MIT 6.875

Foundations of Cryptography
Lecture 19

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.

Two-Party Impossibility

Theorem (folklore):
There is no perfectly / statistically secure two-party protocol for computing the AND function.

Impossibility of 2-Party Secure MPC
(due to Rotem Oshman)

- Alice: $a \in \{0,1\}$, Bob: $b \in \{0,1\}$
- Goal: compute $a \wedge b$
- No information-theoretically secure implementation!

- Fix any protocol Π
- Let $\pi_{a,b}(\tau) =$ probability of transcript τ on input a, b
- w.l.o.g, the transcript contains $a \wedge b$

Impossibility of 2-Party Secure MPC

- Claim: $\pi_{a,b}(\tau) = A(a,\tau) \cdot B(b,\tau)$ for some A, B
- Proof:

$$\pi_{a,b}(\tau) = \prod_{r=1}^{R} \Pr[\, \tau_r \text{ is sent} \mid \tau_{r-1}, \dots, \tau_1, a, b \,]$$

$$= \left(\prod_{r:\text{Alice speaks}} \Pr[\, \tau_r \text{ is sent} \mid \tau_{r-1}, \dots, \tau_1, a, \cancel{b} \,] \right) \quad A(\tau, a)$$

$$\cdot \left(\prod_{r:\text{Bob speaks}} \Pr[\, \tau_r \text{ is sent} \mid \tau_{r-1}, \dots, \tau_1, \cancel{a}, b \,] \right) \quad B(\tau, b)$$

Impossibility of 2-Party Secure MPC

- Claim: $\pi_{a,b}(\tau) = A(a,\tau) \cdot B(b,\tau)$ for some A, B
- From (perfect) security: for every τ,

$$\pi_{1,0}(\tau) = \pi_{0,0}(\tau) = \pi_{0,1}(\tau)$$

$$\Downarrow$$

$$A(1,\tau)B(0,\tau) = A(0,\tau)B(0,\tau) = A(0,\tau)B(1,\tau)$$

$$\Downarrow$$

$$A(0,\tau) = A(1,\tau) \text{ and } B(0,\tau) = B(1,\tau)$$

- But then,

$$\pi_{1,1}(\tau) = A(1,\tau)B(1,\tau) = A(0,\tau)B(0,\tau) = \pi_{0,0}(\tau)$$

The protocol is incorrect!

Extend to statistical security?

Exercise.

Where to Go From Here?

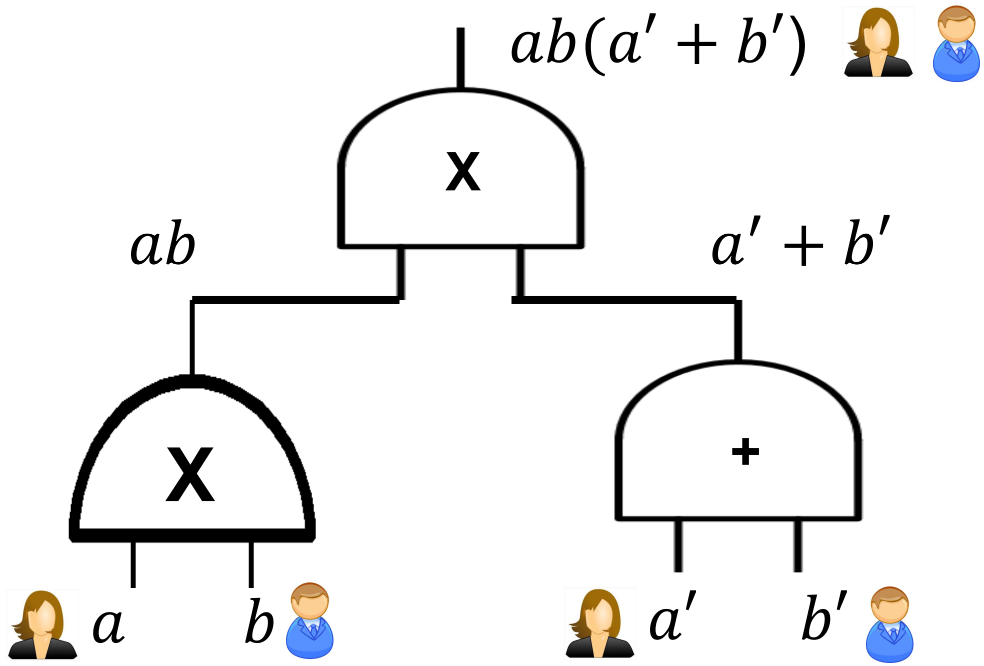
- Option 1: reduce the number of corrupt parties
- Option 2: introduce cryptographic assumptions

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.

How to Compute 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.

Recap: OT $\Rightarrow$ Secret-Shared-AND

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

Alice gets random γ, Bob gets random δ s.t. $\gamma \oplus \delta = \text{ab}$.

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

Output: γ

Output: δ

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

Run an OT protocol

$\longleftrightarrow$

Choice bit b

Alice outputs γ.

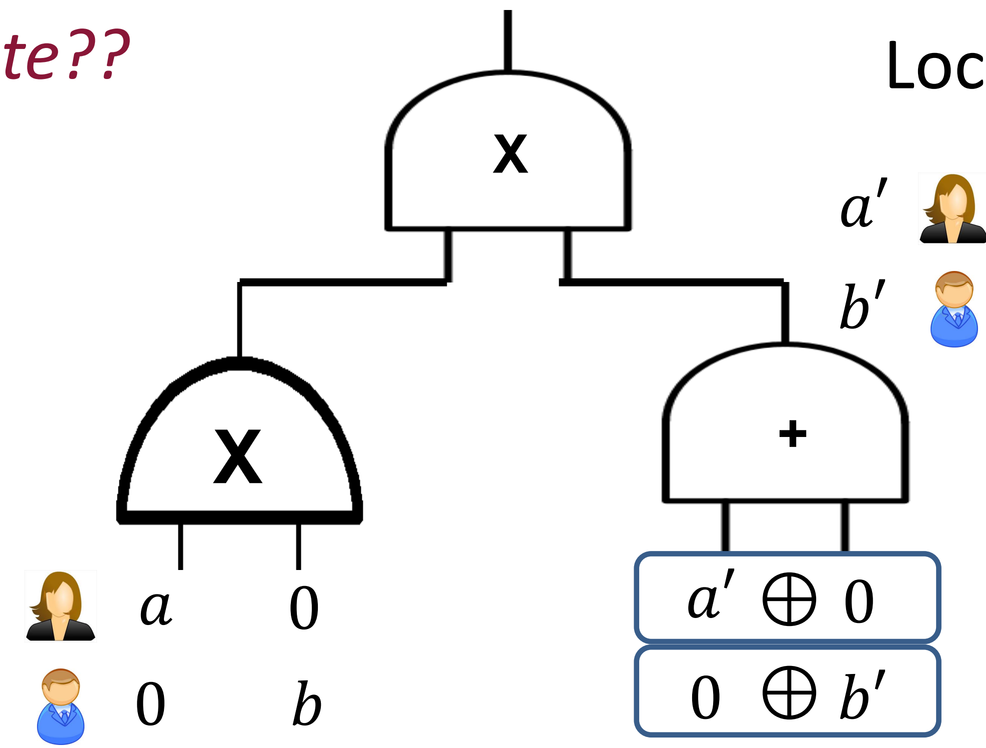
Bob gets $\boldsymbol{x_1 b} + \boldsymbol{x_0(1 \oplus b)} = (\boldsymbol{x_1} \oplus \boldsymbol{x_0})\boldsymbol{b} + \boldsymbol{x_0} = ab \oplus \gamma := \delta$

How to Compute 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.

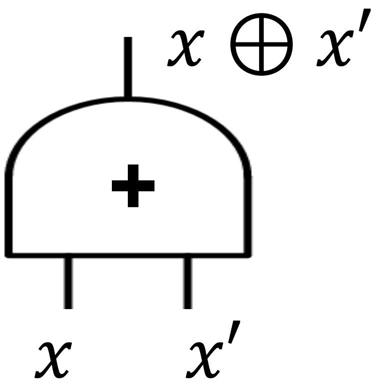
AND gate??

XOR gate:
Locally XOR the shares



Base Case: Input wires

Recap: 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 $\boldsymbol{\beta \oplus \beta'}$.

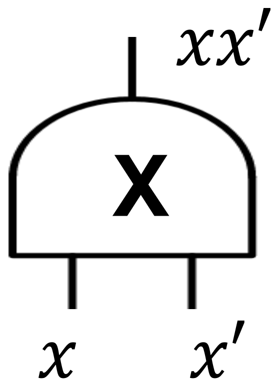
So, we have: $(\alpha \oplus \alpha') \oplus (\beta \oplus \beta')$

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

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):

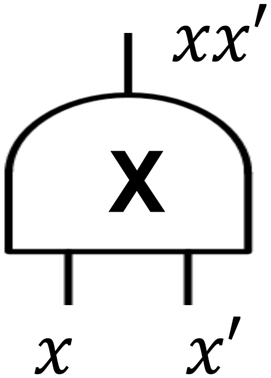
Alice wants $\boldsymbol{\alpha''}$ and Bob wants $\boldsymbol{\beta''}$ s.t. $\boldsymbol{\alpha''} \oplus \boldsymbol{\beta''} = xx'$

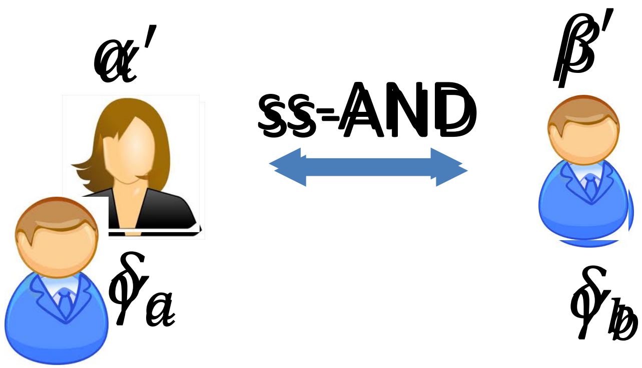
AND gate

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

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

$$\oplus \gamma_b \quad \oplus \delta_b$$



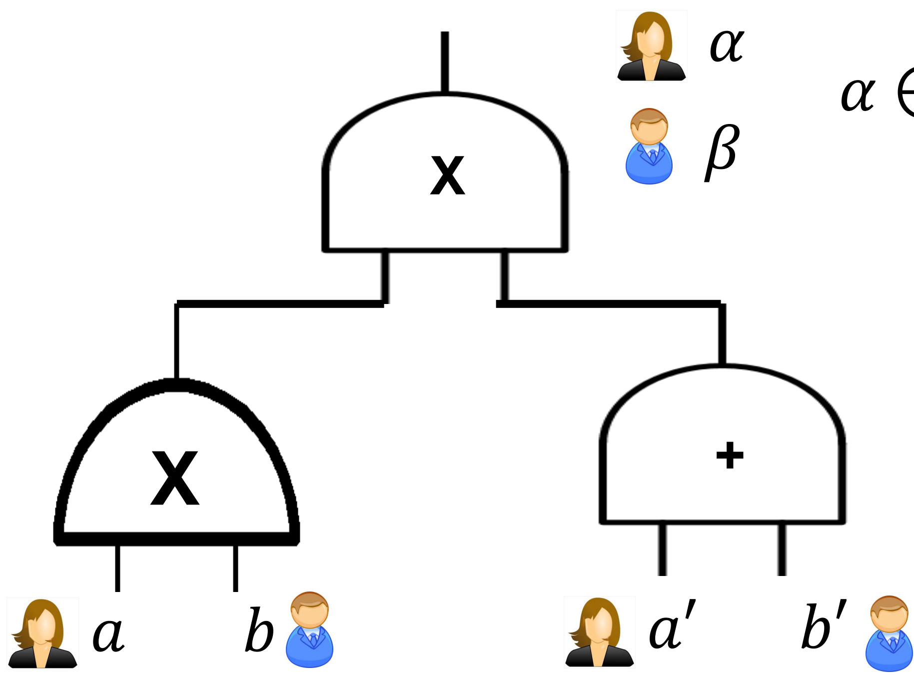


$$\alpha'' = \alpha\alpha' \oplus \gamma_a \oplus \delta_a \qquad \beta'' = \beta\beta' \oplus \gamma_b \oplus \delta_b$$

How to Compute 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.

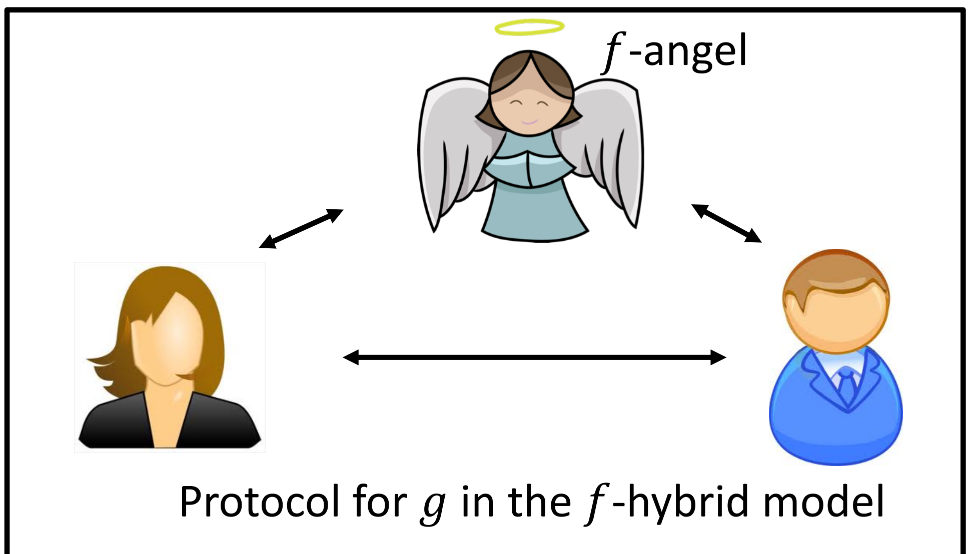
Finally, Alice and Bob exchange the shares at the output wire, and XOR the shares together to obtain the output.



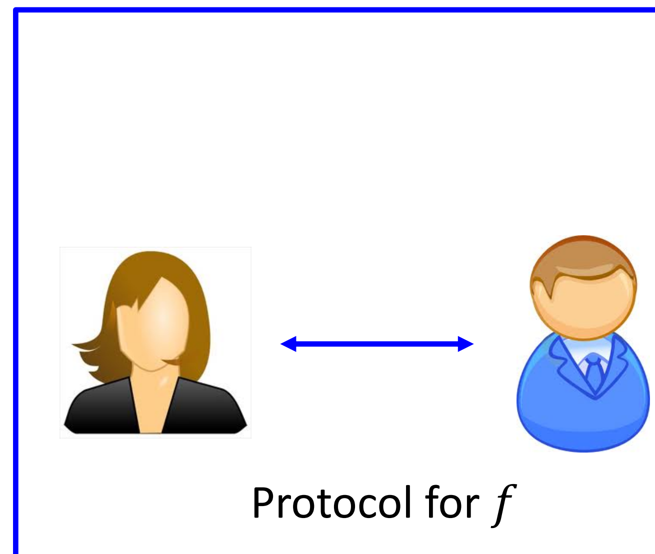
$$\alpha \oplus \beta = ab(a' \oplus b')$$

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.

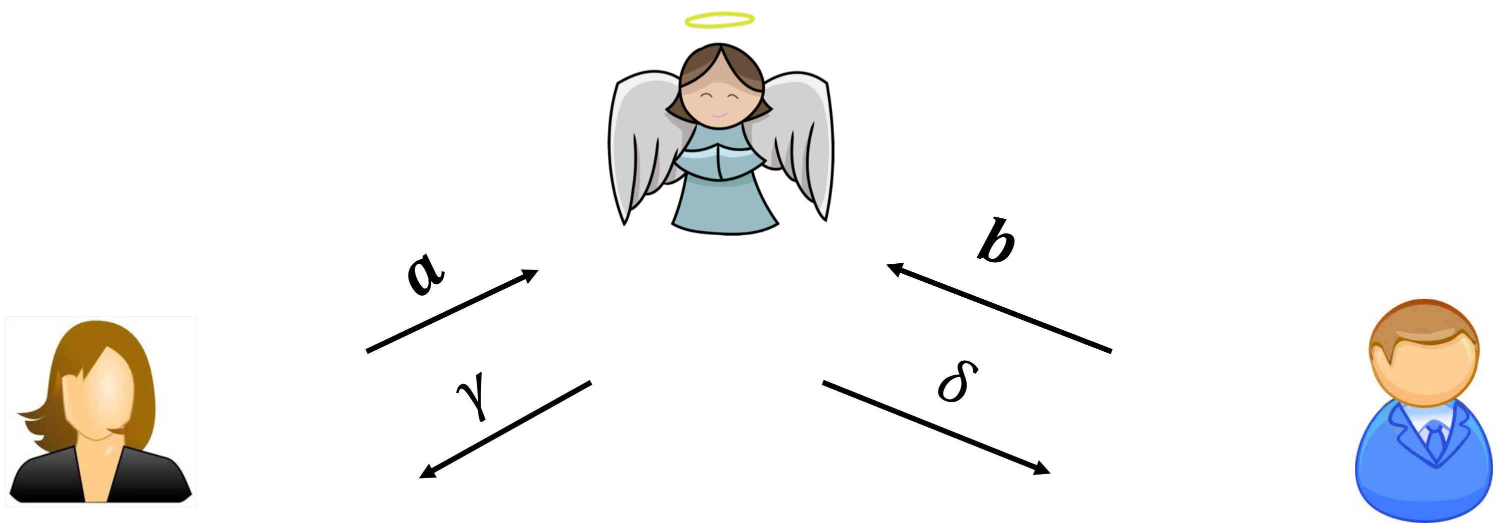


+



Security: Intuition (ss-AND hybrid model)

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



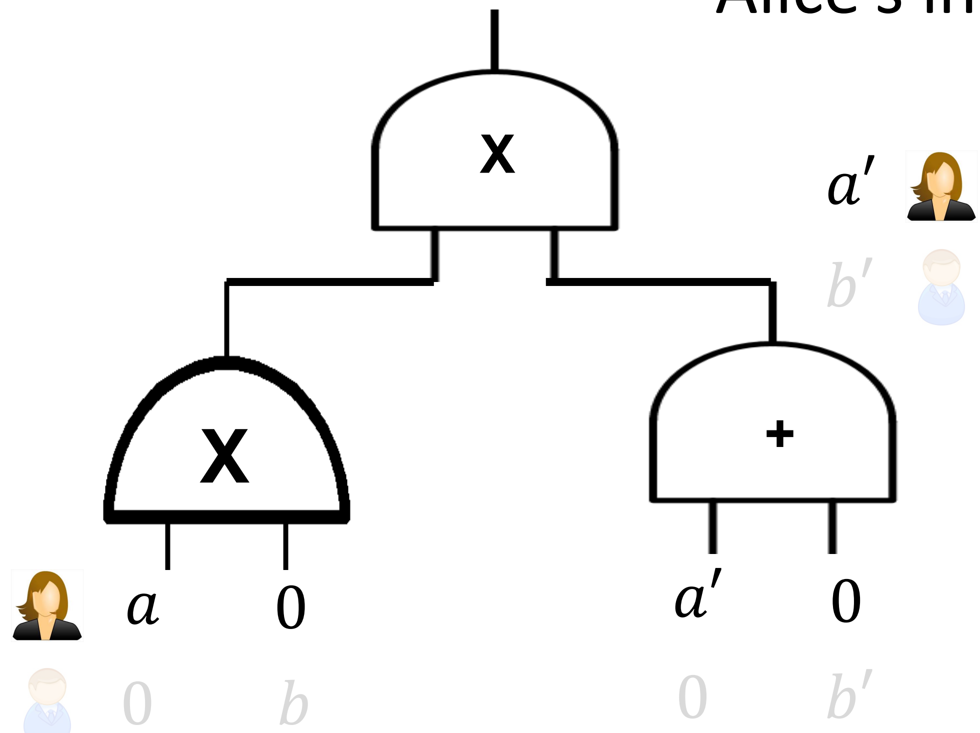
$$\gamma \oplus \delta = \mathrm{ab}$$

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

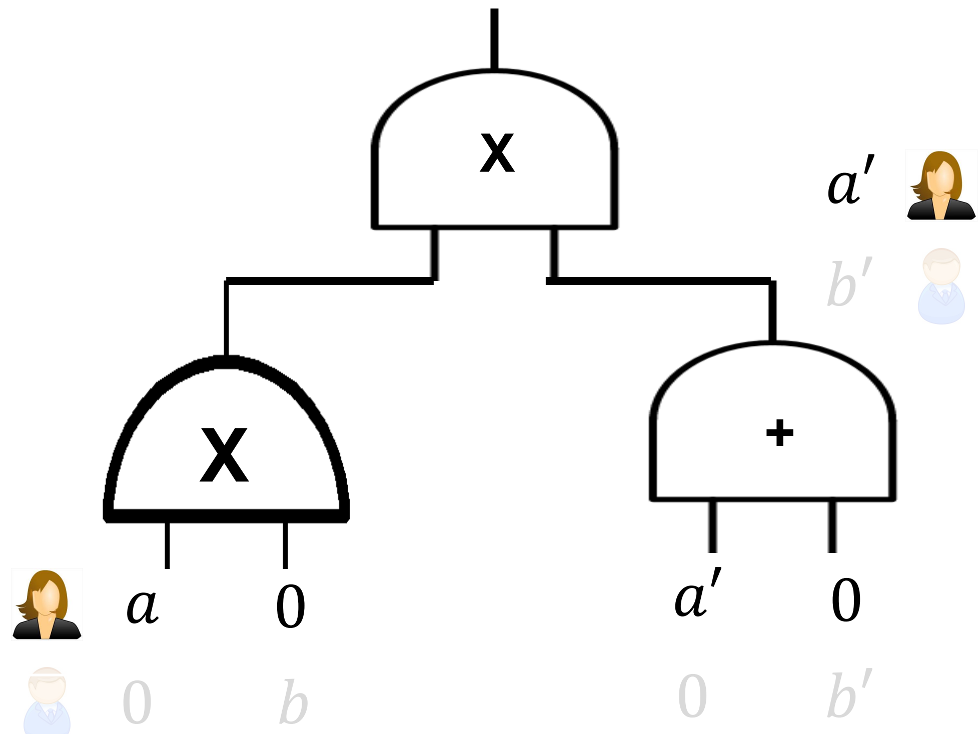
Security: Intuition (ss-AND hybrid model)

Simulator for Alice's view:

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

Alice's share

$$= a \cdot 0 + \gamma_{alice} + \delta_{alice}$$



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

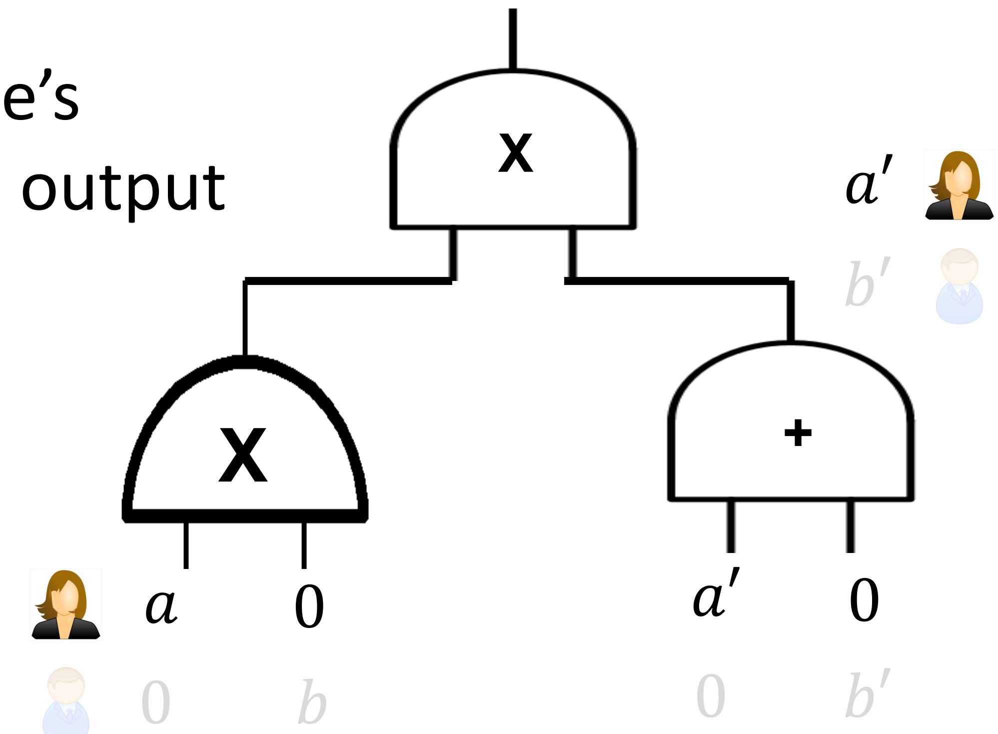
Security: Intuition (ss-AND hybrid model)

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 ⊕ function output

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



Secret-Shared AND protocol

Using the RSA trapdoor permutation.

Input bit: a

Input bit: b

Pick $N = PQ$ and RSA exponent e.

$$\xrightarrow{N, e}$$

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

Choose random s_{1-b}

$$\xleftarrow{s_0, s_1}$$

Let x_0 be random and $x_1 = x_0 \oplus \mathrm{a}$.

Compute r_0, r_1 and one-time pad x_0, x_1 using hardcore bits

$$\xrightarrow[x_1 \oplus HCB(r_1)]{x_0 \oplus HCB(r_0)}$$

Alice outputs x_0

Bob outputs x_b

Secret-Shared AND protocol

Using the RSA trapdoor permutation.

Input bit: a

Input bit: b

Exercise: Construct simulators for Alice and Bob.

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. $\bigoplus_{i=1}^{n} \alpha_i = a$ and $(\beta_1, \ldots, \beta_n)$ s.t. $\bigoplus_{i=1}^{n} \beta_i = b$, 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:

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

Exercise!

Security against Malicious (Active) Adversaries

Secure Two-Party Comp: New Def

(possibly randomized) $F(x, y; r) = (F_A(x, y; r), F_B(x, y; r))$

Input: $\boldsymbol{x}$ **Input:** $\boldsymbol{y}$

Alice Bob

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

$$\left(SIM_A\left(x, F_A(x, y)\right), F(x, y)\right) \cong \left(View_A(x, y), F(x, y)\right)$$

i.e. the joint distribution of the view and the output is correct

Counterexample

Randomized functionality $F(1^n, 1^n) = (r, \perp)$.

Protocol:

Alice picks a random r, outputs it **and sends it to Bob**.

Is this secure?

Secure acc. to old def, insecure acc. to new def.

Ergo, old def is insufficient.

Malicious Parties: Issues to Handle

1. Input (In)dependence: A malicious Alice could choose her input to depend on Bob's, something she cannot do in the ideal world.

Example: $F\big((a, b), x\big) = (\perp, ax + b)$

2. Randomness: A malicious Bob could choose his "random string" in the protocol the way she wants, something she cannot do in the ideal world.

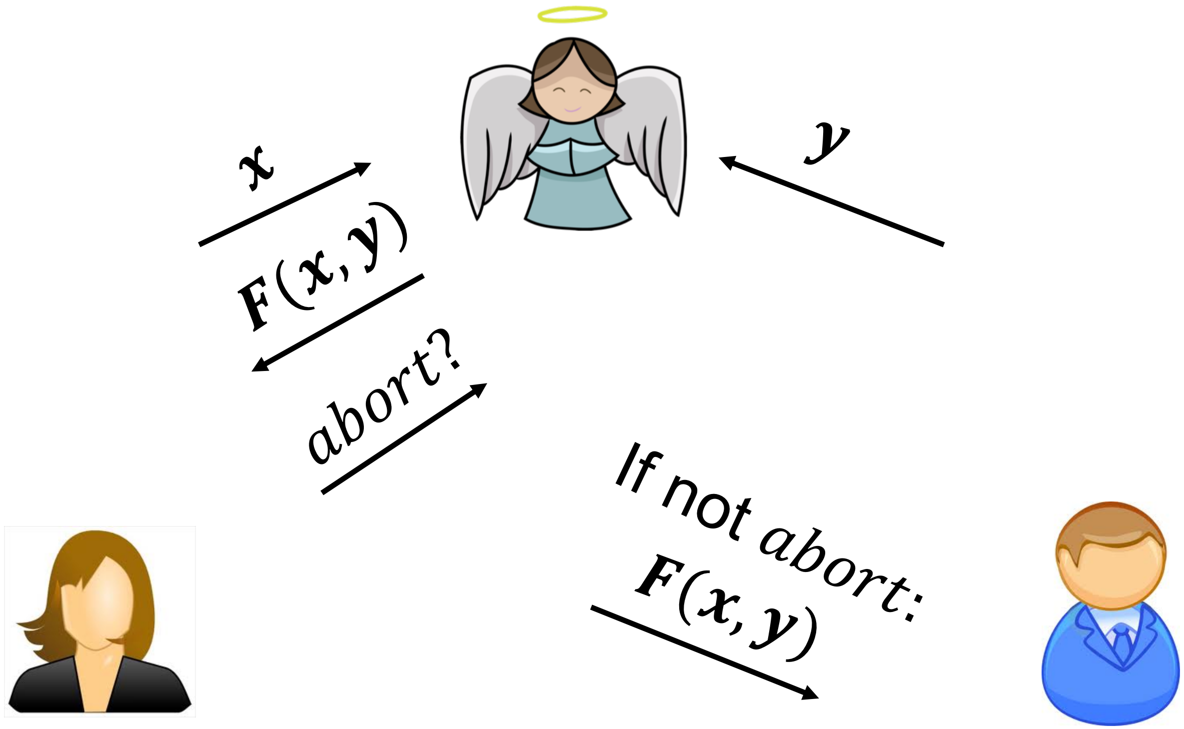
Example: our OT *protocol*

unavoidable

3. (Un)fairness: A malicious party could block the honest party from learning the output, while learning it herself.

4. Deviate from Protocol Instructions.

New (Less) Ideal Model



The “GMW Compiler”

***Theorem* [Goldreich-Micali-Wigderson’87]**: Assuming one-way functions exist, there is a general way to transform any semi-honest secure protocol computing a (possibly randomized) function F into a maliciously secure protocol for F.

Input Independence

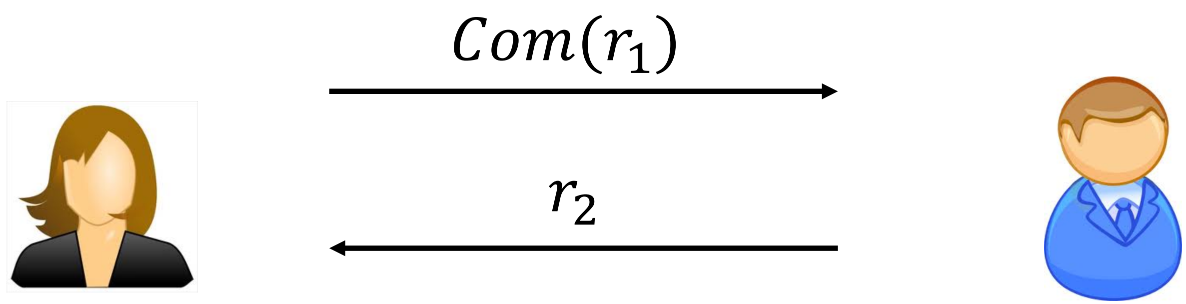
1. Input (In)dependence: A malicious party could choose her input to depend on Bob's, something she cannot do in the ideal world.

Solution: Each party commits to their input in sequence, and provides a **zero-knowledge proof of knowledge** of the underlying input.

Solution: Coin-Tossing Protocol

2. Randomness: A malicious party could choose her "random string" in the protocol the way she wants, something she cannot do in the ideal world.

Def: Realize the functionality $F(1^n, 1^n) = (r, Com(r))$.



Output $r = r_1 \oplus r_2$

Output $(Com(r_1), r_2)$

Zero Knowledge Proofs

4. Deviate from Other Protocol Instructions.

Solution: Each message of each party is a *deterministic* function of their input, their random coins and messages from party B.

When party A sends a message $m = m(x_A, r_A, \overline{msg_B})$, they also prove in zero-knowledge that they did so correctly. That is, they prove in ZK the following NP statement:

$$(m, \overline{msg_B}, XCom, RCom)\text{: } \exists\, x_A, r_A \text{ s.t.}$$
$$m = m(x_A, r_A, \overline{msg_B}) \;\wedge XCom = \mathrm{Com}(x_A) \wedge$$
$$RCom = \mathrm{Com}(r_A)$$

Optimizations

Optimization 1: Preprocessing OTs

Random OT tuple (or AND tuple, or Beaver tuple after D. Beaver): Alice has (α, γ_a) and Bob has (β, γ_b) which are random s.t. $\boldsymbol{\gamma_a \oplus \gamma_b = \alpha\beta}$**.**

Theorem: Given O(1) many *random* OT tuples, we can do OT with information-theoretic security, exchanging O(1) bits.

Optimization 2: OT Extension

Theorem
[Beaver'96, Ishai-Kushilevitz-Nissim-Pinkas'03]:

Given O(λ) many *random* OT tuples, we can generate n OT tuples exchanging O(n) bits --- as opposed to the trivial O($n\lambda$) bits --- and using only symmetric-key crypto.

Complexity of the 2-party solution

Number of OT protocol invocations $= 2 \ * \ \#AND$ gates

Can be made into O(#inputs · λ): Yao's garbled circuits

Number of rounds = AND-depth of the circuit

Can be made into O(1) rounds: Yao's garbled circuits

Communication in bits =

$$O(\#AND \cdot \lambda + \#outputs)$$

Can be made into O(λ #inputs) using FHE: but FHE is computationally more expensive concretely.