RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

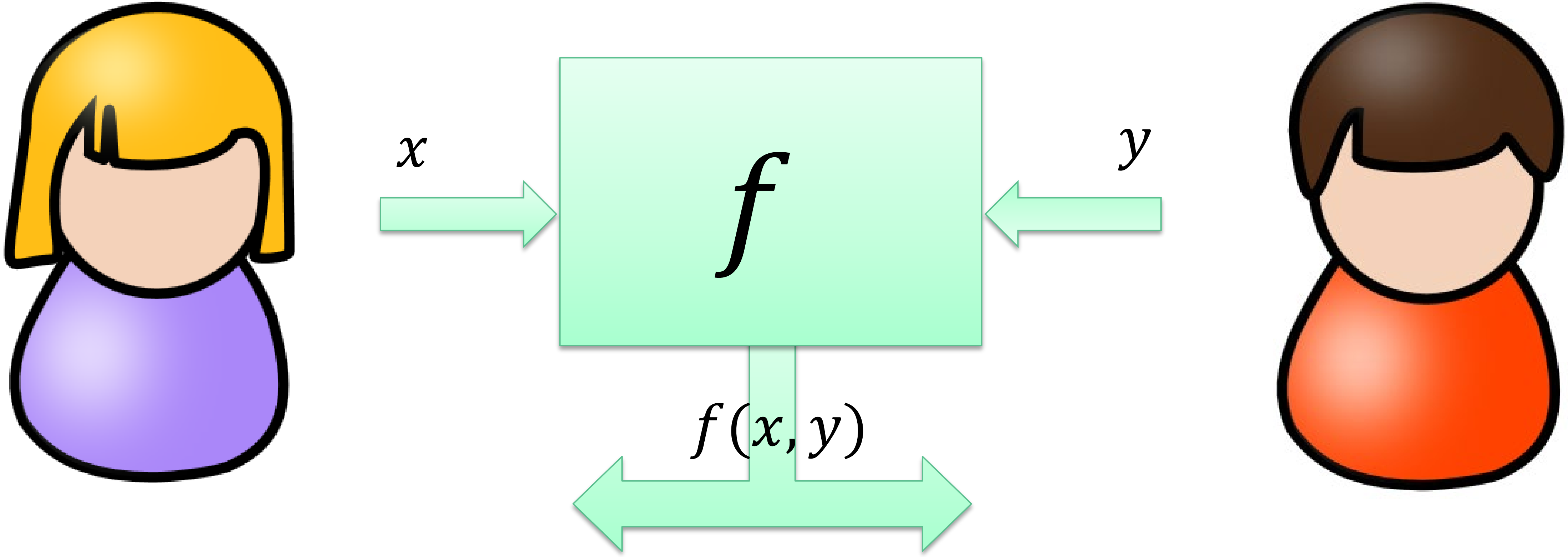
SESSION ID: **CRYP-F02**

Cut-and-Choose for Garbled RAM

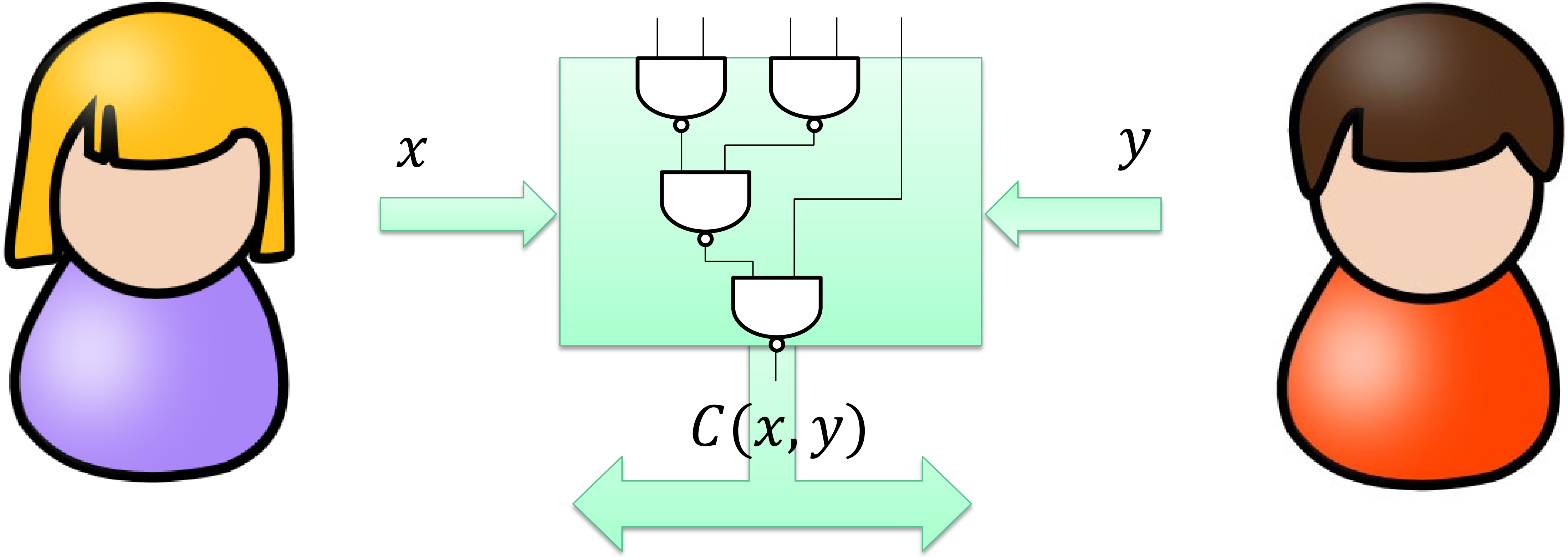
Peihan Miao

Research Scientist
Visa Research

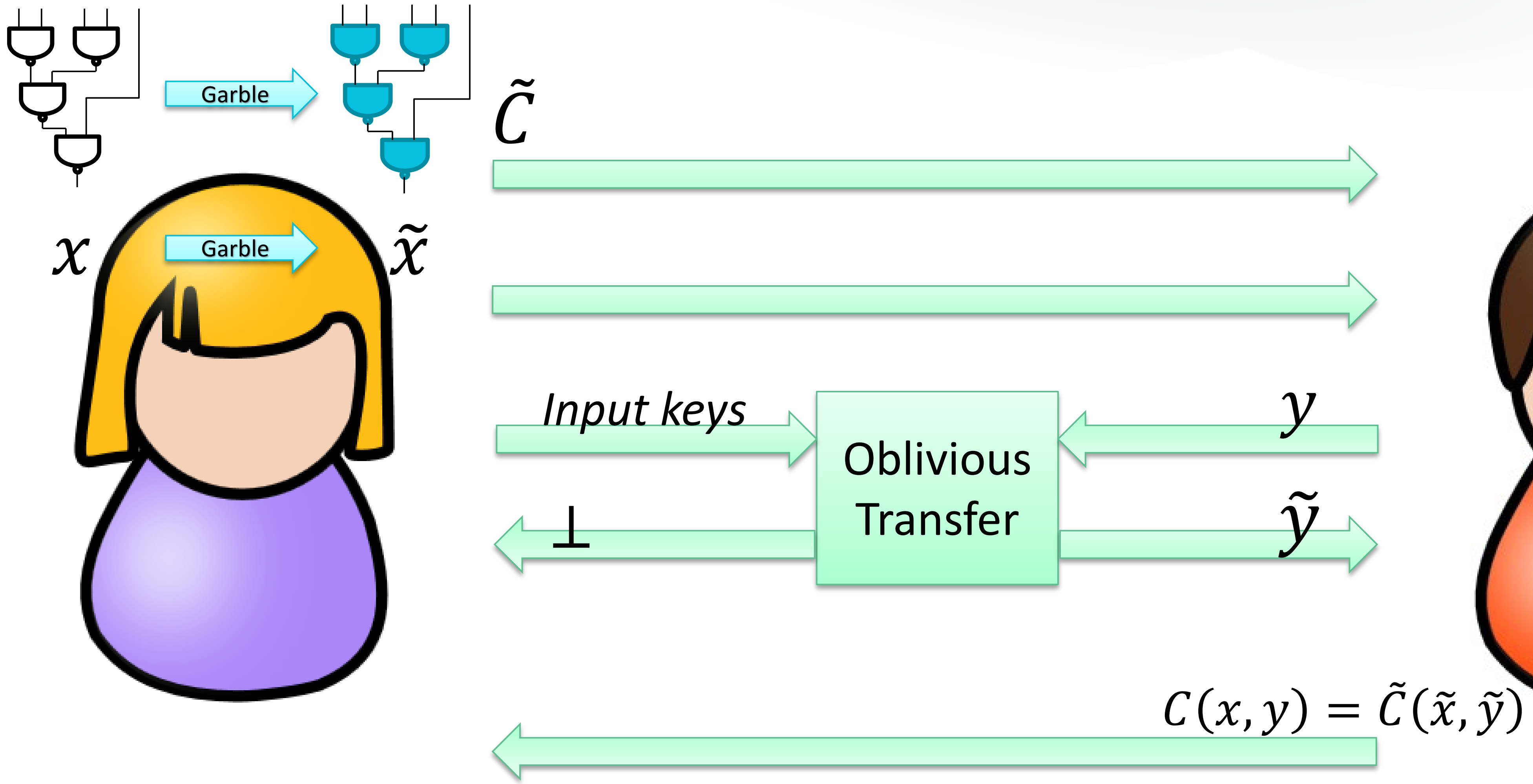
Secure Two-Party Computation



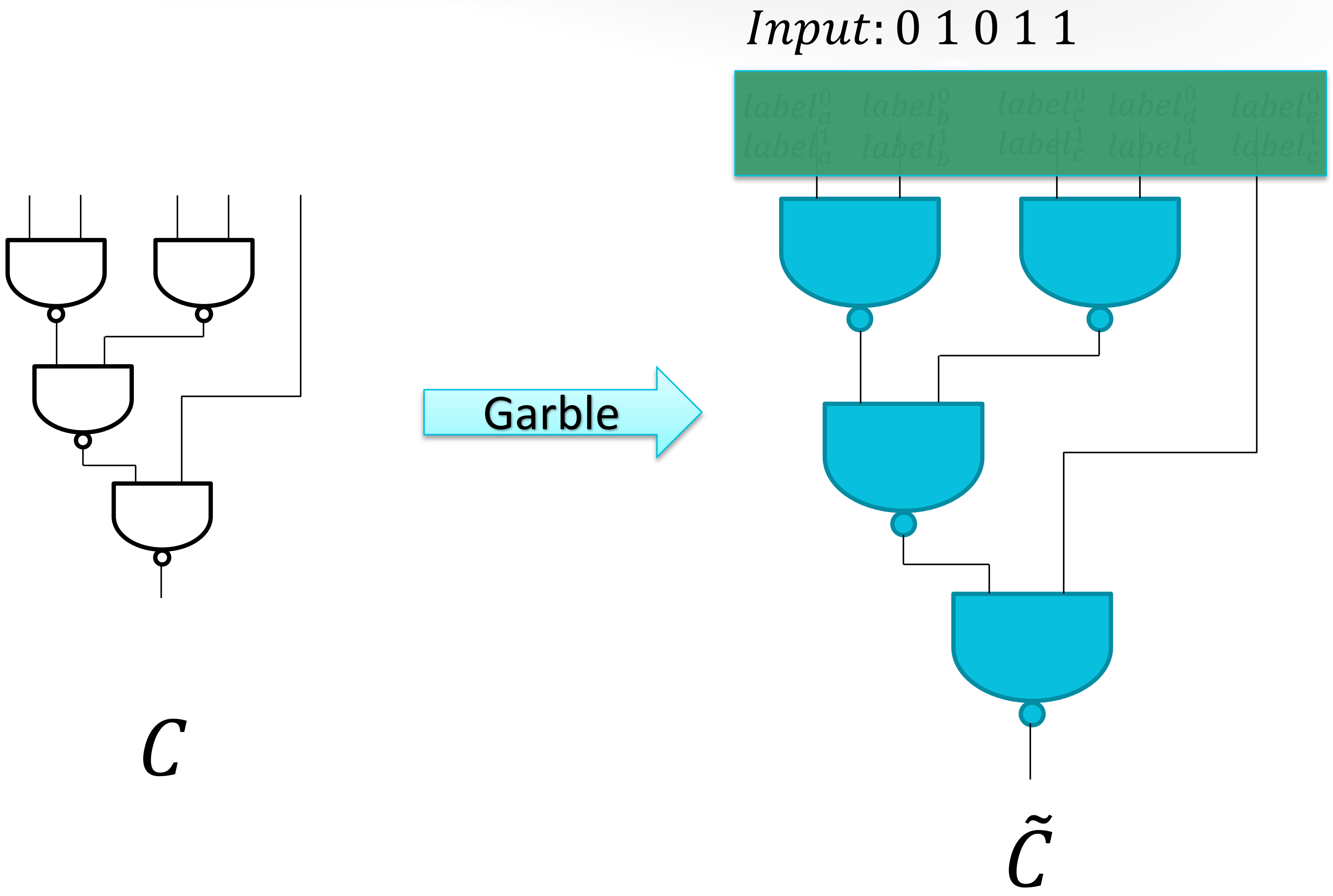
Secure Two-Party Computation for Circuits



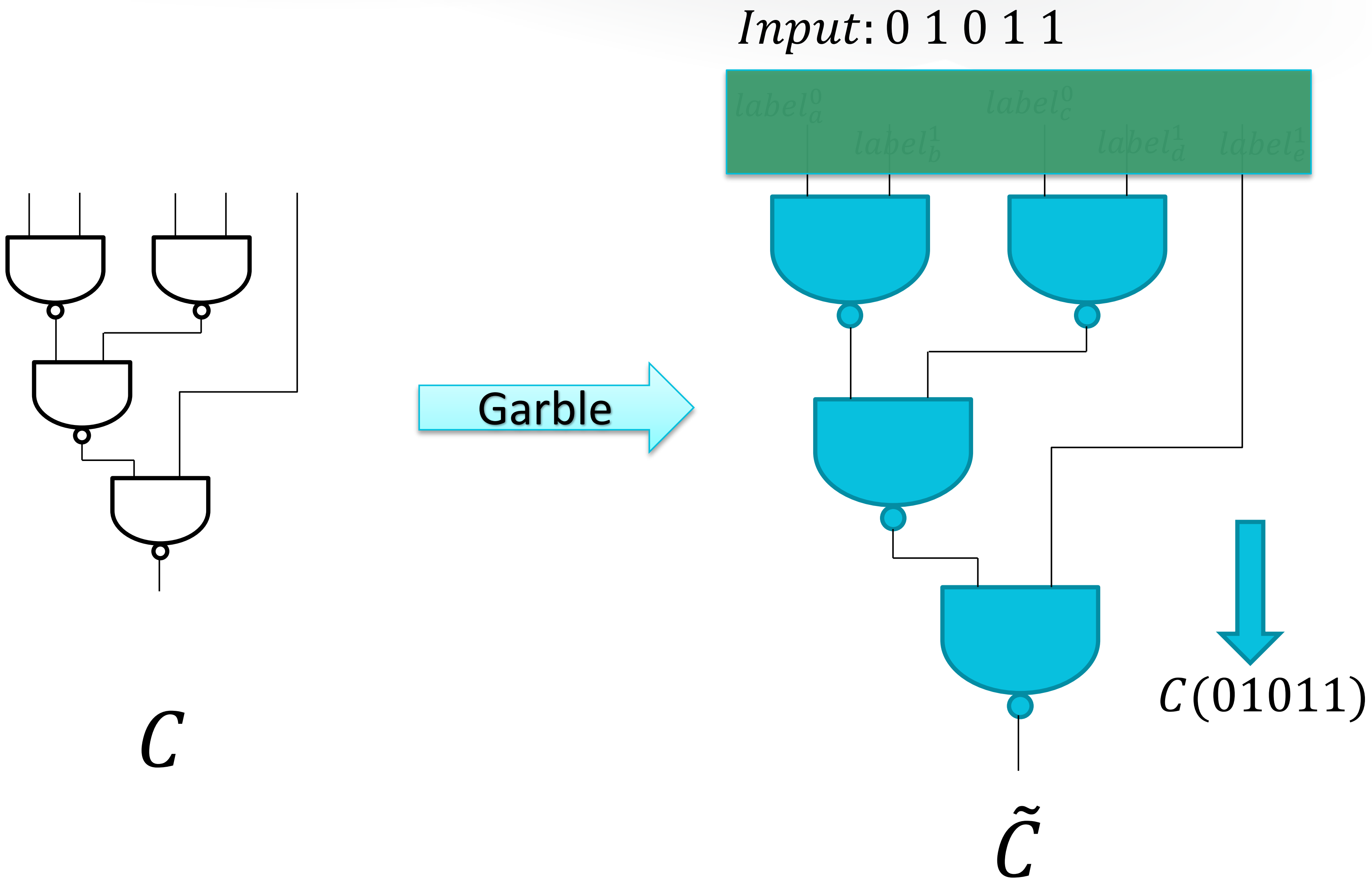
Yao's Garbled Circuit [Yao'86]



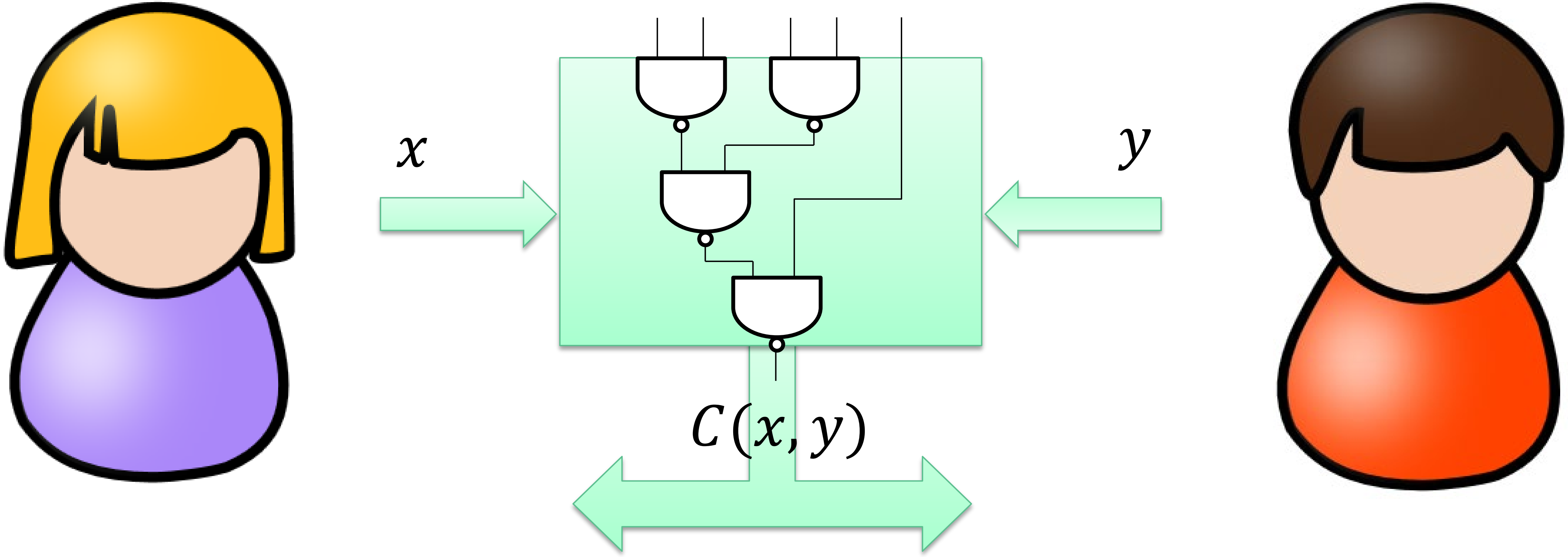
Yao’s Garbled Circuit [Yao’86]


$Input: 0\ 1\ 0\ 1\ 1$
$label_a^0$ $label_b^0$ $label_c^0$ $label_d^0$ $label_e^0$
$label_a^1$ $label_b^1$ $label_c^1$ $label_d^1$ $label_e^1$
C
Garble
$\tilde{C}$

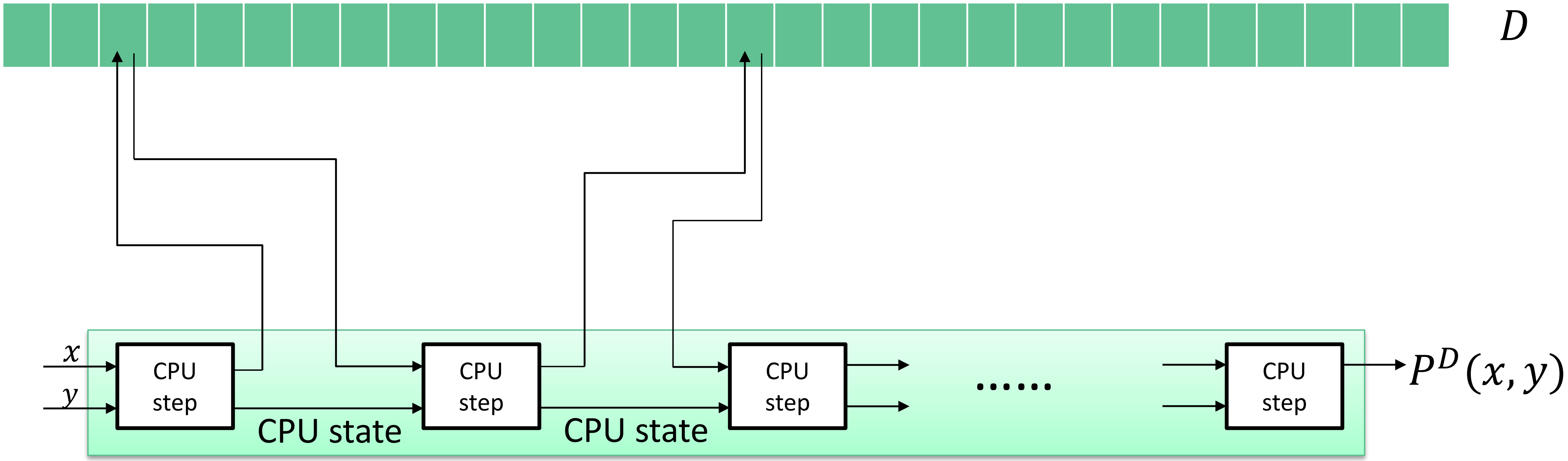
Yao’s Garbled Circuit [Yao’86]



Secure Two-Party Computation for Circuits



RAM (Random-Access Machine) Computation?



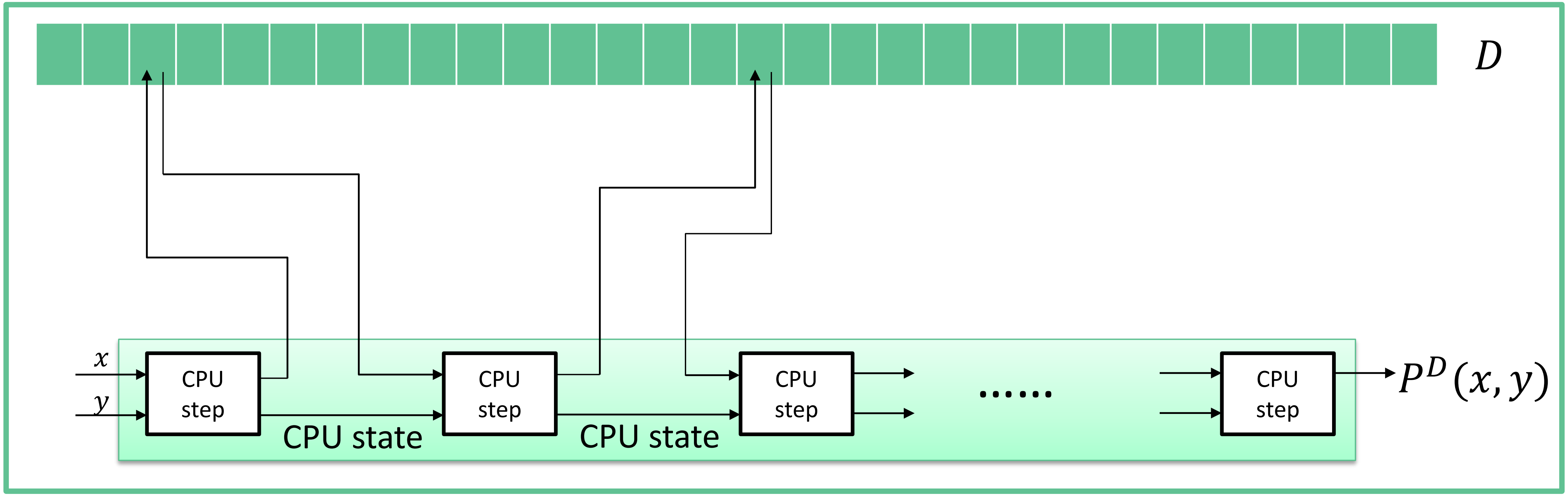
Program P

Secure Two-Party RAM Computation

- Convert RAM program into a circuit?
  - RAM program with running time T
  - Turing machine with running time $O(T^3)$
  - Circuit with size $O(T^3 \log T)$

Secure Two-Party RAM Computation

- Convert RAM program into a circuit?



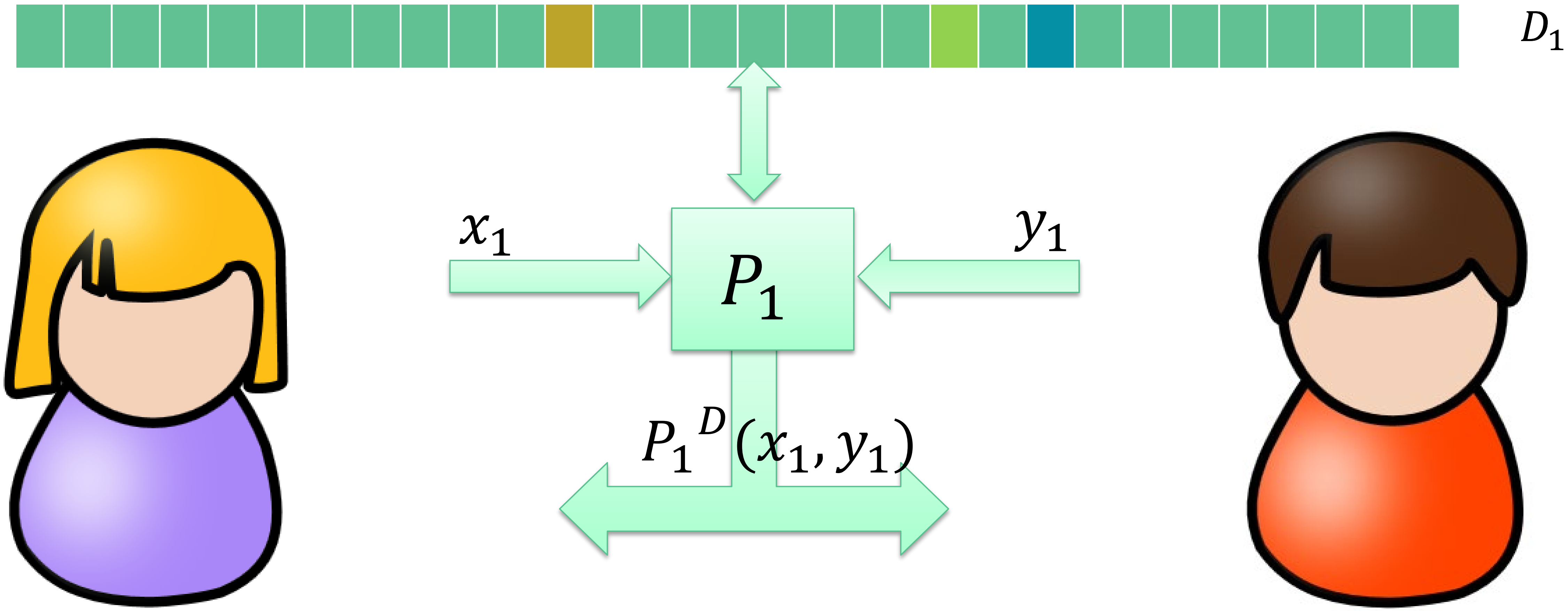
Circuit size could be *exponentially* larger than running time T!

Can we do it more efficiently?

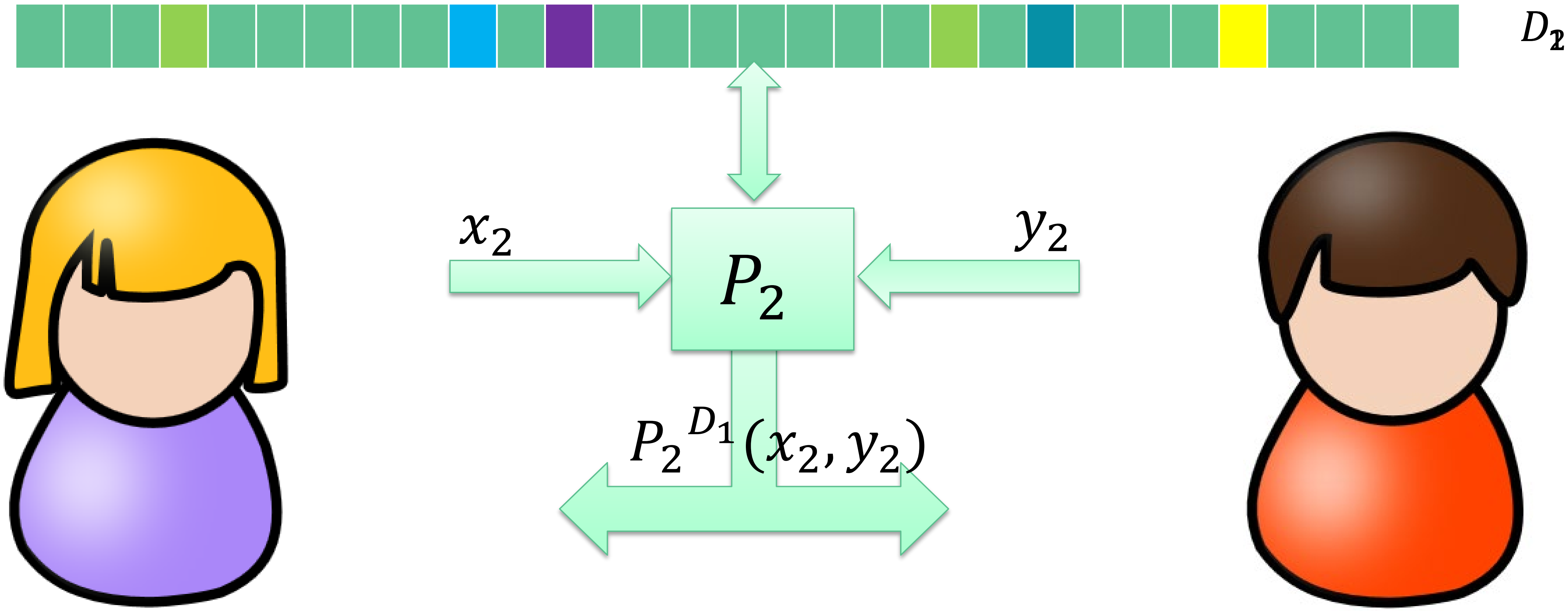
Yes, Garbled RAM [LO’13] !

Secure RAM computation [LO’13, GHLORW’14, GLOS’15, GLO’15, GGMP’16, LO’17, KY’18, HY’19, CQ’19, …]

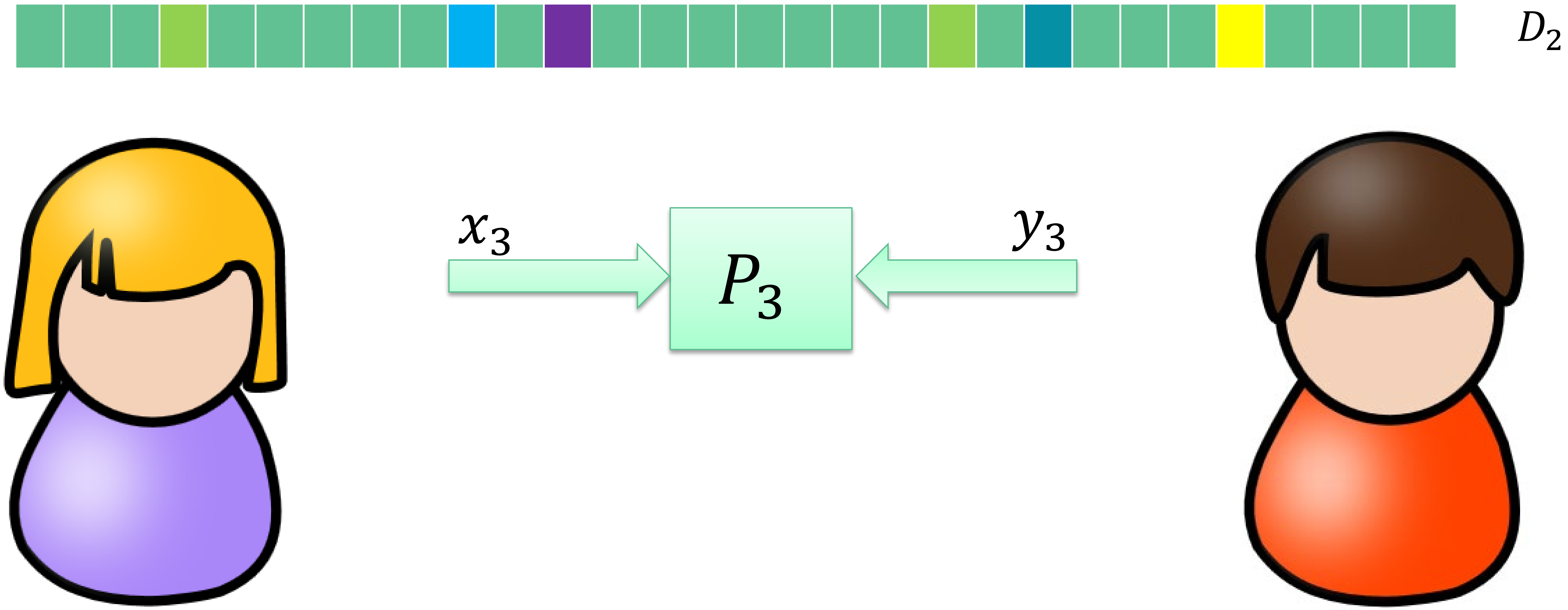
Secure RAM Computation over Persistent Memory



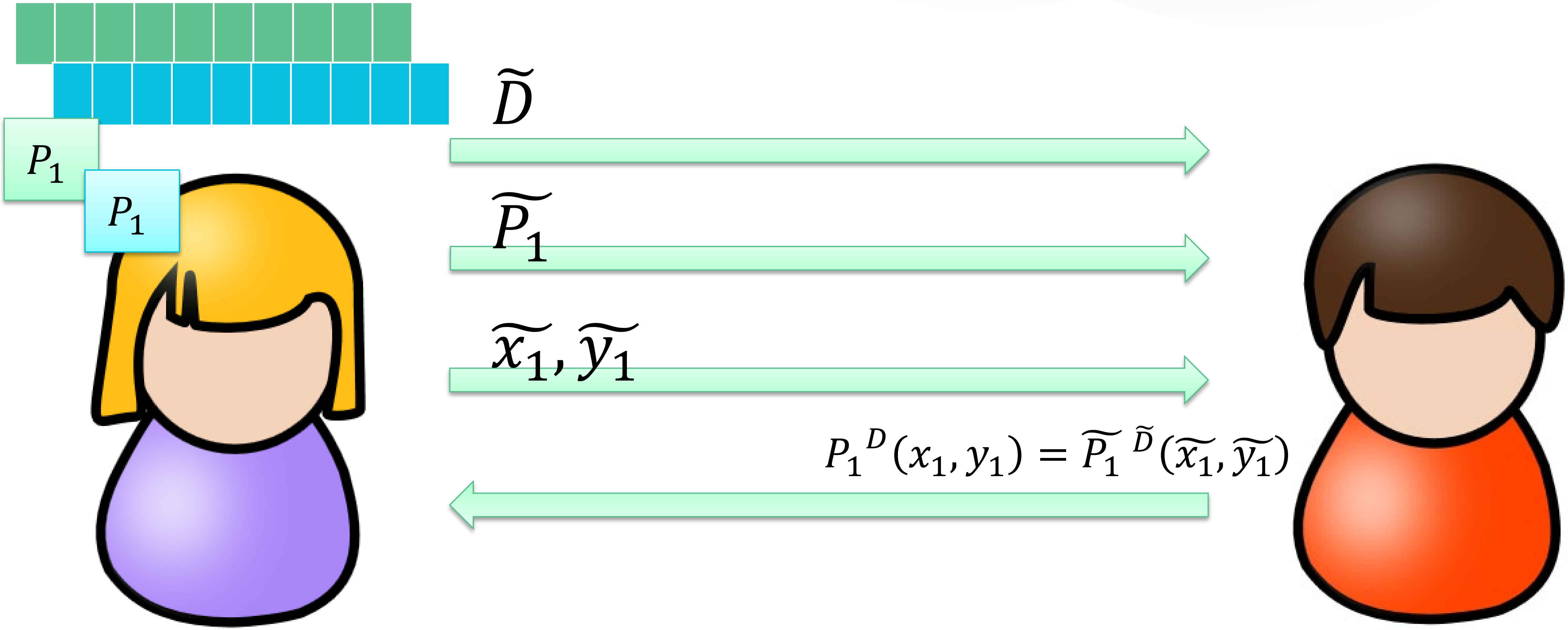
Secure RAM Computation over Persistent Memory



Secure RAM Computation over Persistent Memory



Garbled RAM [LO’13]



P_1

P_1

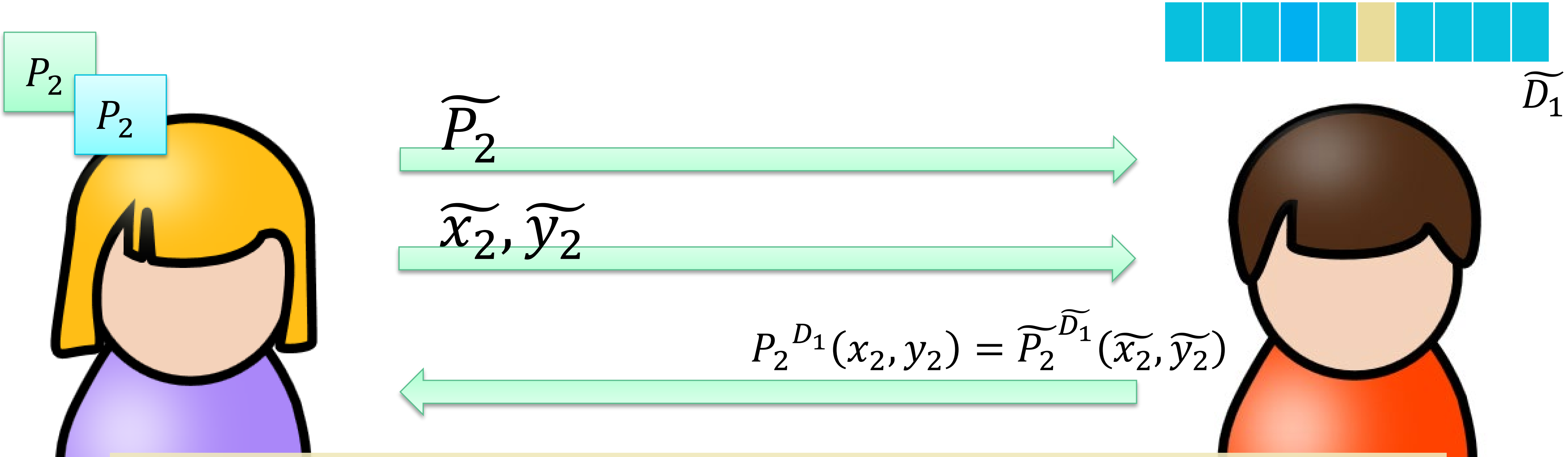
$\widetilde{D}$

$\widetilde{P_1}$

$\widetilde{x_1}, \widetilde{y_1}$

$P_1{}^{D}(x_1, y_1) = \widetilde{P_1}{}^{\widetilde{D}}(\widetilde{x_1}, \widetilde{y_1})$

Garbled RAM [LO'13]


P_2
P_2
$\widetilde{D_1}$
$\widetilde{P_2}$
$\widetilde{x_2}, \widetilde{y_2}$
${P_2}^{D_1}(x_2, y_2) = \widetilde{P_2}^{\widetilde{D_1}}(\widetilde{x_2}, \widetilde{y_2})$

Size of garbled memory $\tilde{D} = \tilde{O}(|D|)$

Size and evaluation time of garbled program $\tilde{P} = \tilde{O}(T)$

$\tilde{O}$ ignores $poly(\lambda) \cdot polylog(|D|, T)$

Can we do it from the weakest cryptographic assumption?

Yes, black-box garbled RAM [GLO'15] !

black-box use of OWFs, but only semi-honest secure

Can we make it maliciously secure?

Yes, [GMW'87] compiler: semi-honest → malicious

requires generic zero-knowledge proofs, non-black-box use of OWFs

Can we make it maliciously secure while still making black-box use of OWFs?

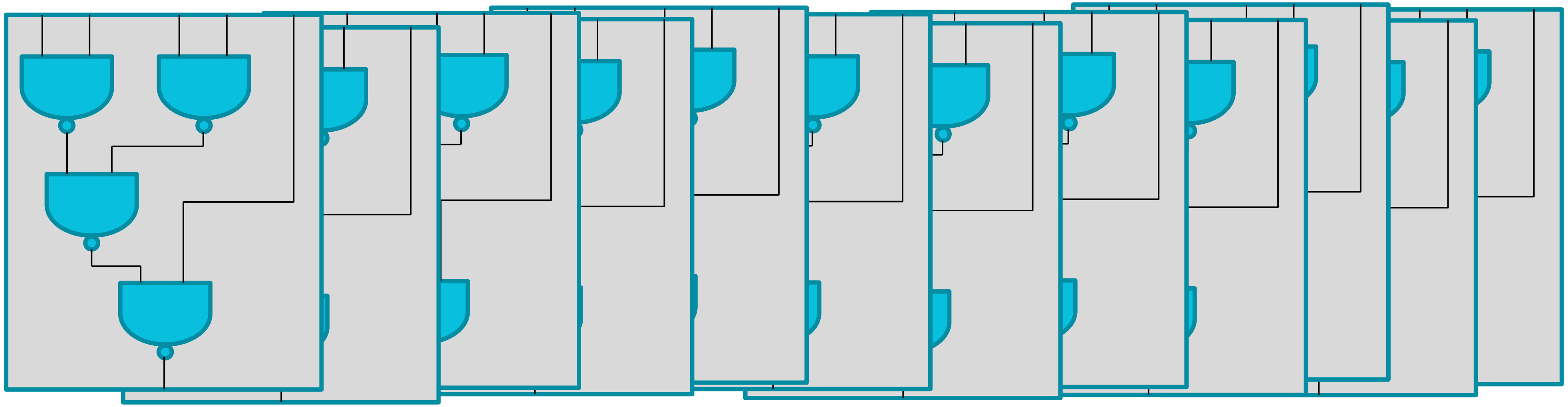
Yes, this work!

Outline

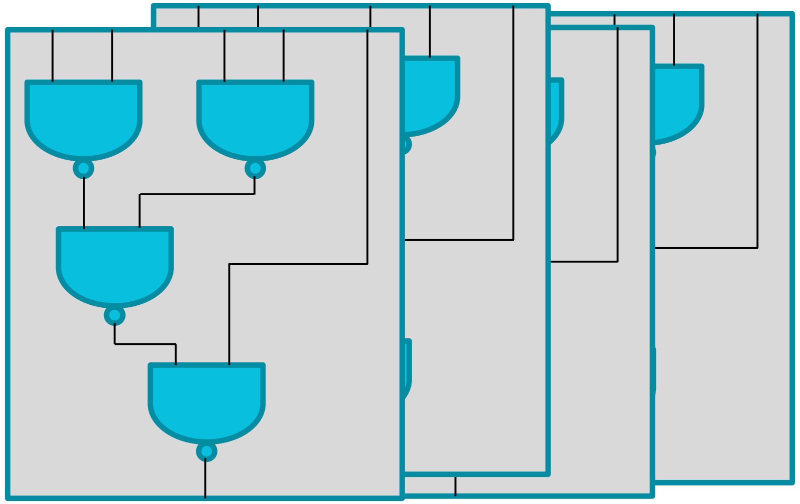
- Secure Two-Party RAM Computation
  - Convert RAM program into a circuit?
- Garbled RAM [LO’13]
- Black-Box Garbled RAM [GLO’15]
- This Work: Malicious Security
  - Consistency Checks by Commitments
  - Cut-and-Choose on Gates

Black-Box Garbled RAM [GLO'15]

$\widetilde{D}$

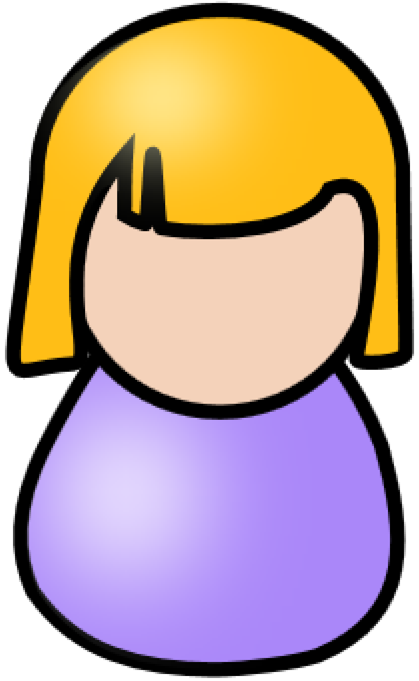


$\widetilde{P}$

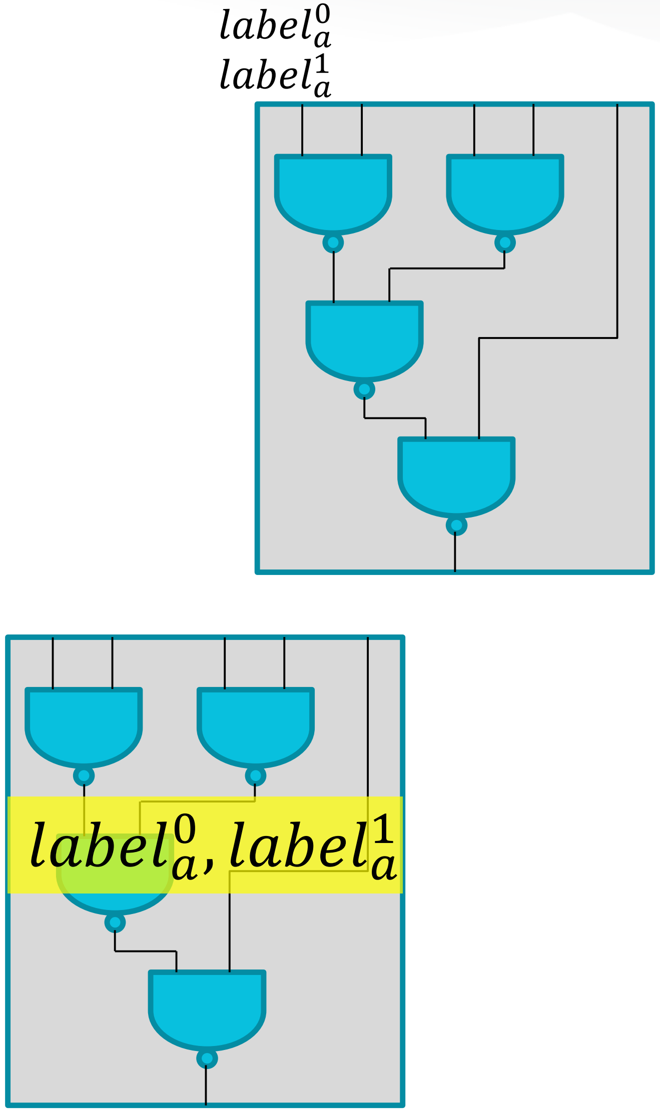


Black-Box Garbled RAM [GLO’15]

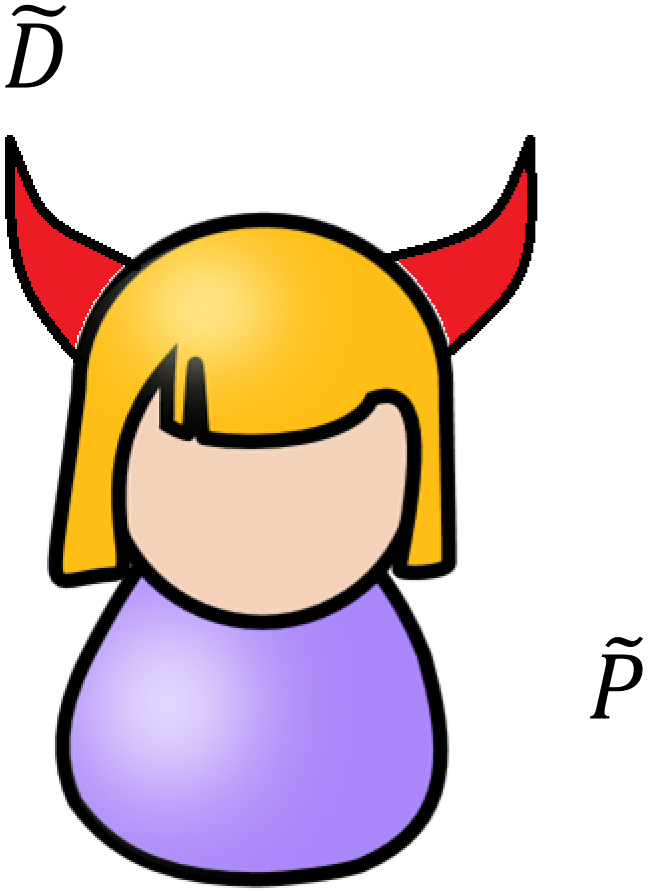
$\widetilde{D}$

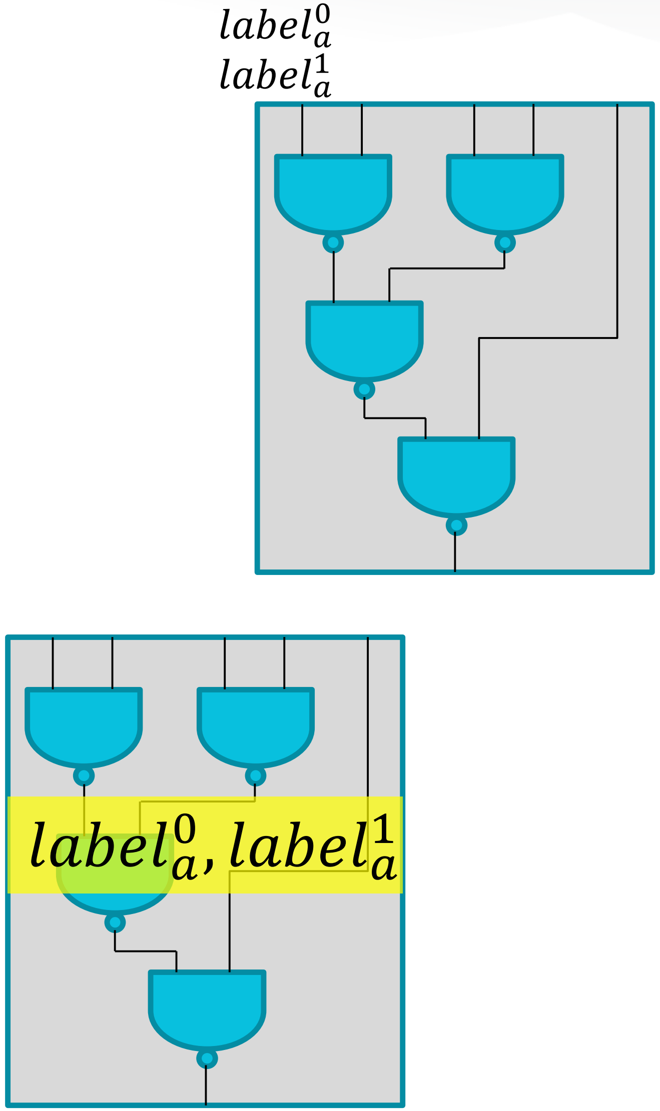


$\tilde{P}$



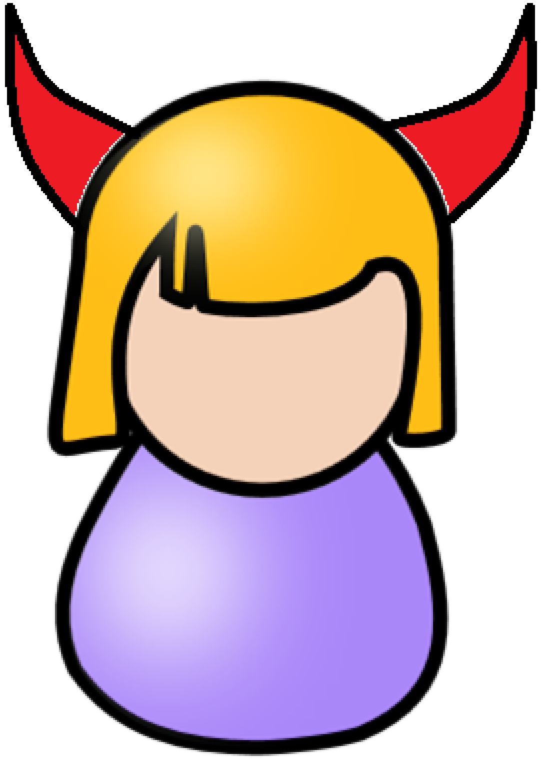
Malicious Alice?



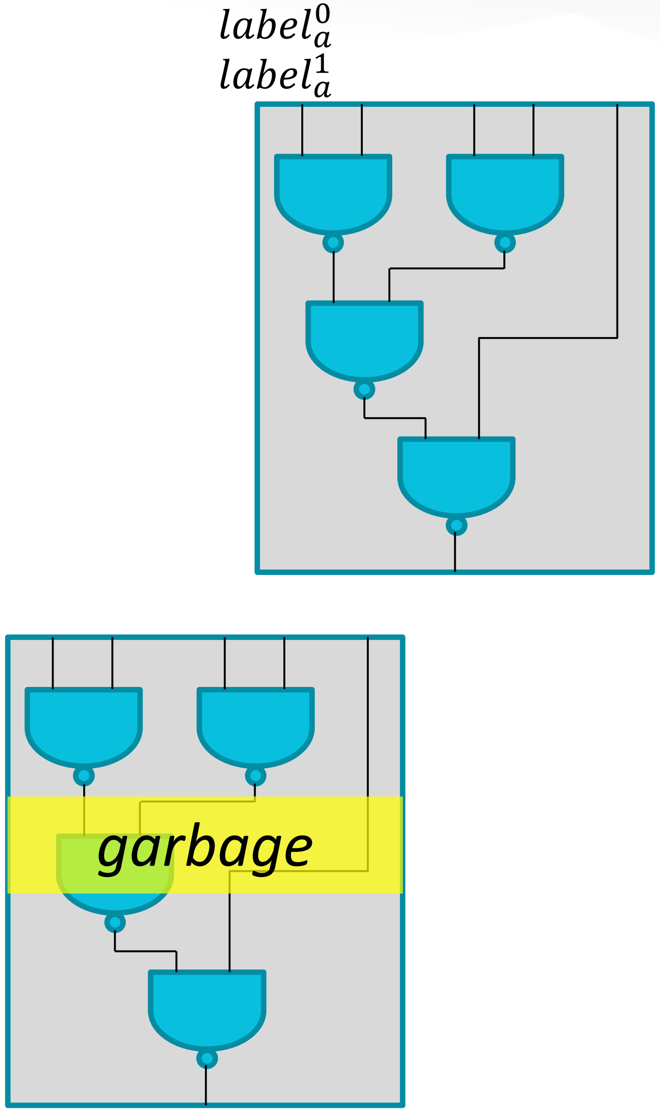


Malicious Alice?

$\widetilde{D}$



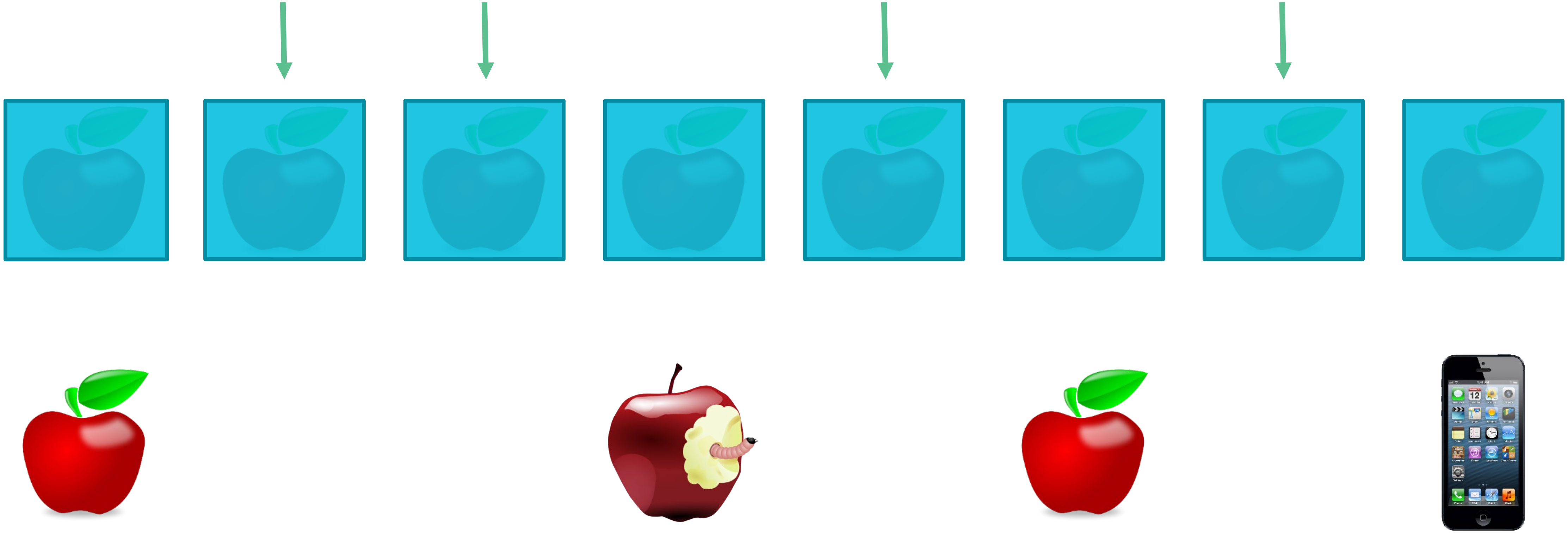
$\widetilde{P}$



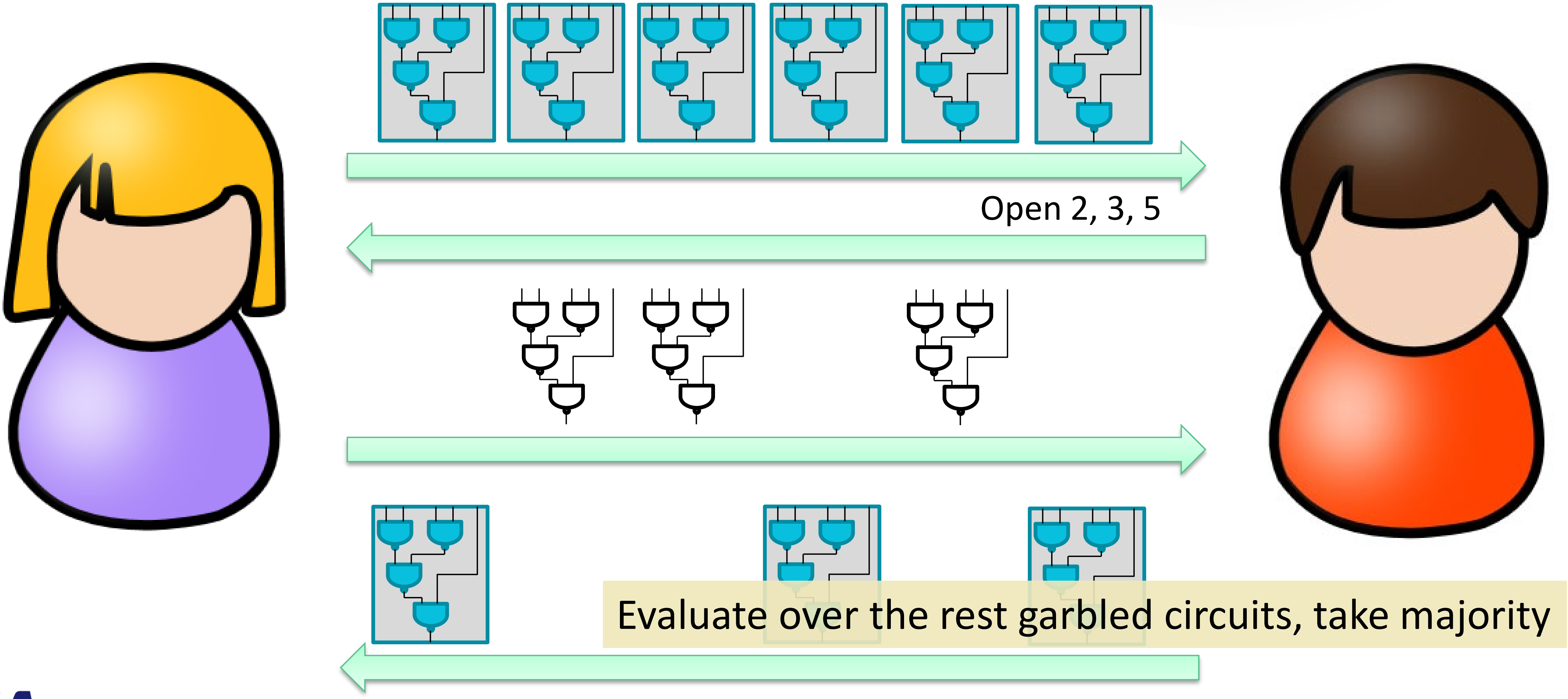
How to avoid Alice cheating?

Cut-and-Choose!

Cut-and-Choose Technique

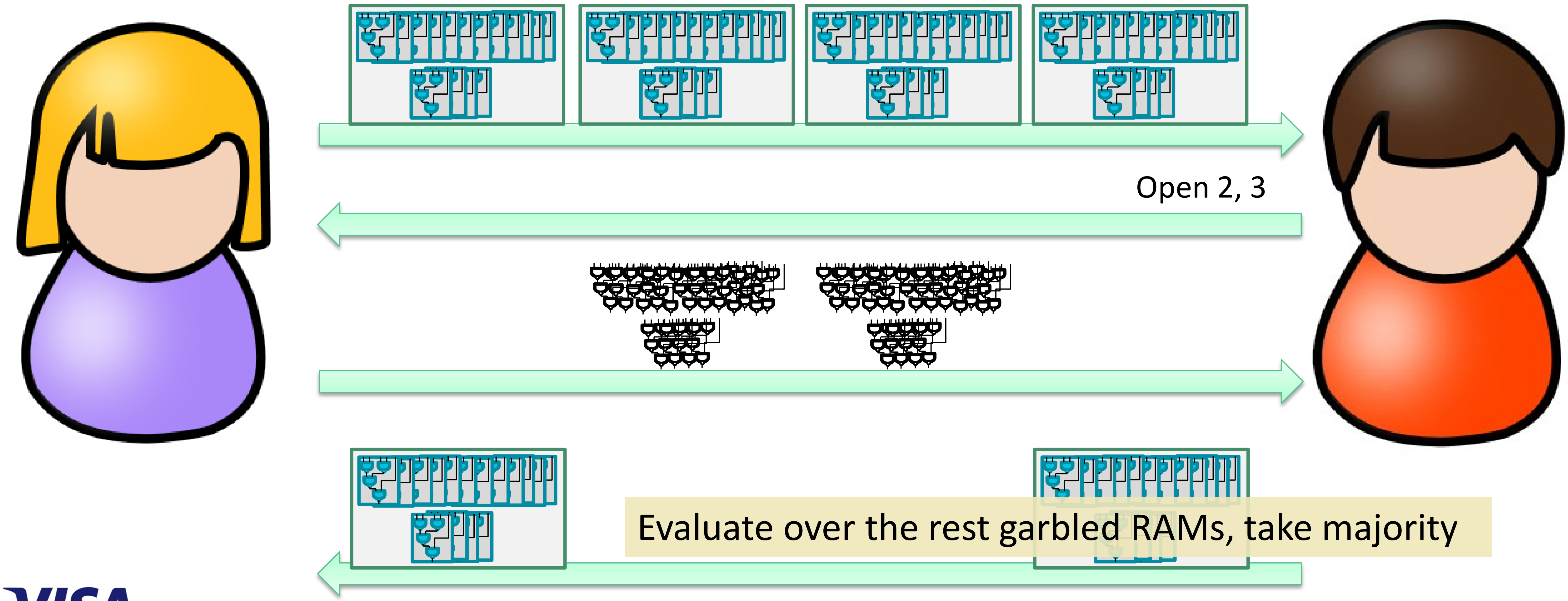


Cut-and-Choose for Yao's Garbled Circuit [LP'07]



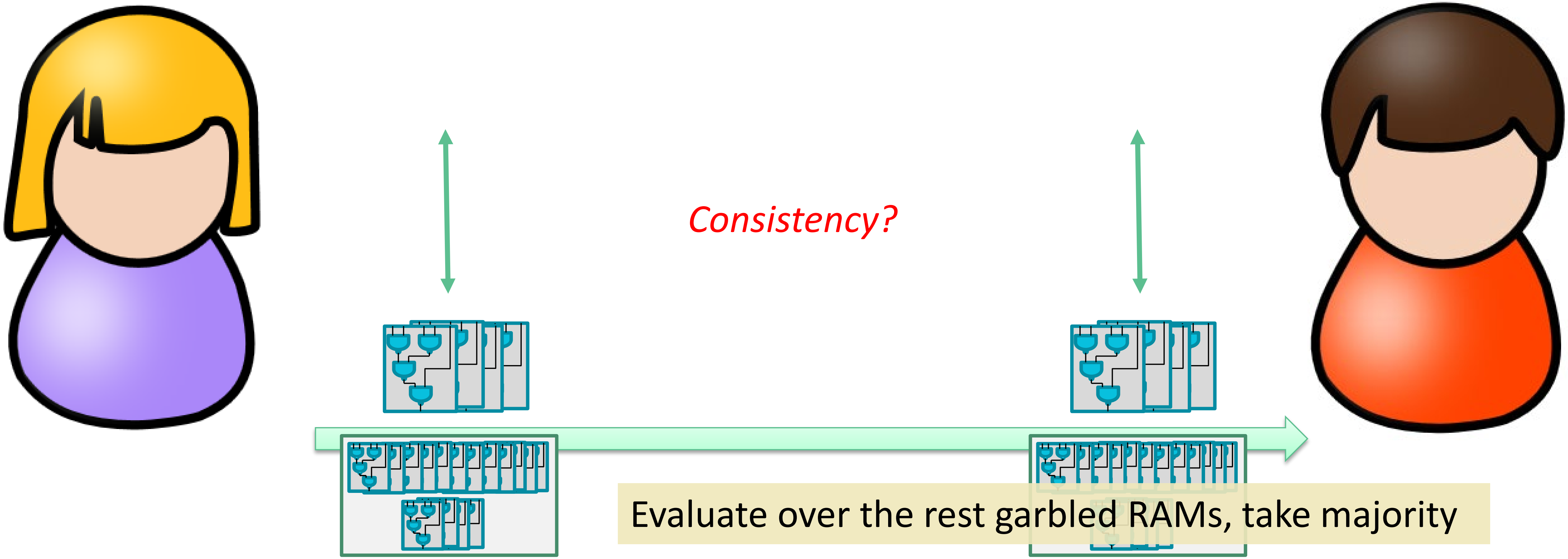
Evaluate over the rest garbled circuits, take majority

Cut-and-Choose for Garbled RAM



Evaluate over the rest garbled RAMs, take majority

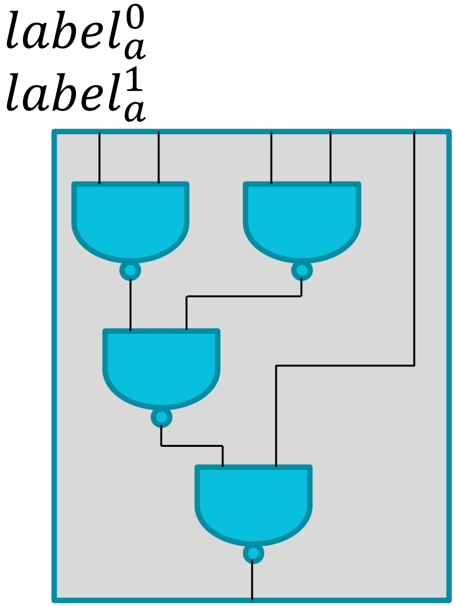
Cut-and-Choose for Garbled RAM

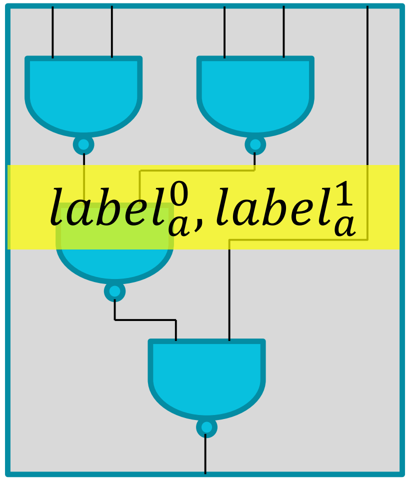


Consistency?

Evaluate over the rest garbled RAMs, take majority

Consistency

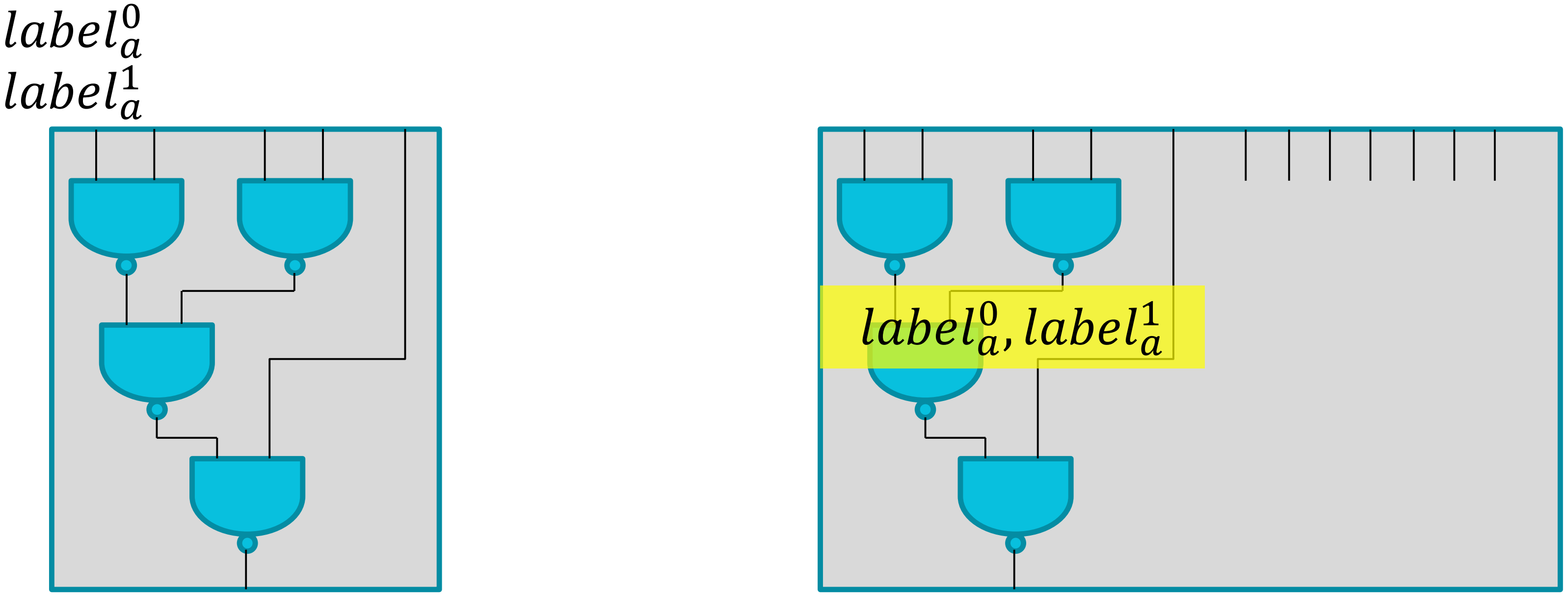




Circuit X

Circuit Y

Consistency



Circuit X

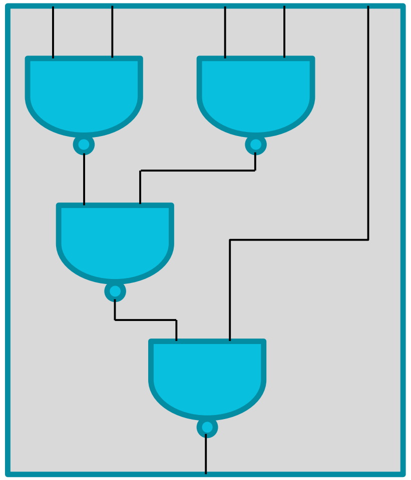
Circuit Y

Consistency

How to enforce Alice to provide $label_w^0$ without revealing the bit 0?

$label_a^0 = 0110$

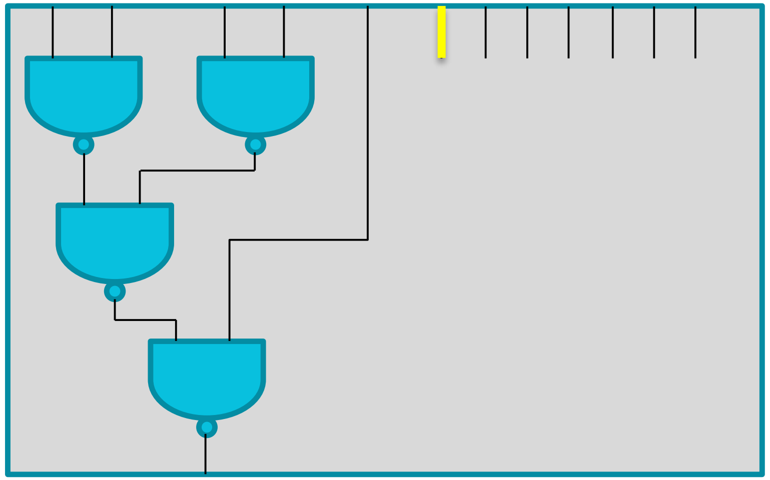
$label_a^1$



Circuit X

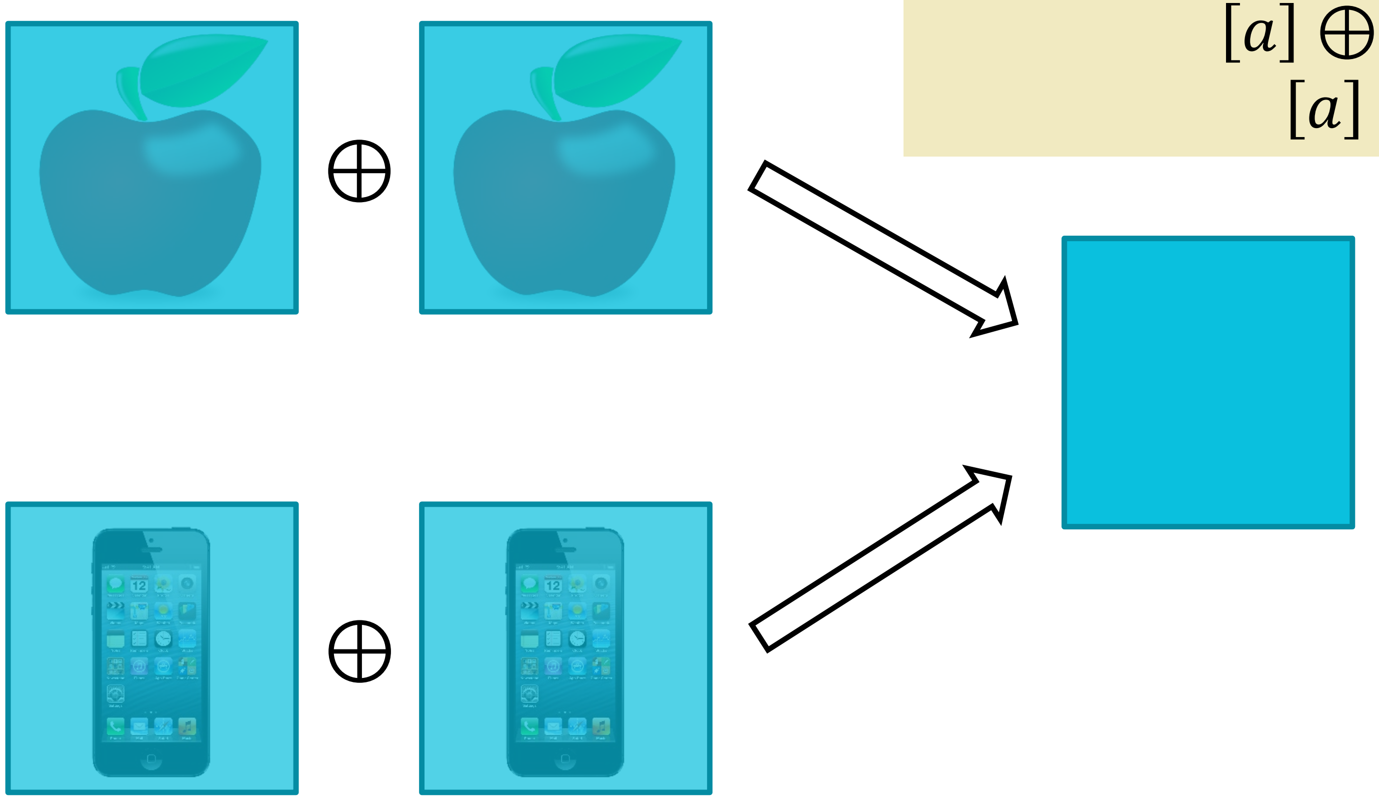
$label_w^0$

$label_w^1$



Circuit Y

Consistency Check by Commitments



XOR-homomorphic commitment [FJNT’15]

$$[a] \oplus [b] = [a \oplus b]$$

$$[a] \oplus [a] = [0]$$

Consistency Check by Commitments

$[b_0], [b_1], [b_2], [b_3]$

$label_a^0 = b_0 b_1 b_2 b_3$

$label_a^1$

$[label_w^0]||[0]$

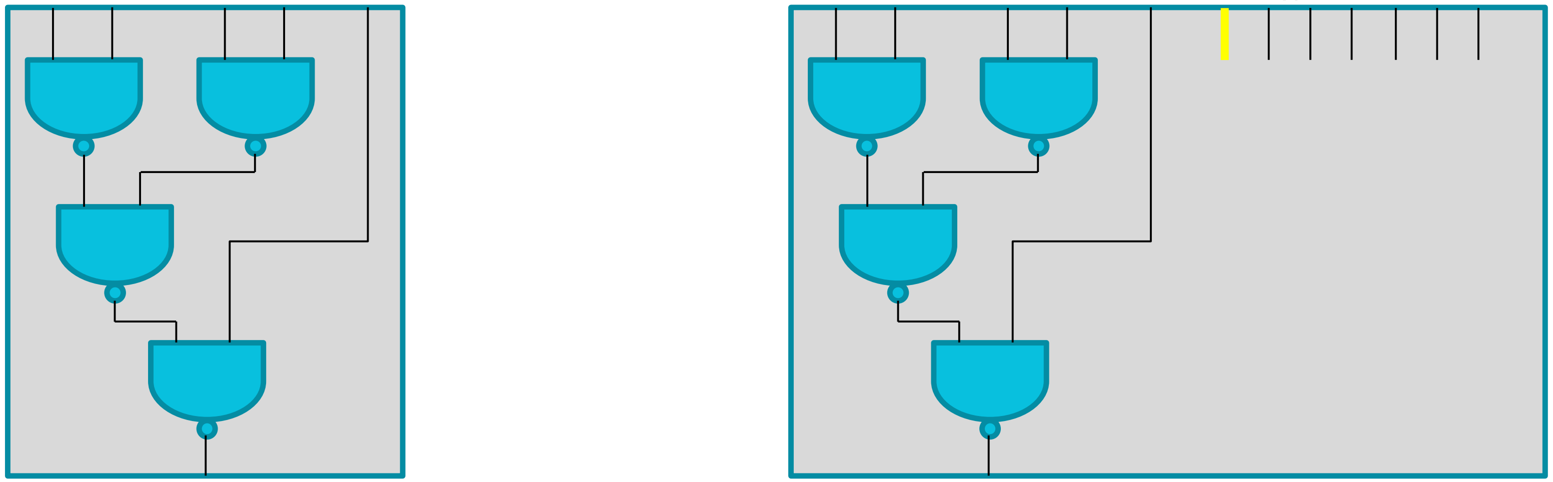
$[label_w^1]||[1]$

$[label_w^r]||[r]$

$[label_w^{\bar r}]||[\bar r]$

$label_w^0$

$label_w^1$



Circuit X

Circuit Y

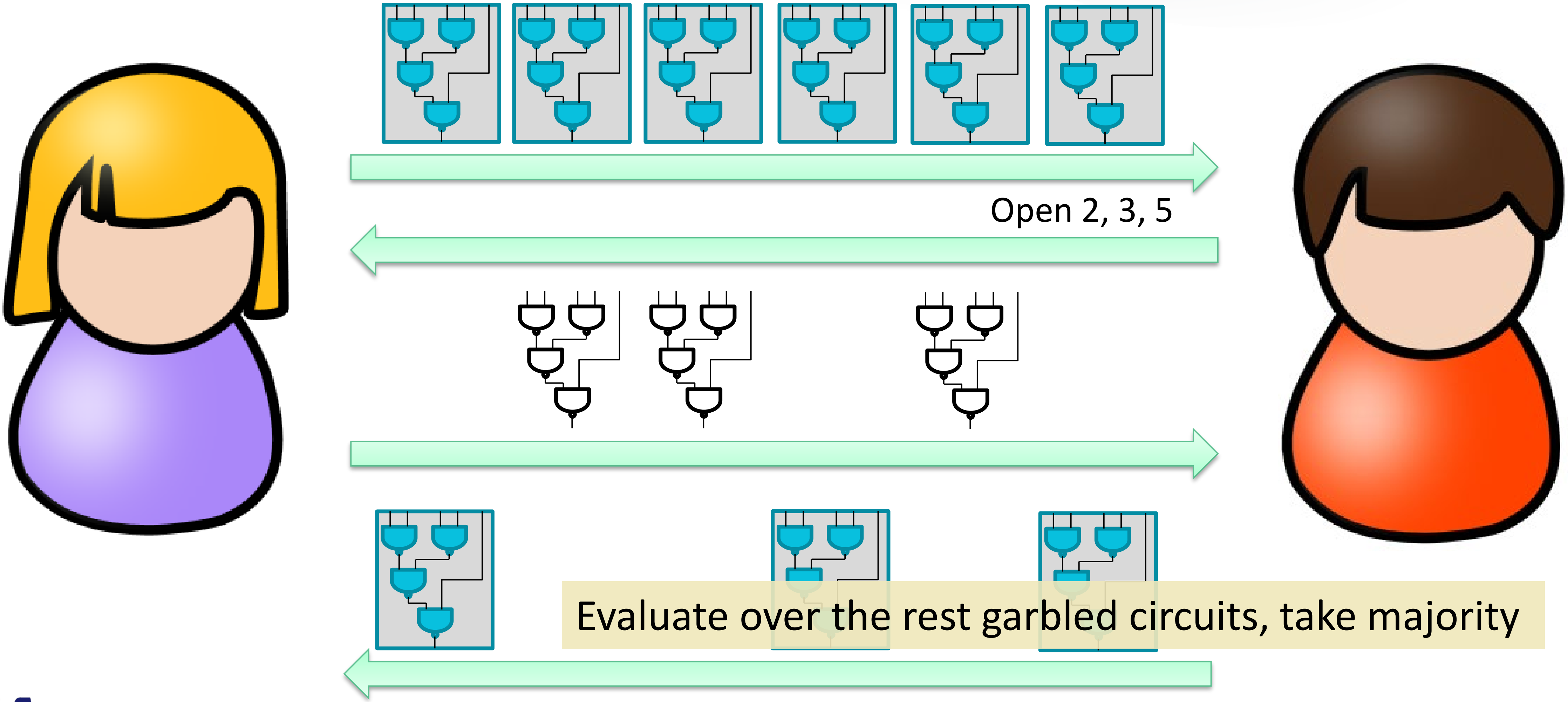
Open $label_w^r$

$[b_0] \oplus [r] \rightarrow [0]$?

Outline

- Secure Two-Party RAM Computation
  - Convert RAM program into a circuit?
- Garbled RAM [LO’13]
- Black-Box Garbled RAM [GLO’15]
- This Work: Malicious Security
  - Consistency Checks by Commitments
  - Cut-and-Choose on Gates

Cut-and-Choose on Circuits?



Open 2, 3, 5

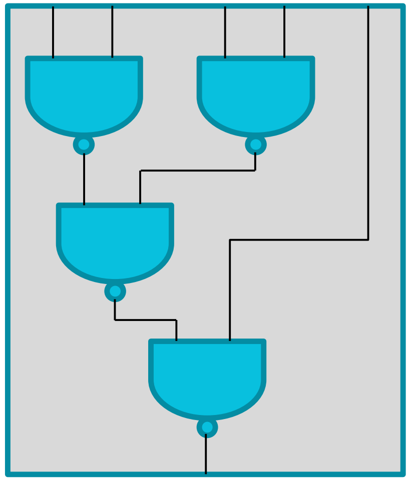
Evaluate over the rest garbled circuits, take majority

Issue 1

$[\overline{b_0}], [b_1], [b_2], [b_3]$

$label_a^0 = b_0 b_1 b_2 b_3$

$label_a^1$



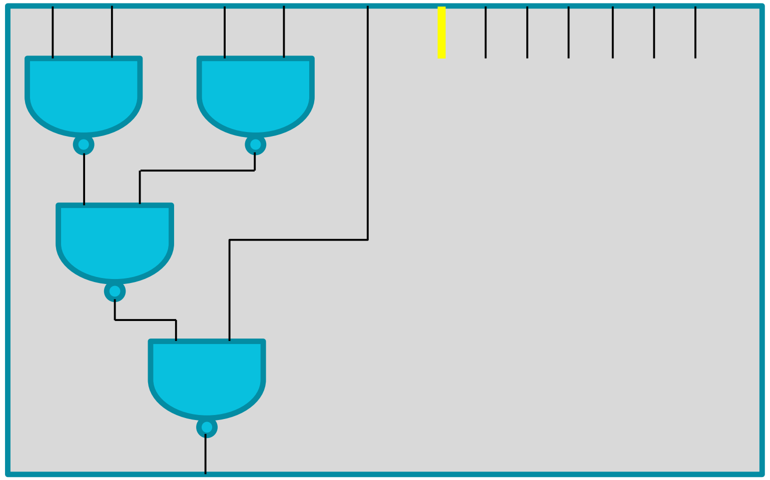
Circuit X

$[label_w^r] \| [r]$

$[label_w^{\bar{r}}] \| [\bar{r}]$

$label_w^0$

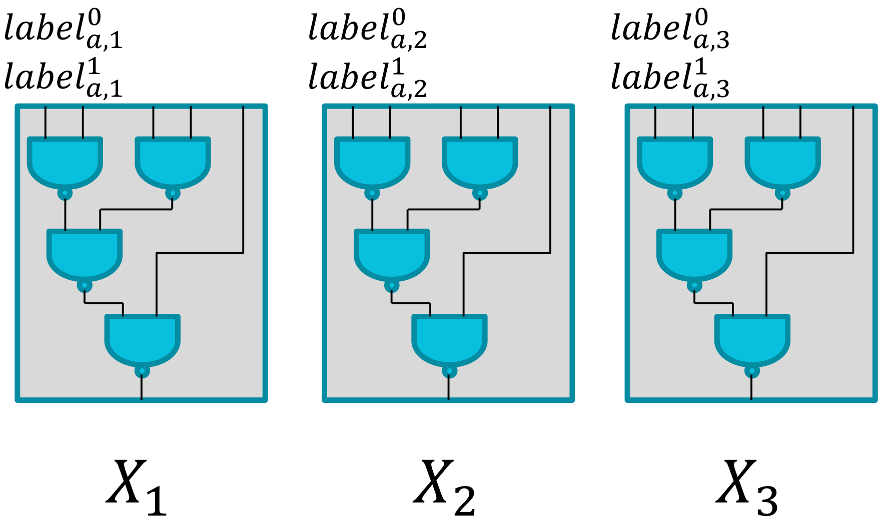
$label_w^1$

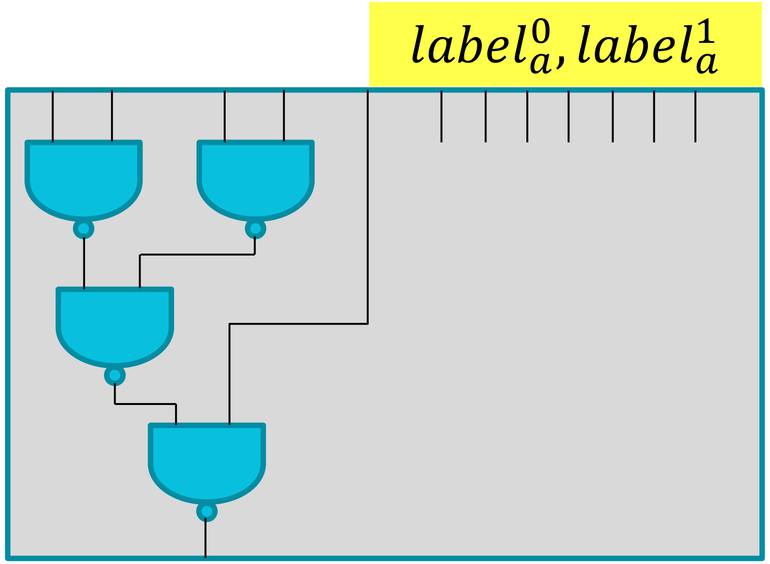


Circuit Y

How to guarantee that Alice has committed correctly?

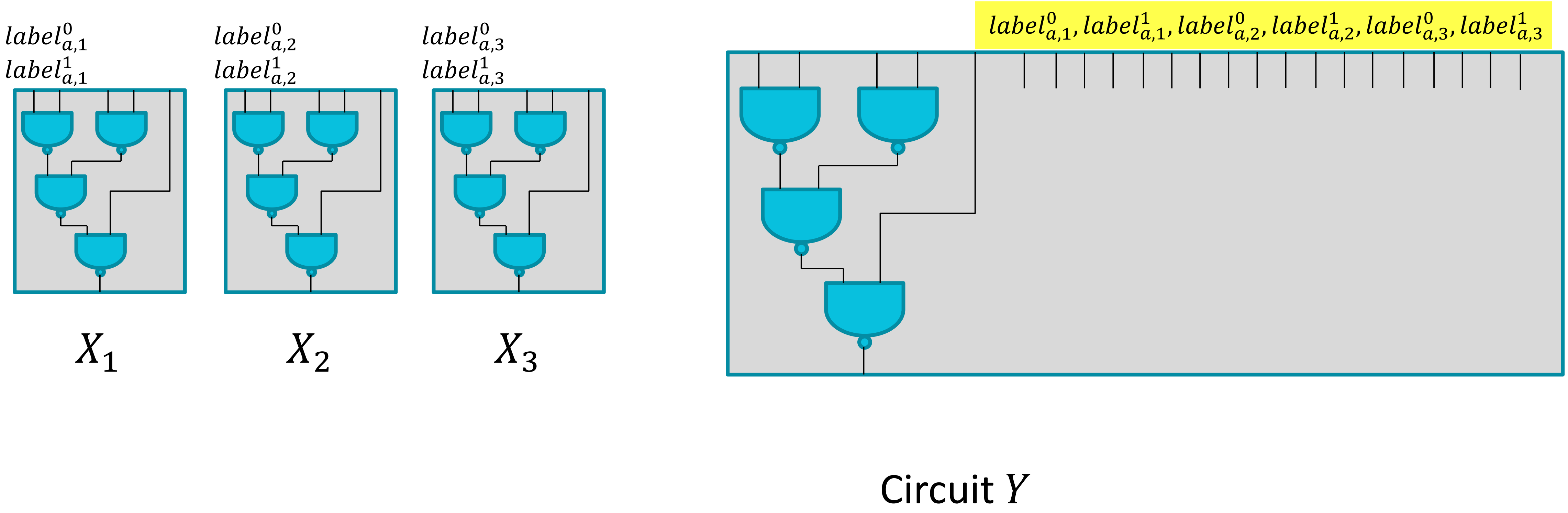
Issue 2





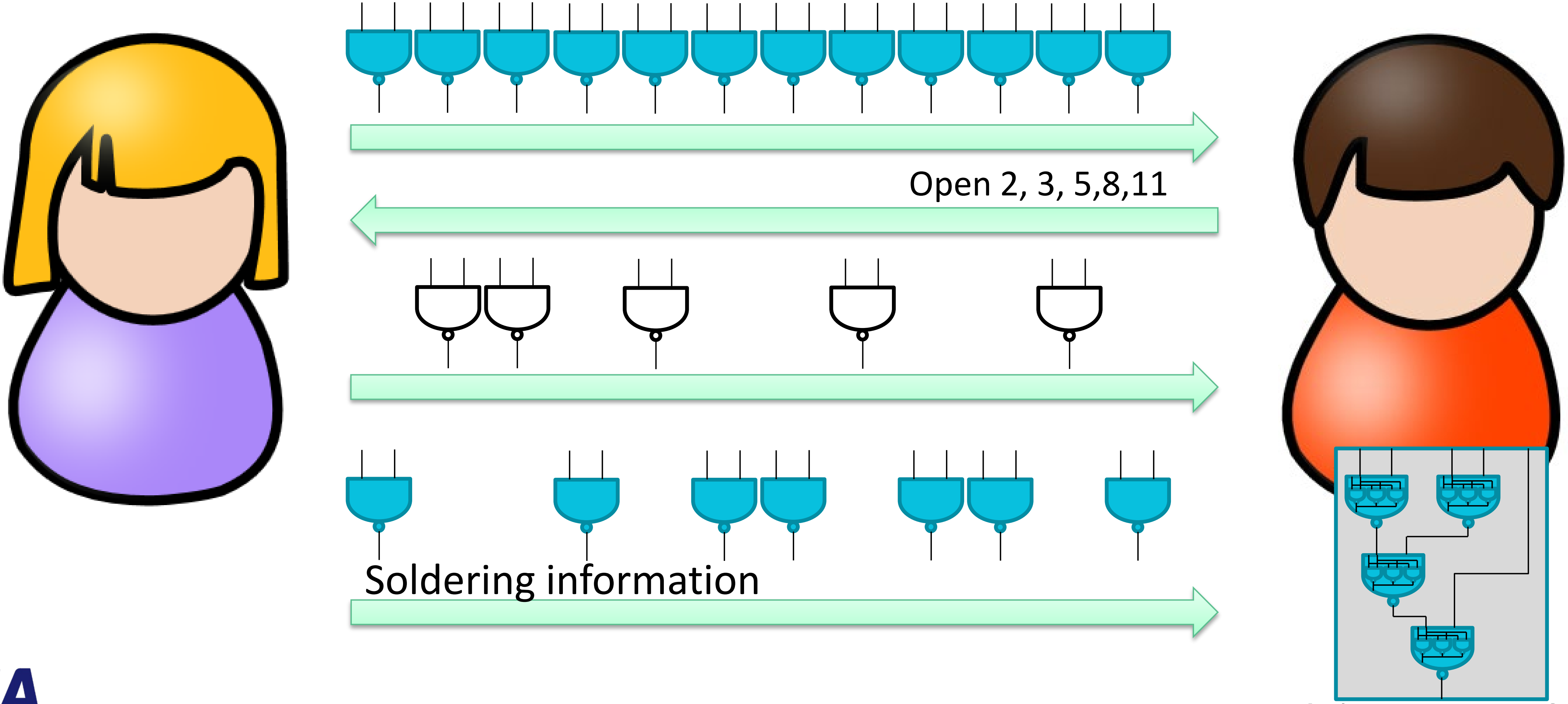
Circuit Y

Issue 2



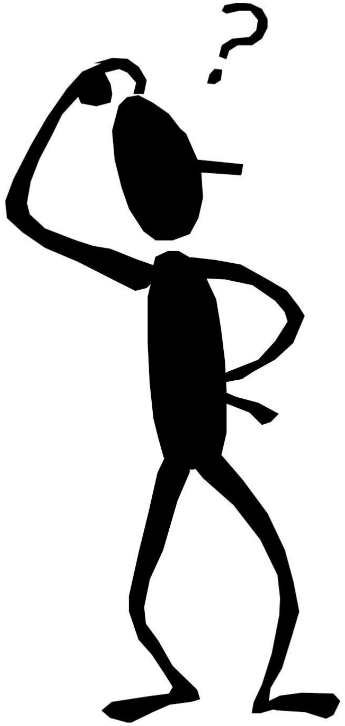
Input size may grow *exponentially* in the number of circuits!

Cut-and-Choose on Gates [NO'09]



Summary

- Secure Two-Party RAM Computation
  - Convert RAM program into a circuit?
- Garbled RAM [LO’13]
- Black-Box Garbled RAM [GLO’15]
- This Work: Malicious Security
  - Consistency Checks by Commitments
  - Cut-and-Choose on Gates



Thank you!