

# Advanced Tools from Modern Cryptography

Lecture 7

Secure 2-Party Computation:  
Yao's Garbled Circuit

Recall

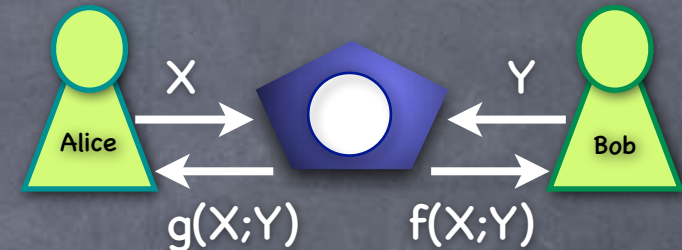
# MPC without Honest-Majority

- Plan (Still sticking with passive corruption):
- Two protocols, that are secure computationally
  - The “passive-GMW” protocol for any number of parties
  - A 2-party protocol using Yao’s Garbled Circuits
  - Both rely on a computational primitive called Oblivious Transfer
- Last time: OT and Passive-GMW
  - (Not exactly the version from the GMW’87 paper.)
- Today: 2-Party protocol using Yao’s Garbled Circuits

# 2-Party SFE

- Secure Function Evaluation (SFE) IDEAL:

- Trusted party takes  $(X;Y)$ . Outputs  $g(X;Y)$  to Alice,  $f(X;Y)$  to Bob



- Randomized Functions:  $g(X;Y;r)$  and  $f(X;Y;r)$  s.t. neither party knows  $r$  (beyond what is revealed by output)
- OT is an instance of a (deterministic) 2-party SFE
  - $g(x_0, x_1; b) = \text{none}; f(x_0, x_1; b) = x_b$
- Single-Output SFE: only one party gets any output

# 2-Party SFE

- Can reduce general SFE (even randomized) to a single-output deterministic SFE

$$f'(X, M, r_1; Y, r_2) = ( g(X; Y; r_1 \oplus r_2) \oplus M, f(X; Y; r_1 \oplus r_2) ).$$

Compute  $f'(X, M, r_1; Y, r_2)$  with random  $M, r_1, r_2$

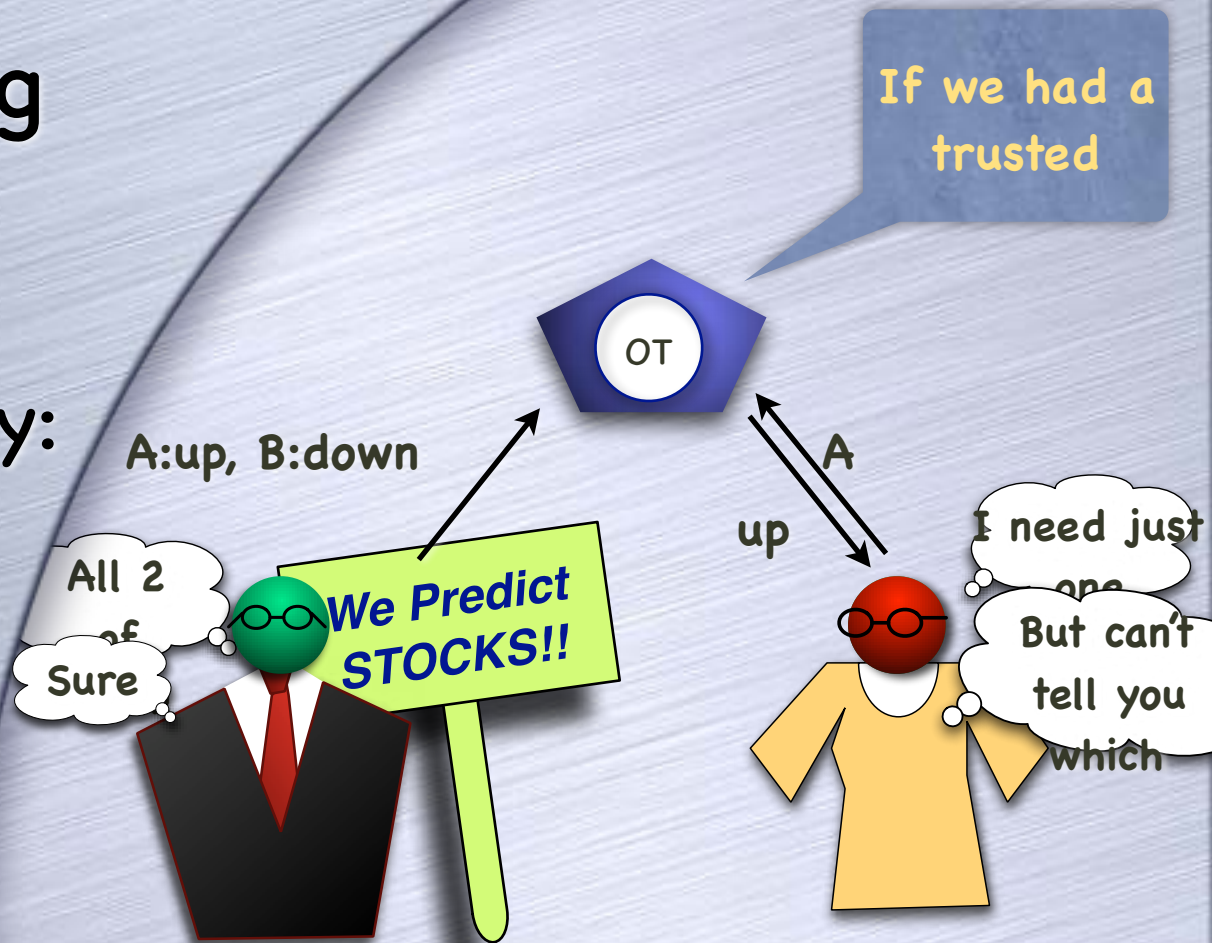
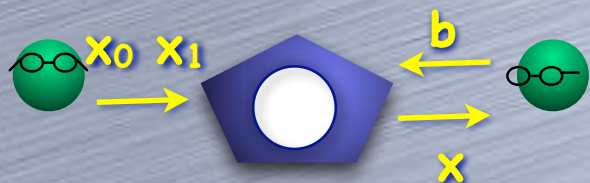
- Bob sends  $g(X, Y; r_1 \oplus r_2) \oplus M$  to Alice
- Passive secure
- For active security too:  $f'$  authenticates (one-time MAC) as well as encrypts  $g(X; Y; r_1 \oplus r_2)$  using keys input by Alice
- Generalizes to more than 2 parties too [Exercise]
- Yao: Reduces single-output deterministic 2-party SFE to OT
- Single round of interaction, but with only computational security (cf. GMW: information-theoretic, but many rounds)

Recall

# Oblivious Transfer

- Pick one out of two, without revealing which

- Intuitive property: transfer partial information



Recall

# Why is OT Useful?

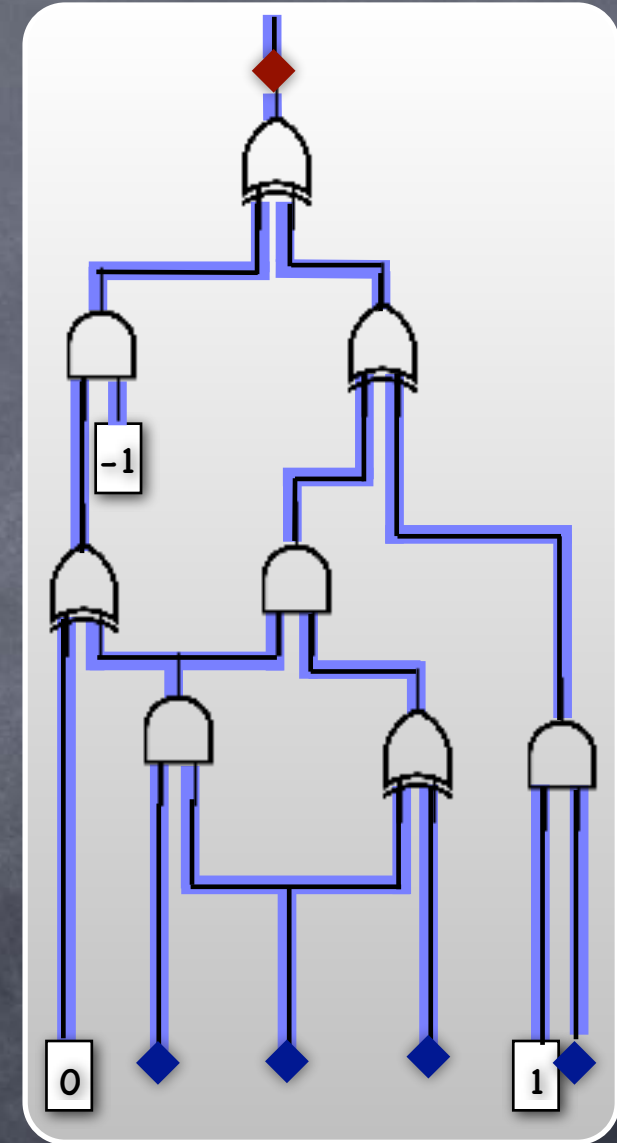
## Naïve 2PC from OT

- Say Alice's input  $x$ , Bob's input  $y$ , and only Bob should learn  $f(x,y)$
- Alice (who knows  $x$ , but not  $y$ ) prepares a table for  $f(x, \cdot)$  with  $D = 2^{|y|}$  entries (one for each  $y$ )
- Bob uses  $y$  to decide which entry in the table to pick up using 1-out-of- $D$  OT (without learning the other entries)
- Bob learns only  $f(x,y)$  (in addition to  $y$ ). Alice learns nothing beyond  $x$ .
- OT captures the essence of MPC:  
**Secure computation of any function  $f$  can be reduced to OT**
- Problem:  $D$  is exponentially large in  $|y|$ 
  - Plan: somehow exploit efficient computation (e.g., circuit) of  $f$

Secure protocol for  $f$  using access to ideal OT

# Functions as Circuits

- Directed acyclic graph
  - Nodes: multiplication and addition gates, constant gates, inputs, output(s)
  - Edges: wires carrying values from  $F$
  - Each wire comes out of a unique gate, but a wire might fan-out
  - Can evaluate wires according to a topologically sorted order of gates they come out of



# 2-Party MPC for General Circuits

	0	1
0	0	1
1	1	1

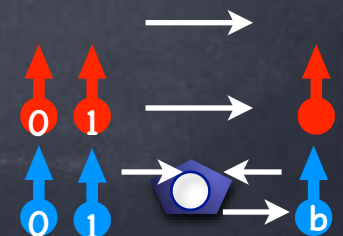
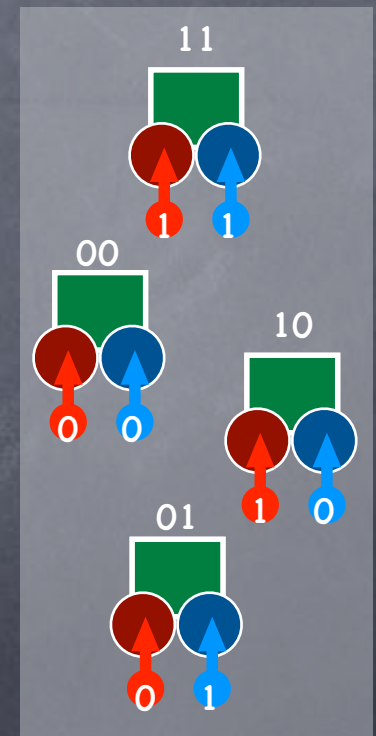
- “General”: evaluate any arbitrary (boolean) circuit
  - One-sided output: both parties give inputs, one party gets outputs
  - Either party maybe corrupted passively
- Consider evaluating OR (single gate circuit)
  - Alice holds  $x=a$ , Bob has  $y=b$ ; Bob should get  $OR(x,y)$



# 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  $K_x$ ).
  - 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.

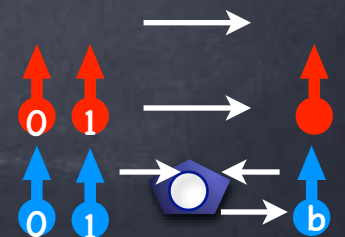
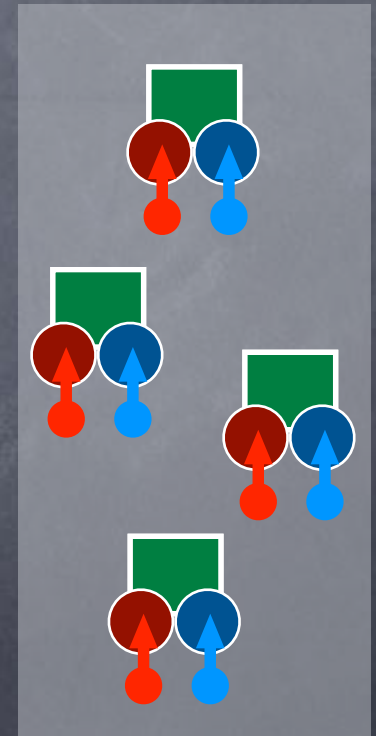
	0	1
0	0	1
1	1	1



# A Physical Protocol

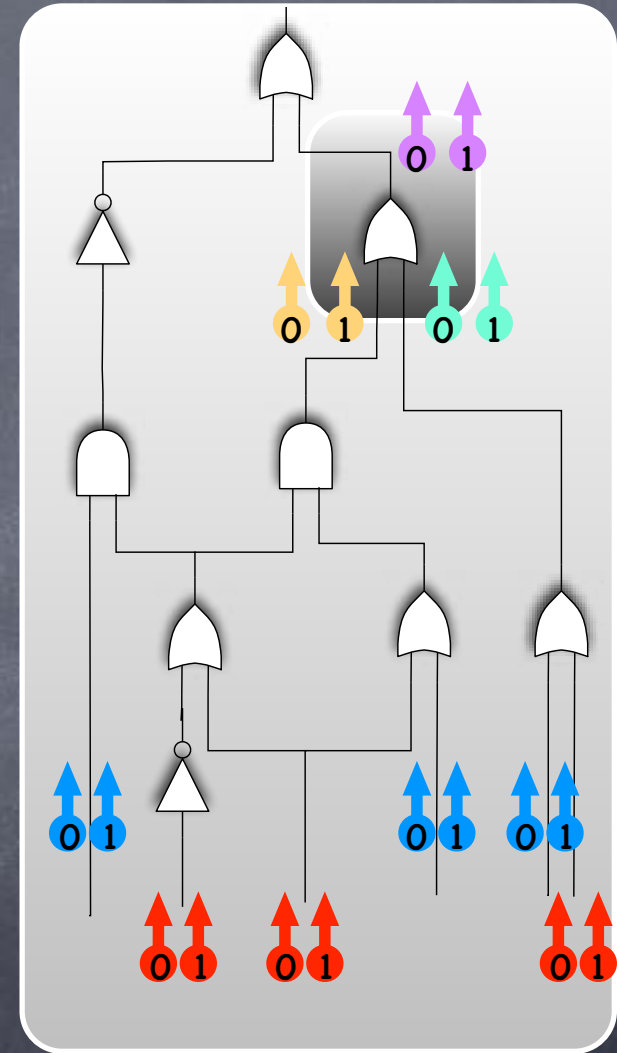
- Secure?
- For curious Alice: only influence from Bob is when he picks up his key  $K_{y=b}$ 
  - But this is done "obliviously", so she learns nothing
- For curious Bob: What he sees is predictable (i.e., simulatable), given the final outcome
  - What Bob sees: His key opens  $K_y$  in two boxes, Alice's 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

	0	1
0	0	1
1	1	1



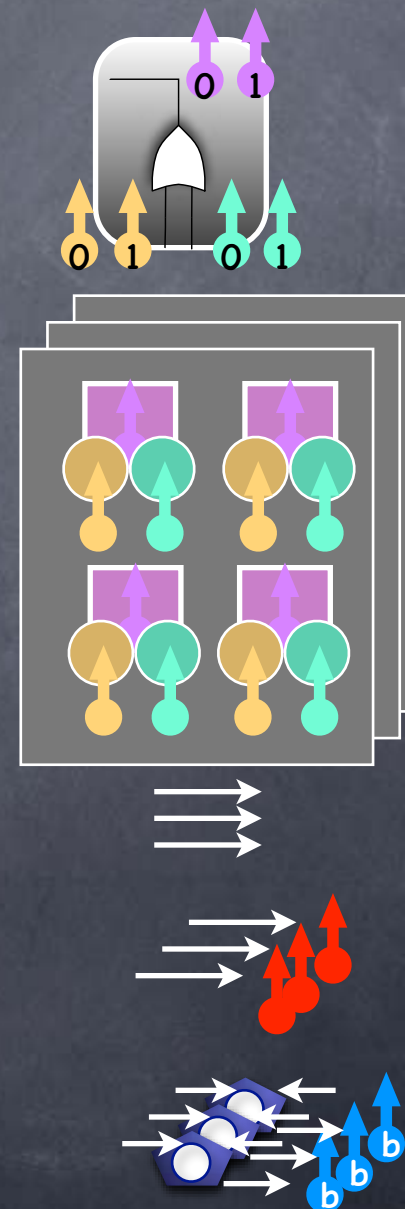
# Larger Circuits

- 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  $K_{w=1}$



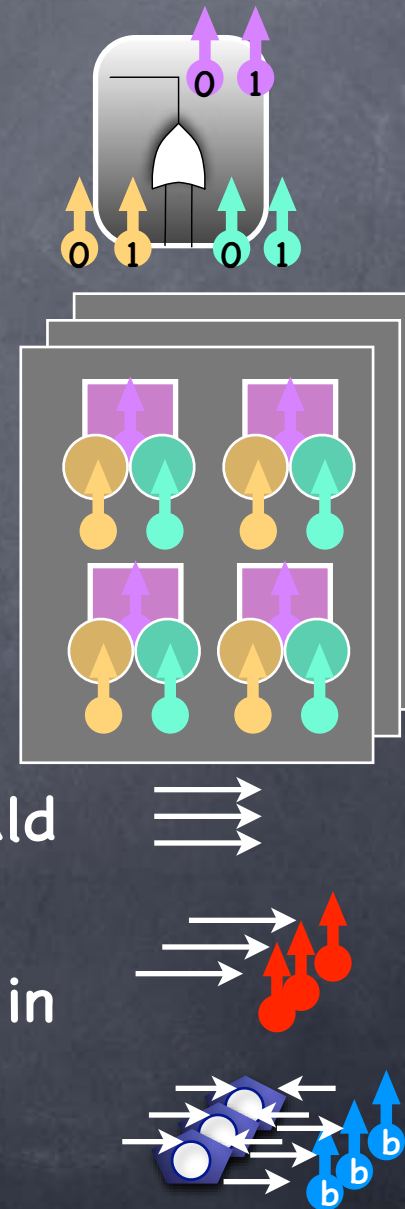
# Larger Circuits

- 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  $K_{v=b}$
- Give to Bob: Boxes for each gate, one key for each of Alice's input wires
  - Obviously: one key for each of Bob's input wires
- Boxes for output gates have values instead of keys



# 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 for the output gate
- Security similar to before
  - Curious Alice sees nothing
  - Bob can simulate his view 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

Coming up

- That was too physical!
- Yao's Garbled circuit: boxes/keys replaced by **Symmetric Key Encryption** (specifically, using a **Pseudorandom Function** or **PRF**)
  - $\text{Enc}_K(m) = \text{PRF}_K(\text{index}) \oplus m$ , where index is a wire index (distinct for different wires fanning-out of the same gate)
  - Double lock:  $\text{Enc}_{K_x}(\text{Enc}_{K_y}(m))$
  - PRF in practice: a block-cipher, like AES
- Uses Oblivious Transfer for strings: For passive security, can just repeat bit-OT several times to transfer longer keys
- Security? Need to first define security when computational primitives are used! (Next time!)

# Garbled Circuit

- One issue when using encryption instead of locks
  - Given four doubly locked boxes (in random order) and two keys, we simply tried opening all locks until one box fully opened
  - With encryption, cannot quite tell if a box opened or not! Outcome of decryption looks random in either case.
  - Simple solution: encode the keys so that wrong decryption does not result in outputs that look like valid encoding of keys
  - Better solution: For each wire 0 & 1 keys have distinct “shape” labels, assigned at random. Each locked box marked with the shape of the two keys needed to unlock it.

