CIS 5560

Cryptography
Lecture 26

Course website:
pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW11** due **Wednesday May 1** at 11:59PM on Gradescope
- Final Exam May 10 9AM-11AM
- Will create and provide a cheat sheet
- Will share sample problems

Recap of Last Lecture

- Secure Multi-party Computation
- Secret Sharing
- Oblivious Transfer

Secure Two-Party Computation

Input: x **Input:** y

Alice Bob

Output: $F_A(x, y)$ **Output**: $F_B(x, y)$

<u>Security:</u>

<u>Semi-honest Security:</u>

- Alice should not learn anything more than x and $F_A(x, y)$.
- Bob should not learn anything more than y and $F_B(x, y)$.

Shamir's t-out-of-n Secret Sharing

Key Idea: Polynomials are Amazing!

1. The dealer picks a uniformly random degree-(t-1) polynomial **(mod p)** whose constant term is the secret b.

$$f(x) = a_{t-1}x^{t-1} + \ldots + a_1 x + b$$

where a_i are uniformly random mod p

2. Compute the shares:

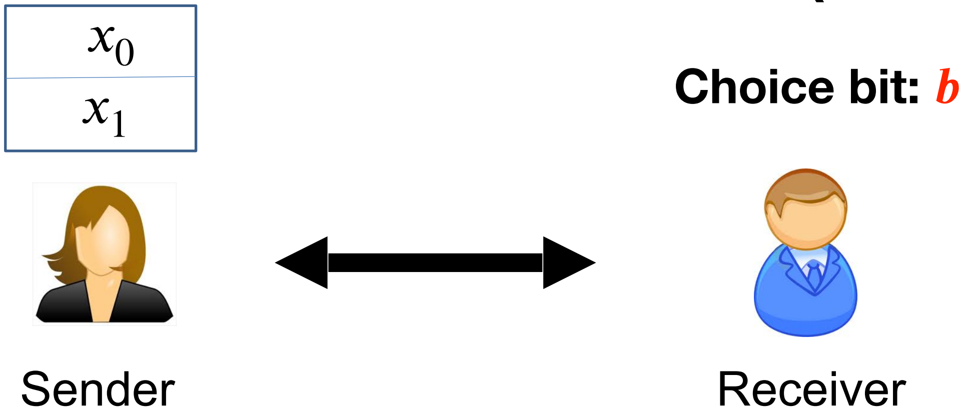
$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

Correctness: can recover secret from any t shares.

Security: the distribution of any $t-1$ shares is independent of the secret.

Note: need p to be larger than the number of parties n.

Oblivious Transfer (OT)



- Sender holds two bits/strings x_0 and x_1.
- Receiver holds a choice bit b.
- Receiver should learn x_b, sender should learn nothing.

(We will consider **honest-but-curious** adversaries; formal definition in a little bit…)

OT Protocol 1: Trapdoor Permutations

For concreteness, let's use the RSA trapdoor permutation.

Input bits: (x_0, x_1)

Choice bit: b

Pick $N = PQ$ and RSA exponent e.

N, e →

Choose random r_b and set $s_b = r_b^e \bmod N$

Choose random s_{1-b}

← s_0, s_1

Compute r_0, r_1 and XOR x_0, x_1 using hardcore bits

$x_0 \bigoplus HCB(r_0)$ → $x_1 \bigoplus HCB(r_1)$

Bob can recover x_b but not x_{1-b}

OT $\Longrightarrow$ Secret-Shared-AND

$\alpha \in \{0,1\}$

Alice gets random γ, Bob gets random δ s.t. $\gamma \oplus \delta = \alpha\beta$.

$\beta \in \{0,1\}$

Output: γ

Output: δ

$x_0 = \gamma$
$x_1 = a \oplus \gamma$

Run an OT protocol

Choice bit $b = \beta$

Alice outputs γ.

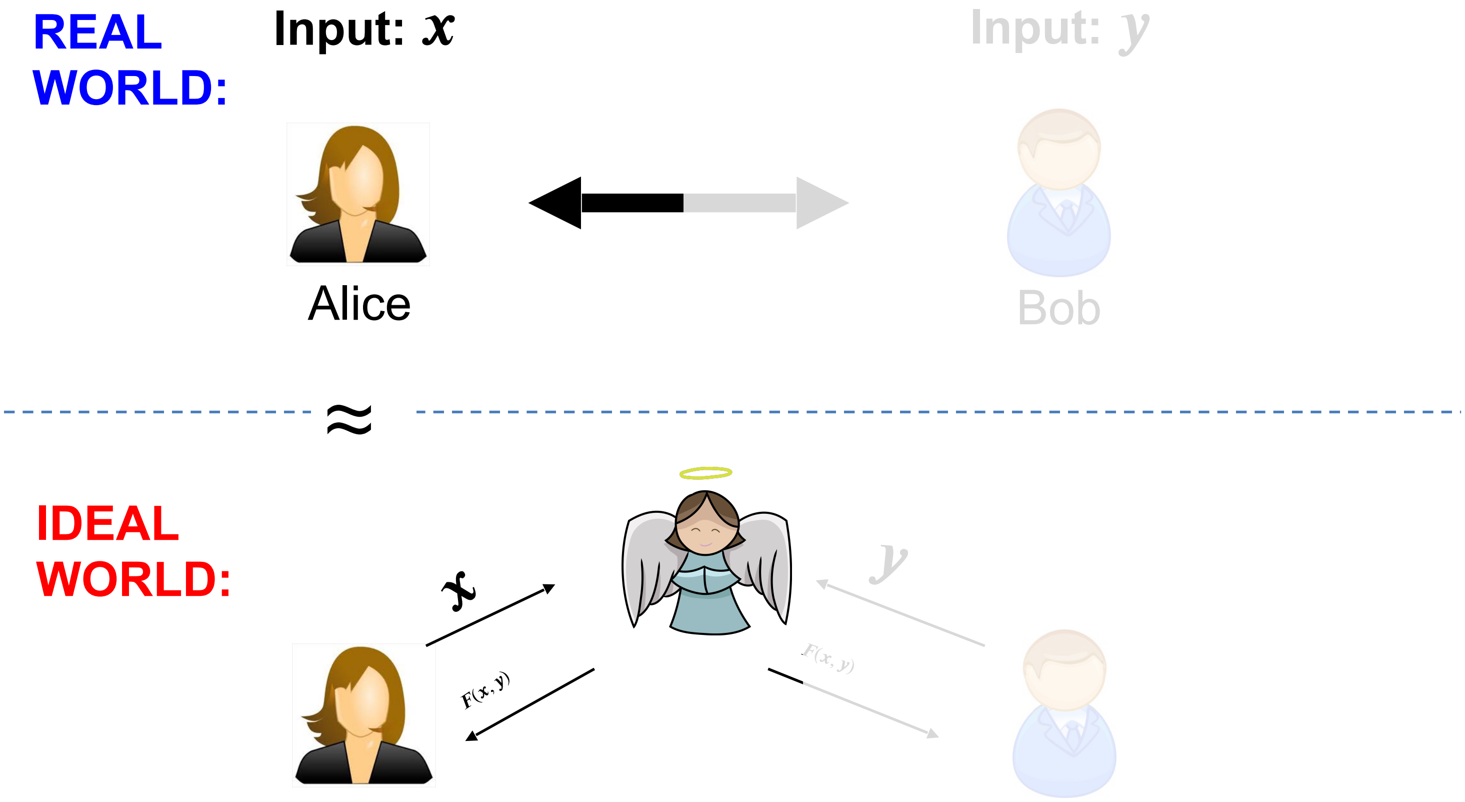
Bob gets $x_1 b + x_0(1 \oplus b) = (x_1 \oplus x_0)b + x_0 = \alpha\beta \oplus \gamma := \delta$

“OT is Complete”

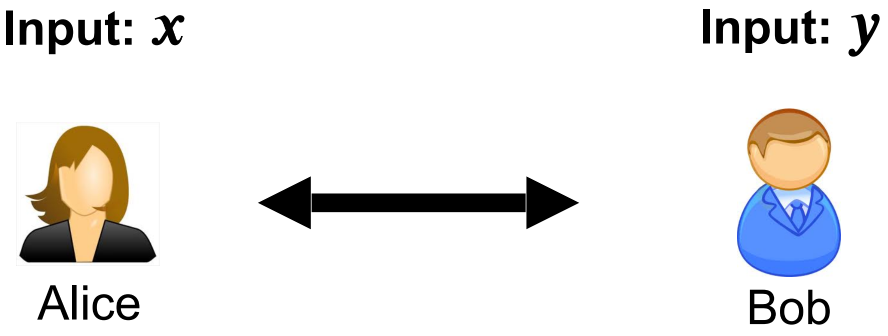
Theorem: OT can solve not just ANDs and money, but ***any*** two-party (and multi-party) problem efficiently.

Defining Security:
The Ideal/Real Paradigm

Secure Two-Party Computation



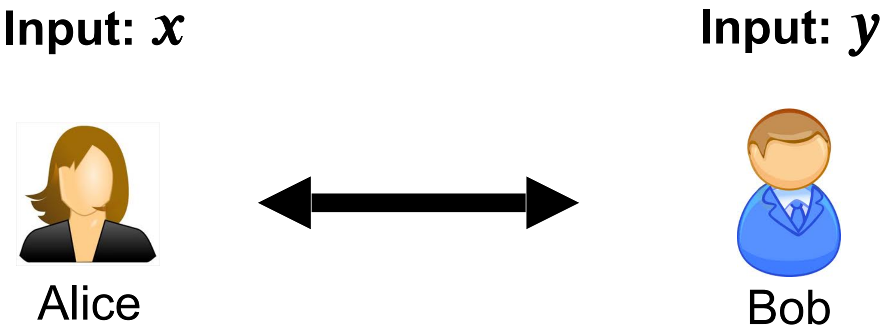
Secure Two-Party Computation



There exists a PPT simulator SIM_A such that for any x and y:

$$SIM_A(x, F(x, y)) \cong View_A(x, y)$$

Secure Two-Party Computation



There exists a PPT simulator SIM_B such that for any x and y:

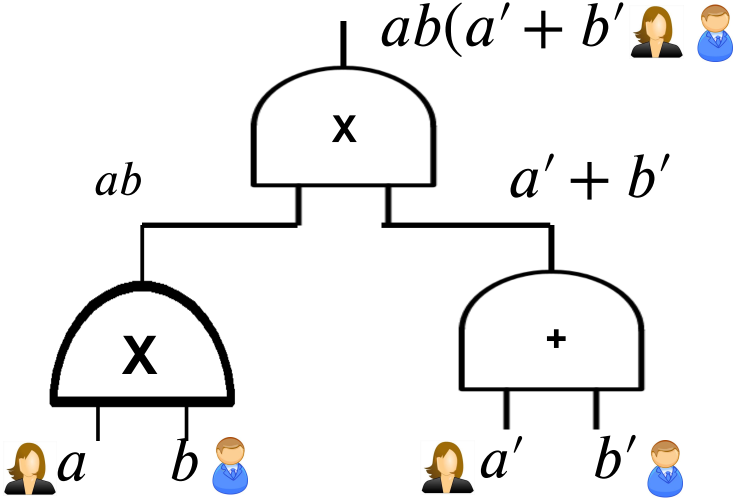
$$SIM_B(y, F(x, y)) \cong View_B(x, y)$$

Secure 2PC from OT

***Theorem* [Goldreich-Micali-Wigderson'87]**:
OT can solve ***any*** two-party computation problem.

Computing Arbitrary Functions

For us, programs = functions = Boolean circuits with XOR ($+ \ mod\ 2$) and AND ($\times\ mod\ 2$) gates.



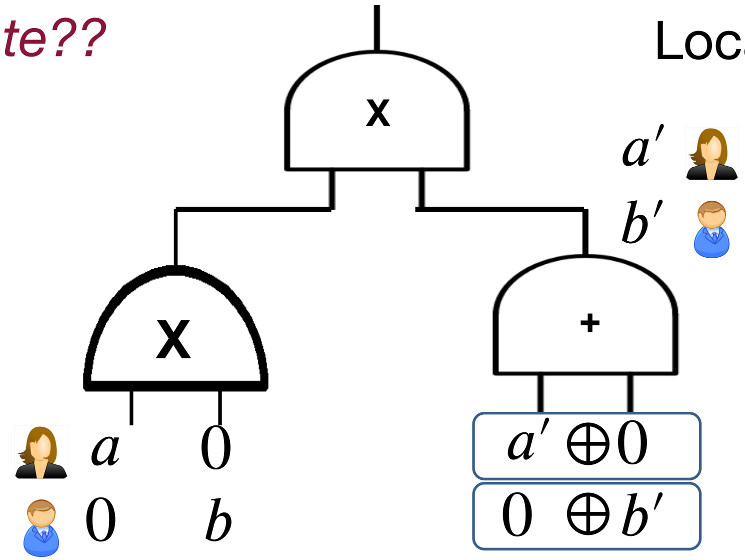
Want: If you can compute XOR and AND *in the appropriate sense*, you can compute everything.

Computing Arbitrary Functions

Secret-sharing Invariant: For each wire of the circuit, Alice and Bob each have a bit whose XOR is the value at the wire.

XOR gate:
Locally XOR the shares

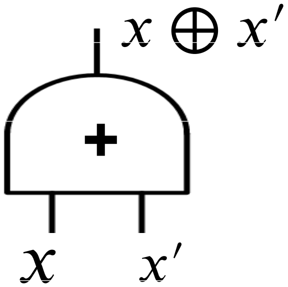
AND gate??



Base Case: Input wires

Computing the XOR gate

Alice has α and Bob has β s.t. $\alpha \oplus \beta = x$



Alice has α' and Bob has β' s.t. $\alpha' \oplus \beta' = x'$

Alice computes $\boldsymbol{\alpha \oplus \alpha'}$ and Bob computes $\boldsymbol{\beta \oplus \beta'}$.

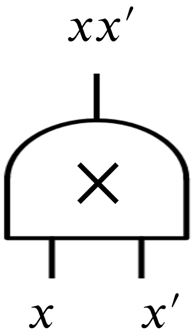
So, we have: $(\alpha \oplus \alpha') \oplus (\beta \oplus \beta')$

$$= (\alpha \oplus \beta) \oplus (\alpha' \oplus \beta') = \mathrm{x} \oplus \mathrm{x}'$$

Computing the AND gate

Alice has α and Bob has β s.t. $\alpha \oplus \beta = x$

Alice has α' and Bob has β' s.t. $\alpha' \oplus \beta' = x'$



Desired output (to maintain invariant):

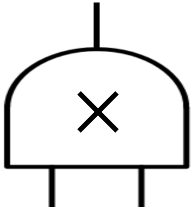
Alice wants $\boldsymbol{\alpha''}$ and Bob wants $\boldsymbol{\beta''}$ s.t. $\boldsymbol{\alpha''} \oplus \boldsymbol{\beta''} = xx'$

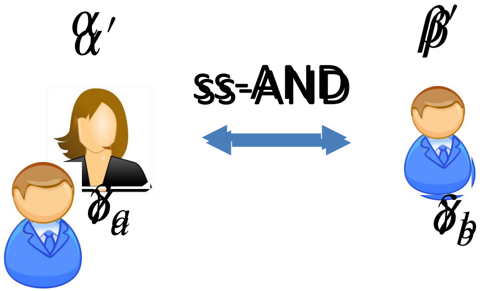
Computing the AND gate

$$xx' = (\alpha \oplus \beta)(\alpha' \oplus \beta')$$

$$= \alpha\alpha' \oplus \gamma_a \oplus \delta_a \oplus \beta\beta'$$

$$\oplus \gamma_b \qquad \oplus \delta_b$$



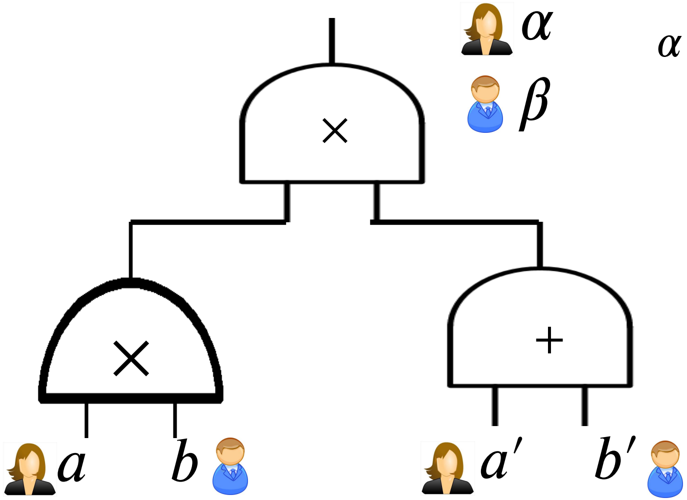


$$\alpha'' = \alpha\alpha' \oplus \gamma_a \oplus \delta_a \qquad \beta'' = \beta\beta' \oplus \gamma_b \oplus \delta_b$$

Computing Arbitrary Functions

Secret-sharing Invariant: For each wire of the circuit, Alice and Bob each have a bit whose XOR is the value at the wire.

Finally, Alice and Bob exchange the shares at the output wire, and XOR the shares together to obtain the output.

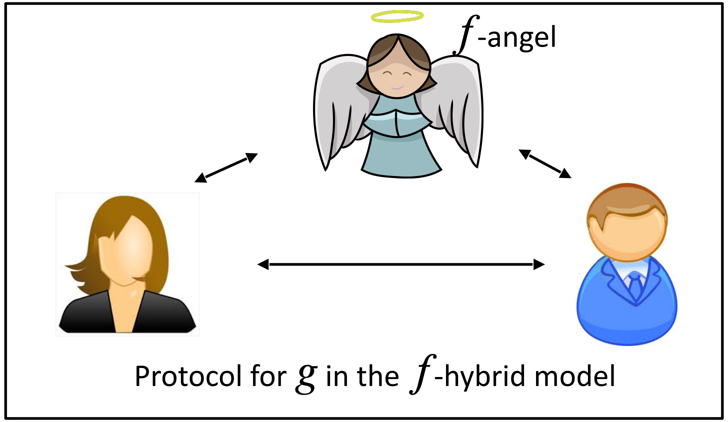


$\alpha \oplus \beta = ab(a' \oplus b')$

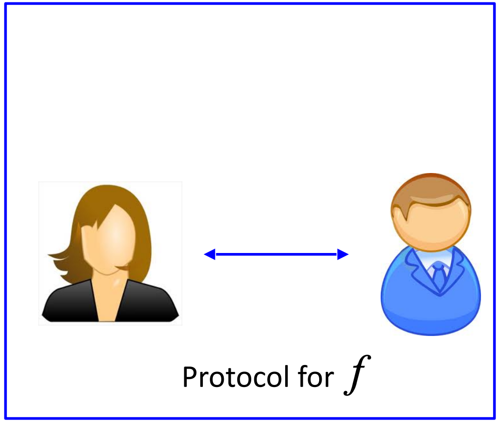
Security by Composition

Theorem:

If protocol Π securely realizes a function g in the "f-hybrid model" and protocol Π' securely realizes f, then $\Pi \circ \Pi'$ securely realizes g.

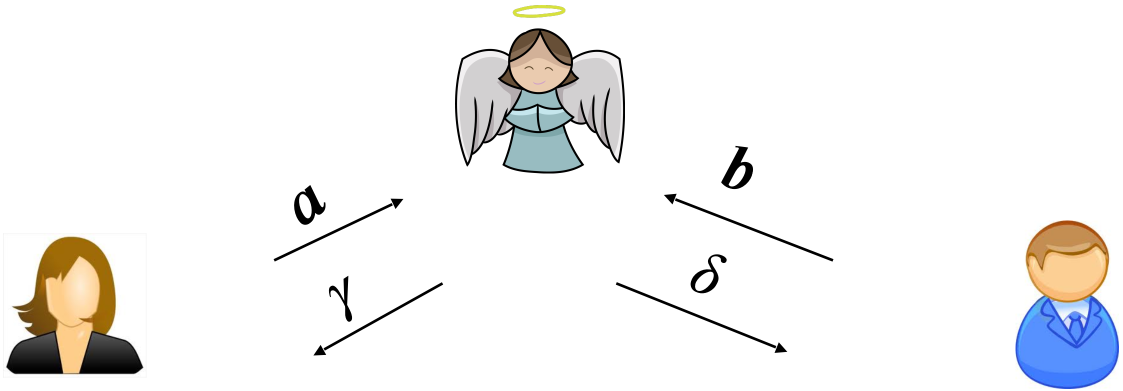


+



Security: Intuition (ss-AND hybrid model)

Imagine that the parties have access to an ss-AND angel.



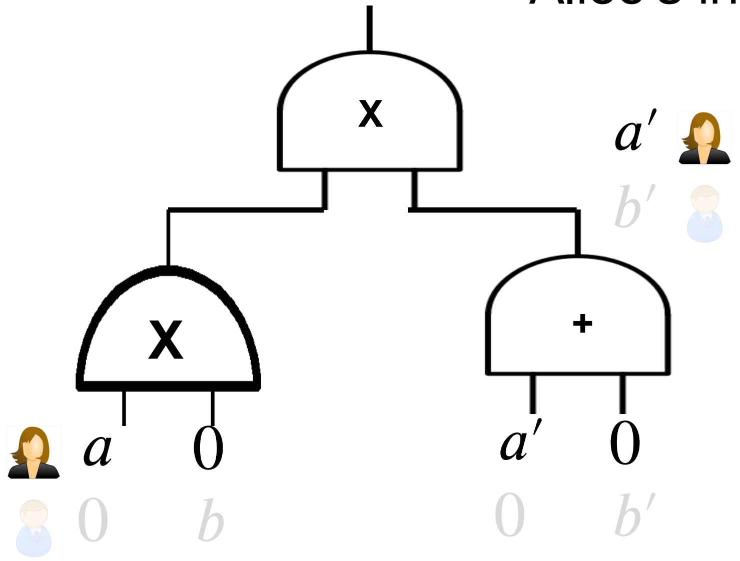
$$\gamma \bigoplus \delta = \mathrm{ab}$$

Security: Intuition (ss-AND hybrid model)

Imagine that the parties have access to an ss-AND angel.

Simulator for Alice's view:

XOR gate: simulate given Alice's input shares



Input wires: can be simulated given Alice's input

Security: Intuition (ss-AND hybrid model)

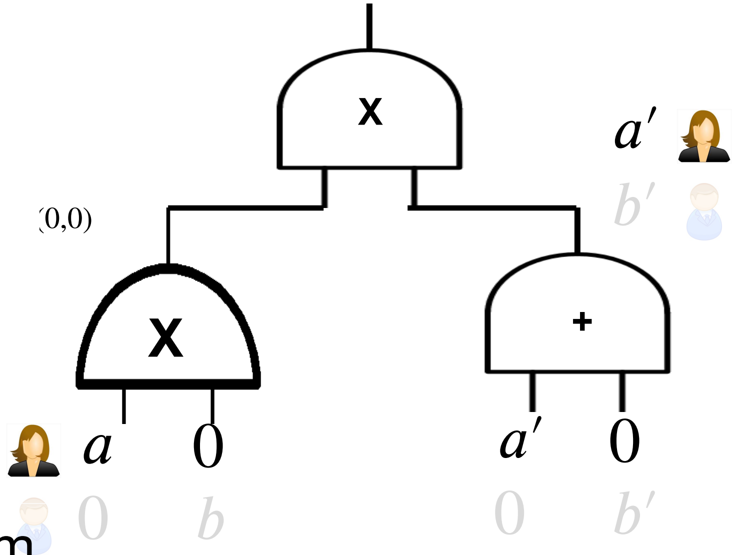
Simulator for Alice's view:

AND gate: simulate given Alice's input shares
& outputs from the ss-AND angel.

Alice's share ✓

$= a \bullet 0 + \gamma_{alice}$ ✓ (0,0)

δ_{alice} ✓



γ_{alice} and δ_{alice} are random,
independent of b

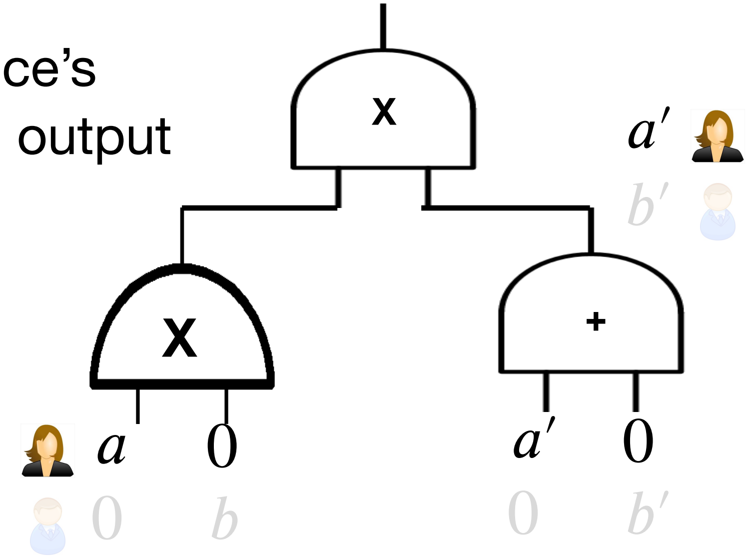
Security: Intuition (ss-AND hybrid model)

Simulator for Alice's view:

Output wire: need to know both Alice and Bob's output shares.

Bob's output share = Alice's output share $\oplus$ function output

Simulator knows the function output, and can compute Bob's output share given Alice's output share.



In summary: Secure 2PC from OT

Theorem* [Goldreich-Micali-Wigderson'87]**: Assuming OT exists, there is a protocol that solves ***any two-party computation problem against semi-honest adversaries.

In fact, GMW does more:

Theorem **[Goldreich-Micali-Wigderson'87]**: Assuming OT exists, there is a protocol that solves any ***multi-party*** computation problem against semi-honest adversaries.

MPC Outline

Secret-sharing Invariant: For each wire of the circuit, **the n parties have a bit each**, whose XOR is the value at the wire.

Base case: input wires.

XOR gate: given input shares $(\alpha_1, \ldots, \alpha_n)$ s.t. $\oplus_{i=1}^{n} \alpha_i = a$ and $(\beta_1, \ldots, \beta_n)$ s.t. $\oplus_{i=1}^{n} \beta_i = b$, compute the shares of the output of the XOR gate:

$$(\alpha_1 + \beta_1, \ldots, \alpha_n + \beta_n)$$

AND gate: given input shares as above, compute the shares of the output of the XOR gate:

$$(o_1, \ldots, o_n) \text{ s.t } \oplus_{i=1}^{n} o_i = ab$$

Exercise!

Course Summary

- We started with a simple goal: secure communication
- Led to discussions about
  - pseudorandomness
  - indistinguishability
  - hardness of computation
- New primitives and security notions:
  - SKE (IND-CPA)
  - MACs (EUF-CMA)
  - AE (Ciphertext Integrity)
  - PKE
  - Signatures
  - Hash functions (CRH)

Course Summary

- With these tools, we started looking at new goals
  - Proving things about hidden data: ZK
  - Computing over hidden data: MPC
- New models:
  - *Interactive* Proofs
- New security paradigms:
  - Simulation

Can do much more with crypto!

- *Efficient proofs* about data (zk optional):
    - Non-interactive ZK
    - Private cryptocurrencies
    - Succinct proofs of computation
- *Efficient* computation on hidden data:
    - Homomorphic encryption
    - Threshold cryptography
- Secure retrieval of outsourced data:
    - “Oblivious” RAM
        - Deployed at Signal for Private Key Discovery
    - Private Information Retrieval

If any of these topics interest you, come speak to me after!

Thank you for a fantastic semester!