CIS 5560

Cryptography
Lecture 25

Course website:

pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW10** due **Thursday Apr 25** at 11:59PM on Gradescope
- **HW11** due **Wednesday May 1** at 11:59PM on Gradescope

Recap of Last Lecture

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

Secure Computation

Input: x **Input:** y

Alice Bob

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

Secure Two-Party Computation

Input: x **Input:** y

Alice Bob

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

Security:

Semi-Honest Security:

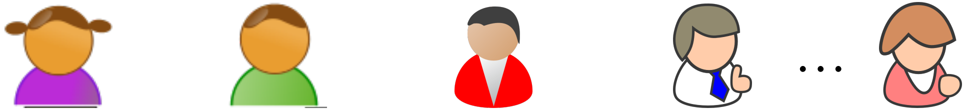
- Alice should not learn anything more than x and $F_A(x, y)$.
- Bob should not learn anything more than y and $F_B(x, y)$.

secret b

Secret Sharing

Dealer

share s_1 share s_2 share s_3 share s_4 share s_n



P_1 P_2 P_3 P_4 P_n

- ❑ Any **"authorized"** subset of players **can recover** b.
- ❑ No other subset of players **has any info** about b.

- ○ Threshold (or t-out-of-n) SS [Shamir'79, Blakley'79]:

  "authorized" subset = has size $\geq$ t.

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:

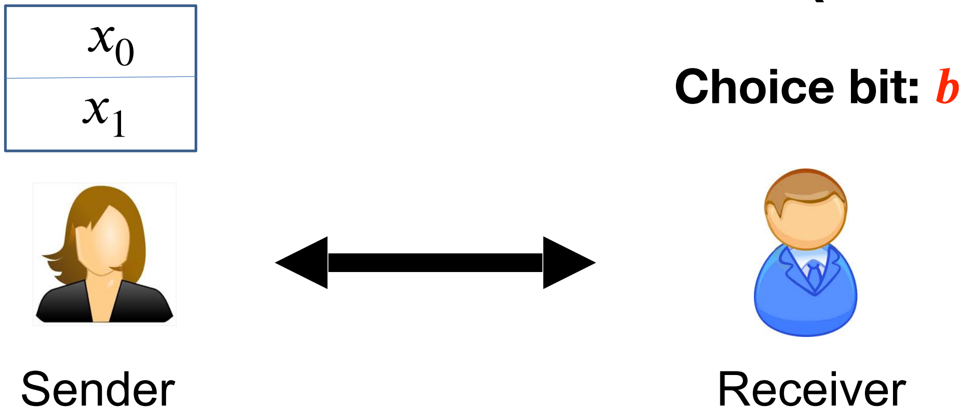
$$s_1 = f(1), s_2 = f(2), \ldots, s_i = f(i), \ldots, s_n = f(n)$$

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…)

Why OT? Computing ANDs

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

Alice and Bob want to compute the AND $\alpha \wedge \beta$.

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

$x_0 = 0$

$x_1 = \alpha$

Run an OT protocol

Choice bit $b = \beta$

Bob gets α if $\beta = 1$, and 0 if $\beta = 0$

Here is a way to write the OT selection function: $\boldsymbol{x_1 b + x_0(1 - b)}$

which, in this case is $= \alpha\beta$.

The Billionaires’ Problem

Who is richer?

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

X

Y

$\cdots$	0	1	0	0	$\cdots$

$\cdots$	0	1	1	1	1	1	1

Unit Vector u_X = 1 in the X^{th} location and 0 elsewhere

Vector v_Y = 1 from the $(Y+1)^{th}$ location onwards

$$f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]$$

~~Compute each AND individually and sum it up~~?

Today's Lecture

- OT for AND of secret-shared bits
- Definition of MPC
- Definition of OT
- Construction of OT from Trapdoor Permutations
-

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

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

$\cdots$	0	1	0	0	$\cdots$

Unit Vector u_X

$\cdots$	0	1	1	1	1	1	1

Vector v_Y

$$\boldsymbol{f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]}$$

1. Alice and Bob run many OTs to get (γ_i, δ_i) s.t. $\gamma_i \bigoplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$

2. Alice computes $\gamma = \oplus_i \gamma_i$ and Bob computes $\delta = \oplus_i \delta_i$

3. Alice reveals γ and Bob reveals δ.

Check (correctness): $\gamma \oplus \delta = \langle u_X, v_Y \rangle = f(X, Y)$.

The Billionaires' Problem

$f(X, Y) = 1$
if and only if $X > Y$

$\cdots$	0	1	0	0	$\cdots$

Unit Vector u_X

$\cdots$	0	1	1	1	1	1	1

Vector v_Y

$$f(X, Y) = \langle u_X, v_Y \rangle = \sum_{i=1}^{U} u_X[i] \wedge v_Y[i]$$

1. Alice and Bob run many OTs to get (γ_i, δ_i) s.t. $\gamma_i \bigoplus \delta_i = u_X[i] \wedge v_Y[i]$

2. Alice computes $\gamma = \oplus_i \gamma_i$ and Bob computes $\delta = \oplus_i \delta_i$

3. Alice reveals γ and Bob reveals δ .

Check (privacy): Alice & Bob get a bunch of random bits.

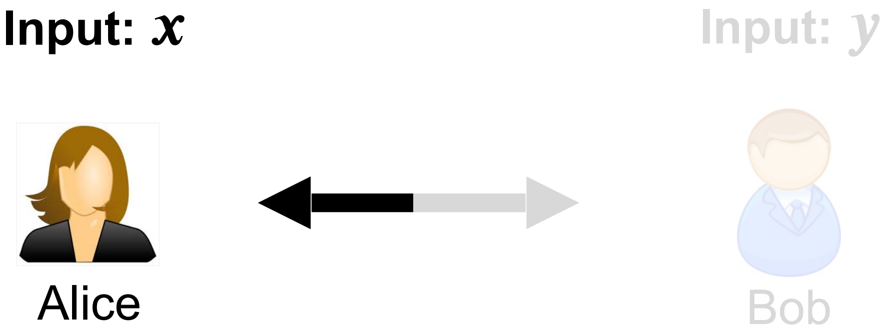
“OT is Complete”

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

Defining Security:
The Ideal/Real Paradigm

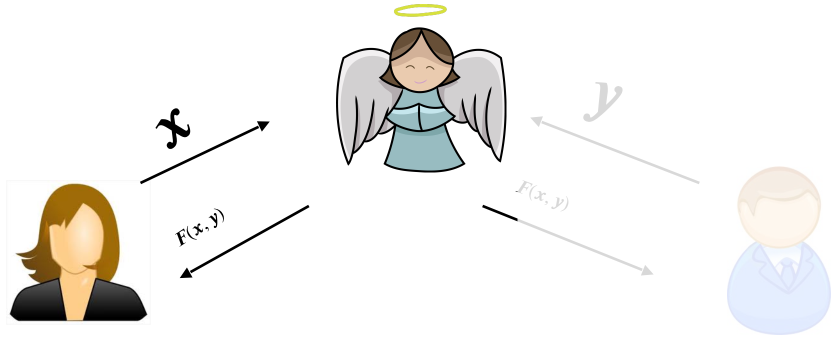
Secure Two-Party Computation

REAL WORLD:



$\approx$

IDEAL WORLD:



Secure Two-Party Computation

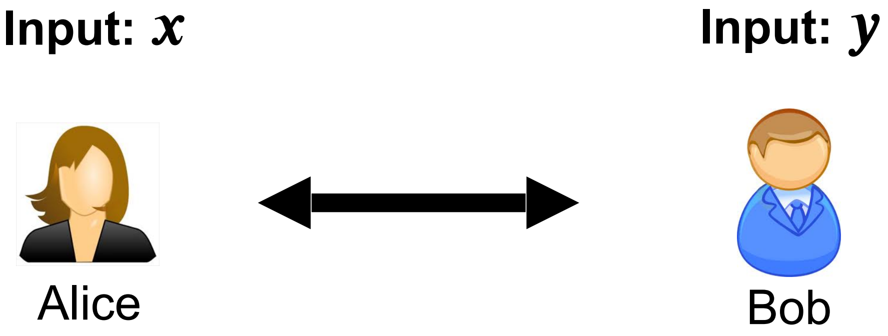
Input: x **Input:** y

Alice Bob

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:

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

OT Definition

x_0
x_1

Choice bit: b

Sender　　　　　　　　　　Receiver

Receiver Security: Sender should not learn b.

Define Sender's view $View_S(x_0, x_1, b)$ = her random coins and the protocol messages.

OT Definition

x_0
x_1

Choice bit: b

Sender

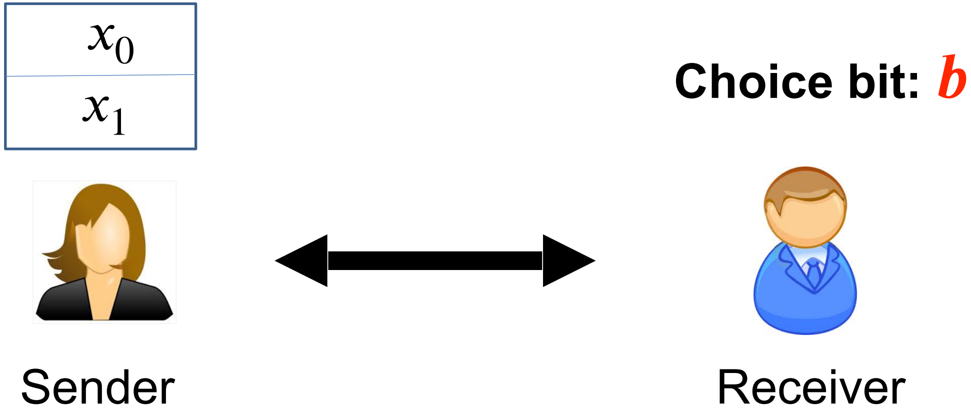
Receiver

Receiver Security: Sender should not learn b.

There exists a PPT simulator SIM_S such that for any x_0, x_1 and b:

$$SIM_S(x_0, x_1) \cong View_S(x_0, x_1, b)$$

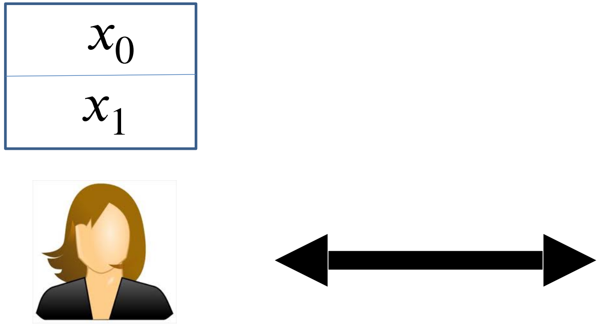
OT Definition



Sender Security: Receiver should not learn x_{1-b}.

Define Receiver’s view $View_R(x_0, x_1, b)$ = his random coins and the protocol messages.

OT Definition



Choice bit: $\boldsymbol{b}$

Sender

Receiver

Sender Security: Receiver should not learn x_{1-b}**.**

There exists a PPT simulator SIM_R such that for any x_0, x_1 and b:

$$SIM_R(b, x_b) \cong View_R(x_0, x_1, b)$$

OT Protocols

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.

$\xrightarrow{N, e}$

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

Choose random s_{1-b}

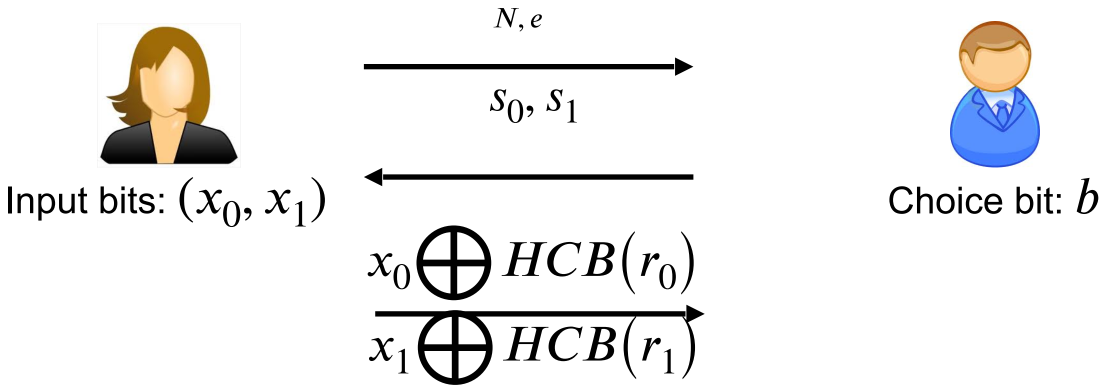
$\xleftarrow{s_0, s_1}$

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

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

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

OT Protocol 1: Trapdoor Permutations



How about Bob's security
(a.k.a. Why does Alice not learn Bob's choice bit)?

Alice's view is s_0, s_1 one of which is chosen randomly from Z_N^* and the other by raising a random number to the e-th power. They look exactly the same!

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

$\xrightarrow{N, e}$

$\xleftarrow{s_0,\, s_1}$

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

Choice bit: b

How about Bob's security
(a.k.a. Why does Alice not learn Bob's choice bit)?

Exercise: Show how to construct the simulator.

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

N, e $\longrightarrow$

s_0, s_1 $\longleftarrow$

$x_0 \bigoplus HCB(r_0)$

$x_1 \bigoplus HCB(r_1)$ $\longrightarrow$

Choice bit: b

How about Alice's security
(a.k.a. Why does Bob not learn both of Alice's bits)?

Assuming Bob is semi-honest, he chose s_{1-b} uniformly at random, so the hardcore bit of $s_{1-b} = r_{1-b}^d$ is computationally hidden from him.

Many More Constructions of OT

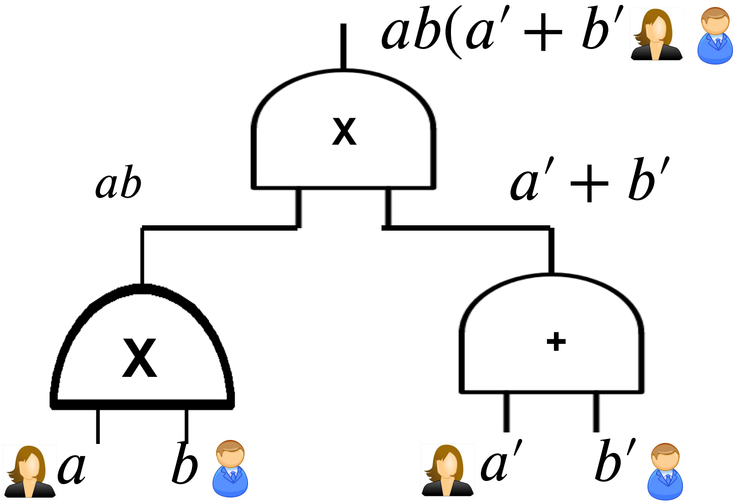
***Theorem**:* OT protocols can be constructed based on the hardness of the Diffie-Hellman problem, factoring, quadratic residuosity, LWE, elliptic curve isogeny problem etc. etc.

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.

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

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

Output: γ

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

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

Output: δ

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

Run an OT protocol

Choice bit $b = \beta$

Alice outputs γ.

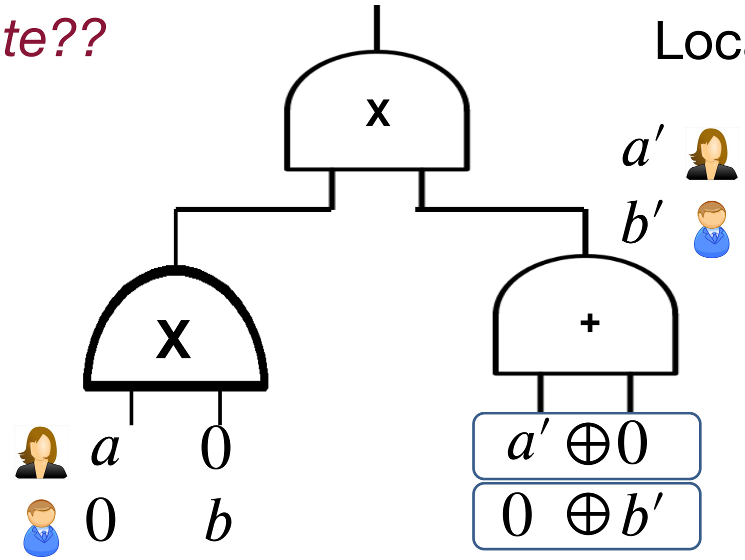
Bob gets $x_1 b + x_0(1 \oplus b) = (x_1 \oplus x_0)b + x_0 = \alpha\beta \oplus \gamma := \delta$

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

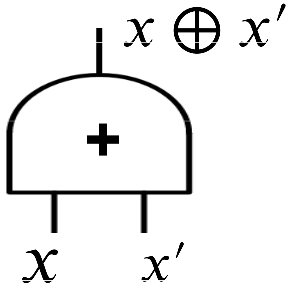
AND gate??



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

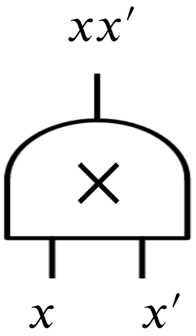
So, we have: $(\alpha \oplus \alpha') \oplus (\beta \oplus \beta')$

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

Computing the AND gate

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

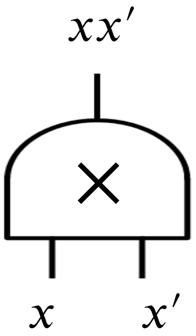
Alice has α' and Bob has β' s.t. $\alpha' \oplus \beta' = x'$



Desired output (to maintain invariant):

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

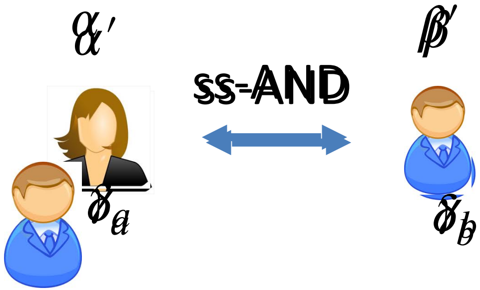
Computing the AND gate



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

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

$\oplus$ γ_b $\oplus$ δ_b

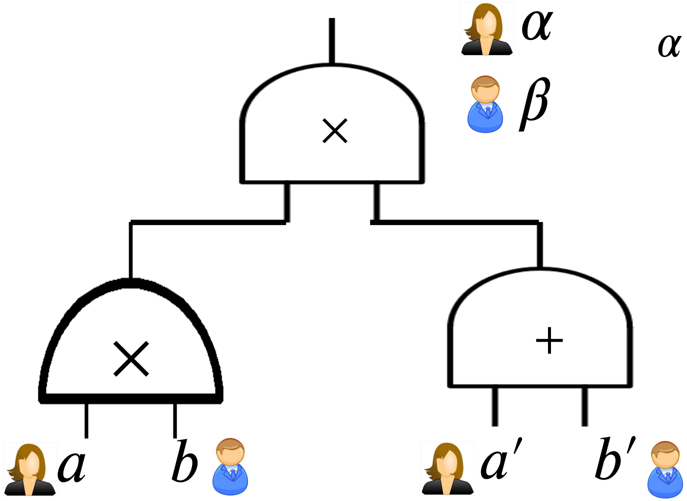


$$\alpha'' = \alpha\alpha' \oplus \gamma_a \oplus \delta_a \qquad \beta'' = \beta\beta' \oplus \gamma_b \oplus \delta_b$$

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.

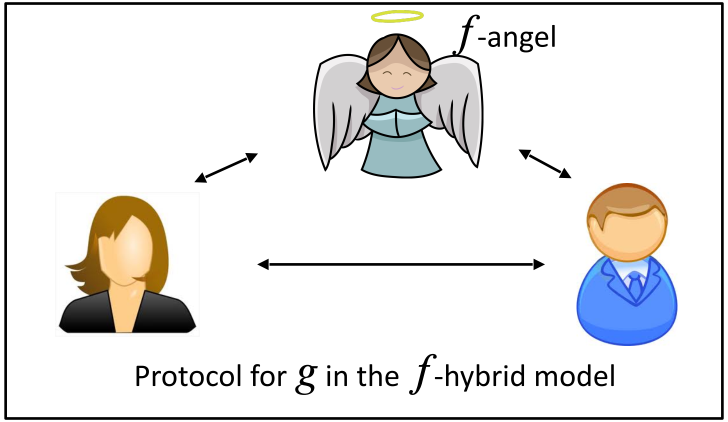


$$\alpha \oplus \beta = ab(a' \oplus b')$$

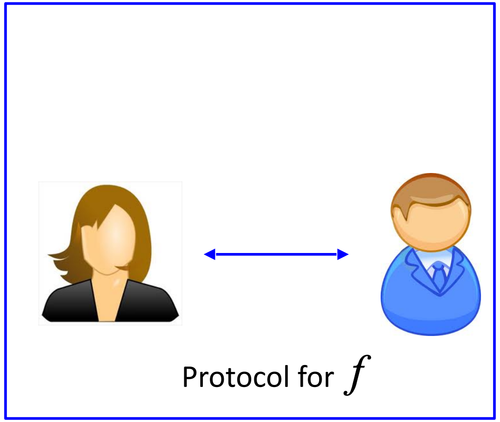
Security by Composition

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.

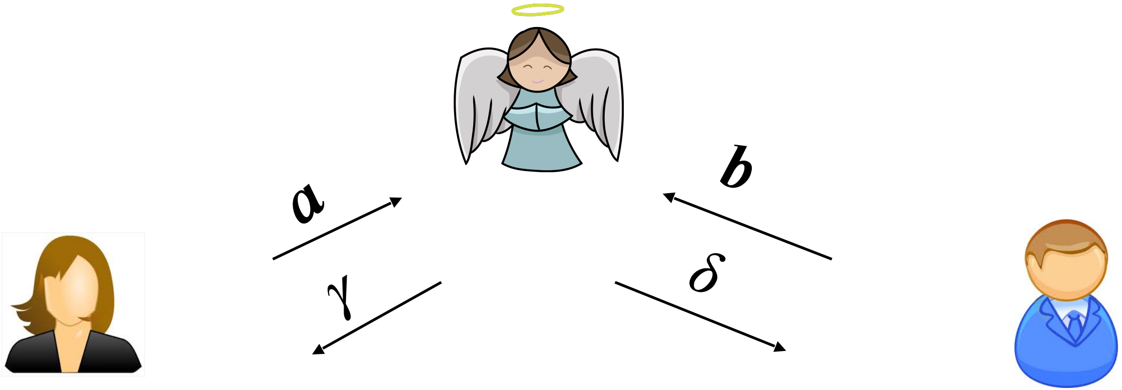


+



Security: Intuition (ss-AND hybrid model)

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



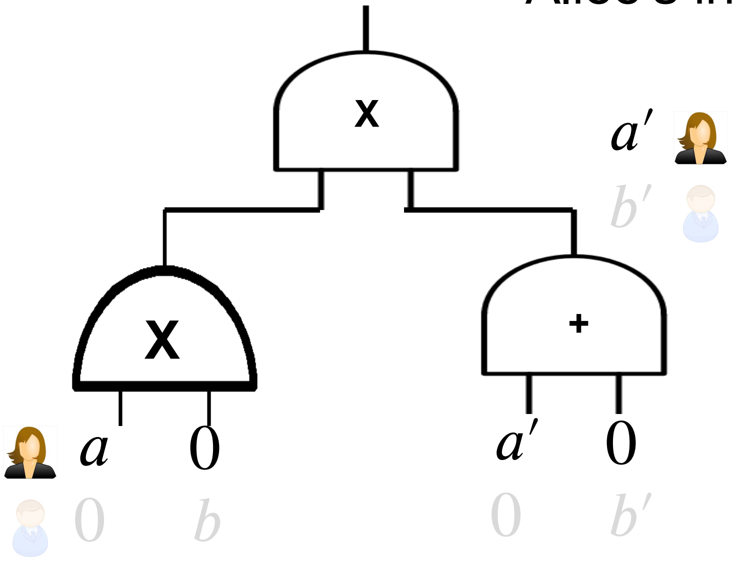
$$\gamma \bigoplus \delta = ab$$

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:

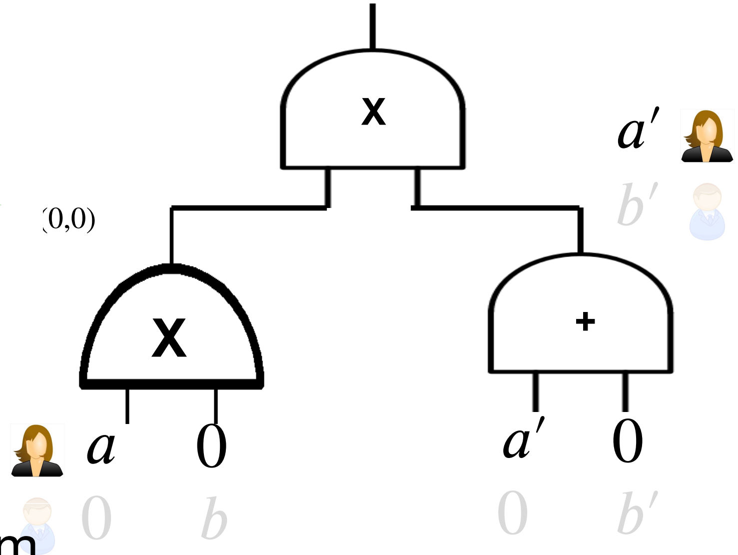
AND gate: simulate given Alice's input shares & outputs from the ss-AND angel.

Alice's share

$= a \bullet 0 + \gamma_{alice}$

δ_{alice}

(0,0)



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

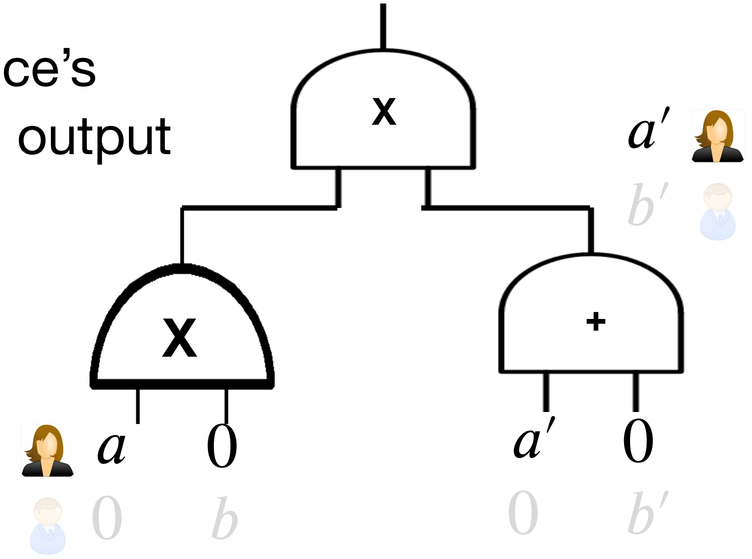
Security: Intuition (ss-AND hybrid model)

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.

N, e →

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

Choose random s_{1-b}

← s_0, s_1

Let x_0 be random and $x_1 = x_0 \bigoplus \text{a}$.

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

$x_0 \bigoplus HCB(r_0)$

$x_1 \bigoplus HCB(r_1)$ →

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!