MIT 6.875

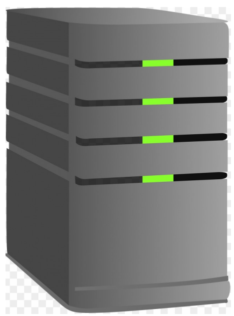
Foundations of Cryptography
Lecture 21

TODAY: Oblivious Transfer and Private Information Retrieval

Basic Problem: Database Access

Database D

0	x_0
1	x_1
2	x_2
3	x_3
4	x_4
5	x_5
6	x_6
7	x_7



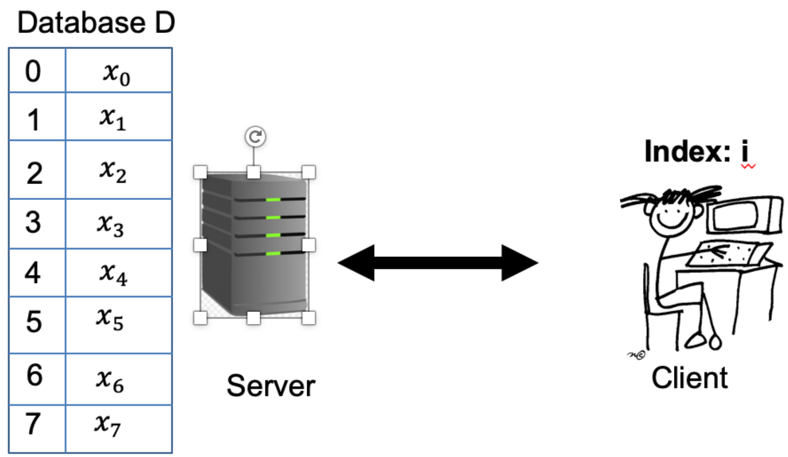
Server

Index: i

Client

Correctness: Client gets $D[i]$.

Privacy (for client): Server gets no information about i.



Here is a "solution": The server sends the DB to the client.

Two ways to be secure non-trivially

Oblivious Transfer (OT)

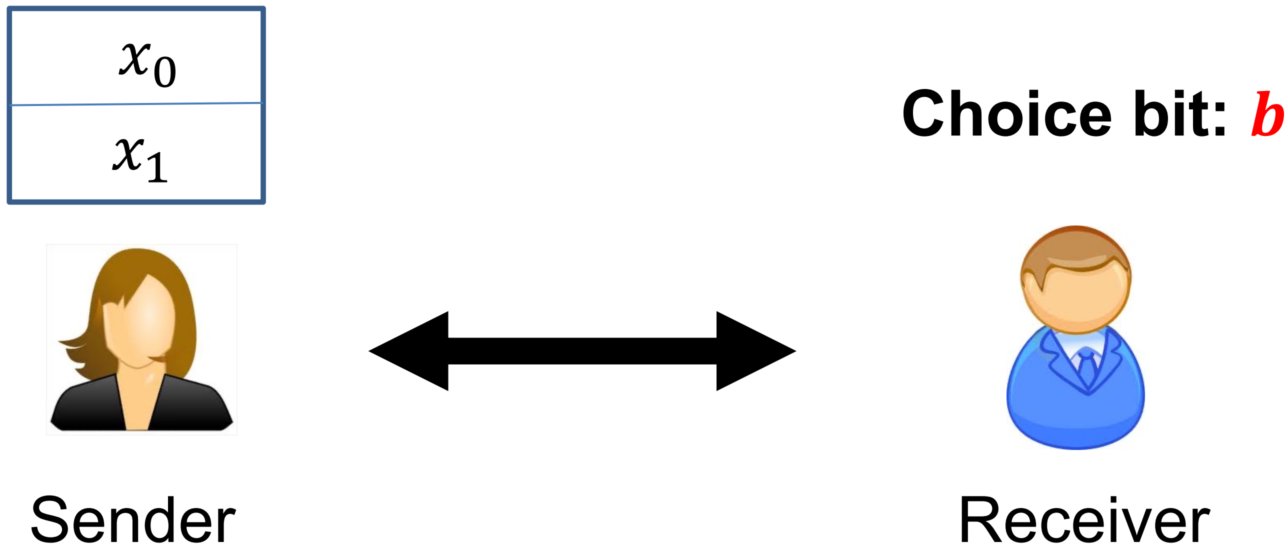
Add'l property: server privacy

Symmetric PIR =
Succinctness +
Server privacy

Private Information Retrieval (PIR)

Add'l property: succinctness

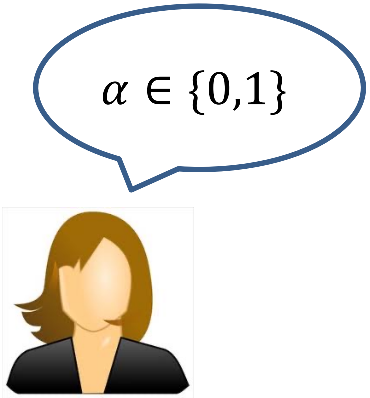
Oblivious Transfer (OT)



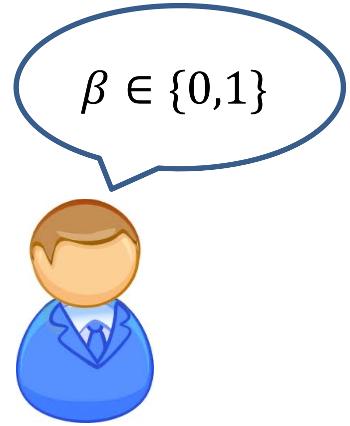
- Sender holds two bits x_0 and x_1.
- Receiver holds a choice bit b.
- Receiver should learn x_b, sender should learn nothing.

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

Why OT? The Dating Problem



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



Why OT? The Dating Problem

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

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 β=1, and 0 if β=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

⋯	0	1	0	0	⋯

⋯	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

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

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

Detour: OT $\Rightarrow$ 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

$x_0 = \gamma$
$x_1 = \alpha \oplus \gamma$

Run an OT protocol

Choice bit $b = \beta$

Alice outputs γ.

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

The Billionaires' Problem

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

...	0	1	0	0	...

Unit Vector u_X

...	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.

$$\gamma_i \oplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$$

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): $\gamma \oplus \delta = \boldsymbol{\langle u_X, v_Y \rangle = f(X, Y)}$.

The Billionaires' Problem

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

…	0	1	0	0	…

Unit Vector u_X

…	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.

$$\gamma_i \oplus \delta_i = \boldsymbol{u_X[i] \wedge v_Y[i]}$$

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

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

“OT is Complete”

Theorem *(lec22-24):* OT can solve not just love and money, but ***any*** two-party (and multi-party) problem efficiently (complexity prop. To circuit size of f).

OT Definition

x_0
x_1

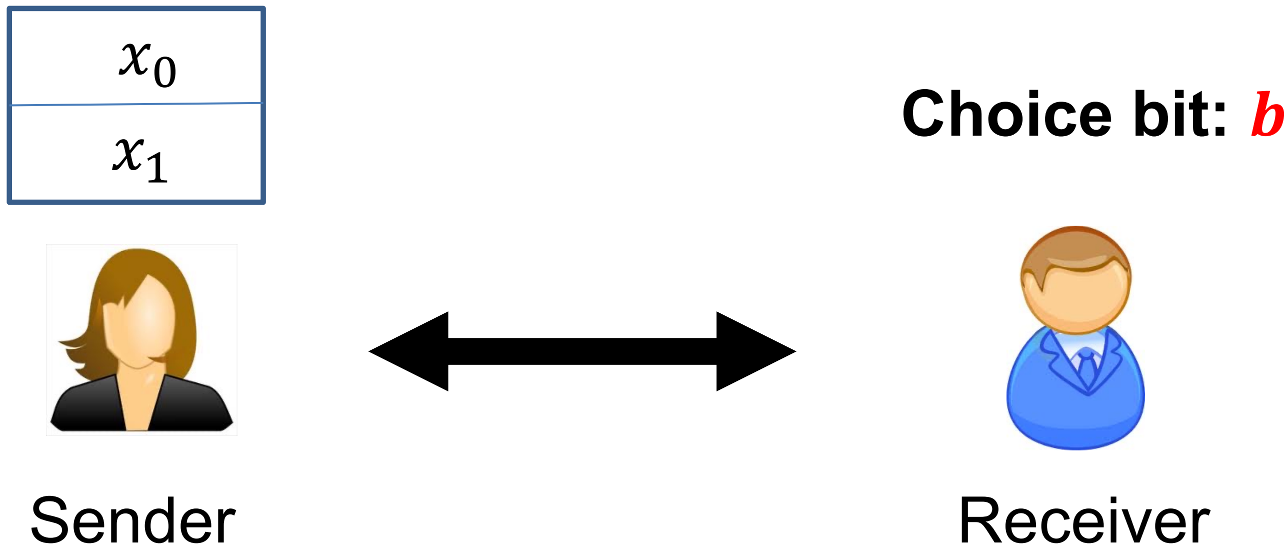
Choice bit: b

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

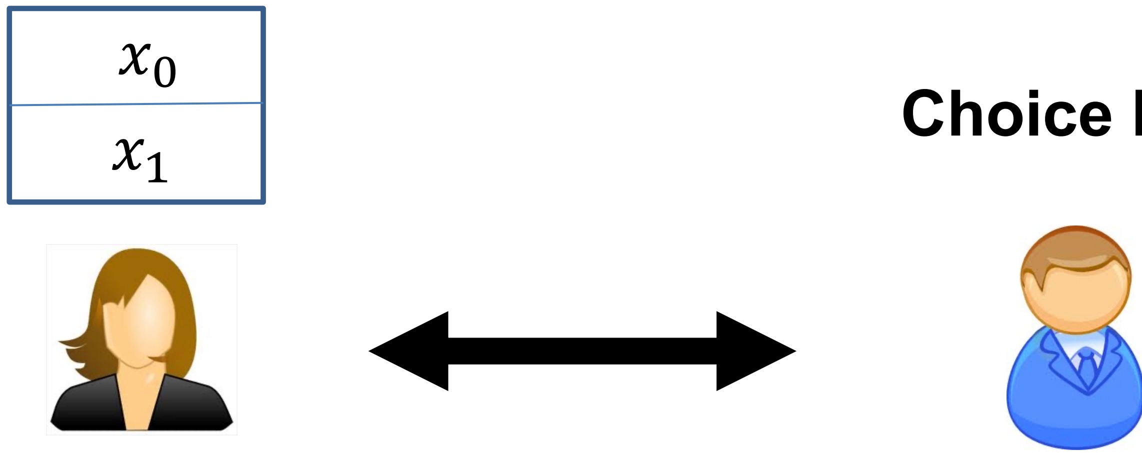


Receiver Security: Sender should not learn b.

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

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

OT Definition



Choice bit: $\boldsymbol{b}$

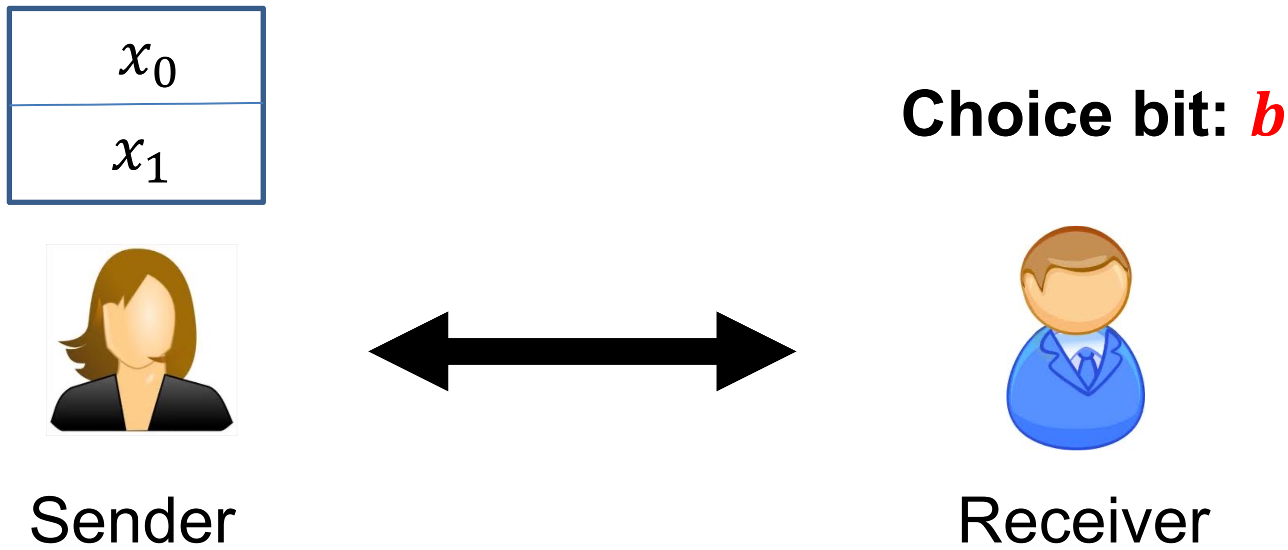
Sender

Receiver

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



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 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 $s_b = r_b^e \bmod N$

← s_0, s_1

Choose random s_{1-b}

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

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

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

OT Protocol 1: Trapdoor Permutations

Input bits: (x_0, x_1)

N, e

s_0, s_1

$x_0 \oplus HCB(r_0)$

$x_1 \oplus HCB(r_1)$

Choice bit: b

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)

$$\xrightarrow{N, e}$$

$$\xleftarrow{s_0, s_1}$$

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

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 →

← s_0, s_1

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

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.

OT from Trapdoor Permutations

Input bits: (x_0, x_1)

$$N, e \longrightarrow$$

$$s_0, s_1 \longleftarrow$$

$$x_0 \oplus HCB(r_0) \longrightarrow$$
$$x_1 \oplus HCB(r_1)$$

Choice bit: b

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

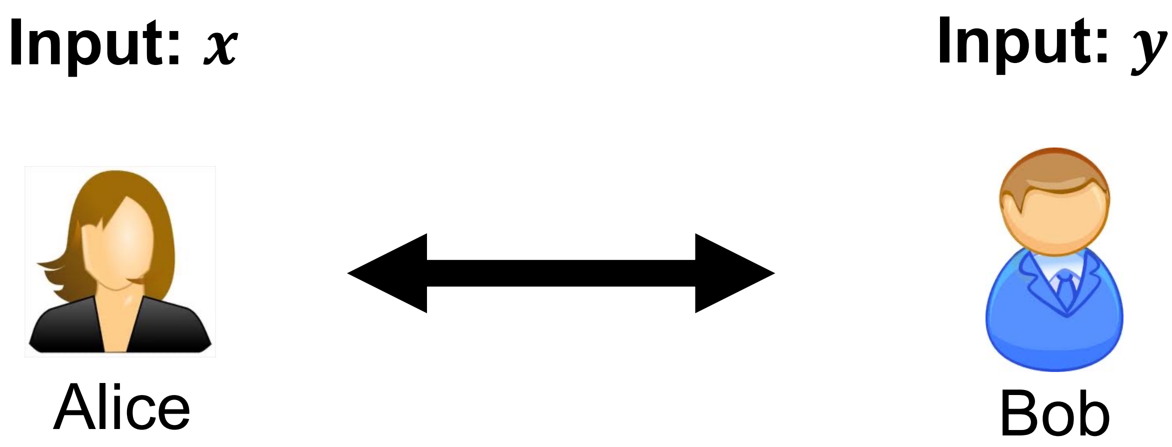
Exercise: Show how to construct the simulator.

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 Two-Party Computation

Secure Two-Party Computation

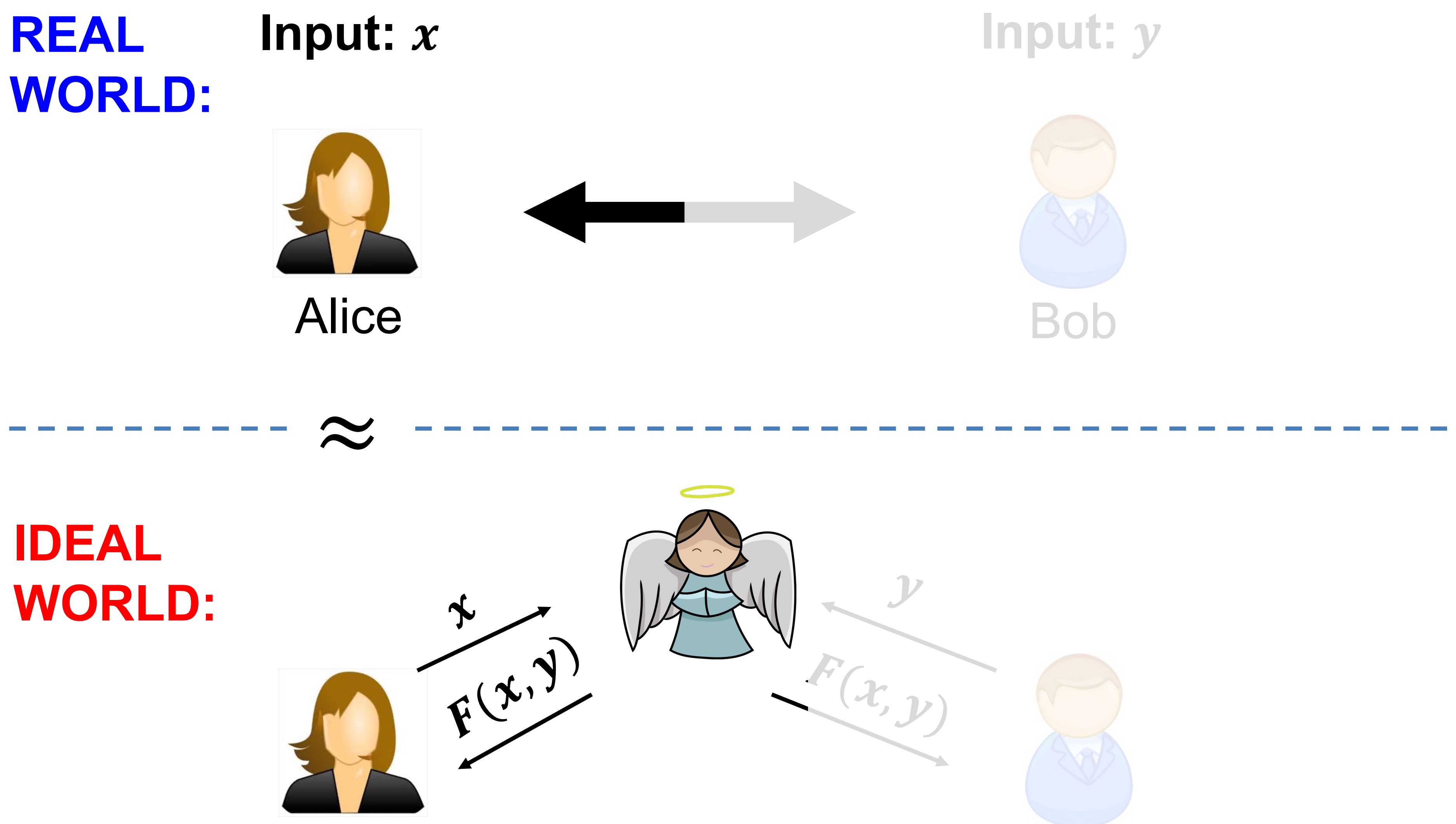


- Alice and Bob want to compute $F(x, y)$.

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

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

Secure Two-Party Computation



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

Input: x **Input:** y

Alice Bob

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

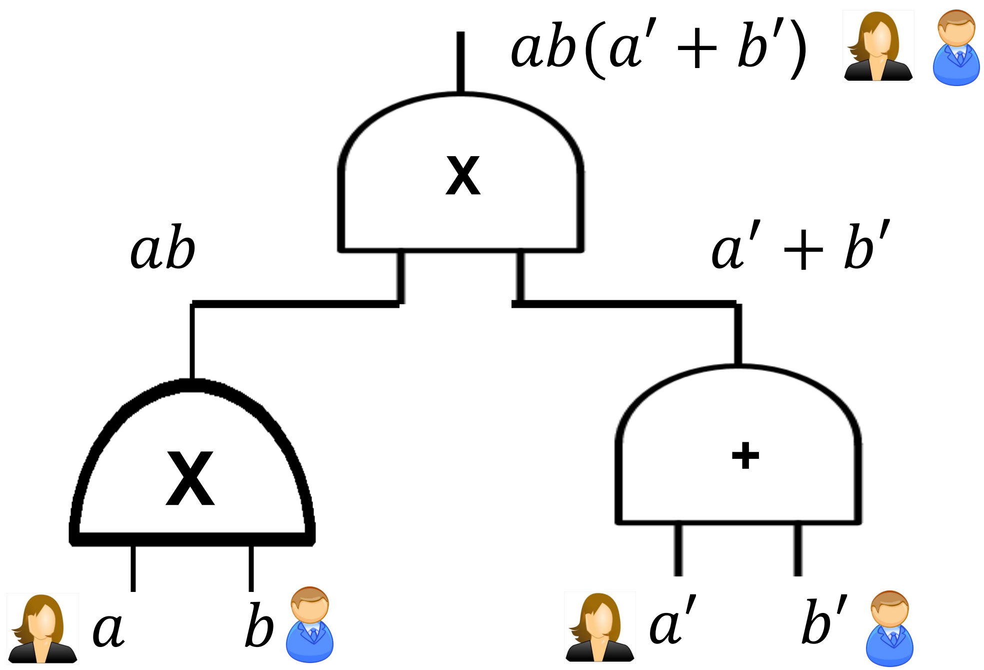
$$SIM_B(y, F(x,y)) \cong View_B(x,y)$$

Secure 2PC from OT

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

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.

Basic Secret-Sharing

A secret (bit) s is shared between Alice and Bob if Alice holds a bit α and Bob holds a bit β s.t. $\alpha \oplus \beta = s$

α and β are (typically) individually random, so neither Alice nor Bob knows any information about s. Together, however, they can recover s.

Recap: OT ⇒ 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

Choice bit b

Alice outputs γ.

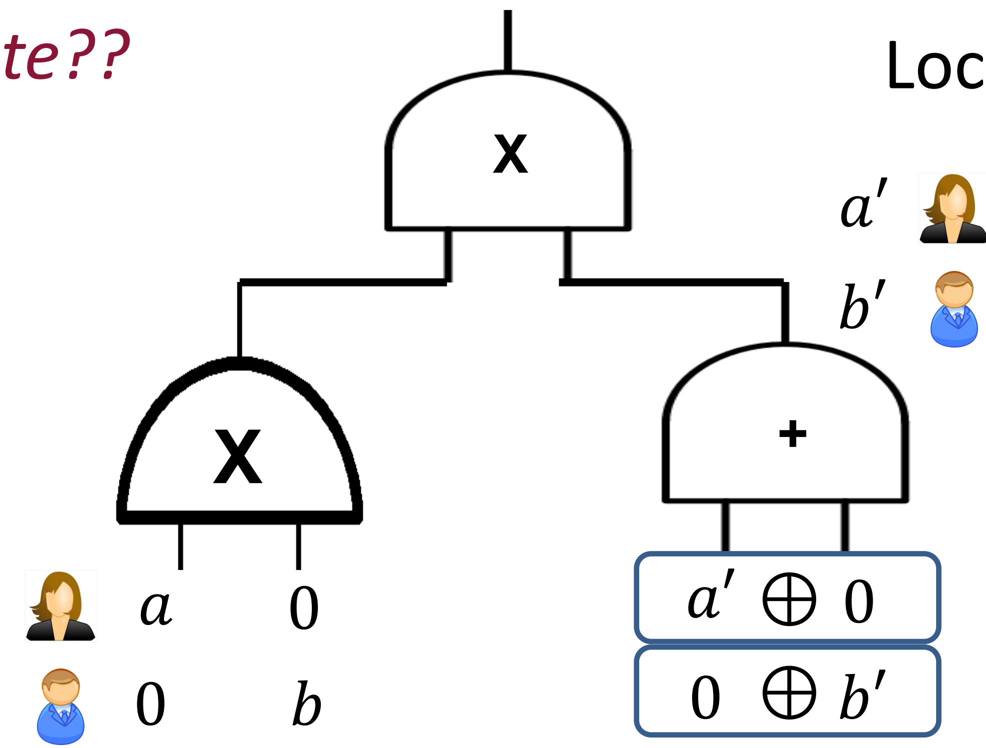
Bob gets $\boldsymbol{x_1 b} + \boldsymbol{x_0 (1 \oplus b)} = (\boldsymbol{x_1} \oplus \boldsymbol{x_0})\boldsymbol{b} + \boldsymbol{x_0} = ab \oplus \gamma := \delta$

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

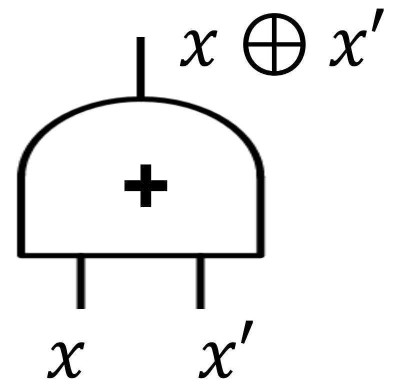


Base Case: Input wires

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: $(\alpha \oplus \alpha') \oplus (\beta \oplus \beta')$

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

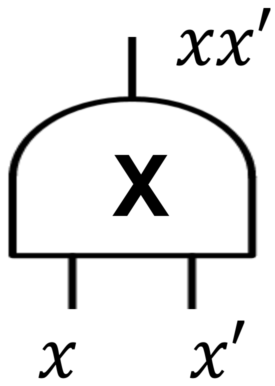
AND gate

Alice has α and Bob has β s.t.

$$\alpha \oplus \beta = x$$

Alice has α' and Bob has β' s.t.

$$\alpha' \oplus \beta' = x'$$

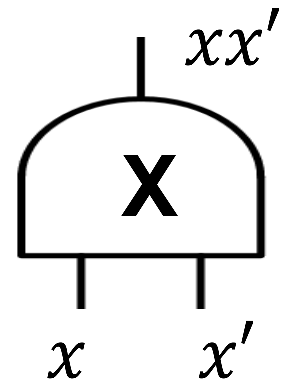


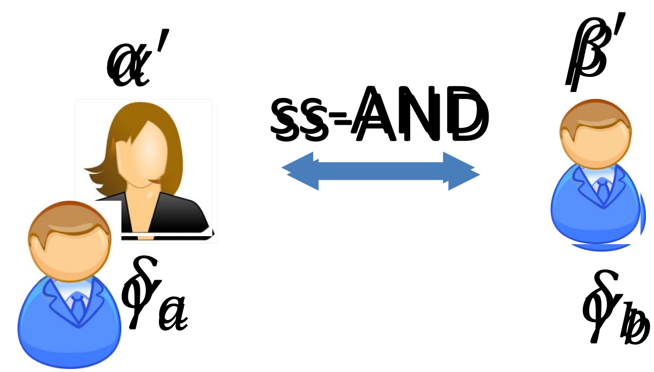
Desired output (to maintain invariant):
Alice wants $\boldsymbol{\alpha''}$ and Bob wants $\boldsymbol{\beta''}$ s.t. $\boldsymbol{\alpha''} \oplus \boldsymbol{\beta''} = xx'$

AND gate

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

$$\begin{array}{ccccccc} = \alpha\alpha' & \oplus & \gamma_a & \oplus & \delta_a & \oplus & \beta\beta' \\ & & \oplus & & \oplus & & \\ & & \gamma_b & & \delta_b & & \end{array}$$



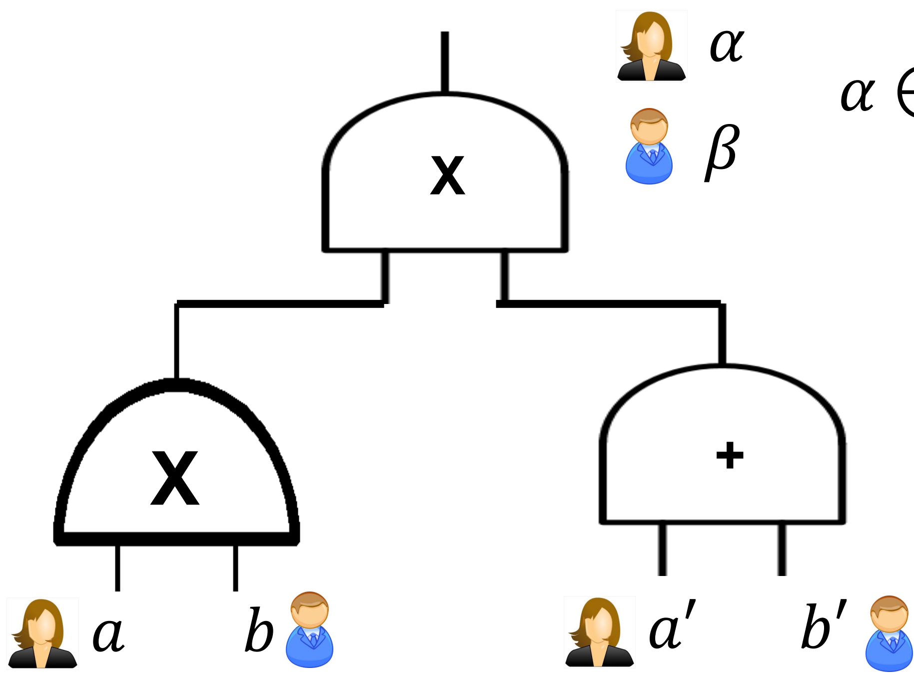


$$\alpha'' = \alpha\alpha' \oplus \gamma_a \oplus \delta_a \qquad \beta'' = \beta\beta' \oplus \gamma_b \oplus \delta_b$$

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.



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