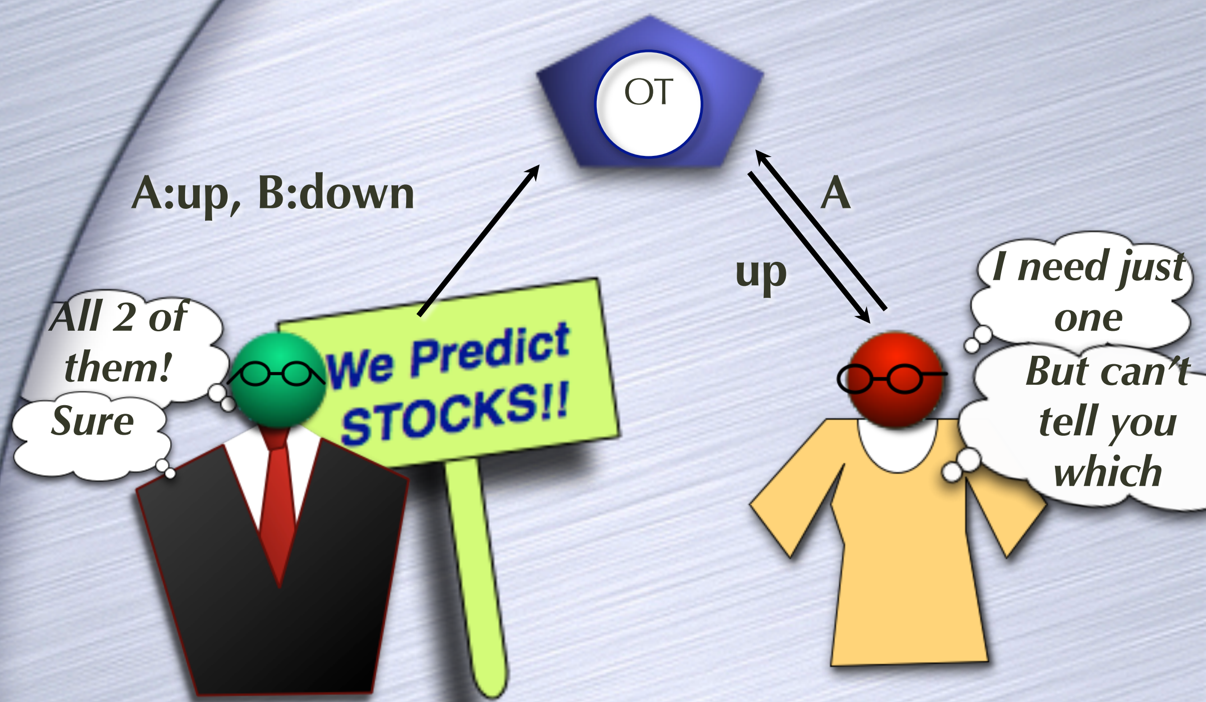
SFE: Yao's Garbled Circuit

RECALL

Oblivious Transfer

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

IDEAL World

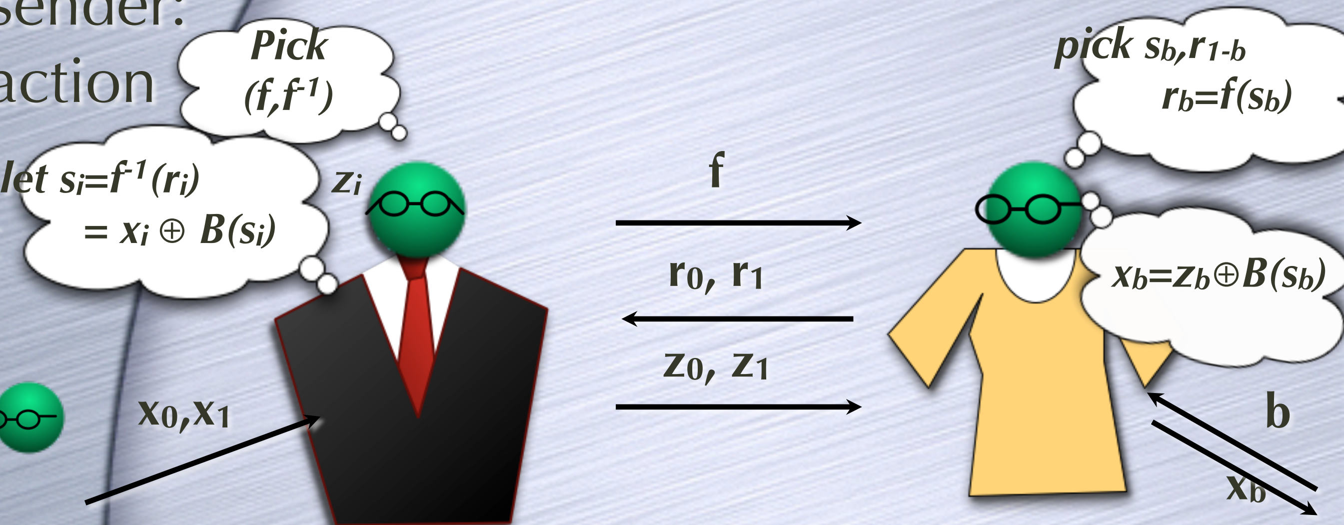


RECALL

An OT Protocol against Passive Adversary

REAL World

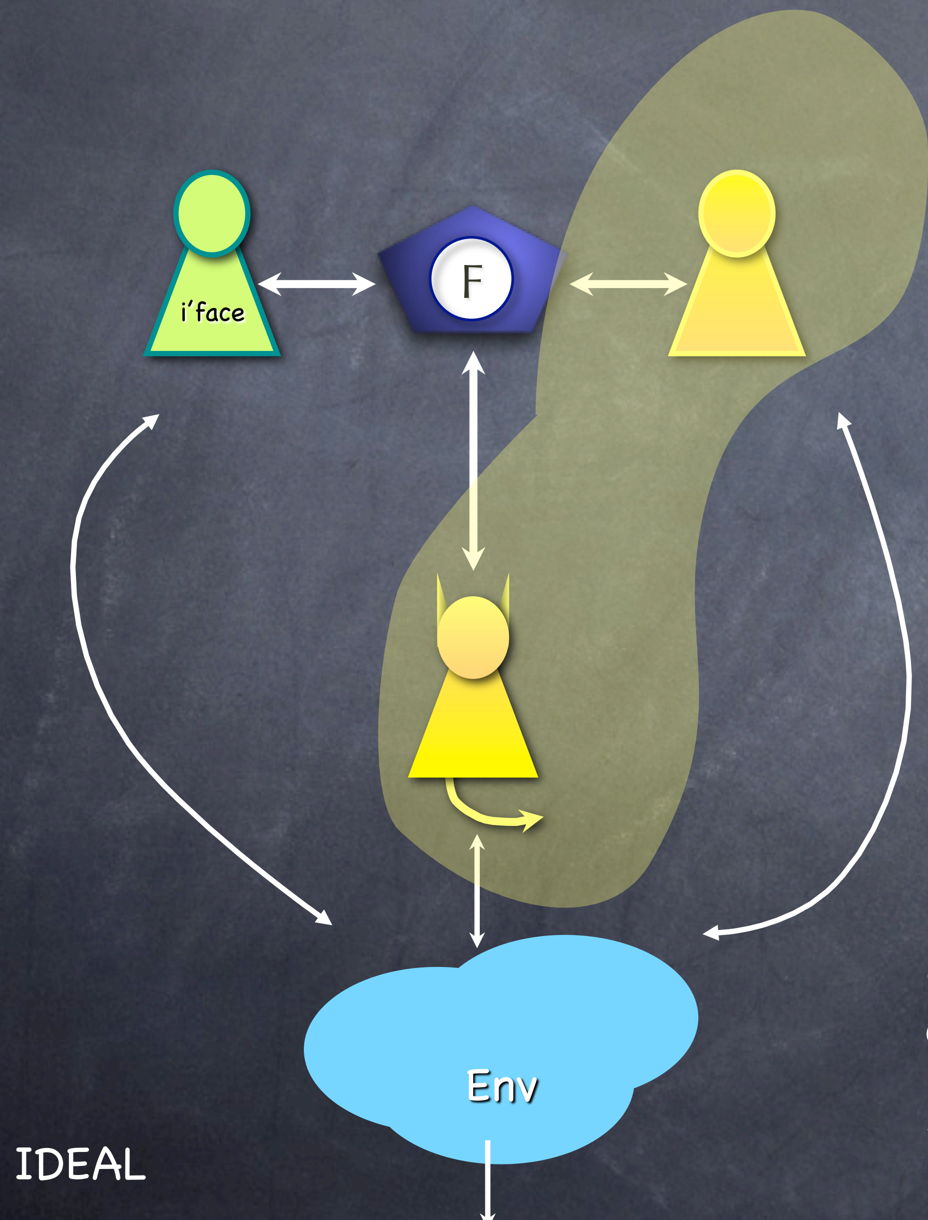
- Using a TOWP
  - Depends on receiver to pick x_0, x_1 as prescribed
- Simulation for corrupt receiver: Must simulate z_0, z_1 knowing only x_b (use random z_{1-b})
- Simulation for corrupt sender: Extract x_0, x_1 from interaction (pick s_{1-b} also); works even if actively corrupt

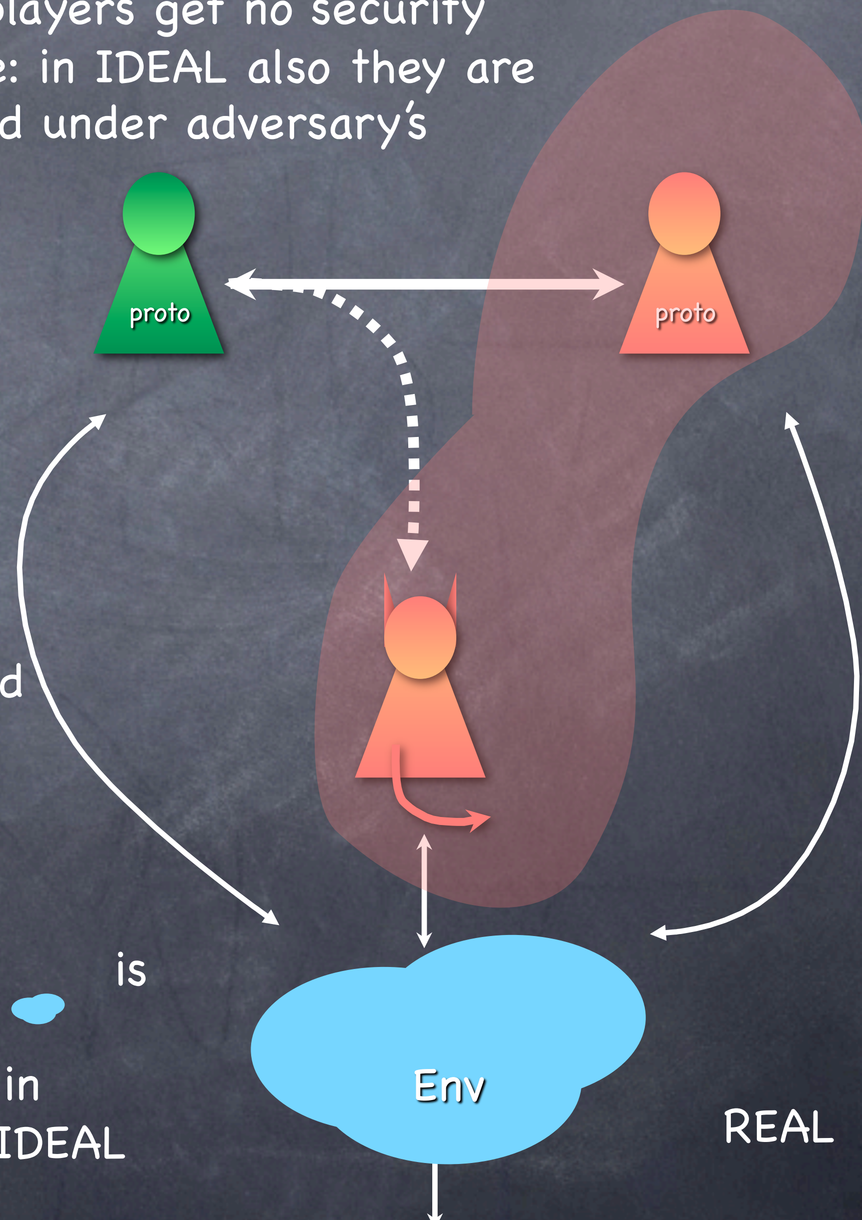


RECALL

SIM-Secure MPC

Corrupt players get no security guarantee: in IDEAL also they are considered under adversary's control





Secure (and correct) if:

∀

∃ s.t.

∀

output of is distributed identically in REAL and IDEAL

Adversary

- REAL-adversary can corrupt any set of players
  - In security requirement IDEAL-world adversary should corrupt the same set of players
    - Equivalently, environment "knows" set of corrupt players
- More sophisticated notion: adaptive adversary which corrupts players dynamically during/after the execution
  - We'll stick to static adversaries
- Passive adversary: gets only read access to the internal state of the corrupted players (and can use that information during the execution)

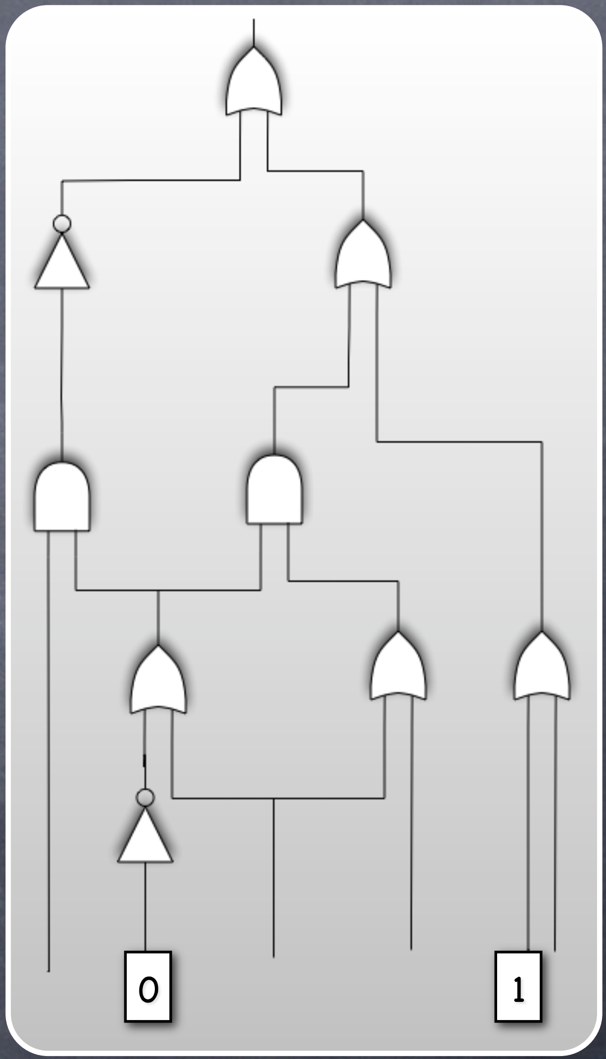
RECALL

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
  - $f(x_0,x_1;b)$ = 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 [How?]
- So, for passive security, enough to consider one-sided SFE

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

- 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 AND (NOT y_1)) OR (NOT(x_1 OR y_1) AND (x_0 AND (NOT y_0))
- Can convert any "program" into a (reasonably "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, with size of circuit exponentially large
  - In general, finding a small/smallest circuit from truth-table is notoriously hard
  - But problems already described as succinct programs/circuits

	00	01	10	11
00	0	0	0	0
01	1	0	0	0
10	1	1	0	0
11	1	1	1	0

2-Party SFE using General Circuits

	0	1
0	0	1
1	1	1

- "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 OR(x,y)
  - Any ideas?

Scrambled OR gate

- Alice creates 4 keys:

  $K_{x=0}$, $K_{x=1}$, $K_{y=0}$, $K_{y=1}$

- Alice creates 4 "boxes" for each of the table entries

  $B_{00} = 0$, $B_{01}=1$, $B_{10}=1$, $B_{11}=1$

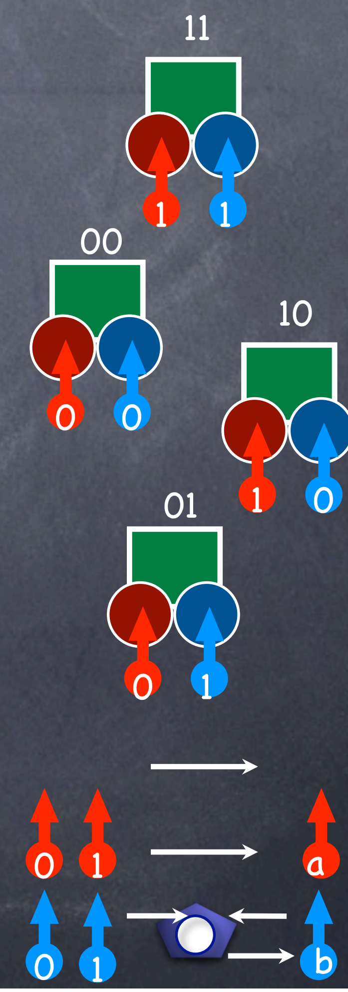
- Each box is encrypted with the two keys corresponding to the inputs

  $E(K_{x=0}||K_{y=0}, B_{00})$, $E(K_{x=0}||K_{y=1}, B_{01})$

  $E(K_{x=1}||K_{y=0}, B_{10})$, $E(K_{x=1}||K_{y=1}, B_{11})$

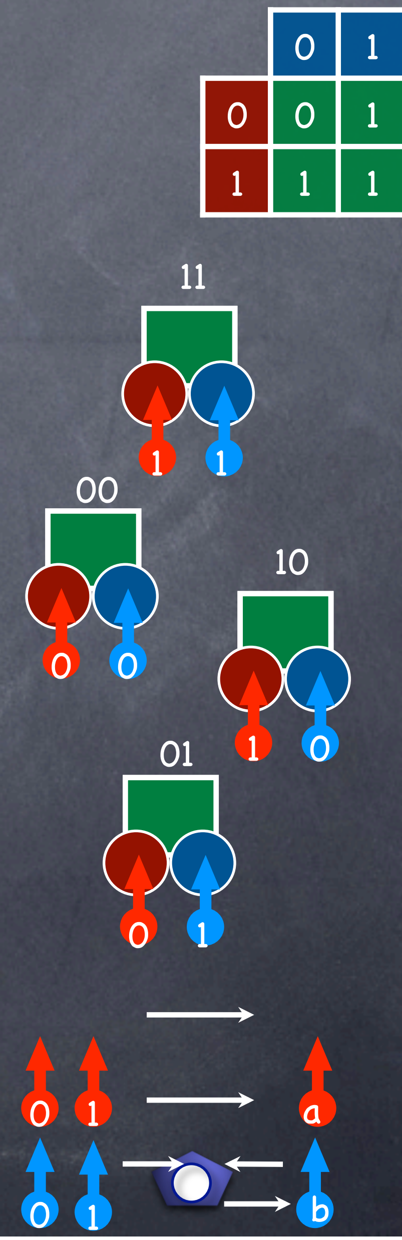
- Boxes permuted, sent to Bob
- Bob gets $K_{x=a}$ from Alice, uses OT to get $K_{y=b}$
- Bob decrypts the only box he can (B_{ab})

	0	1
0	0	1
1	1	1

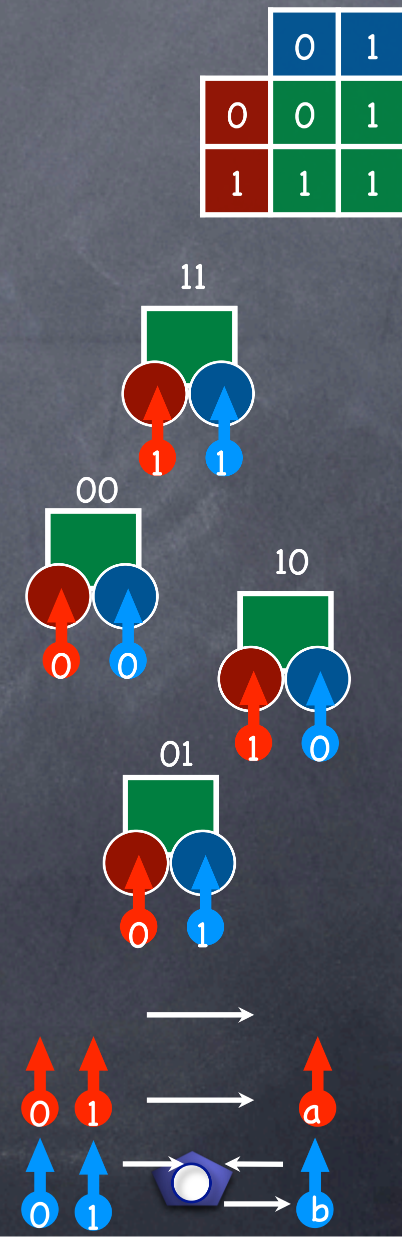


OR gate security

- Passive (honest-but-curious) adversary
  - Adversary learns state of corrupted parties, but does not modify protocol
- Alice learns nothing about Bob's input
  - Oblivious transfer
- Bob only learns contents of output box
  - Formally, can model other box encryptions as garbage
- What kind of encryption do we need?
  - IND-CPA, IND-CCA?



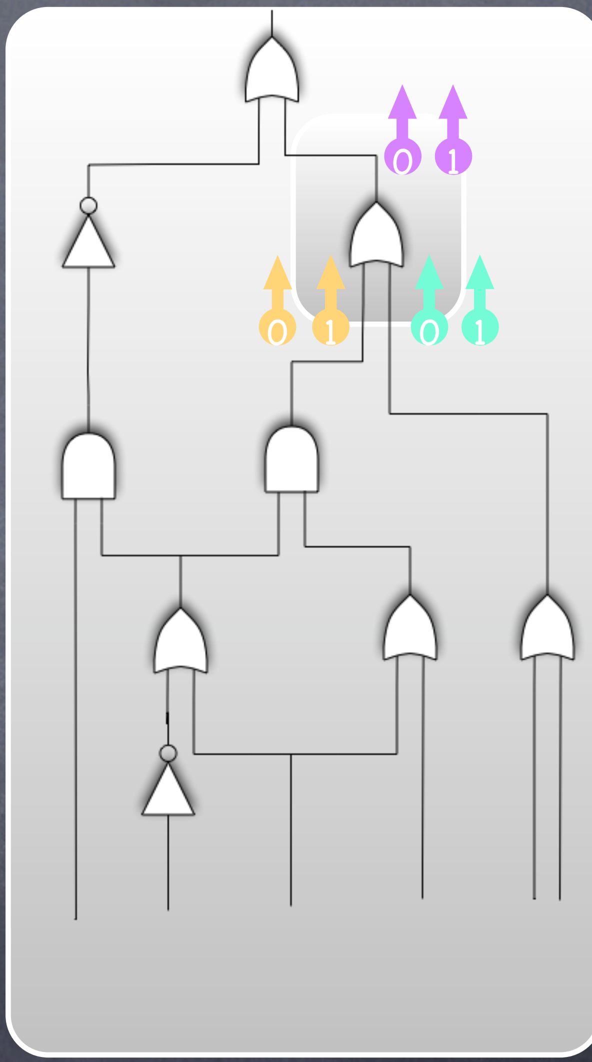
Active Adversaries?



- What can an active adversary accomplish?
- Alice: encrypt a different circuit
- Bob: learn Alice's input
  - Note: this is true in ideal world, too!

Larger Circuits

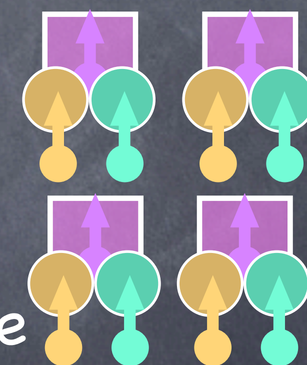
- Idea: For each gate in the circuit Alice will prepare locked boxes, but will use it to keep keys for the next gate
- For each wire w in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$



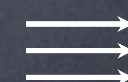
Larger Circuits

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

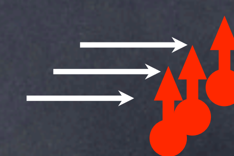
- For each wire w in the circuit (i.e., input wires, or output of a gate) pick 2 keys $K_{w=0}$ and $K_{w=1}$



  - 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 $K_{v=b}$

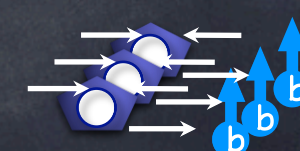


  - Give to Bob: Boxes for each gate, one key for each of Alice's input wires



    - Obliviously: one key for each of Bob's input wires

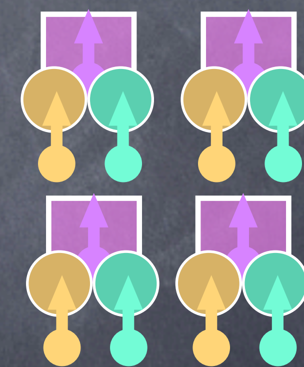
  - Boxes for output gates have values instead of keys



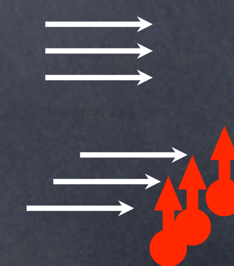
Larger Circuits

- 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

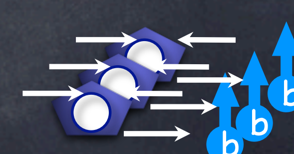
  - Gets output from a box in the output gate



- 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



Security

- How do we make sure Alice gives the correct circuit?
- Cut-and-choose:
  - Alice prepares m circuits
  - Bob picks one to execute
  - Alice reveals secrets for all others
- Multiple circuits
  - Bob evaluates k out of m circuits, verifies the others
  - Note: must ensure Bob's inputs for all circuits are the same

FairPlay

- Implementation of SFE
- Function specified as programs
- Compiler converts it to circuits

```
program Millionaires {
   type int = Int<4>; // 4-bit
   integer
   type AliceInput = int;
   type BobInput = int;
   type AliceOutput = Boolean; type
   BobOutput = Boolean;
   type Output = struct { AliceOutput
   alice, BobOutput bob};
   type Input = struct { AliceInput
   alice, BobInput bob};

   function Output out(Input inp)
   { out.alice = inp.alice > inp.bob;
   out.bob = inp.bob > inp.alice; }

}
```

FairPlay Performance

Function	Gates	OTs
AND	32	8
Billionaires	254	32
KDS	1229	6
Median	4383	160

Function	LAN	WAN
AND	0.41	2.57
Billionaires	1.25	4.01
KDS	0.49	3.38
Median	7.09	16.63

Universal Circuits

- What if Bob wants to evaluate secret function over Alice's input?
  - No fly list
  - Credit report check
- Use a universal circuit
  - $UC(C,x,y) = C(x,y)$
- Have either Alice or Bob provide circuit as input
- Can be made "reasonably" efficient

Today

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