Advanced Tools from Modern Cryptography

Lecture 6

Secure Multi-Party Computation:

Yao's Garbled Circuit

MPC for Passive Corruption

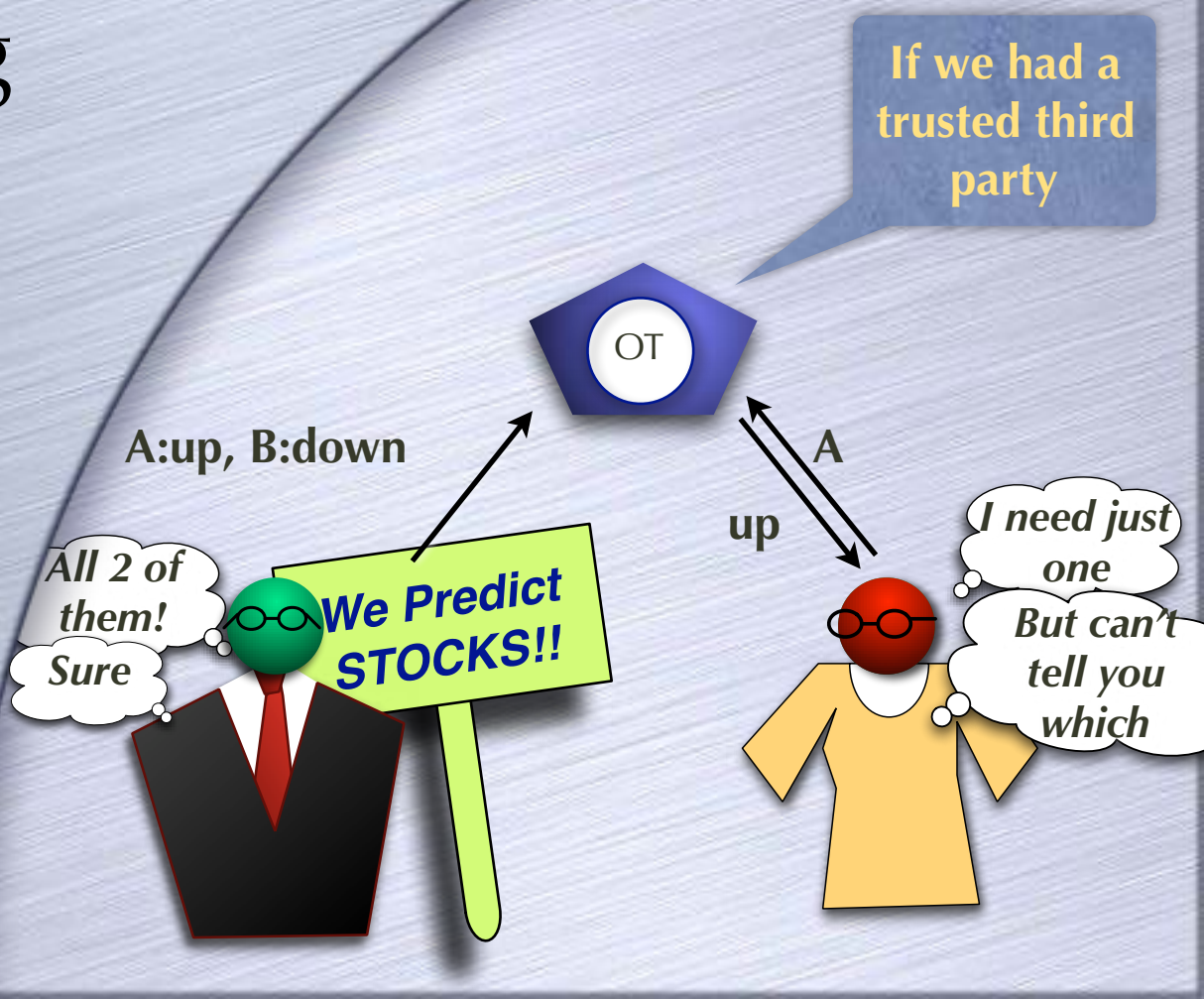
- Story so far:
  - For honest-majority: Information-theoretically secure protocol, using Shamir secret-sharing [BGW]
  - Without honest-majority: Using Oblivious Transfer (OT), using additive secret-sharing [GMW]
- Today
  - A 2-party protocol (so no honest-majority) using Oblivious Transfer and Yao's Garbled Circuits
    - Uses additional computational primitives and is limited to arithmetic circuits over small fields (e.g., boolean circuits)
    - Needs just one round of interaction
    - Garbled Circuits have other applications too

Oblivious Linear-function Evaluation (OLE) for large fields (Exercise)

Recall

Oblivious Transfer

- Pick one out of two, without revealing which
  - Intuitive property: transfer partial information "obliviously"



Recall

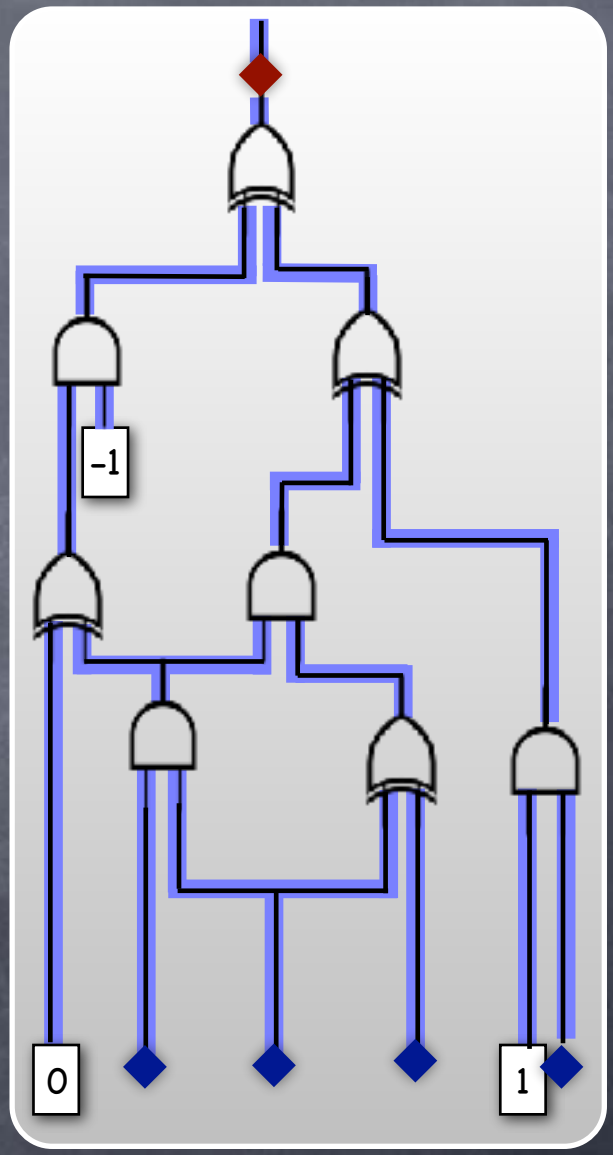
Naïve 2PC from OT

- 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
- Problem: D is exponentially large in |y|

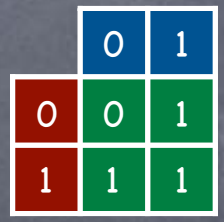
Recall

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



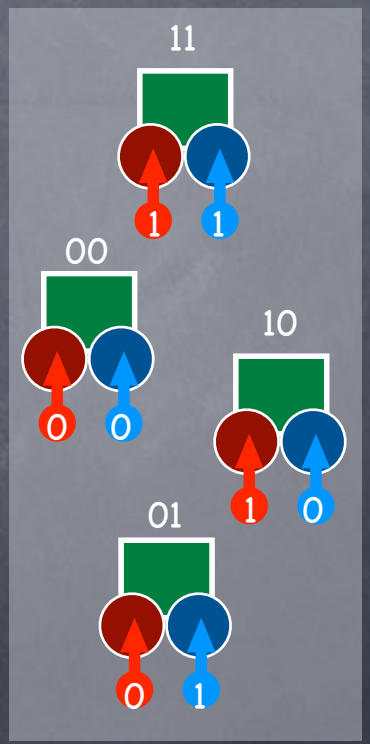
2-Party MPC for General Circuits

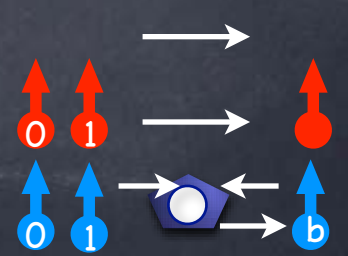


- "General": evaluate any arbitrary (boolean) circuit
  - One-sided output: both parties give inputs, one party gets outputs
  - Either party maybe corrupted passively
- Consider evaluating OR (single gate circuit)
  - Alice holds x=a, Bob has y=b; Bob should get OR(x,y)

A Physical Protocol

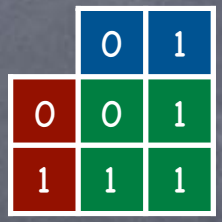
- Alice prepares 4 boxes B_{xy} corresponding to 4 possible input scenarios, and 4 padlocks/keys $K_{x=0}$, $K_{x=1}$, $K_{y=0}$ and $K_{y=1}$
- Inside $B_{xy=ab}$ she places the bit OR(a,b) and locks it with two padlocks $K_{x=a}$ and $K_{y=b}$ (need to open both to open the box)
- She un-labels the 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.

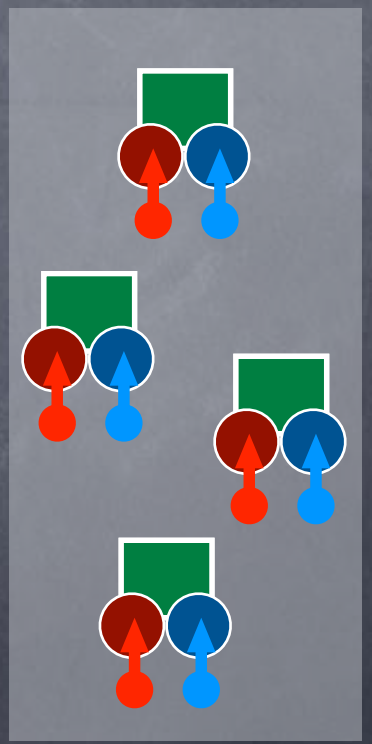


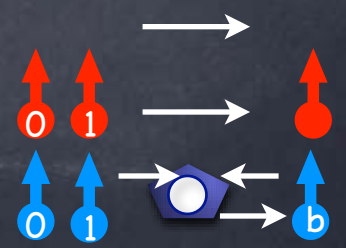


A Physical Protocol

- 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

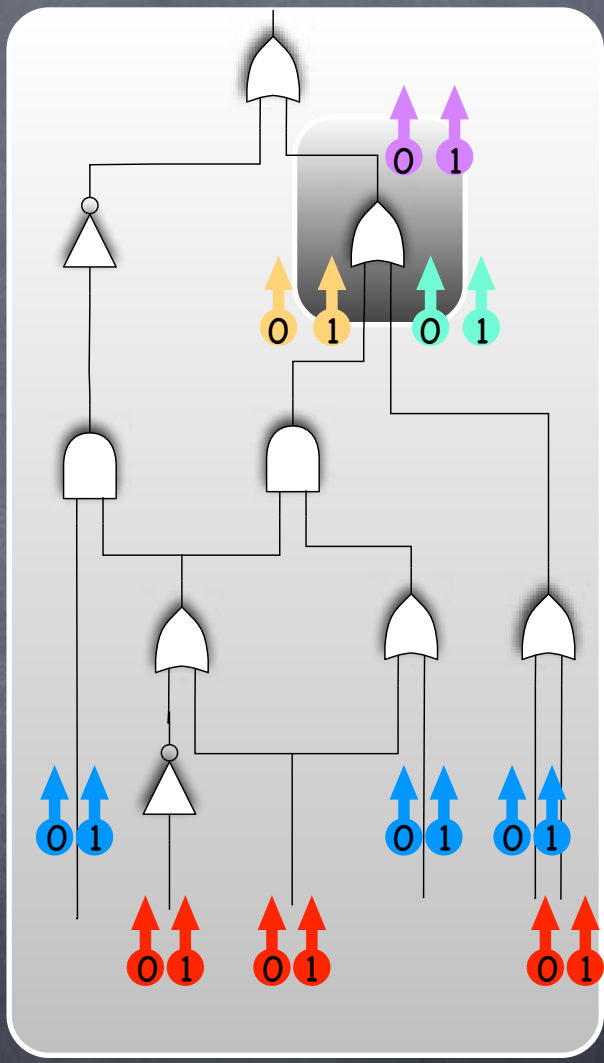






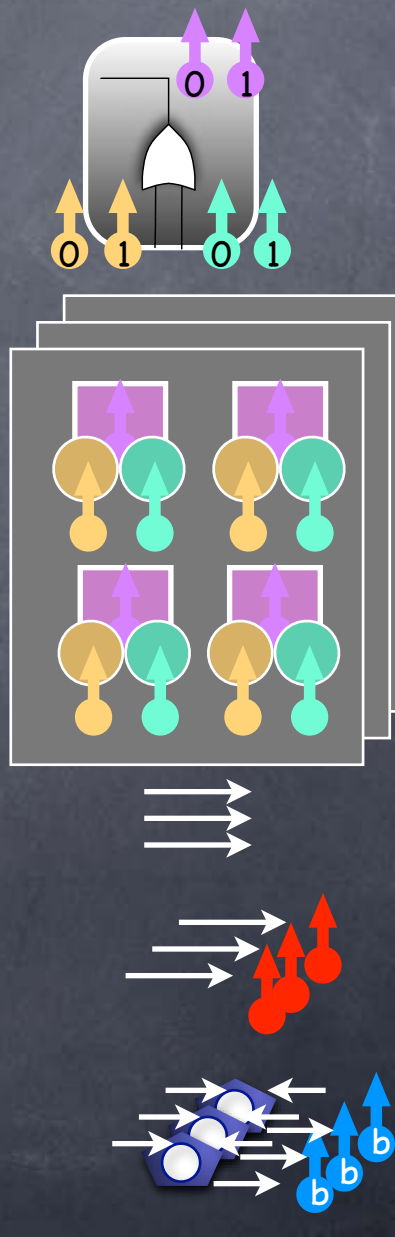
Larger Circuits

- 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 $K_{w=1}$



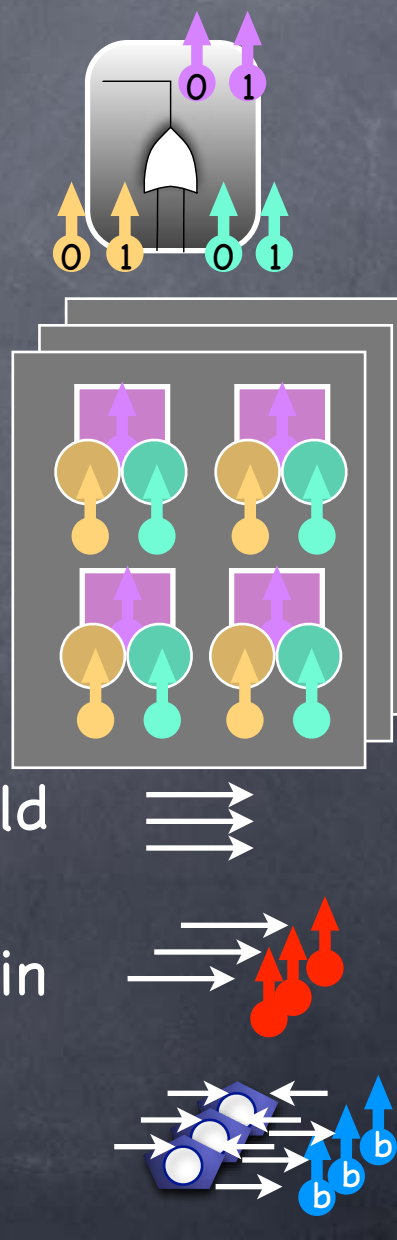
Larger Circuits

- 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
    - Obliviously: 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

- That was too physical!
- Yao's Garbled circuit: boxes/keys replaced by Symmetric Key Encryption (specifically, using a Pseudorandom Function or PRF)
  - $Enc_K(m) = PRF_K(index) \oplus m$, where index is a wire index (distinct for different wires fanning-out of the same gate)
  - Double lock: $Enc_{Kx}(Enc_{Ky}(m))$
  - 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 minor 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: attach a "pointer" label (random, distinct) for each key. (A single bit suffices, since a key's wire is known.) Locked boxes marked with the pointers of the two keys needed to unlock them.