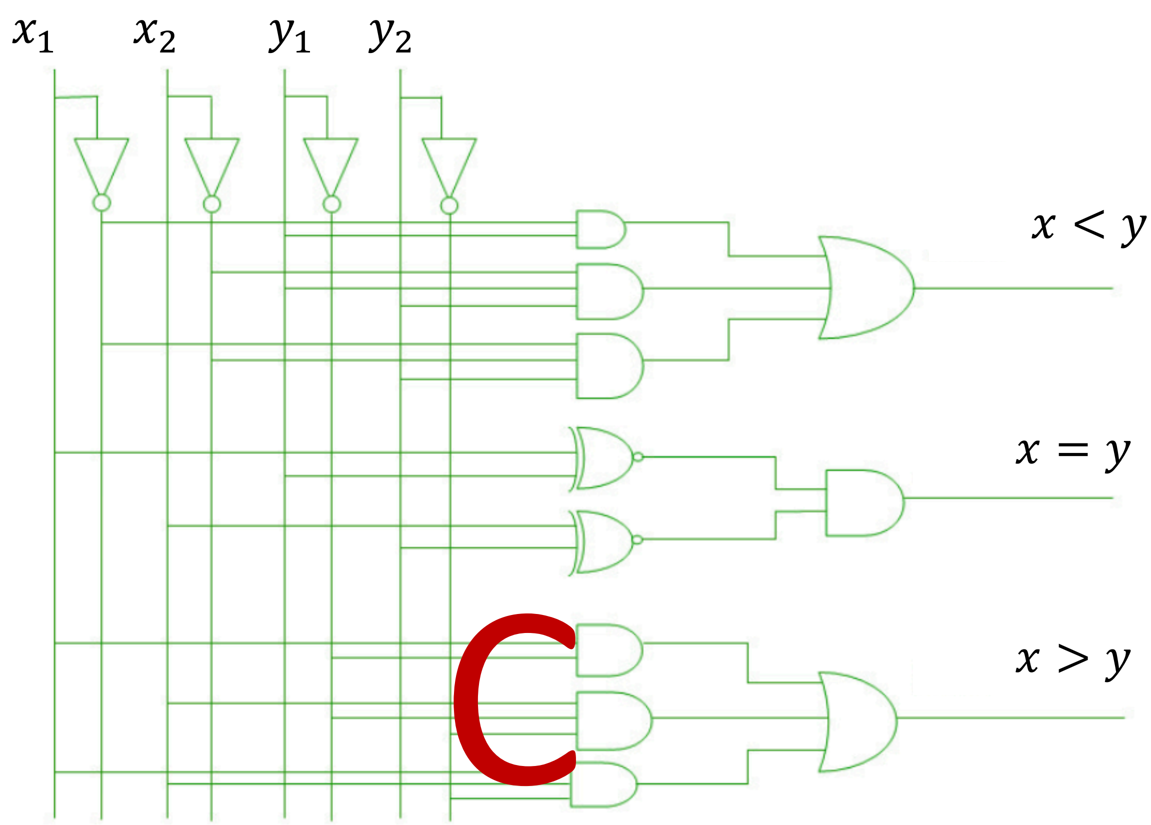
Or use comparison circuit

$f(x, y) = 1$
if and only if $x > y$

x

y

Magnitude comparator for 2-bit numbers



Garbling Boolean Circuits

- **Input:** Boolean circuit $C: \{0,1\}^n \rightarrow \{0,1\}$
- **Output:** Garbled circuit $G(C)$ and input labels $\{(L_1^0, L_1^1), \dots, (L_n^0, L_n^1)\}$

Goal: Given $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}$

- It is possible to compute $C(x_1 \cdots x_n)$
- It is not possible to learn any additional information other than size of circuit or input

For example, for $x = 010$, labels are L_1^0, L_2^1, L_3^0

Using garbled circuits for secure 2-party computation

Input will be x, y

Common input: $C: \{0,1\}^{2n} \to \{0,1\}$

Input: $x \in \{0,1\}^n$

Compute $G(C)$ and labels $\{(L_i^0, L_i^1)\}_{i \in [2n]}$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x, y)$

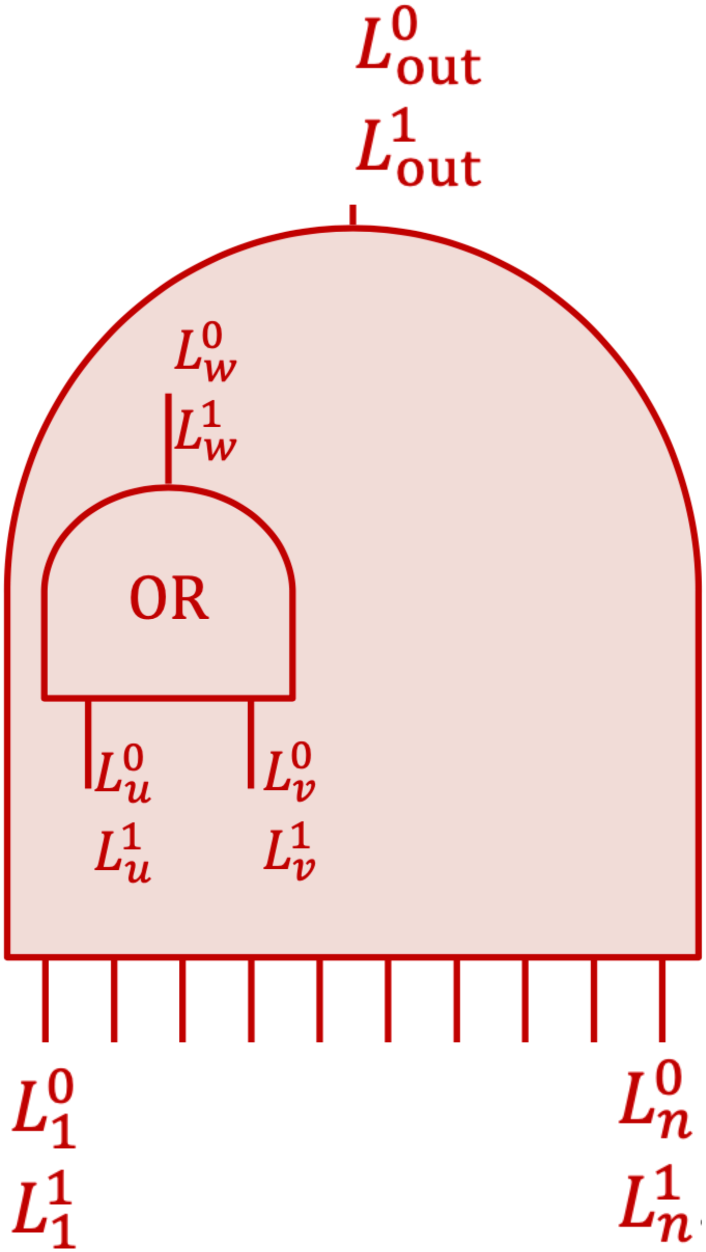
Input: $y \in \{0,1\}^n$

Compute $C(x, y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w

Garbled circuit



The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows

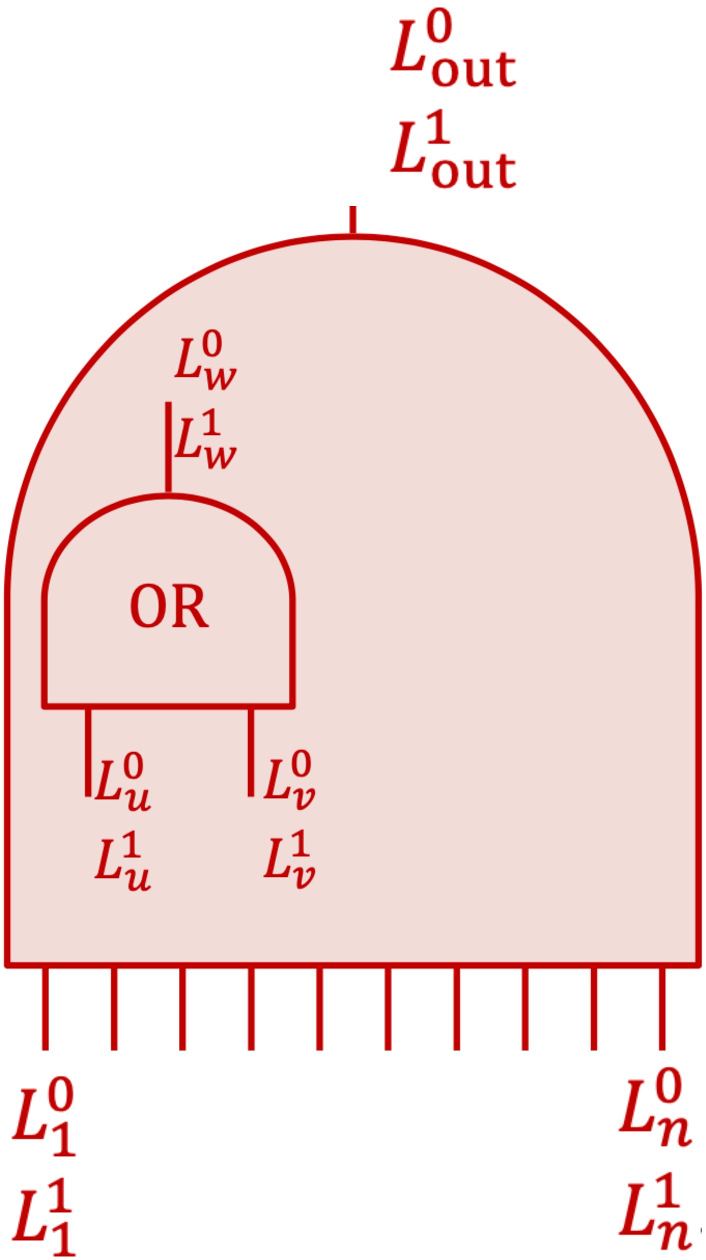
0	0	$g(0,0)$
0	1	$g(0,1)$
1	0	$g(1,0)$
1	1	$g(1,1)$

The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows

L_u^0	L_v^0	$L_w^{g(0,0)}$
L_u^0	L_v^1	$L_w^{g(0,1)}$
L_u^1	L_v^0	$L_w^{g(1,0)}$
L_u^1	L_v^1	$L_w^{g(1,1)}$

The garbling procedure



The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows

L_u^0	L_v^0	$L_w^{g(0,0)}$
L_u^0	L_v^1	$L_w^{g(0,1)}$
L_u^1	L_v^0	$L_w^{g(1,0)}$
L_u^1	L_v^1	$L_w^{g(1,1)}$

Why can't I leave the output labels this way?
Because they leak (e.g. type of gate)

The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows

L_u^0	L_v^0	$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$
L_u^0	L_v^1	$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(0,1)}\right)\right)$
L_u^1	L_v^0	$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(1,0)}\right)\right)$
L_u^1	L_v^1	$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(1,1)}\right)\right)$

The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows

$(\mathrm{G}, \mathrm{E}, \mathrm{D})$ has elusive & efficiently verifiable range

Given L_u^α and L_v^β can identify the row corresponding to inputs (α, β) and compute $L_w^{g(\alpha,\beta)}$

$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(1,1)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(0,1)}\right)\right)$
$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(1,0)}\right)\right)$

The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows
- Construct an output translation table

0	L_{out}^0
1	L_{out}^1

Can handle any number of output wires by constructing a table for each one

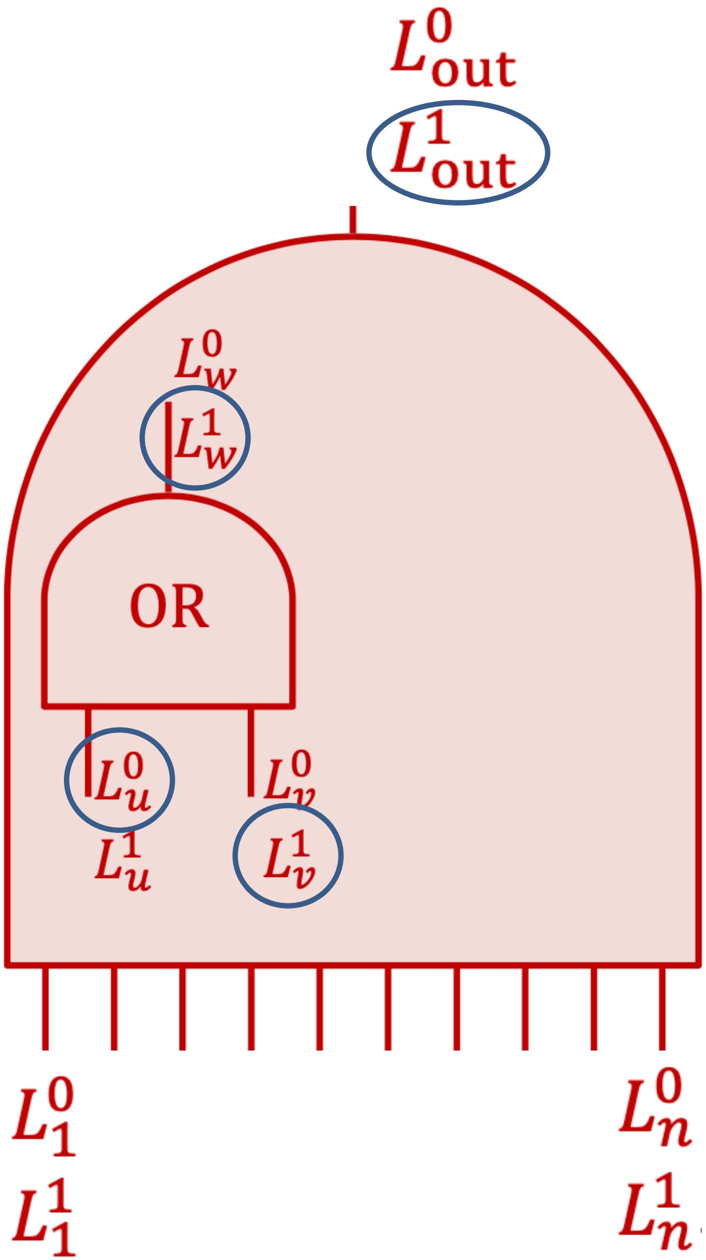
The garbling procedure

- Assign two random labels (L_w^0, L_w^1) to each wire w
  - $L_w^0 \leftarrow \mathrm{G}(1^n)$ corresponds to value 0 on wire w
  - $L_w^1 \leftarrow G(1^n)$ corresponds to value 1 on wire w
- For each gate g construct a doubly-encrypted translation table with randomly permuted rows
- Construct an output translation table
- Output all tables

0	L_{out}^0
1	L_{out}^1

$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(1,1)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(0,1)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$

The garbling procedure



Yao's protocol

Common input: $C: \{0,1\}^{2n} \rightarrow \{0,1\}$

Input: $x \in \{0,1\}^n$

Compute $G(C)$ and labels $\{(L_i^0, L_i^1)\}_{i \in [2n]}$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x, y)$

Input: $y \in \{0,1\}^n$

Compute $C(x, y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

Recall:Security in the semi-honest model

Protocol (A, B)

Alice: x
random tape: r_a

Bob: y
random tape: r_b

Definition: An efficient protocol $\langle A, B\rangle$ securely computes a deterministic function $f = (f_1, f_2)$ in the semi-honest model if there exist PPT simulators S_A and S_B such that for every $\{x, y\} \in \{0,1\}^*$, the following hold:

Correctness:

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

Security against semi-honest Alice:

$$S_A(x, f_1(x, y)) \approx_c view_A(\langle A(x), B(y)\rangle)$$

Security against semi-honest Bob: symmetric

Alice's simulator

Input: $x \in \{0,1\}^n$

Compute $G(C)$ and labels $\{(L_i^0, L_i^1)\}_{i \in [2n]}$

Garbled circuit $G(C)$ →

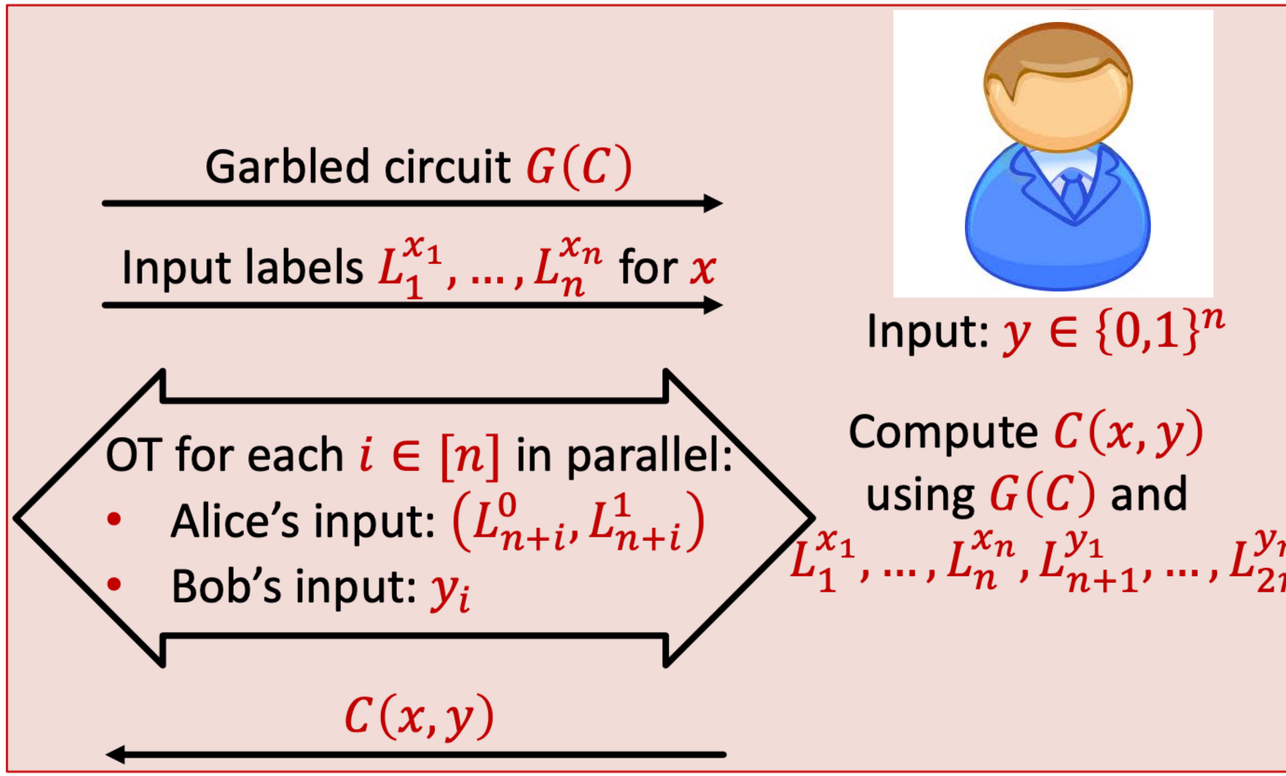
Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x,y)$

Bob’s simulator



Bob's simulator: step 1

Replace Bob's view in the OTs with the assumed OT simulator $\mathcal{S}_B^{\mathrm{OT}}$

Indistinguishable from Bob's original view by the security of the OT (standard hybrid argument over the n OTs)

From this point on, $\mathcal{S}_B$ needs to know $L_{n+i}^{y_i}$ but does not use $L_{n+i}^{1-y_i}$

Input: $y \in \{0,1\}^n$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: $\left(L_{n+i}^0, L_{n+i}^1\right)$
- Bob's input: y_i

Compute $C(x, y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

← $C(x, y)$

Bob's simulator: step 2

Replace $G(C)$ with an indistinguishable $\tilde{G}(C)$ that evaluates to $C(x,y)$ on all input labels

Intuition: Bob should not notice that $\tilde{G}(C)$ computes a constant function since he knows only one of (L_{n+i}^0, L_{n+i}^1) by the security of the OT

Input: $y \in \{0,1\}^n$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

Compute $C(x,y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

← $C(x,y)$

Bob's simulator: step 1

Replace Bob's view in the OTs with the assumed OT simulator $\mathcal{S}_B^{\mathrm{OT}}$

Can now replace $L_1^{x_1}, \dots, L_n^{x_n}$ with $L_1^0, \dots, L_n^0$

This view can be generated given y and $C(x, y)$, and without knowing x

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:
- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x, y)$

Input: $y \in \{0,1\}^n$

Compute $C(x, y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

The fake $\tilde{G}(C)$

- Assign two random labels (L_w^0, L_w^1) to each wire w
- For each gate g construct a randomly permuted translation table doubly-encrypting the zero label

Real table	Fake table
$E_{L_u^1}\left(E_{L_v^1}\left(L_w^{g(1,1)}\right)\right)$	$E_{L_u^1}\left(E_{L_v^1}\left(L_w^0\right)\right)$
$E_{L_u^0}\left(E_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$	$E_{L_u^0}\left(E_{L_v^0}\left(L_w^0\right)\right)$
$E_{L_u^0}\left(E_{L_v^1}\left(L_w^{g(0,1)}\right)\right)$	$E_{L_u^0}\left(E_{L_v^1}\left(L_w^0\right)\right)$
$E_{L_u^1}\left(E_{L_v^0}\left(L_w^{g(1,0)}\right)\right)$	$E_{L_u^1}\left(E_{L_v^0}\left(L_w^0\right)\right)$

Leverage CPA security of (G, E, D)

Real and fake tables are indistinguishable because only one label is known from each pair (L_u^0, L_u^1) and (L_v^0, L_v^1)

(Subtle hybrid argument due to dependencies between tables corresponding to different gates)

Real table

$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(1,1)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(0,0)}\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^{g(0,1)}\right)\right)$
$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^0}\left(L_w^{g(1,0)}\right)\right)$

Fake table

$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^0}\left(L_w^0\right)\right)$

The fake $\tilde{G}(C)$

- Assign two random labels (L_w^0, L_w^1) to each wire w
- For each gate g construct a randomly permuted translation table doubly-encrypting the zero label
- Construct an output translation table where L_{out}^0 is translated to $C(x, y)$

0	$L_{\mathrm{out}}^{C(x,y)}$
1	$L_{\mathrm{out}}^{1-C(x,y)}$

The fake $\tilde{G}(C)$

- Assign two random labels (L_w^0, L_w^1) to each wire w
- For each gate g construct a randomly permuted translation table doubly-encrypting the zero label
- Construct an output translation table where L_{out}^0 is translated to $C(x,y)$
- Output all tables

0	$L_{\text{out}}^{C(x,y)}$
1	$L_{\text{out}}^{1-C(x,y)}$

$\mathrm{E}_{L_u^1}\left(\mathrm{E}_{L_v^1}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^1}\left(L_w^0\right)\right)$
$\mathrm{E}_{L_u^0}\left(\mathrm{E}_{L_v^0}\left(L_w^0\right)\right)$

Bob's simulator

Garbled circuit $\tilde{G}(C)$ →

Input labels $L_1^0, \dots, L_n^0$ for x →

Input: $y \in \{0,1\}^n$

For each $i \in [n]$
invoke $\mathcal{S}_B^{\mathrm{OT}}(y_i, L_{n+i}^{y_i})$ ⟺

Compute $C(x,y)$
using $\tilde{G}(C)$ and
$L_1^0, \dots, L_n^0, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

← $C(x,y)$

Yao's protocol

Common input: $C: \{0,1\}^{2n} \to \{0,1\}$

Input: $x \in \{0,1\}^n$

Compute $G(C)$ and labels $\{(L_i^0, L_i^1)\}_{i\in[2n]}$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \dots, L_n^{x_n}$ for x →

⇔ OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x,y)$

Input: $y \in \{0,1\}^n$

Compute $C(x,y)$ using $G(C)$ and $L_1^{x_1}, \dots, L_n^{x_n}, L_{n+1}^{y_1}, \dots, L_{2n}^{y_n}$

Theorem:

Yao's protocol securely computes any $C: \{0,1\}^{2n} \to \{0,1\}$ in the semi-honest model

Efficiency

- Garbling and evaluation tend to be very efficient because it can be implemented via AES, which is in hardware
- Creating a circuit from a program often results in a big circuit

Questions

Q: Say Alice and Bob want to compare their European and US funds. Can they reuse the garbled circuit?

A: No! Yao garbled circuits are **one-time.** Insecure with multiple input encodings.

Your three instructors had a paper (STOC'11) on how to reuse garbled circuits. Great proof of concept but a very inefficient scheme with nesting of heavy schemes like FHE or ABE.

Q: What are two inputs that reveal all values of $f(x, y)$?

A: 00000… and 11111.. because Bob receives all possible labels.

Summary

We learned about secure two-party computation

- definition for semi-honest adversaries, and
- a construction via Yao garbled circuits and OT