MIT 6.875

Foundations of Cryptography
Lecture 25

Credit: Some slides adapted from Raluca Ada Popa and Vinod Vaikuntanathan

$\boldsymbol{O(1)}$-Round
Two-Party Computation

Secure Two-Party Computation

Input: x **Input:** y

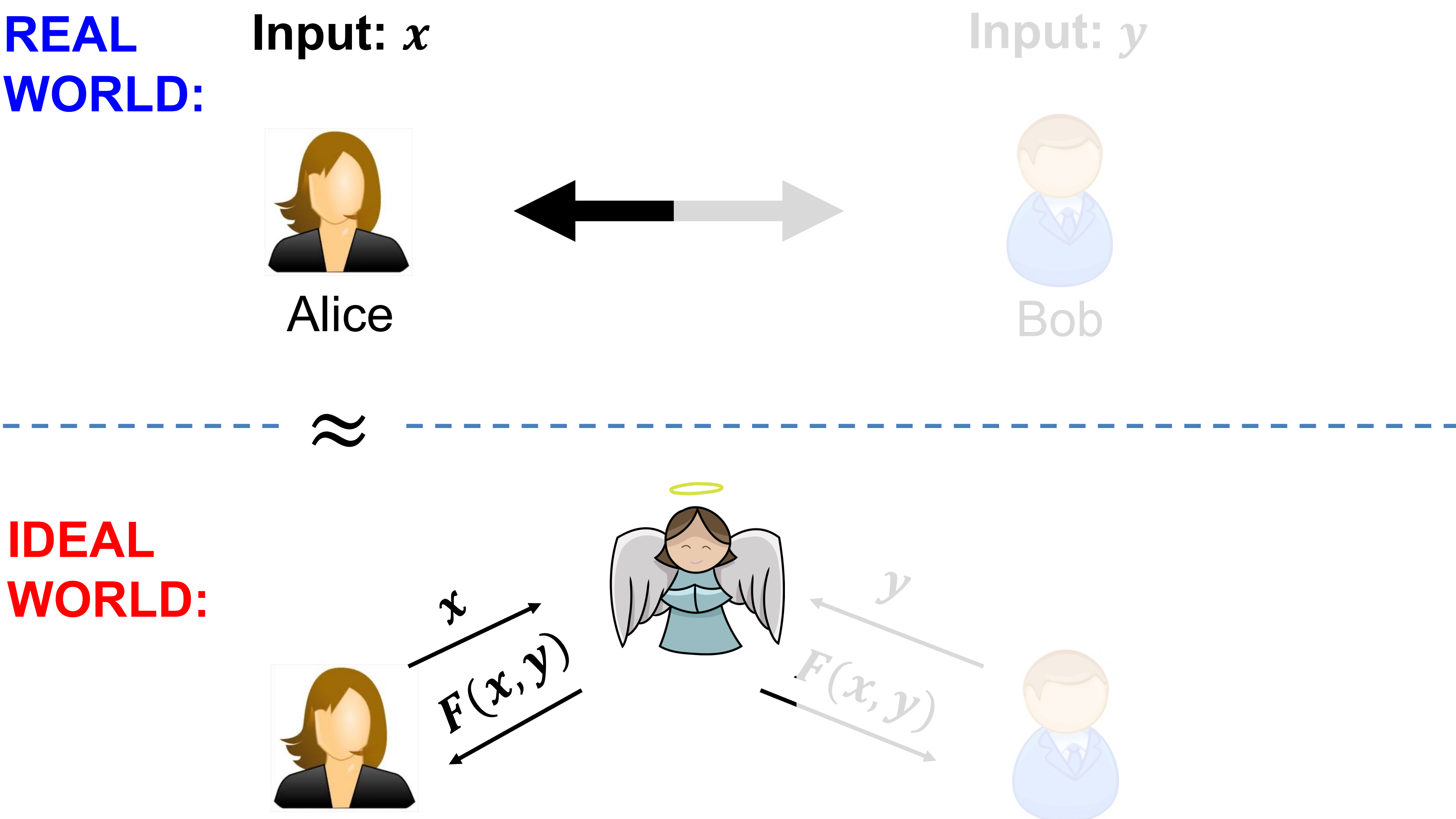
Alice Bob

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

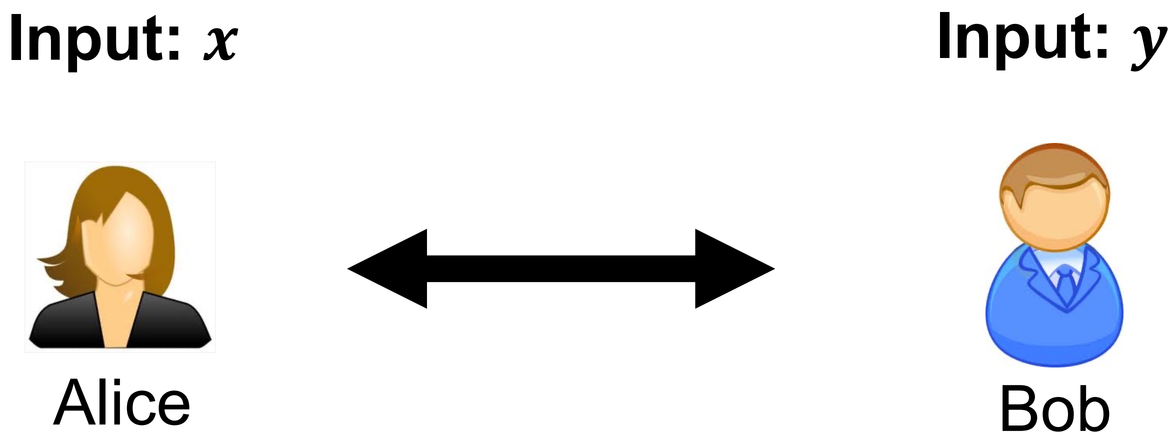
<u>Semi-honest Security:</u>

Parties should not learn anything more than their inputs and $F(x, y)$.

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

Input: x

Input: y

Alice

Bob

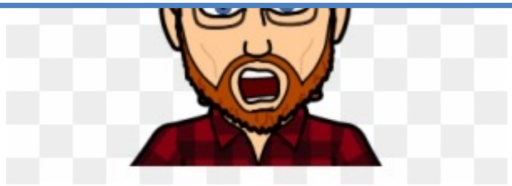
There exists a PPT simulator SIM_B such that for any x and y:

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

Secure MPC

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

One year before '87…
Theorem (Yao'86):
OT**+OWF** solve ***any*** **two**-party computation problem.



Secure 2PC from OT

- **Constant Round!!**
- Groundbreaking generic solution
- Inspired GMW'87 and more
- **Beyond secure computation**
  - Computing on encrypted data
  - Secure function evaluation
  - Parallel cryptography
  - …
  - Garbling as **Randomized Encoding of functions** [IK'00,IK'02,AIK'04,AIK'06]…

Secure Function Evaluation

Alice's private input is $C: \{0,1\}^n \to x$
Bob's private input is x

Goal: Compute $C(x)$

Q: Is iO a solution?

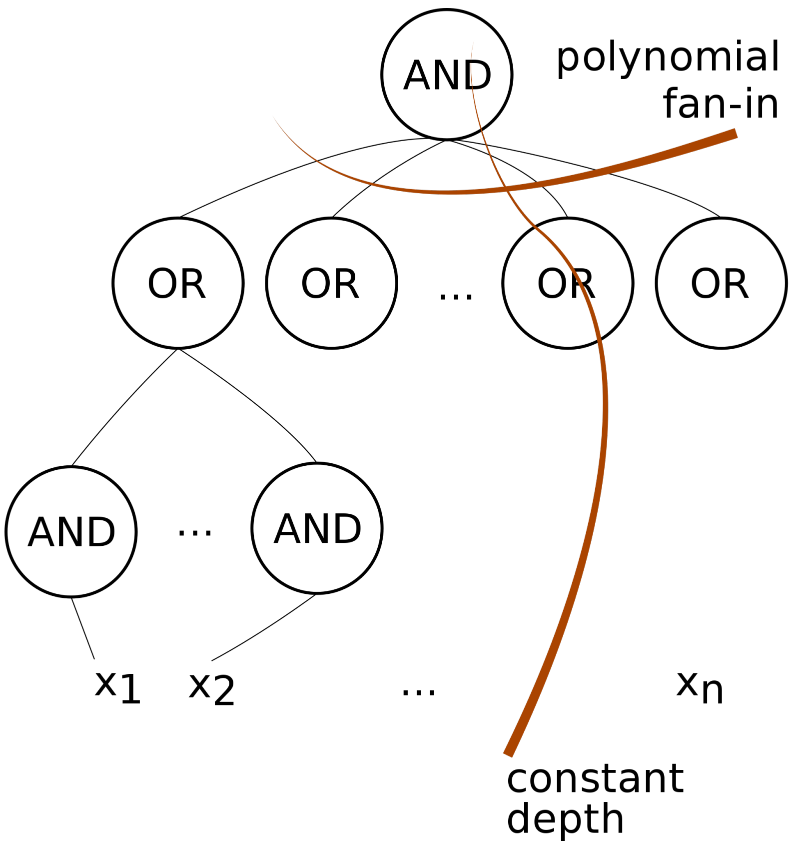
A: depends…
❌ SFE needs interaction
✅ But SFE gives one-time evaluation *only*.

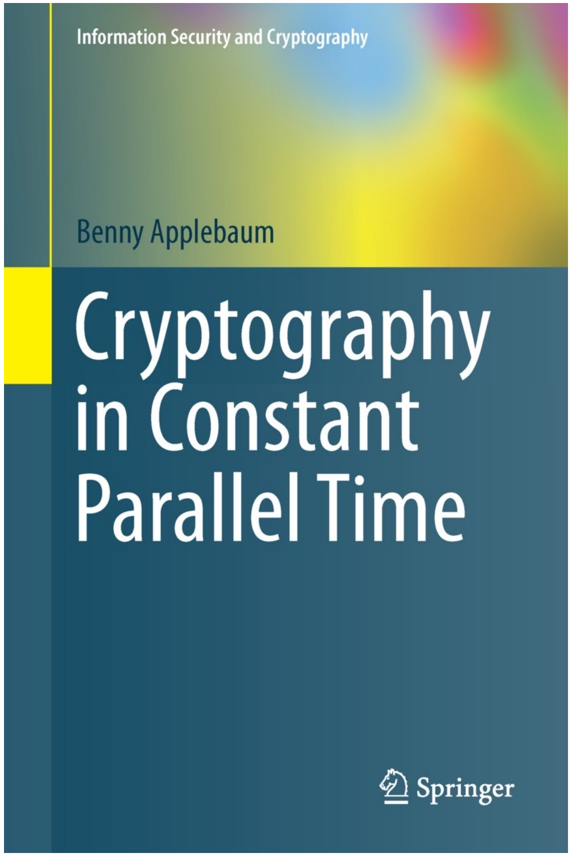
Secure 2PC from OT

- **Constant Round!!**
- Groundbreaking generic solution
- Inspired GMW'87 and more
- **Beyond secure computation**
  - Computing on encrypted data
  - Secure function evaluation
  - Parallel cryptography
  - …
  - Garbling as **Randomized Encoding of functions** [IK'00,IK'02,AIK'04,AIK'06]…

Parallel Cryptography

Can we do super-fast cryptography?





Complexity of the 2-party solution

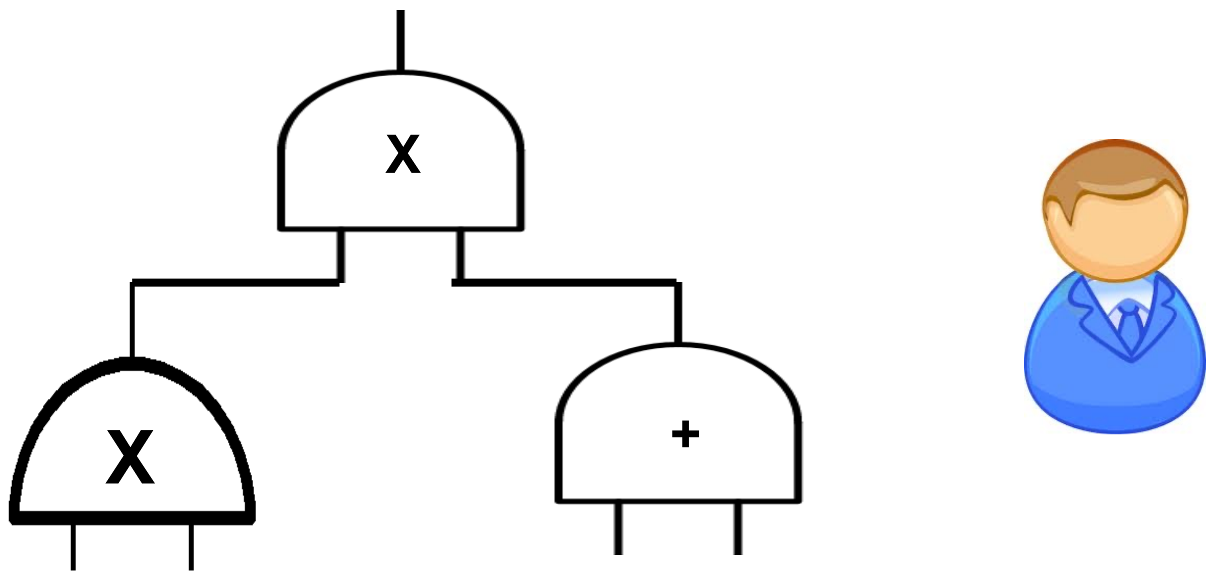
f computed by circuit C

$$C(x, y) \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$$

2PC efficiency	GMW'87	Garbled Circuit
# OT	$O(\# \wedge)$	$O(n \cdot \lambda)$
# Rounds	$\wedge$ -depth	$O(1)$
# Comm	$O(\# \wedge \cdot \lambda + m)$	$O(\lambda \cdot \# \wedge)$

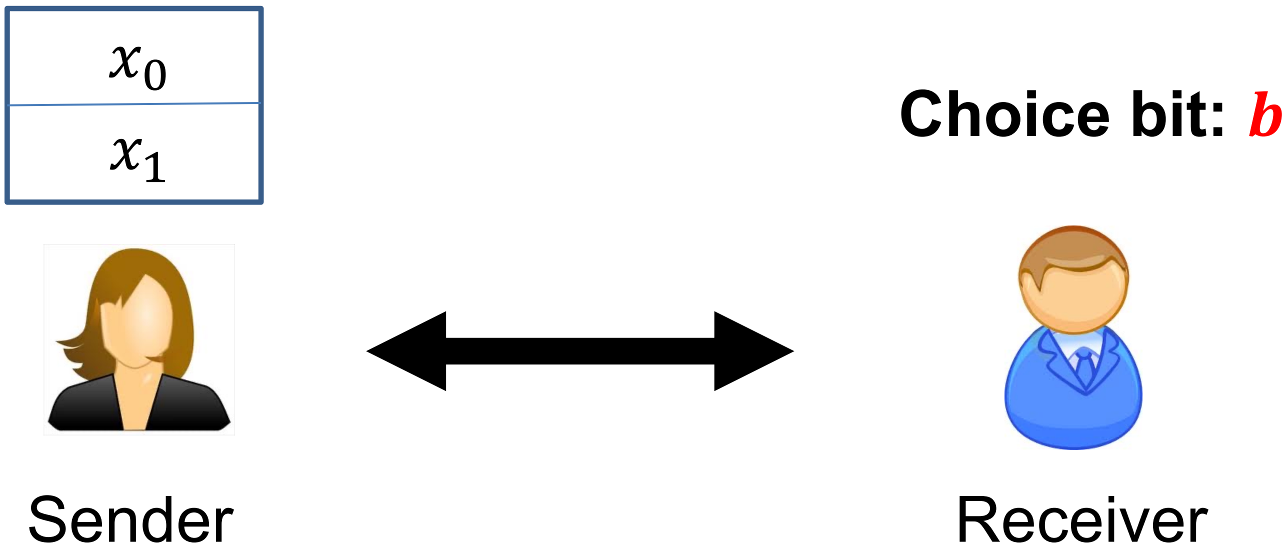
How to Compute Arbitrary Functions

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



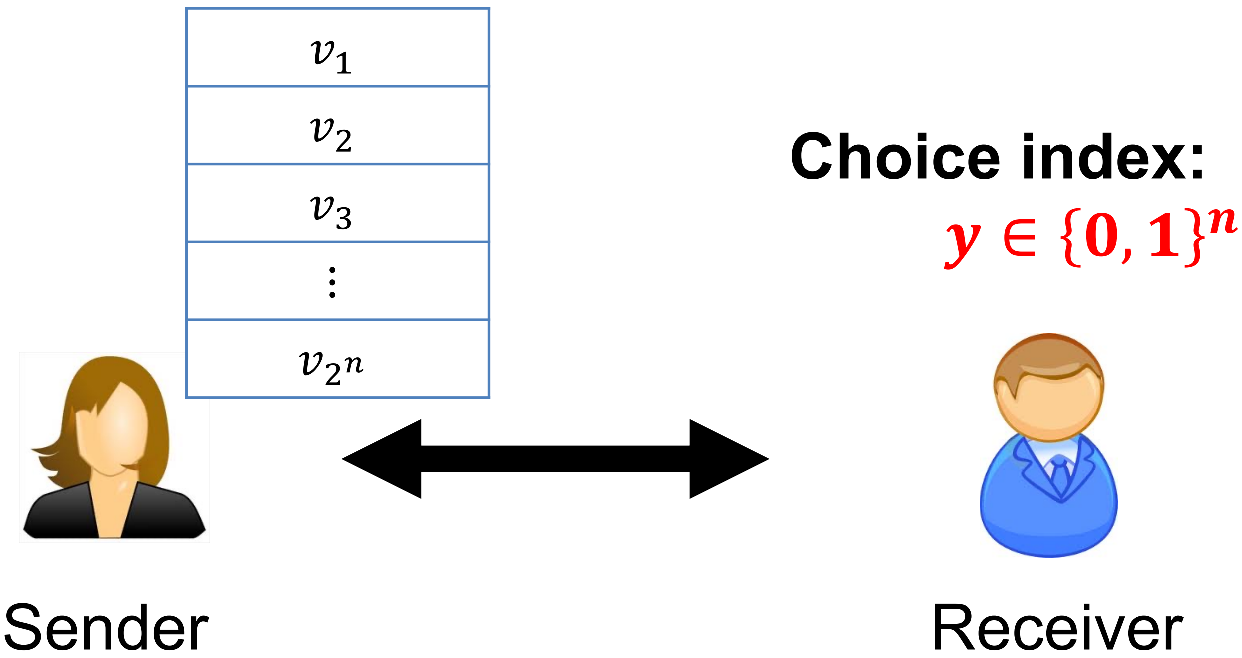
Goal: Compute every gate without knowing what the inputs and outputs are

Tool: 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.

What if we have 1-out-of-2^n OT?



- Sender holds 2^n bits

= PIR except we also require that receiver learns nothing but x_y.

- Receiver should learn x_y, sender should learn nothing.

OT on Truth table?

$f(x, \mathbf{0}^n)$
$f(x, \mathbf{0}^{n-1}\mathbf{1})$
$f(x, \mathbf{0}^{n-2}\mathbf{10})$
$\vdots$
$f(x, \mathbf{1}^n)$

Input: $x \in \{0,1\}^n$

Alice

↔

Input: $y \in \{0,1\}^n$

Bob

- itself is already inefficient!

Gate-by-gate on circuit of f!

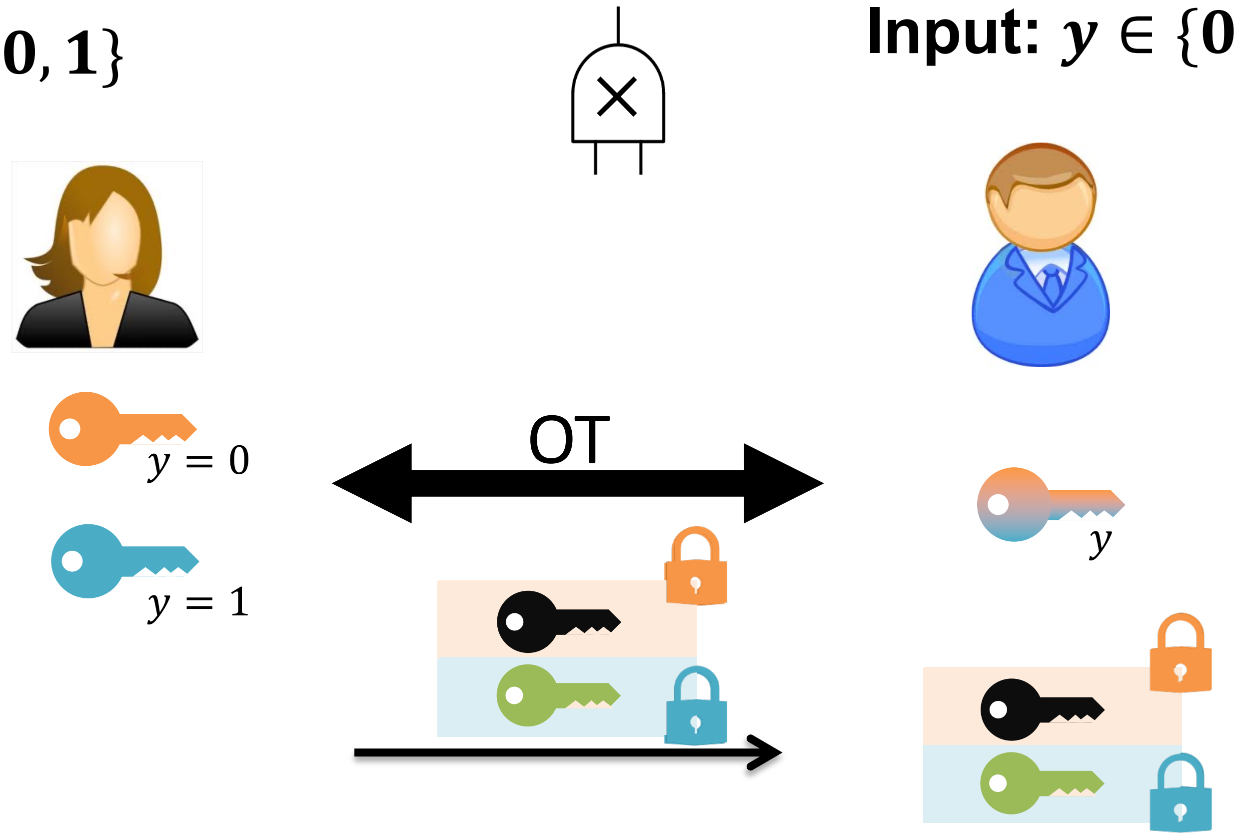
Instead of deriving the OT-by-OT protocol of GMW'87…

Fun with Lockboxes

Physical 2PC

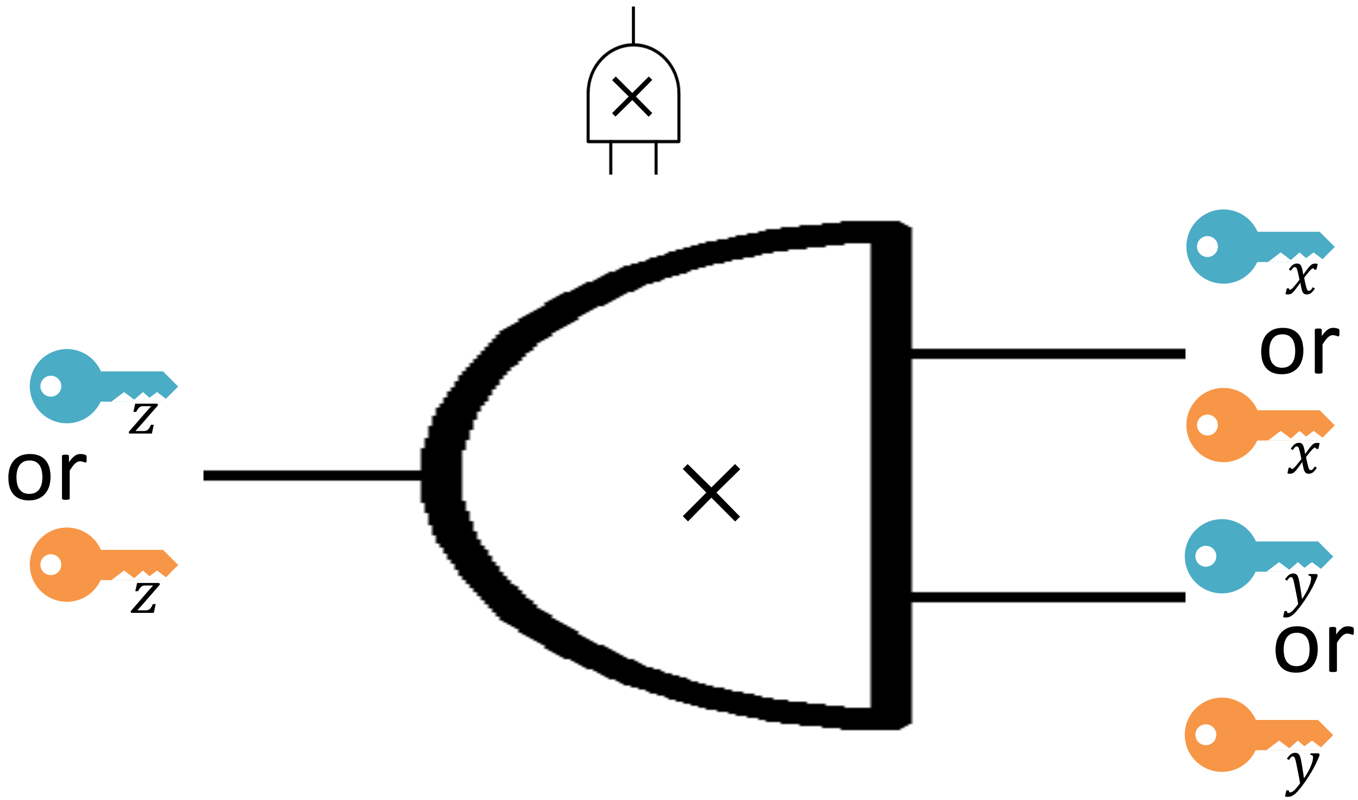
Input: $x \in \{0, 1\}$

Input: $y \in \{0, 1\}$



Possession of or is information!

Physical 2PC

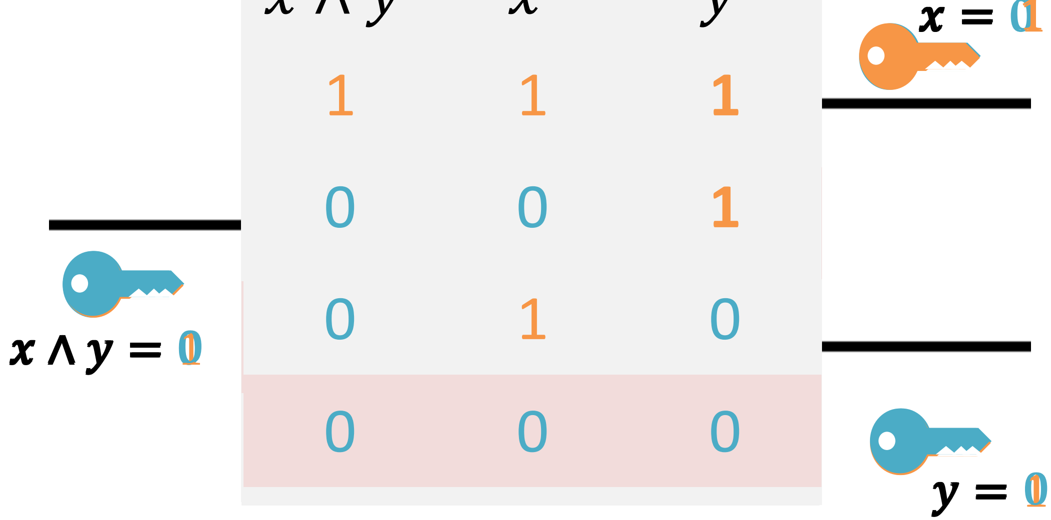


•

•

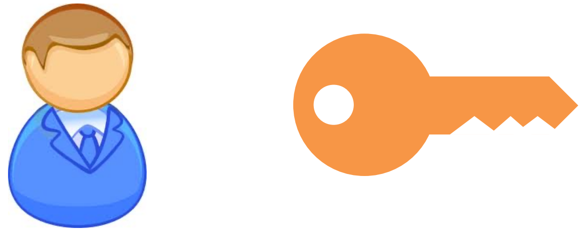
Blue/orange means 1/0, but keys on different wires, even with same colors, are different

Idea

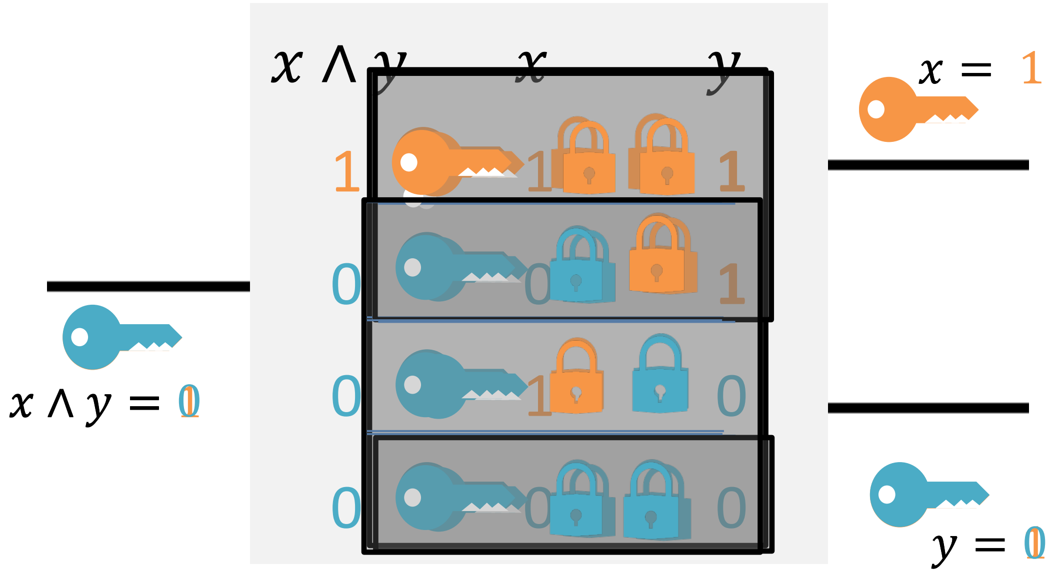


$x \wedge y$	x	y
1	1	1
0	0	1
0	1	0
0	0	0

Key Invariant: For each wire w of the circuit, generate a pair of keys k_w^0, k_w^1. The possession of $k_w^b \Leftrightarrow$ the value carried on the wire is b.

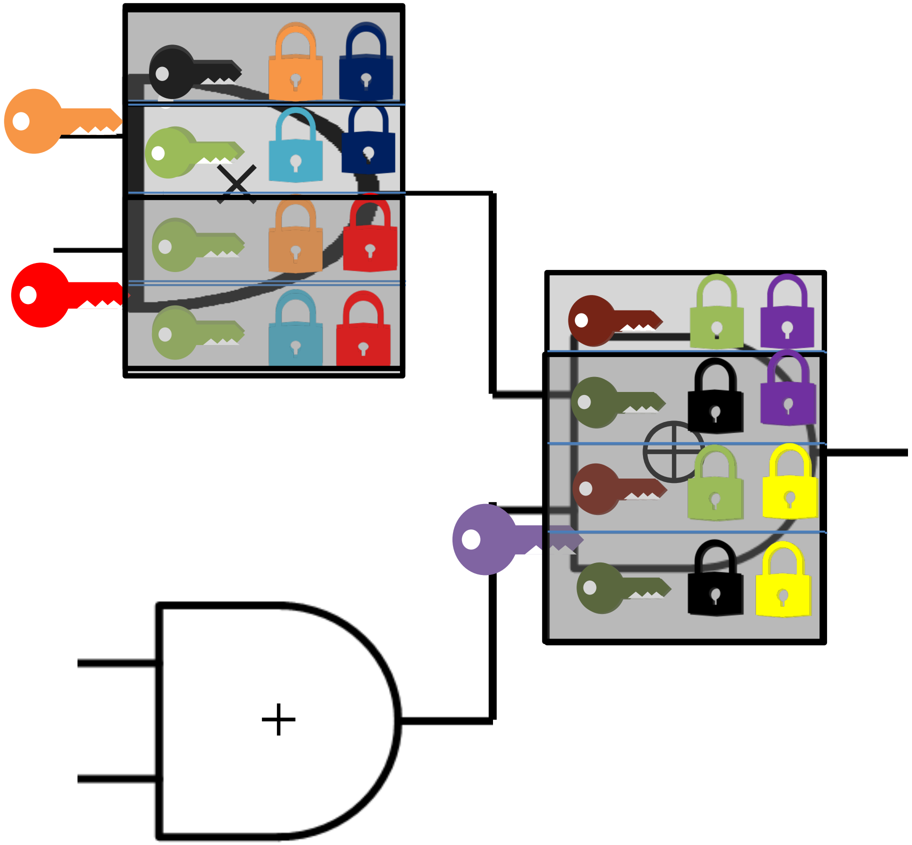


Idea



Key Invariant: For each wire w of the circuit, generate a pair of keys k_w^0, k_w^1. The possession of $k_w^b \Leftrightarrow$ the bit b is carried on w.

Garbled Evaluation

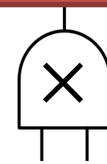


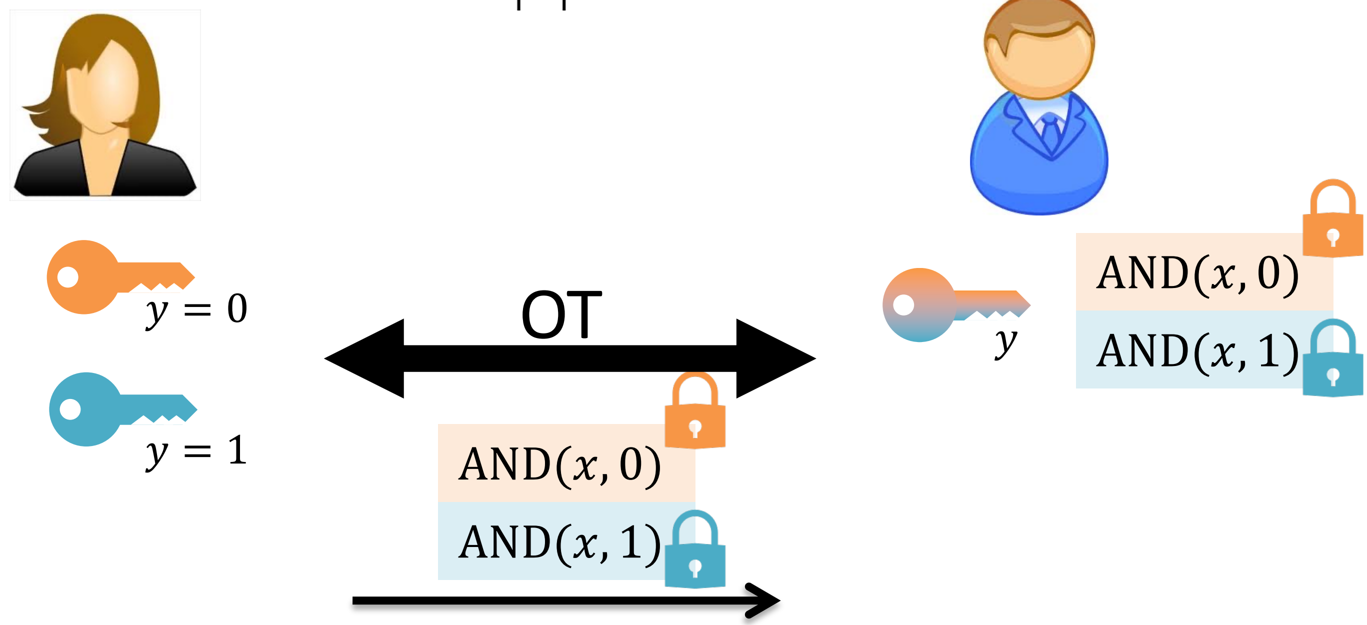
Crypto Lockboxes

Crypto 2PC

Input: $x \in \{0,1\}^n$

Input: $y \in \{0,1\}\%$





- Alice puts results in IND-CPA encryption
- Bob gets key via some crypto OT mechanism
- Bob tries to open both boxes using the key he received.

What about last point?

Tool: *Special* CPA Encryption

CPA-secure secret-key encryption $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ that satisfies

1. **Elusive range** --- Let $k \leftarrow_R \mathrm{Gen}(1^{\mathrm{n}})$. Any p.p.t. adversary A cannot generate ciphertext encrypted under k, w/o k.
2. **Efficiently verifiable range** --- there exists an algo $\mathrm{Check}(1^n, k, c)$ that checks if c is encrypted under k.

Tool: *Special* CPA Encryption

1. **Elusive range** --- Let $k \leftarrow_R \mathrm{Gen}(1^{\mathrm{n}})$. Any p.p.t. adversary A cannot generate ciphertext encrypted under k, w/o k.
2. **Efficiently verifiable range** --- there exists an algo $\mathrm{Check}(1^n, k, c)$ that checks if c is encrypted under k.

Construction:

Let $F_k: \{0,1\}^n \to \{0,1\}^{2n}$ be a PRF,

$$\mathrm{E}_k(m; r) := F_k(r) \oplus (m||0^n).$$

Crypto Lockboxes

x	y	z
1	1	1
0	1	0
1	0	0
0	0	0

Garbled gate

$$\mathrm{E}_{k_x^1}(\mathrm{E}_{k_y^1}(k_z^1))$$
$$\mathrm{E}_{k_x^0}(\mathrm{E}_{k_y^1}(k_z^0))$$
$$\mathrm{E}_{k_x^1}(\mathrm{E}_{k_y^0}(k_z^0))$$
$$\mathrm{E}_{k_x^0}(\mathrm{E}_{k_y^0}(k_z^0))$$

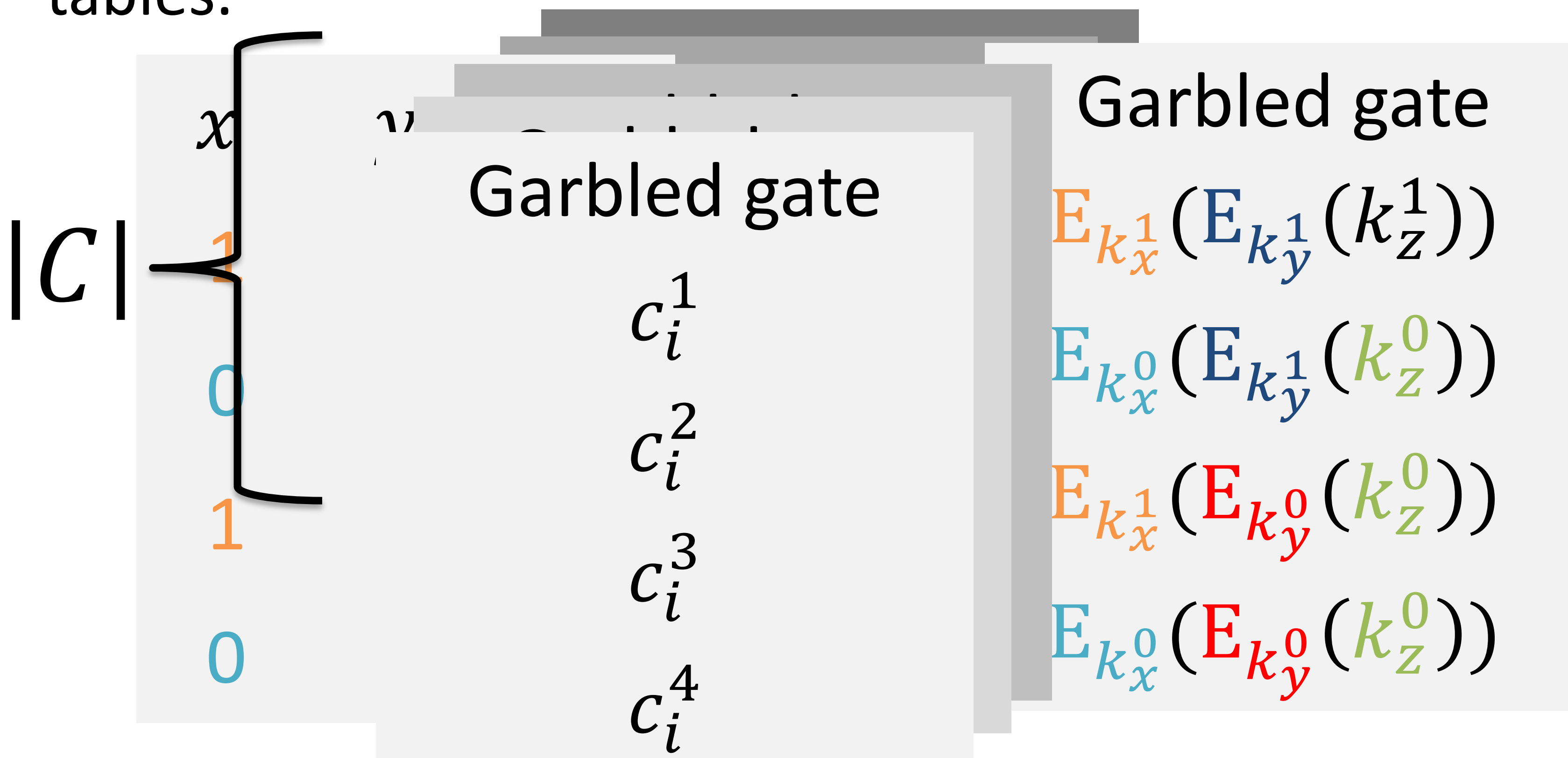
Efficiently verifiable range

$\mathrm{Check}(1^n, k, c)$ checks if c is encrypted under k.

Protocol Sketch

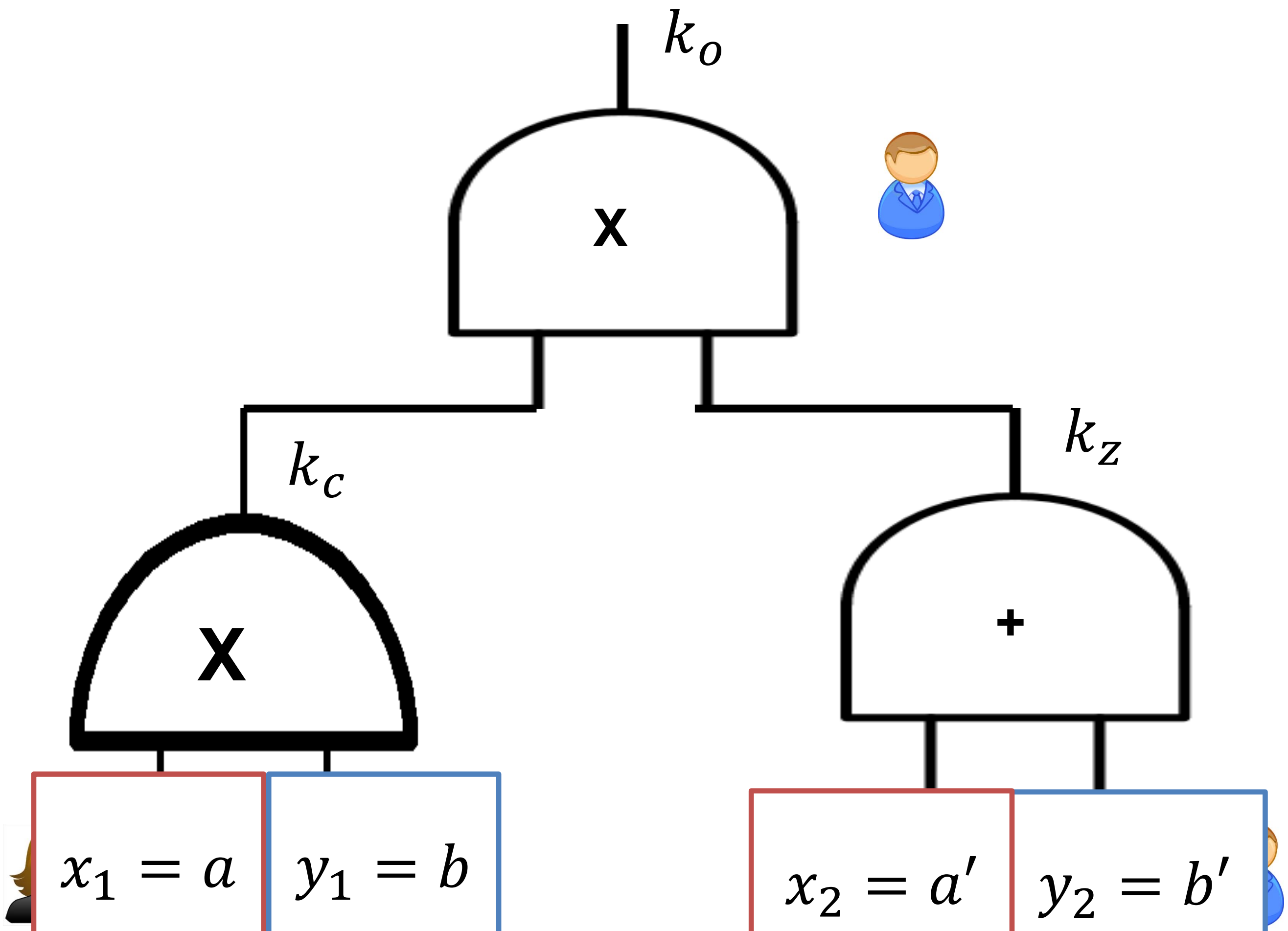
Yao's protocol: Garbling

- *Keys generation:* For each wire w of C, Alice generates a pair of keys k_w^0, k_w^1.
- *Gate Garbling:* For each gate $z \leftarrow G(x, y)$, compute the tables.



Yao's protocol: Evaluator Bob

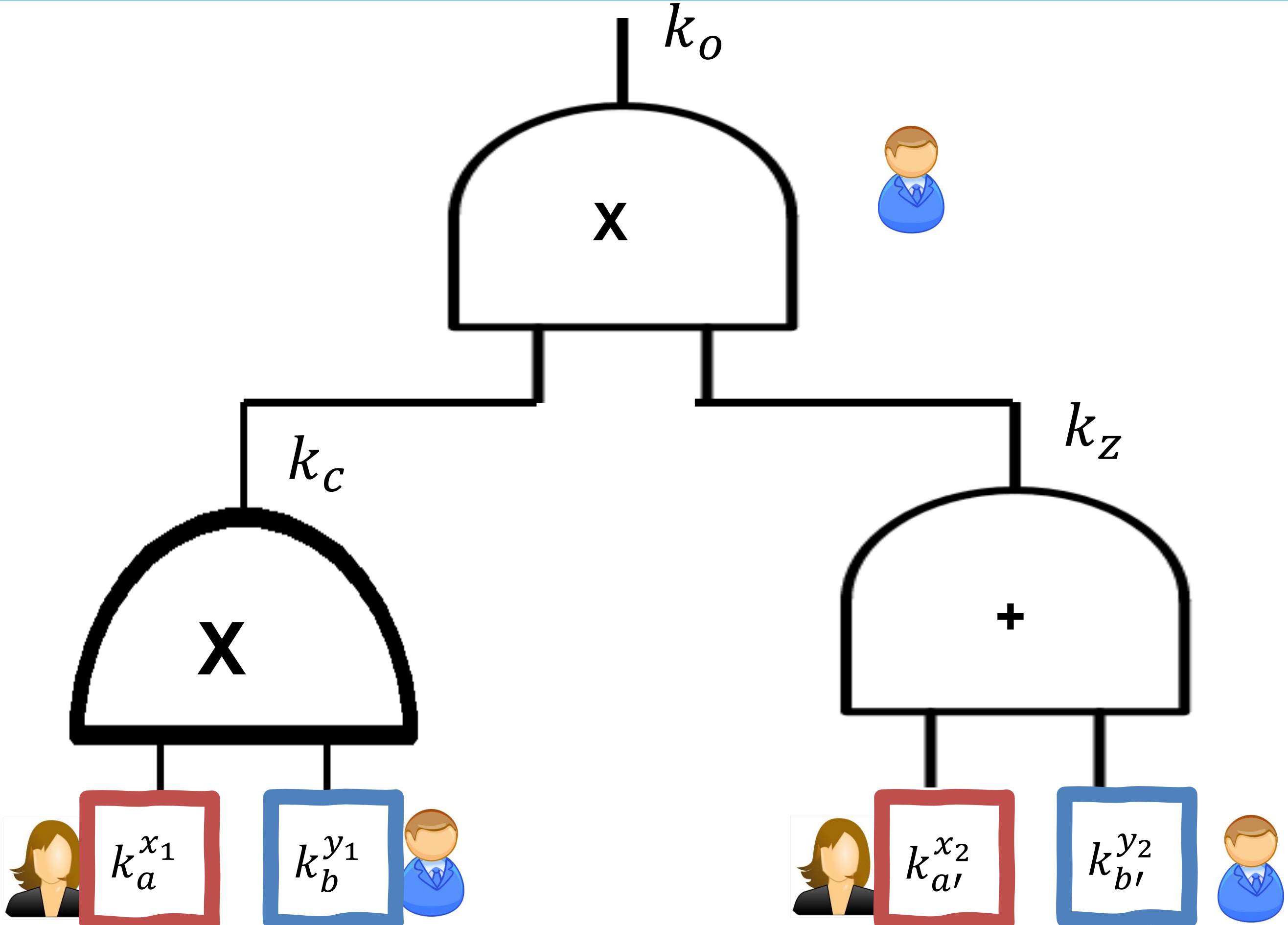
Key Invariant: For each wire w of the circuit, Bob can obtain exactly one of the two keys from k_w^0, k_w^1.



Yao's protocol: Evaluator Bob

Base case:

- Alice's input $x_i \in \{0,1\}$: Alice send the correct key $k_i^{x_i}$.
- Bob's input $y_i \in \{0,1\}$: they runs OT on $((k_i^0, k_i^1), y_i)$.



Yao's protocol: Evaluator Bob

Inductive step:

Assume: Bob has one key for each input wire

- Bob can get exactly one key for the output wire by trying all four ciph[…]

$k_o^{(x_1+x_2)(x_1\wedge x_2)}$

Efficiently verifiable range

$\mathrm{Check}(1^n, k, c) \in \{0,1\}$ checks if c is encrypted under k.

Evaluating One Gate

Inductive step:

Oops..

This procedure, as-is, is actually insecure.

trying all four ciphertexts.

Recall:

Given $\boldsymbol{k_x^{b_x}, k_y^{b_y}}$,

- Try all four rows to obtain $k_z^{b_x \wedge b_y}$

Garbled gate

$\mathrm{E}_{k_x^1}(\mathrm{E}_{k_y^1}(k_z^1))$

$\mathrm{E}_{k_x^0}(\mathrm{E}_{k_y^1}(k_z^0))$

$\mathrm{E}_{k_x^1}(\mathrm{E}_{k_y^0}(k_z^0))$

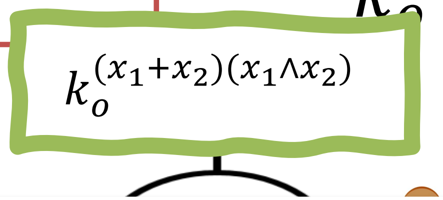
$\mathrm{E}_{k_x^0}(\mathrm{E}_{k_y^0}(k_z^0))$

Reconstructing Output

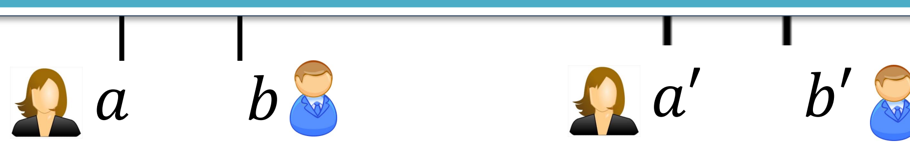
Key Invariant: For each wire of the circuit, Bob can obtain exactly one of the two keys associated with each wire.

After evaluation, Bob learns $k_o \in \{k_o^0, k_o^1\}$.

- Bob simply asks Alice if k_o is k_o^0 or k_o^1.



Can we avoid this final round?



Garbling as a Standalone Tool

Q: Difference with $i\mathcal{O}$?

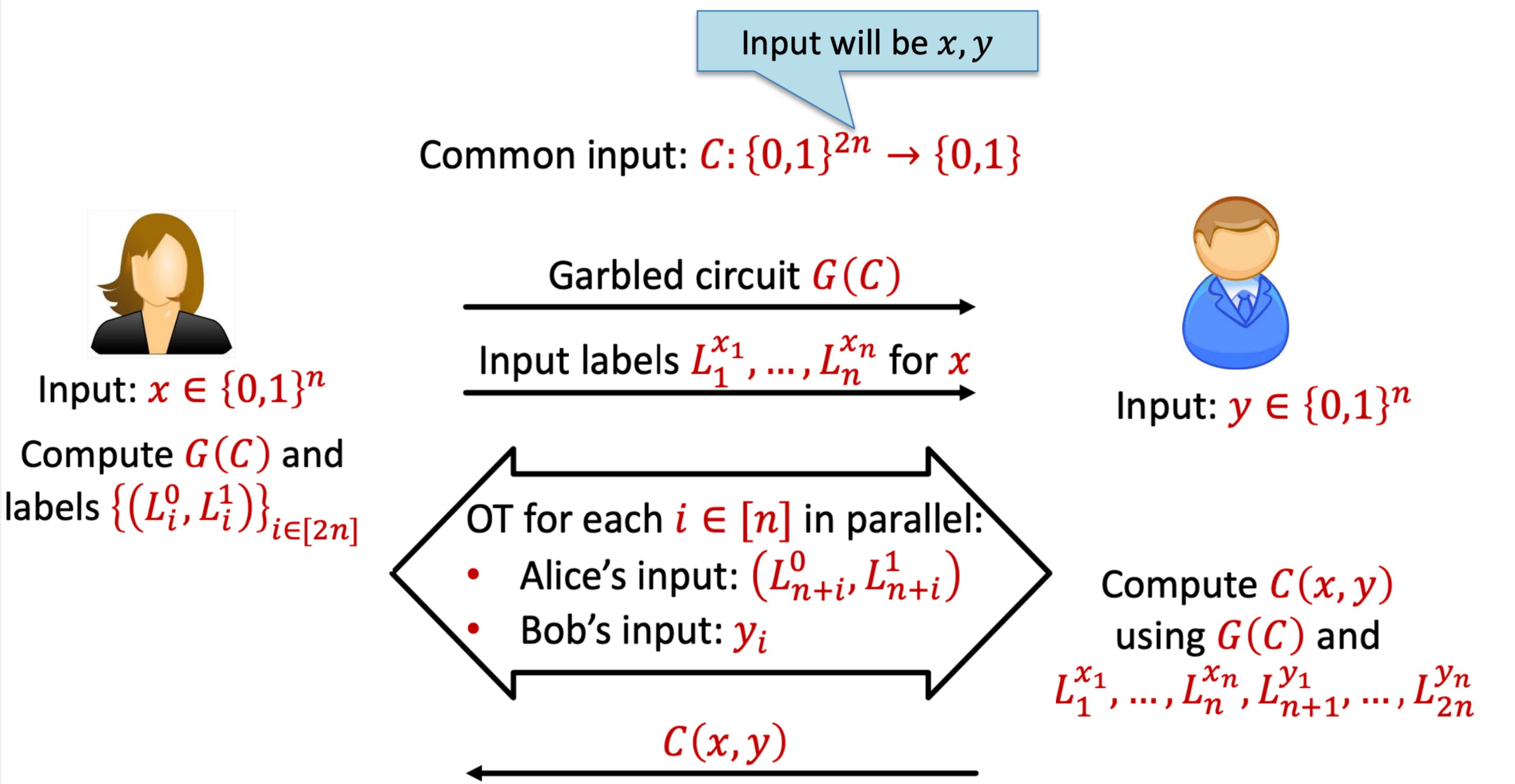
- **Input:** Boolean circuit $C: \{0,1\}^n \rightarrow \{0,1\}$
- **Output:** Garbled circuit $G(C)$ and input labels $\{(L_1^0, L_1^1), \ldots, (L_n^0, L_n^1)\}$

For example, for $x = 010$, labels are L_1^0, L_2^1, L_3^0

Goal: Given $G(C)$ and $L_1^{x_1}, \ldots, L_n^{x_n}$

- It is possible to compute $C(x_1 \cdots x_n)$
- It is not possible to learn any additional information other than size of circuit or input

2PC Using Garbled Circuits

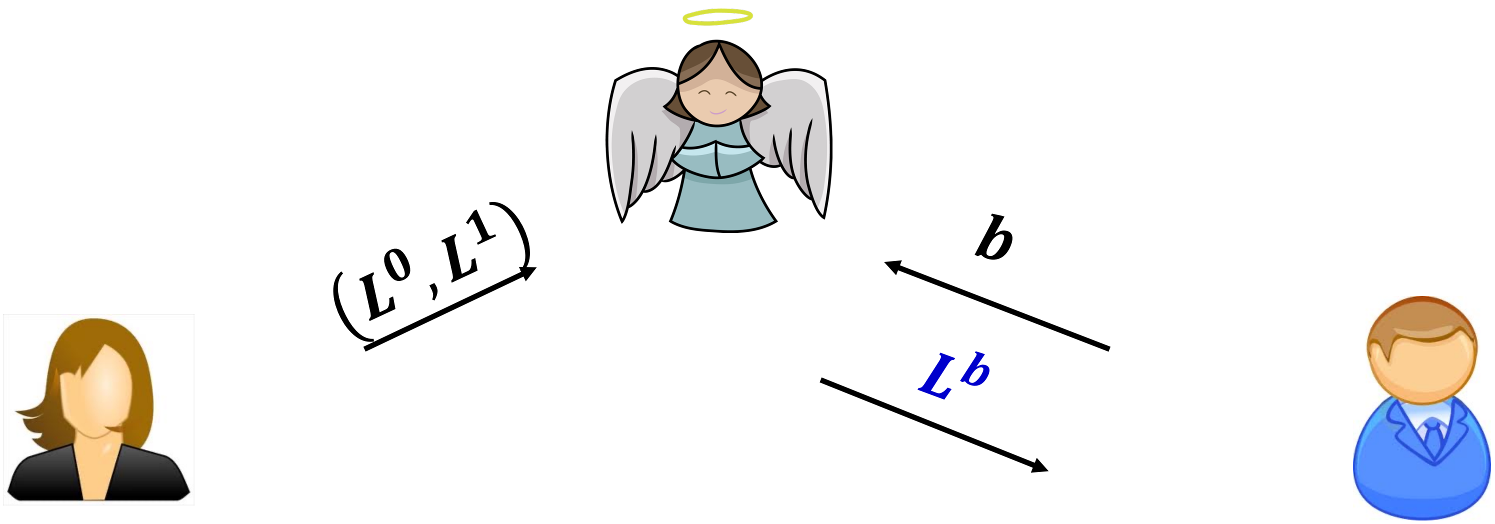


Simulation Proof Sketch

Simulating Alice

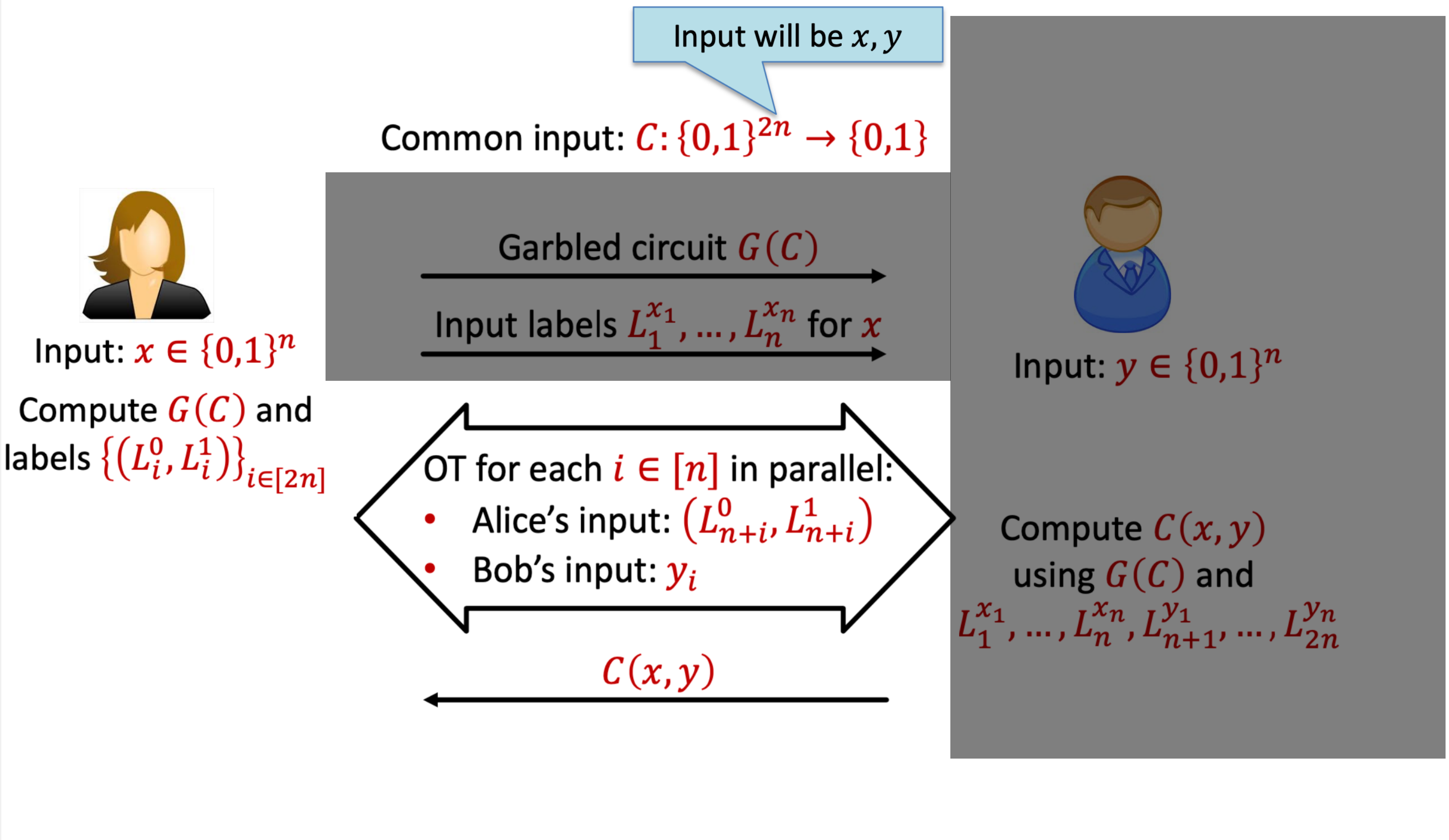
Imagine that the parties have access to an OT angel

- Implemented by secure simulatable OT.



Simulating Alice

Imagine that the parties have access to an OT angel.



Simulating Alice

Alice's View
- OT transcripts
- $C(x,y)$

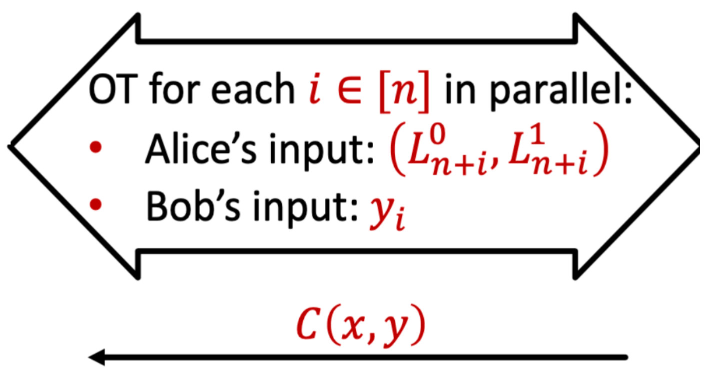
$\approx$

$\text{Sim}\big(C, x, f(x,y)\big)$:

Output

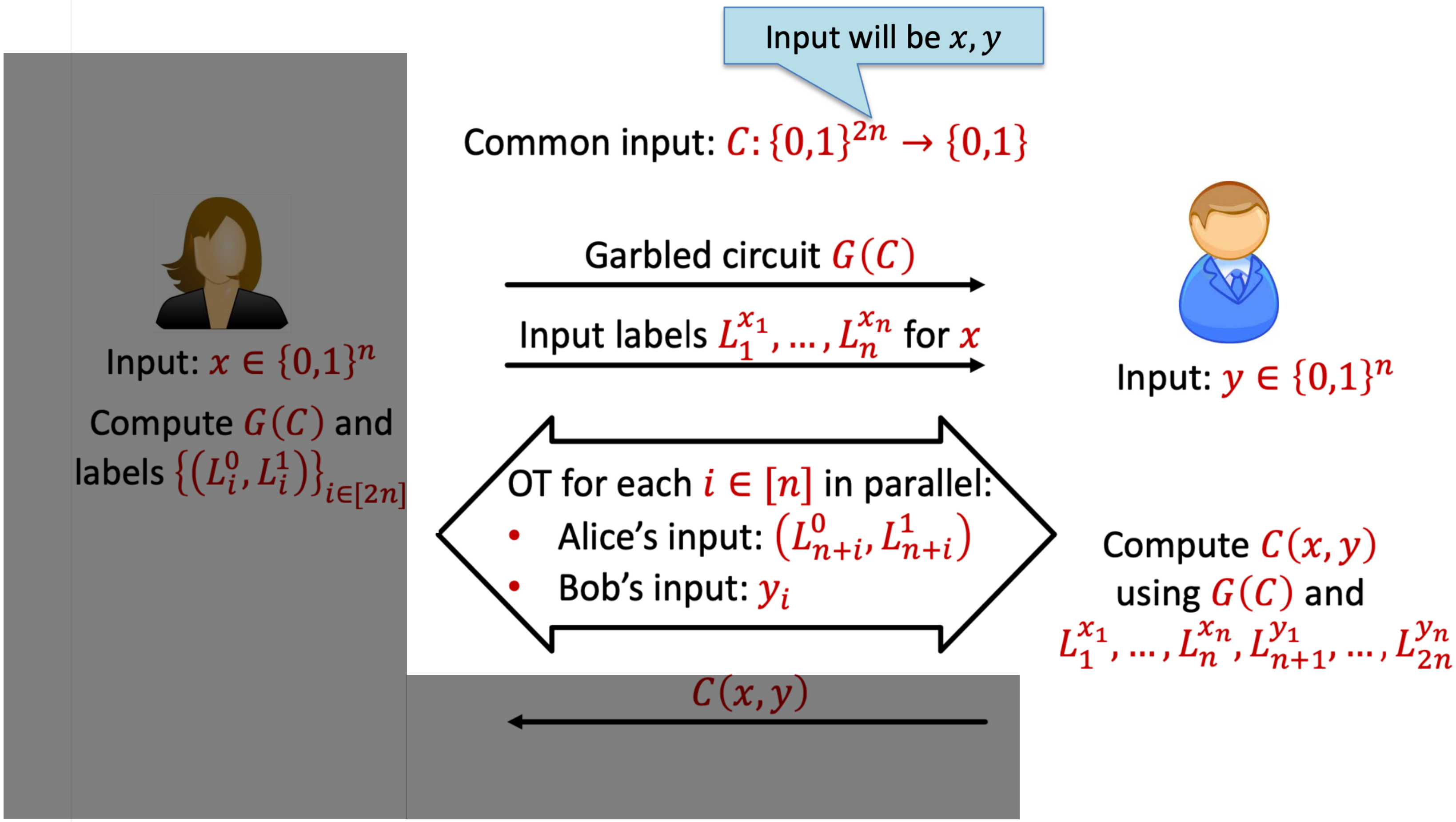
$\{\{\text{Sim}^{A}_{\text{OT}}\big(L^{0}_{n+i}, L^{1}_{n+i}\big)\}_i$

$f(x,y)\}$



Simulating Bob

Imagine that the parties have access to an OT angel.



Simulating Bob

OT Simulation:

$$\mathrm{View}_{\mathrm{OT}}^{\mathrm{B}} \approx \mathrm{Sim}_{\mathrm{OT}}^{\mathrm{B}}\left(1^n, y_i, L_{n+i}^{y_i}\right).$$

OT for each $i \in [n]$ in parallel:

- Alice's input: $\left(L_{n+i}^{0}, L_{n+i}^{1}\right)$
- Bob's input: y_i

Assume we already simulated L_{n+i}^{B} with the correct distribution.

Simulating Bob

Input will be x, y

Common input: $C: \{0,1\}^{2n} \rightarrow \{0,1\}$

Input: $x \in \{0,1\}^n$

Compute $G(C)$ and labels $\{(L_i^0, L_i^1)\}_{i \in [2n]}$

Garbled circuit $G(C)$ →

Input labels $L_1^{x_1}, \ldots, L_n^{x_n}$ for x →

OT for each $i \in [n]$ in parallel:

- Alice's input: (L_{n+i}^0, L_{n+i}^1)
- Bob's input: y_i

← $C(x, y)$

Input: $y \in \{0,1\}^n$

Compute $C(x, y)$ using $G(C)$ and $L_1^{x_1}, \ldots, L_n^{x_n}, L_{n+1}^{y_1}, \ldots, L_{2n}^{y_n}$

Simulating Bob

Garbled circuit $G(C)$

Input labels $L_1^{x_1}, \ldots, L_n^{x_n}$ for x

Input: $y \in \{0,1\}^n$

Step 1: Generate Dummy Labels

- *Label generation:* For each wire w of C, generates a pair of keys L_w^0, L_w^1.
- *Label simulation:* For all input wire i, let $\widetilde{L_i^{x_i}} := L_i^0$.

Sanity Check:
This replacement is fine because keys are randomly generated.

Step 2.a: Simulate Fake Gates

- *Garbled gate simulation:* Replace intermediate ciphertexts with junks.

$$\sigma \cdot \begin{array}{c}\text{Garbled gate}\\ \mathrm{E}_{L_x^1}(\mathrm{E}_{L_y^1}(L_z^1))\\ \mathrm{E}_{L_x^0}(\mathrm{E}_{L_y^1}(L_z^0))\\ \mathrm{E}_{L_x^1}(\mathrm{E}_{L_y^0}(L_z^0))\\ \mathrm{E}_{L_x^0}(\mathrm{E}_{L_y^0}(L_z^0))\end{array} \approx \tau \cdot \begin{array}{c}\text{Garbled gate}\\ \mathrm{E}_{L_x^1}(\mathrm{E}_{L_y^1}(L_z^0))\\ \mathrm{E}_{L_x^0}(\mathrm{E}_{L_y^1}(L_z^0))\\ \mathrm{E}_{L_x^1}(\mathrm{E}_{L_y^0}(L_z^0))\\ \mathrm{E}_{L_x^0}(\mathrm{E}_{L_y^0}(L_z^0))\end{array}$$

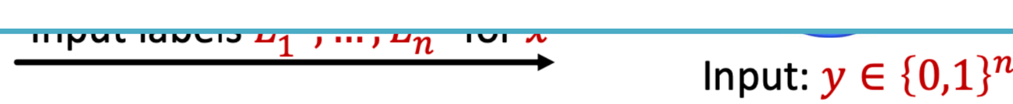
- the rows are randomly permuted ($\sigma, \tau \in Perm([4])$)
- only a random row can be decrypted
- the junk entries are w.h.p. non-decryptable.

Step 2.b: Simulate Output

- *Generate the following decoding table*

Output label	Decoded result
L_o^0	$C(x,y)$
L_o^1	$1 - C(x,y)$

Sanity Check: This is fine because the label L_o^0 might as well be encoding 1.



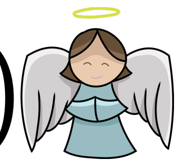
Simulating Bob

Bob's View

- Wire labels
- Garbled tables
- Final decoding table
- OT transcripts

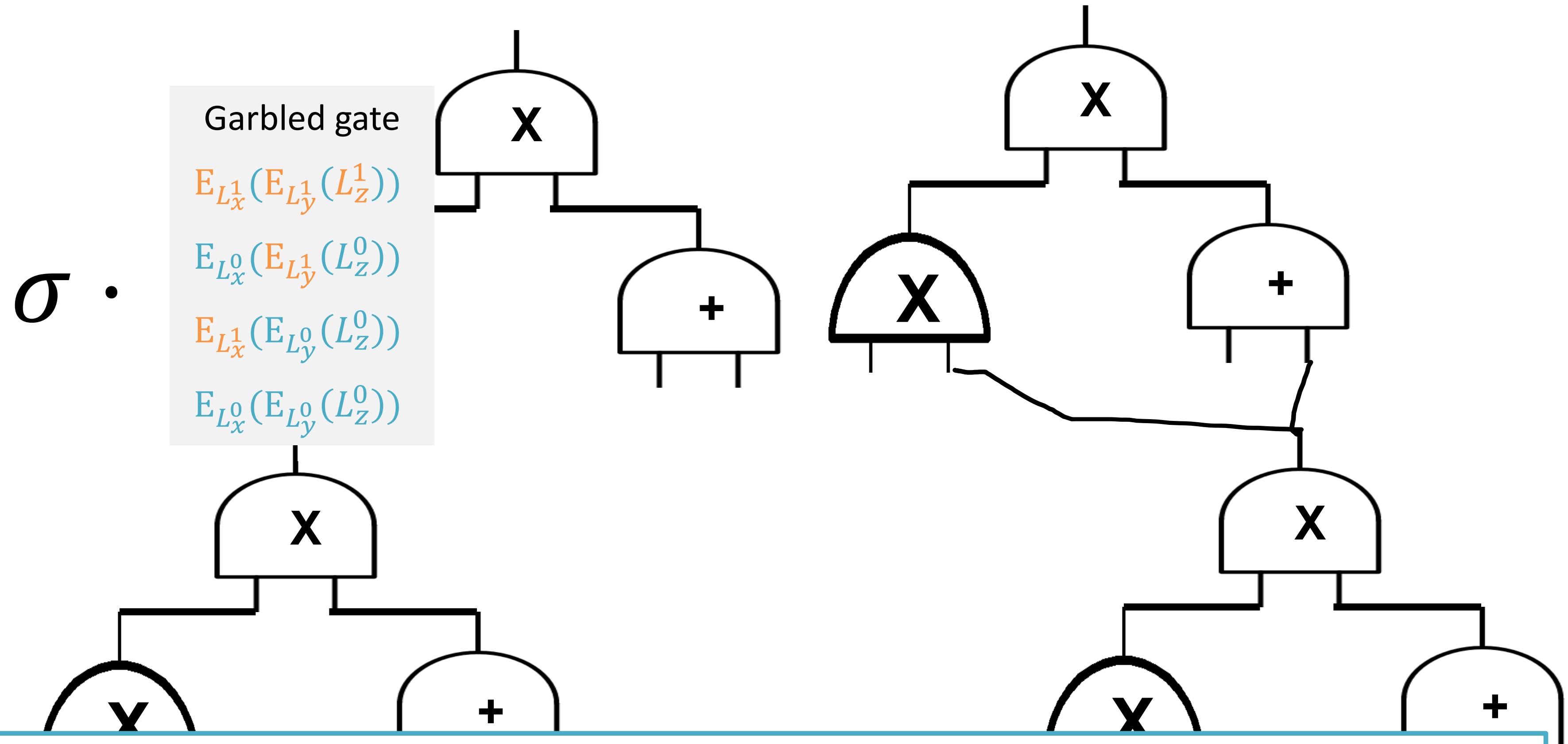
$\approx$

$\mathrm{Sim}\big(C, y, f(x, y)\big)$:

- ➢ Simulate labels
- ➢ $\mathrm{Sim}_{\mathrm{OT}}^{\mathrm{B}}\left(L_y^{y_i}\right)$ 
- ➢ Simulate garbled gates
- ➢ Simulate final decoding table.

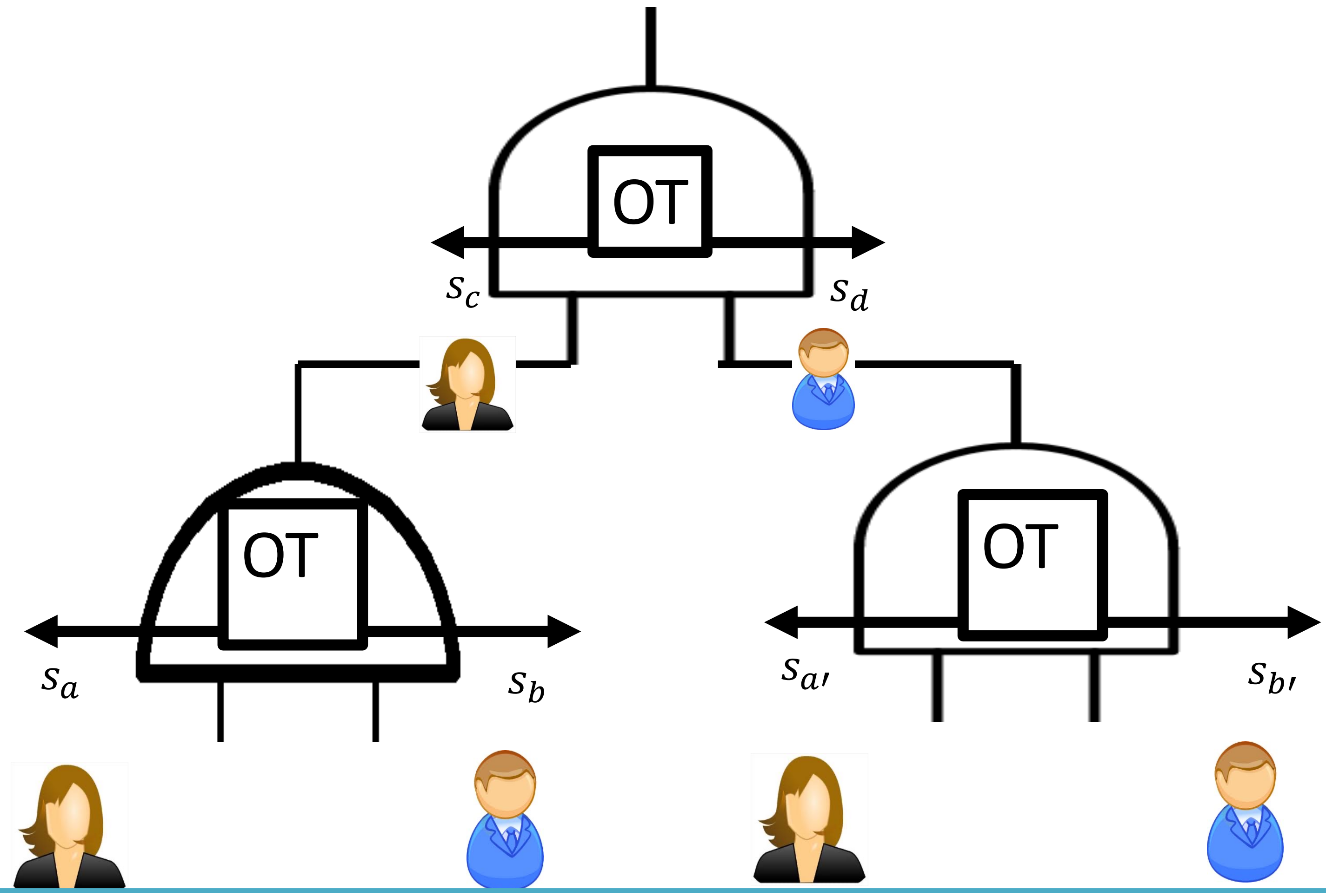
Efficiency

Garbling is parallelizable



Parallelism: Each garbled gate is computed locally (only depends on $(L_x^0, L_x^1), \left(L_y^0, L_y^1\right), (L_z^0, L_z^1)$, generated at the very beginning).

Why is GMW sequential?



Sequentiality: Input to next OT is output from previous OT.

Garbled-circuit 2PC

f computed by circuit C

$$C(x, y): \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$$

2PC efficiency	GMW'87	Garbled Circuit
# OT	$O(\# \wedge)$	$O(n \cdot \lambda)$
# Rounds	$\wedge$ -depth	$O(1)$
# Comm	$O(\# \wedge \cdot \lambda + m)$	$O(\lambda \cdot \# \wedge)$

Optimization 1: Free XOR trick

In GMW or BGW, linear (XOR) gates are free (no communication).
Can we say something for Garbled circuits?

Theorem [Kolesnikov, Schneider'08]: If we generate labels *carefully,* then there is no need to send garbled XOR tables.

Observation: do not use so much randomness.

Optimization 1: Free XOR trick

Rough intuition:

Acceptable correlations of labels:

- Pick global R, a random value hidden from evaluator
- Generate non-XOR-output wire w subject to

$$L_w^b = L_w^{1-b} \oplus R$$

Now if $z = x \oplus y$, we define $L_z = L_x \oplus L_y$.

1. $L_z^1 = L_x^0 \oplus L_y^1 = L_x^0 \oplus L_y^0 \oplus R = L_x^1 \oplus L_y^0$.
2. $L^0 = L^1 \oplus L^1 = L^0 \oplus L^0$

Observation: do not use so much randomness.

Optimization 2: Half-Gates

Half-gate trick [Zahur, Rosulek, Evans'15]: Keeping XOR gates free, AND gate can be 2 ct each.

	size per gate		calls to H per gate			
			generator		evaluator	
technique	XOR	AND	XOR	AND	XOR	AND
classical [31]	4	4	4	4	4	4
point-permute [3]	4	4	4	4	1	1
row reduction (GRR3) [27]	3	3	4	4	1	1
row reduction (GRR2) [28]	2	2	4	4	1	1
free XOR + GRR3 [20]	0	3	0	4	0	1
fleXOR [19]	$\{0, 1, 2\}$	2	$\{0, 2, 4\}$	4	$\{0, 1, 2\}$	1
half gates [this work]	0	2	0	4	0	2

Table 1. Optimizations of garbled circuits. Size is number of "ciphertexts" (multiples of k bits).

Credit: Table taken from proceedings version of [ZRE'15].

Optimization 3: Beyond Half-Gates

Slicing & Dicing [Rosulek, Roy'21]: Keeping XOR gates free, AND gate can be 1.5 ct plus 5 bits each.

scheme	GC size (κ bits / gate) AND	XOR	calls to H per gate: garbler AND	garbler XOR	evaluator AND	evaluator XOR	assump.
unoptimized textbook Yao	8	8	4	4	2.5	2.5	PRF
Yao + point-permute [BMR90]	4	4	4	4	1	1	PRF
$4 \to 3$ row reduction [NPS99]	3	3	4	4	1	1	PRF
$4 \to 2$ row reduction [PSSW09]	2	2	4	4	1	1	PRF
free-XOR [KS08]	3	0	4	0	1	0	CCR
fleXOR [KMR14]	2	$\{0, 1, 2\}$	4	$\{0, 2, 4\}$	1	$\{0, 1, 2\}$	CCR
half-gates [ZRE15]	2	0	4	0	2	0	CCR
[GLNP15]	2	1	4	3	2	1.5	PRF
ours	**1.5**	**0**	≤ 6	0	≤ 3	0	CCR

Credit: Table taken from proceedings version of [RR'21].

Malicious Alice

What can a malicious garbler do?

Simple Generic Defense
Cut-and-choose

Rough sketch:

- Alice commits to q garbled gates and the randomness in generating them.
- Bob opens all but one instances, including all the labels, and check that gates are garbled correctly; if not, abort.
- Use the unopened GC to compute.

Note: Use of commitment is crucial:

How do we get soundness error: $2^{\Omega(q)}$?

Malicious Bob

What can a malicious evaluator do?

Next class:

Quantum Cryptography