Secure
2-Party Computation

Lecture 14
Yao’s Garbled Circuit
SIM-Secure MPC

Secure (and correct) if:

∀ ∃ s.t. output of is distributed identically in REAL and IDEAL
Passive Adversary

- Gets **only read access** to the internal state of the corrupted players (and can use that information in talking to environment)
  - Also called "Honest-But-Curious" adversary
  - Will require that simulator also corrupts passively

- Simplifies several cases
  - e.g. coin-tossing [why?], commitment [coming up]

- Oddly, sometimes security against a passive adversary is more demanding than against an active adversary
  - Active adversary: too pessimistic about what guarantee is available even in the IDEAL world
  - e.g. 2-party SFE for OR, with output going to only one party (trivial against active adversary; impossible without computational assumptions against passive adversary)
Oblivious Transfer

Pick one out of two, without revealing which

Intuitive property: transfer partial information “obliviously”

A: up, B: down

IDEAL World

RECALL
2-Party (Passive)
Secure Function Evaluation
2-Party (Passive) Secure Function Evaluation

Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
- \(f(x_0,x_1;b) = \text{none}; \ g(x_0,x_1;b) = x_b\)
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob.
- OT is an instance of 2-party SFE:
  \[ f(x_0, x_1; b) = \text{none}; \quad g(x_0, x_1; b) = x_b \]
- **Symmetric SFE**: both parties get the same output.
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
  \[
f(x^0_0, x^1_1; b) = \text{none}; \quad g(x^0_0, x^1_1; b) = x^b_1
  \]
- Symmetric SFE: both parties get the same output
  - e.g. \(f(x^0_0, x^1_1; b, z) = g(x^0_0, x^1_1; b, z) = x^b_1 \oplus z\) [OT from this! How?]
2-Party (Passive)
Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
  - \(f(x_0, x_1; b) = \text{none}; g(x_0, x_1; b) = x_b\)
- Symmetric SFE: both parties get the same output
  - e.g., \(f(x_0, x_1; b, z) = g(x_0, x_1; b, z) = x_b \oplus z\) [OT from this! How?]
- General SFE from appropriate symmetric SFE [How?]
2-Party (Passive) Secure Function Evaluation

Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob

OT is an instance of 2-party SFE

\[
f(x_0, x_1; b) = \text{none}; \quad g(x_0, x_1; b) = x_b
\]

Symmetric SFE: both parties get the same output

e.g. \(f(x_0, x_1; b, z) = g(x_0, x_1; b, z) = x_b \oplus z\) [OT from this! How?]

General SFE from appropriate symmetric SFE [How?]

One-sided SFE: only one party gets any output
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
  - \(f(x_0,x_1;b) = \text{none} \); \(g(x_0,x_1;b) = x_b\)
- **Symmetric SFE:** both parties get the same output
  - e.g. \(f(x_0,x_1;b,z) = g(x_0,x_1;b,z) = x_b \oplus z\) [OT from this! *How?*
  - General SFE from appropriate symmetric SFE [How?]
- **One-sided SFE:** only one party gets any output
  - Symmetric SFE from one-sided SFE (passive secure) [How?]
2-Party (Passive) Secure Function Evaluation

- Functionality takes \((X;Y)\) and outputs \(f(X;Y)\) to Alice, \(g(X;Y)\) to Bob
- OT is an instance of 2-party SFE
  - \(f(x_0, x_1; b) = \text{none}; g(x_0, x_1; b) = x_b\)
- Symmetric SFE: both parties get the same output
  - e.g. \(f(x_0, x_1; b, z) = g(x_0, x_1; b, z) = x_b \oplus z\) [OT from this! How?]
  - General SFE from appropriate symmetric SFE [How?]
- One-sided SFE: only one party gets any output
  - Symmetric SFE from one-sided SFE (passive secure) [How?]
- So, for passive security, enough to consider one-sided SFE
2-Party Secure Function Evaluation
2-Party Secure Function Evaluation

- Randomized Functions: f(X;Y;r)
Randomized Functions: \( f(X;Y;r) \)

Neither party should know \( r \) (beyond what is revealed by output)
2-Party Secure Function Evaluation

Randomized Functions: \( f(X;Y;r) \)

- Neither party should know \( r \) (beyond what is revealed by output)
- Evaluating \( f'(X,a;Y,b) := f(X;Y;a \oplus b) \) with random \( a, b \) works
2-Party Secure Function Evaluation

- Randomized Functions: \( f(X;Y;r) \)
  - Neither party should know \( r \) (beyond what is revealed by output)
  - Evaluating \( f'(X,a;Y,b) := f(X;Y;a \oplus b) \) with random \( a,b \) works
  - Note \( f' \) is deterministic
2-Party Secure Function Evaluation

Randomized Functions: \( f(X;Y;r) \)

- Neither party should know \( r \) (beyond what is revealed by output)

Evaluating \( f'(X,a;Y,b) := f(X;Y;a \oplus b) \) with random \( a,b \) works

- Note \( f' \) is deterministic

For passive security, realizing deterministic, one-sided SFE enough for all SFE
2-Party Secure Function Evaluation

Randomized Functions: \( f(X;Y;r) \)

Neither party should know \( r \) (beyond what is revealed by output)

Evaluating \( f'(X,a;Y,b) := f(X;Y;a \oplus b) \) with random \( a,b \) works

Note \( f' \) is deterministic

For passive security, realizing deterministic, one-sided SFE enough for all SFE

Can we do “general” deterministic, one-sided SFE (i.e., for all functions)?
Boolean Circuits
Boolean Circuits

Directed acyclic graph
Boolean Circuits

- Directed acyclic graph
- Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
Boolean Circuits

- Directed acyclic graph
  - Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
  - Edges: Boolean valued wires
Boolean Circuits

- Directed acyclic graph
  - Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
  - Edges: Boolean valued wires
  - Each wire comes out of a unique gate
Boolean Circuits

- Directed acyclic graph
  - Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
  - Edges: Boolean valued wires
  - Each wire comes out of a unique gate
  - But a wire might fan-out
Boolean Circuits

- Directed acyclic graph
  - Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
  - Edges: Boolean valued wires
  - Each wire comes out of a unique gate
    - But a wire might fan-out
  - Acyclic: output well-defined
Boolean Circuits

- Directed acyclic graph
  - Nodes: AND, OR, NOT, CONST gates, inputs, output(s)
  - Edges: Boolean valued wires
  - Each wire comes out of a unique gate
    - But a wire might fan-out
  - Acyclic: output well-defined
    - Note: no memory gates
Circuits and Functions
Circuits and Functions

e.g.: OR (single gate, 2 input bits, 1 bit output)
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: $X > Y$ for two bit inputs $X=x_1x_0$, $Y=y_1y_0$:
  
  $$(x_1 \text{ AND } (\text{NOT } y_1)) \text{ OR } (\text{NOT}(x_1 \text{ XOR } y_1) \text{ AND } (x_0 \text{ AND } (\text{NOT } y_0)))$$
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: X > Y for two bit inputs $X=x_1x_0$, $Y=y_1y_0$:
  $$ (x_1 \text{ AND} (\text{NOT} y_1)) \text{ OR} (\text{NOT}(x_1 \text{ XOR} y_1) \text{ AND} (x_0 \text{ AND} (\text{NOT} y_0))) $$
- Can convert any (“efficient”) program into a (“small”) circuit
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: $X > Y$ for two bit inputs $X=x_1x_0$, $Y=y_1y_0$:
  $$(x_1 \text{ AND } \neg y_1) \text{ OR } (\neg(x_1 \text{ XOR } y_1) \text{ AND } (x_0 \text{ AND } \neg y_0))$$

Can convert any ("efficient") program into a ("small") circuit

- Size of circuit: number of wires (as a function of number of input wires)
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: $X > Y$ for two bit inputs $X=x_1x_0$, $Y=y_1y_0$:
  $\left( x_1 \text{ AND } (\text{NOT } y_1) \right) \text{ OR } \left( \text{NOT}(x_1 \text{ XOR } y_1) \text{ AND } (x_0 \text{ AND } \text{NOT } y_0) \right)$
- Can convert any (“efficient”) program into a (“small”) circuit
- Size of circuit: number of wires (as a function of number of input wires)
- Can convert a truth-table into a circuit

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: $X > Y$ for two bit inputs $X=x_1x_0$, $Y=y_1y_0$:
  $$(x_1 \text{AND (NOT } y_1)) \text{ OR (NOT}(x_1 \text{ XOR } y_1) \text{ AND }(x_0 \text{ AND (NOT } y_0))$$
- Can convert any ("efficient") program into a ("small") circuit
- Size of circuit: number of wires (as a function of number of input wires)
- Can convert a truth-table into a circuit
- Directly: circuit size exponential in input size
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)
- e.g.: $X > Y$ for two bit inputs $X=x_1x_0$, $Y=y_1y_0$: $(x_1 \text{ AND } (\text{NOT } y_1)) \text{ OR } (\text{NOT}(x_1 \text{ XOR } y_1) \text{ AND } (x_0 \text{ AND } (\text{NOT } y_0)))$
- Can convert any ("efficient") program into a ("small") circuit
  - Size of circuit: number of wires (as a function of number of input wires)
- Can convert a truth-table into a circuit
  - Directly: circuit size exponential in input size
- In general, finding a small/smallest circuit from truth-table is notoriously hard
Circuits and Functions

- e.g.: OR (single gate, 2 input bits, 1 bit output)

- e.g.: \( X > Y \) for two bit inputs \( X=x_1x_0, Y=y_1y_0: \)
\[
(x_1 \text{ AND } (\text{NOT } y_1)) \text{ OR } (\text{NOT}(x_1 \text{ XOR } y_1) \text{ AND } (x_0 \text{ AND } (\text{NOT } y_0)))
\]

- Can convert any (“efficient”) program into a (“small”) circuit

- Size of circuit: number of wires (as a function of number of input wires)

- Can convert a truth-table into a circuit

- Directly: circuit size exponential in input size

- In general, finding a small/smallest circuit from truth-table is notoriously hard

- Often problems already described as succinct programs/circuits
2-Party SFE using General Circuits
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit

One-sided output: both parties give inputs, one party gets outputs
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit

One-sided output: both parties give inputs, one party gets outputs

Either party maybe corrupted passively
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit

One-sided output: both parties give inputs, one party gets outputs

Either party maybe corrupted passively

Consider evaluating OR (single gate circuit)
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit

One-sided output: both parties give inputs, one party gets outputs

Either party maybe corrupted passively

Consider evaluating OR (single gate circuit)

Alice holds $x=a$, Bob has $y=b$; Bob should get $\text{OR}(x,y)$
2-Party SFE using General Circuits

“General”: evaluate any arbitrary circuit

One-sided output: both parties give inputs, one party gets outputs

Either party maybe corrupted passively

Consider evaluating OR (single gate circuit)

Alice holds $x=a$, Bob has $y=b$; Bob should get $\text{OR}(x,y)$

Any ideas?
A Physical Protocol
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$.
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$.

Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box).
A Physical Protocol

- Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$.

- Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box).
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)

She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$.

Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box).

She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).
A Physical Protocol

- Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$
- Inside $B_{xy=ab}$ she places the bit OR(a,b) and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)
- She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).
A Physical Protocol

Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

Inside $B_{xy}=ab$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)

She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).
A Physical Protocol

- Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

- Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)

- She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).

- So far Bob gets no information
A Physical Protocol

- Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

- Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)

- She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).

  - So far Bob gets no information

- Bob “obliviously picks up” $K_{y=b}$, and tries the two keys $K_x, K_y$ on the four boxes. For one box both locks open and he gets the output.
A Physical Protocol

- Alice prepares 4 boxes $B_{xy}$ corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$

- Inside $B_{xy=ab}$ she places the bit $\text{OR}(a,b)$ and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)

- She un-labels the four boxes and sends them in random order to Bob. Also sends the key $K_{x=a}$ (labeled only as $K_x$).

  - So far Bob gets no information

- Bob “obliviously picks up” $K_{y=b}$, and tries the two keys $K_x,K_y$ on the four boxes. For one box both locks open and he gets the output.
A Physical Protocol
A Physical Protocol

Secure?
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_{y=b}$.
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y=b$

But this is done “obliviously”, and so she learns nothing
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y = b$

But this is done “obliviously”, and so she learns nothing

For curious Bob: Everything is predictable (i.e., simulatable), given the final outcome
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y = b$

But this is done "obliviously", and so she learns nothing

For curious Bob: Everything is predictable (i.e., simulatable), given the final outcome

What Bob sees: His key opens $K_y$ in two boxes, Alice's opens $K_x$ in two boxes; only one random box fully opens. It has the outcome.
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y=b$

But this is done “obliviously”, and so she learns nothing

For curious Bob: Everything is predictable (i.e., simulatable), given the final outcome

What Bob sees: His key opens $K_y$ in two boxes, Alice’s opens $K_x$ in two boxes; only one random box fully opens. It has the outcome.

Note when $y=1$, cases $x=0$ and $x=1$ appear same
A Physical Protocol

Secure?

For curious Alice: only influence from Bob is when he picks up his key $K_y = b$

But this is done "obliviously", and so she learns nothing

For curious Bob: Everything is predictable (i.e., simulatable), given the final outcome

What Bob sees: His key opens $K_y$ in two boxes, Alice’s opens $K_x$ in two boxes; only one random box fully opens. It has the outcome.

Note when $y=1$, cases $x=0$ and $x=1$ appear same

Formally, easy to simulate (can stuff unopenable boxes randomly)
Larger Circuits
Larger Circuits
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.
Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$. 
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire w in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.

For each gate G with input wires (u,v) and output wire w, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.

For each gate $G$ with input wires $(u,v)$ and output wire $w$, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.

Give to Bob: Boxes for each gate, one key for each of Alice's input wires.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

- For each wire $w$ in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.
- For each gate $G$ with input wires $(u,v)$ and output wire $w$, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.
- Give to Bob: Boxes for each gate, one key for each of Alice’s input wires.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire w in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.

For each gate G with input wires (u,v) and output wire w, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.

Give to Bob: Boxes for each gate, one key for each of Alice’s input wires.

Obliviously: one key for each of Bob’s input wires.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire \( w \) in the circuit (i.e., input wires, or output of a gate) pick 2 keys \( K_{w=0} \) and \( K_{w=1} \).

For each gate \( G \) with input wires \((u,v)\) and output wire \( w \), prepare 4 boxes \( B_{uv} \) and place \( K_{w=G(a,b)} \) inside box \( B_{uv=ab} \). Lock \( B_{uv=ab} \) with keys \( K_{u=a} \) and \( K_{v=b} \).

Give to Bob: Boxes for each gate, one key for each of Alice’s input wires.

Obliviously: one key for each of Bob’s input wires.
Larger Circuits

Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate.

For each wire w in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$.

For each gate G with input wires (u,v) and output wire w, prepare 4 boxes $B_{uv}$ and place $K_{w=G(a,b)}$ inside box $B_{uv=ab}$. Lock $B_{uv=ab}$ with keys $K_{u=a}$ and $K_{v=b}$.

Give to Bob: Boxes for each gate, one key for each of Alice’s input wires.

Obliviously: one key for each of Bob’s input wires.

Boxes for output gates have values instead of keys.
Larger Circuits
Larger Circuits

Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds
Larger Circuits

Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds.

Gets output from a box in the output gate.
Larger Circuits

Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds

- Gets output from a box in the output gate
- Security similar to before
Larger Circuits

- Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds.

- Gets output from a box in the output gate.

- Security similar to before.

- Curious Alice sees nothing (as Bob picks up keys obliviously).
Evaluation: Bob gets one key for each input wire of a gate, opens one box for the gate, gets one key for the output wire, and proceeds

- Gets output from a box in the output gate

Security similar to before

- Curious Alice sees nothing (as Bob picks up keys obliviously)

- Everything is simulatable for curious Bob given final output: Bob could prepare boxes and keys (stuffing unopenable boxes arbitrarily); for an output gate, place the output bit in the box that opens
Garbled Circuit
Garbled Circuit

That was too physical!
Garbled Circuit

That was too physical!

Yao’s Garbled circuit: boxes/keys replaced by IND-CPA secure SKE (i.e., using PRF, and independent randomness when key reused)
Garbled Circuit

- That was too physical!

- Yao’s Garbled circuit: boxes/keys replaced by IND-CPA secure SKE (i.e., using PRF, and independent randomness when key reused)

- Double lock: $\text{Enc}_{K_x}(\text{Enc}_{K_y}(m))$
That was too physical!

Yao’s Garbled circuit: boxes/keys replaced by IND-CPA secure SKE (i.e., using PRF, and independent randomness when key reused)

Double lock: $\text{Enc}_{K_x}(\text{Enc}_{K_y}(m))$

Need proof to ensure that this suffices for indistinguishability of simulation. (In fact, milder security for Enc suffices)
Garbled Circuit

That was too physical!

Yao’s Garbled circuit: boxes/keys replaced by IND-CPA secure SKE (i.e., using PRF, and independent randomness when key reused)

Double lock: $\text{Enc}_{K_x}(\text{Enc}_{K_y}(m))$

Need proof to ensure that this suffices for indistinguishability of simulation. (In fact, milder security for Enc suffices)

Oblivious Transfer: We already saw for one bit (using T-OWP); with passive adversaries, just repeat bit-OT several times to transfer longer keys
Garbled Circuit

- That was too physical!

- Yao’s Garbled circuit: boxes/keys replaced by IND-CPA secure SKE (i.e., using PRF, and independent randomness when key reused)

- Double lock: $\text{Enc}_{K_x}(\text{Enc}_{K_y}(m))$

- Need proof to ensure that this suffices for indistinguishability of simulation. (In fact, milder security for Enc suffices)

- Oblivious Transfer: We already saw for one bit (using T-OWP); with passive adversaries, just repeat bit-OT several times to transfer longer keys

- Can we really compose? Yes, for passive security.
Today
Today

- 2-Party SFE secure against passive adversaries
Today

- 2-Party SFE secure against passive adversaries
- Yao's Garbled Circuit
Today

- 2-Party SFE secure against passive adversaries
- Yao's Garbled Circuit
- Using OT and IND-CPA encryption
Today

- 2-Party SFE secure against passive adversaries
  - Yao’s Garbled Circuit
  - Using OT and IND-CPA encryption
  - OT using TOWP
Today

- 2-Party SFE secure against passive adversaries
- Yao's Garbled Circuit
- Using OT and IND-CPA encryption
  - OT using TOWP
- Composition (implicitly)
Today

- 2-Party SFE secure against passive adversaries
  - Yao’s Garbled Circuit
  - Using OT and IND-CPA encryption
    - OT using TOWP
  - Composition (implicitly)

Coming up: Zero-Knowledge proofs and general multi-party computation, more protocols (for different settings).
Universal Composition