Advanced Tools from Modern Cryptography

Lecture 5

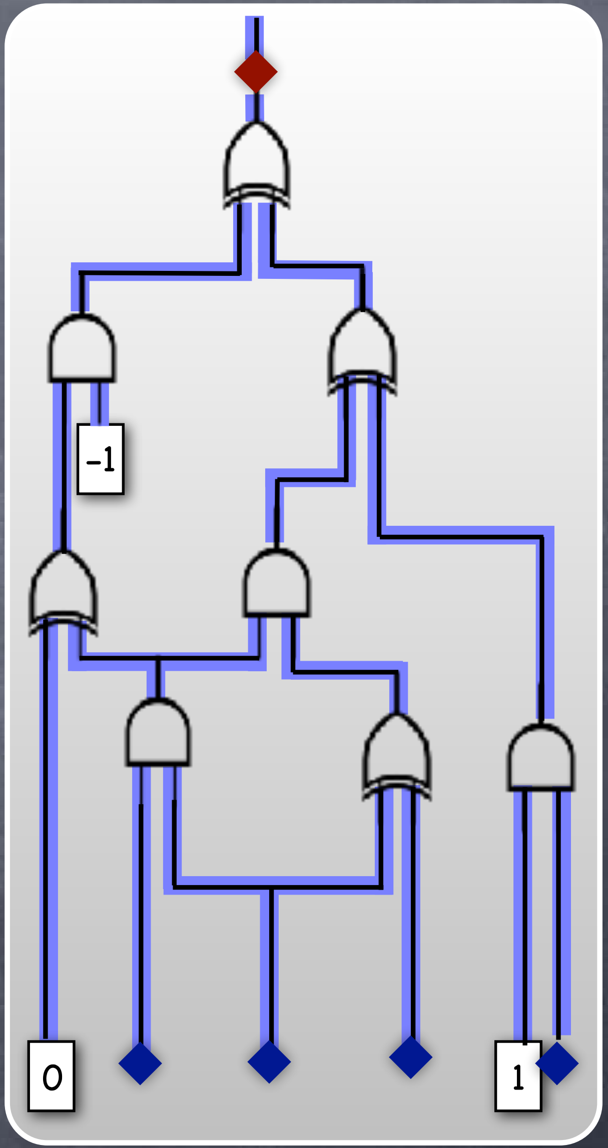
Secure Multi-Party Computation: Passive Corruption

MPC: Honest-Majority + Passive-Corruption

- Can achieve information-theoretic security for any function
- Function should be given as an arithmetic circuit over a large enough field (|F| > #parties)
  - Gate-by-gate evaluation, under Shamir secret-sharing of wires

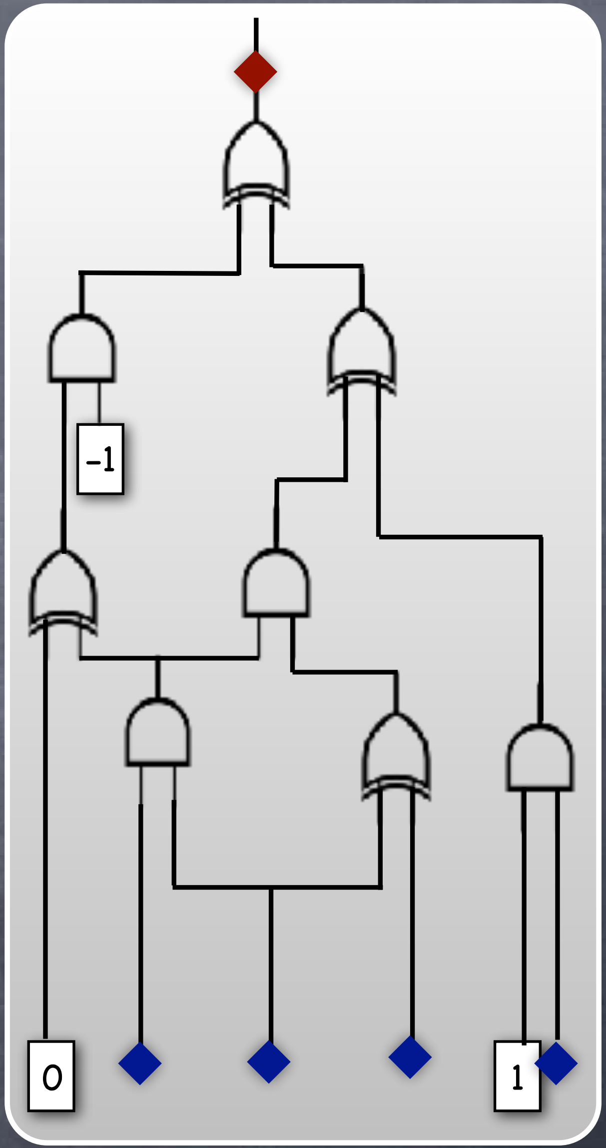
Functions as Circuits

- 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



Gate-by-Gate Evaluation

- Wire values will be kept Shamir-secret-shared among all parties
- Linear operations "free" (no communication)
- Multiplication involves **degree reduction**: reshare the higher-degree "product" shares and locally reconstruct shares of the original degree
- Efficiency proportional to the number of multiplication gates in the circuit (not counting multiplication with constants)
- To use degree d Shamir secret-sharing need N > 2d parties. Can tolerate only d < N/2 corrupt parties.



MPC: Honest-Majority + Passive-Corruption

- Can achieve information-theoretic security for any function
- Function should be given as an arithmetic circuit over a large enough field (|F| > #parties)
- Can tolerate corruption of strictly less than N/2 parties
  - e.g., 1 party out of 3, or 2 parties out of 5
  - No security in a 2-party setting!
- Q: For which functions can we obtain information-theoretic security against N/2 (or more) corruption?
  - Not all functions!
  - Exactly known for N=2 (later)
  - General case is still an open problem!

Information-Theoretic MPC: No Honest-Majority

- Need honest majority for computing AND
- Enough to show that 2 parties cannot compute AND securely
  - Because, if there were an N-party AND protocol tolerating N/2 corrupt parties, we can convert it into a 2-party protocol for AND as follows:
    - Alice runs $P_1,\ldots,P_{N/2}$ "in her head", with her input as P_1's input and 1 as input for the others. Bob runs the remaining parties similarly.
    - View of the parties in Alice's head don't reveal anything about Bob's input, other than what the AND reveals

Information-Theoretic MPC: No Honest-Majority

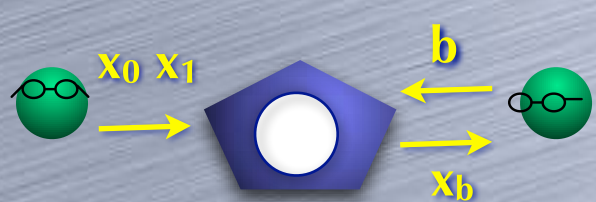
- Need honest majority for computing AND
- Enough to show that 2 parties cannot compute AND securely
  - Suppose there is a 2-party protocol for AND. Consider a transcript m such that $\Pr[m|x=0,y=0] = p > 0$.
  - By security against Alice, $\Pr[m|x=0,y=1] = p$.
    And by security against Bob, $\Pr[m|x=1,y=0] = p$.
  - How about $\Pr[m|x=1,y=1]$? Should be 0, for correctness
    - Suppose $m=m_1m_2\ldots m_t$, with Alice sending the first message. Alice with $x=1$ will send m_1 with positive probability because $\Pr[m|x=1,y=0] > 0$. Bob with $y=1$, and given m_1 will send m_2 with positive probability, etc. Hence $\Pr[m|x=1,y=1] > 0$!

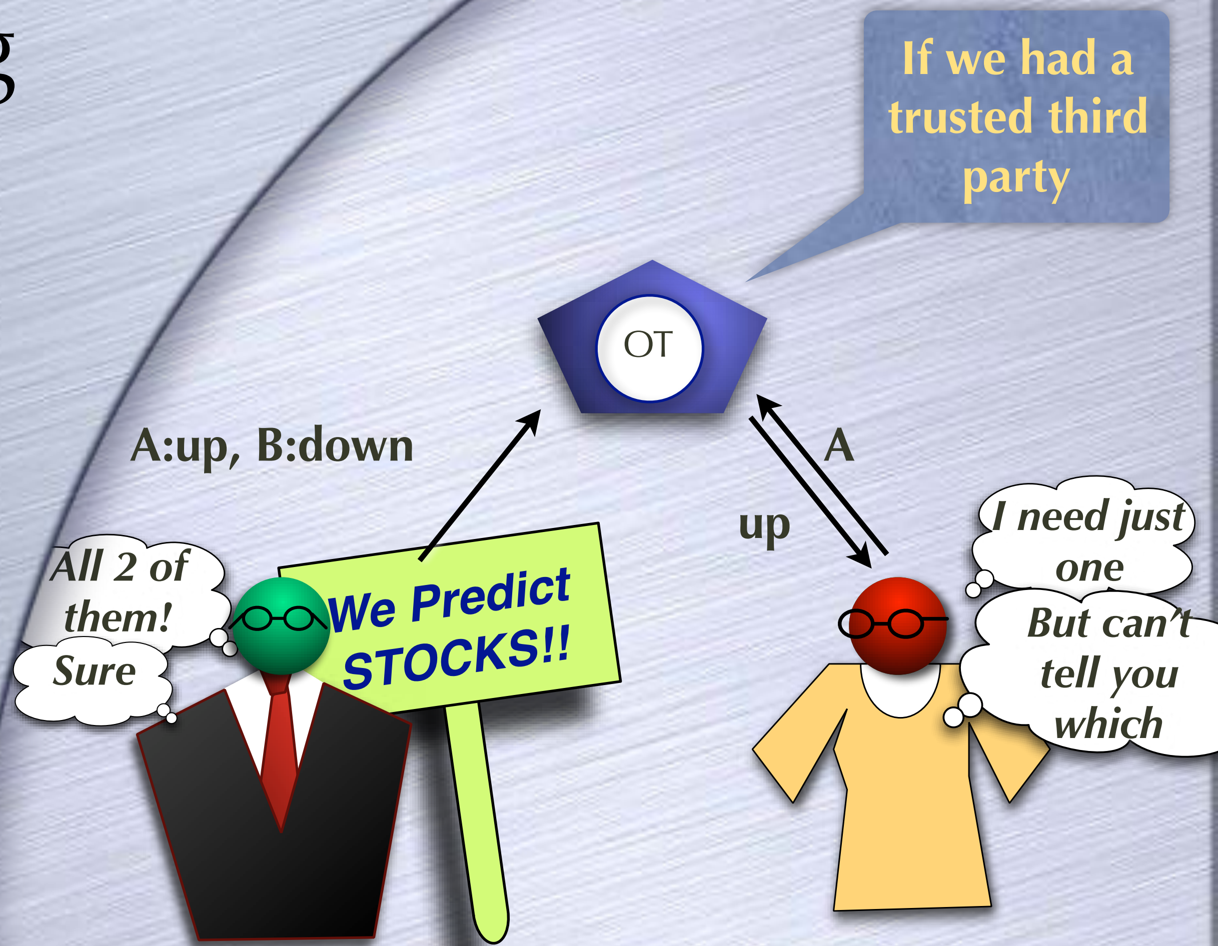
MPC without Honest-Majority

- 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 <u>Oblivious Transfer</u>
- Today: OT and Passive-GMW
  - (Not exactly the version from the GMW’87 paper.)

Oblivious Transfer

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





Why is OT Useful?

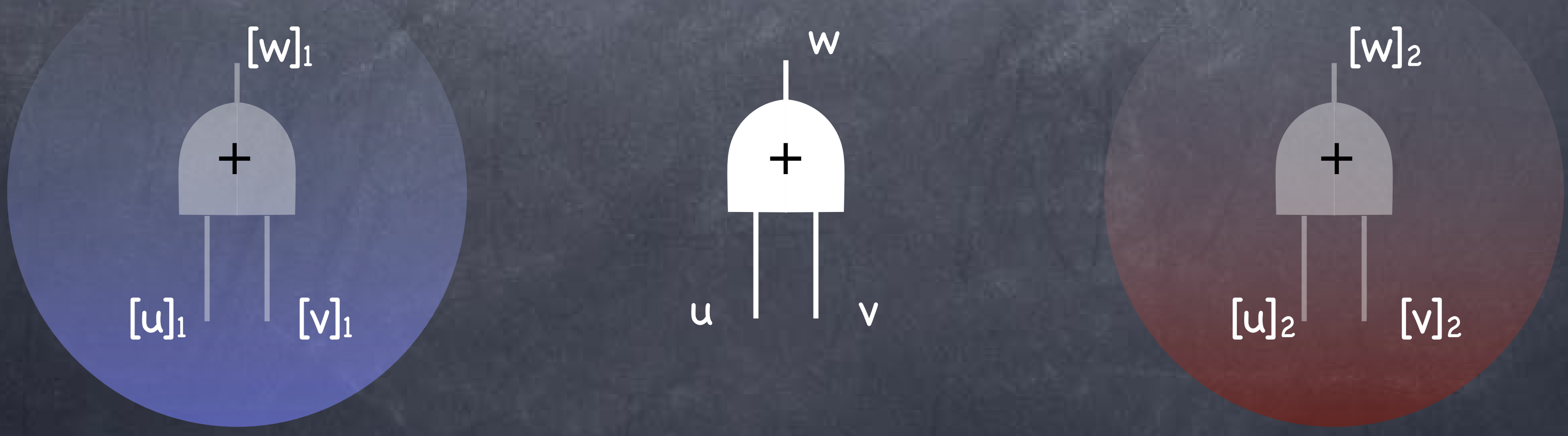
- Say Alice's input x, Bob's input y, and only Bob should learn f(x,y)
- Alice (who knows x, but not y) prepares a table for $f(x,\cdot)$ with $D = 2^{|y|}$ entries (one for each y)
- Bob uses y to decide which entry in the table to pick up using 1-out-of-D OT (without learning the other entries)
- Bob 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|
  - Plan: somehow exploit efficient computation (e.g., circuit) of f

Passive GMW

- Adapted from the famous Goldreich-Micali-Wigderson (1987) protocol (due to Goldreich-Vainish, Haber-Micali,...)
- Passive secure MPC based on OT, without any other computational assumptions
  - Will assume that a trusted party is available to carry out OT between any pair of parties (replaced by a cryptographic protocol, later)
  - Tolerates any number of corrupt parties
- Idea: Computing on **additively secret-shared values**
  - For a variable (wire value) s, will write $[s]_i$ to denote its share held by the i^{th} party

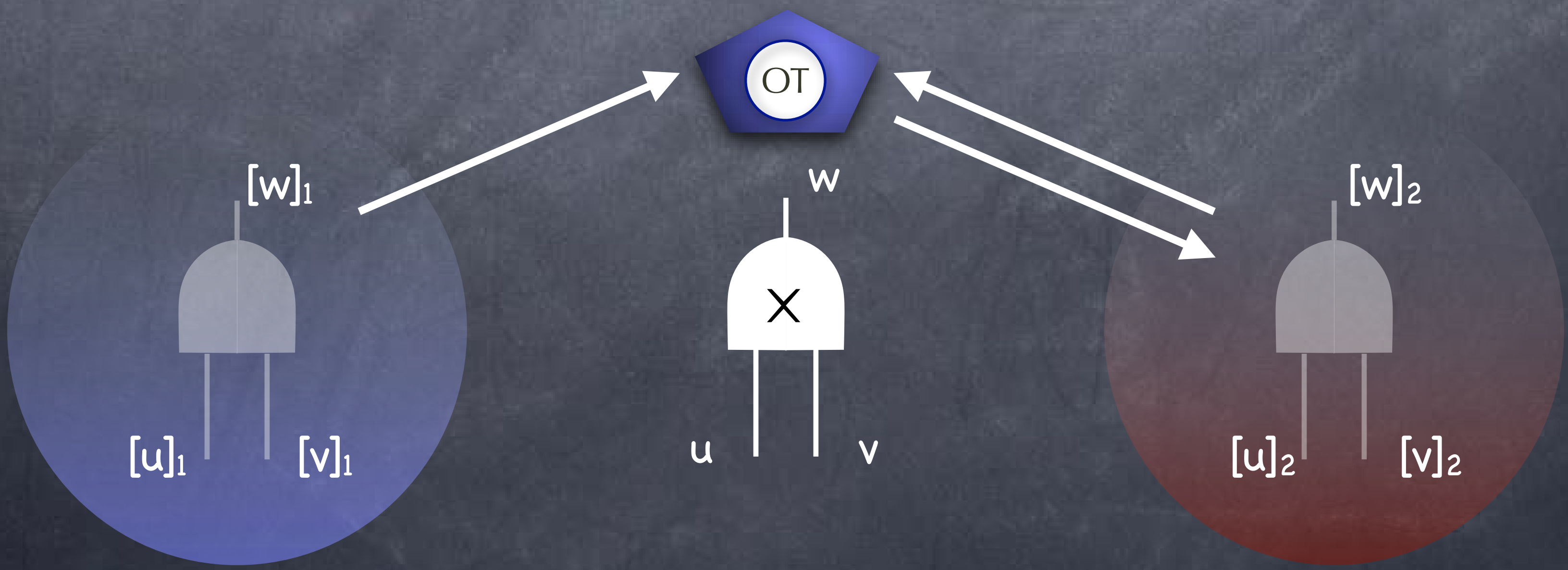
Computing on Shares: 2 Parties

- Let gates be + & × (XOR & AND for Boolean circuits)
- Plan: Similar to BGW: shares of each wire value will be computed, with Alice holding one share and Bob the other. At the end, Alice sends her share of output wire to Bob.
- $w = u + v$: Each one locally computes $[w]_i = [u]_i + [v]_i$



Computing on Shares: 2 Parties

- What about $w = u \times v$?
  - $[w]_1 + [w]_2 = ([u]_1 + [u]_2) \times ([v]_1 + [v]_2)$
  - Alice picks $[w]_1$ and lets Bob compute $[w]_2$ using the naive (proof-of-concept) protocol
    - Note: Bob's input is $([u]_2,[v]_2)$. Over the binary field, this requires a single 1-out-of-4 OT.



Passive GMW

- Secure?
- View of Alice:
  - Input x and random values it picks through out the protocol ✓
- View of Bob:
  - Input y and random values it picks through out the protocol
  - A random value (picked via OT) for each wire out of a $\times$ gate
  - f(x,y) - own share, for the output wire
- This distribution is the same for x, x′ if f(x,y)=f(x′,y) ✓
- Exercise: What goes wrong in the above claim if Alice reuses $[w]_1$ for two $\times$ gates?

Computing on Shares: m Parties

- m-way sharing: $s = [s]_1 +...+ [s]_m$
- Addition, local as before
- Multiplication: For $w = u \times v$
  $[w]_1 +..+ [w]_m = ([u]_1 +..+ [u]_m) \times ([v]_1 +..+ [v]_m)$
  - Party i computes $[u]_i[v]_i$
  - For every pair (i,j), i≠j, Party i picks random a_{ij} and lets Party j securely compute b_{ij} s.t. $a_{ij} + b_{ij} = [u]_i[v]_j$ using the naive protocol (a single 1-out-of-2 OT)
  - Party i sets $[w]_i = [u]_i[v]_i + \Sigma_j (a_{ij} + b_{ji})$