Multi-Party Computation

Constructions & Properties

MPC - objectives

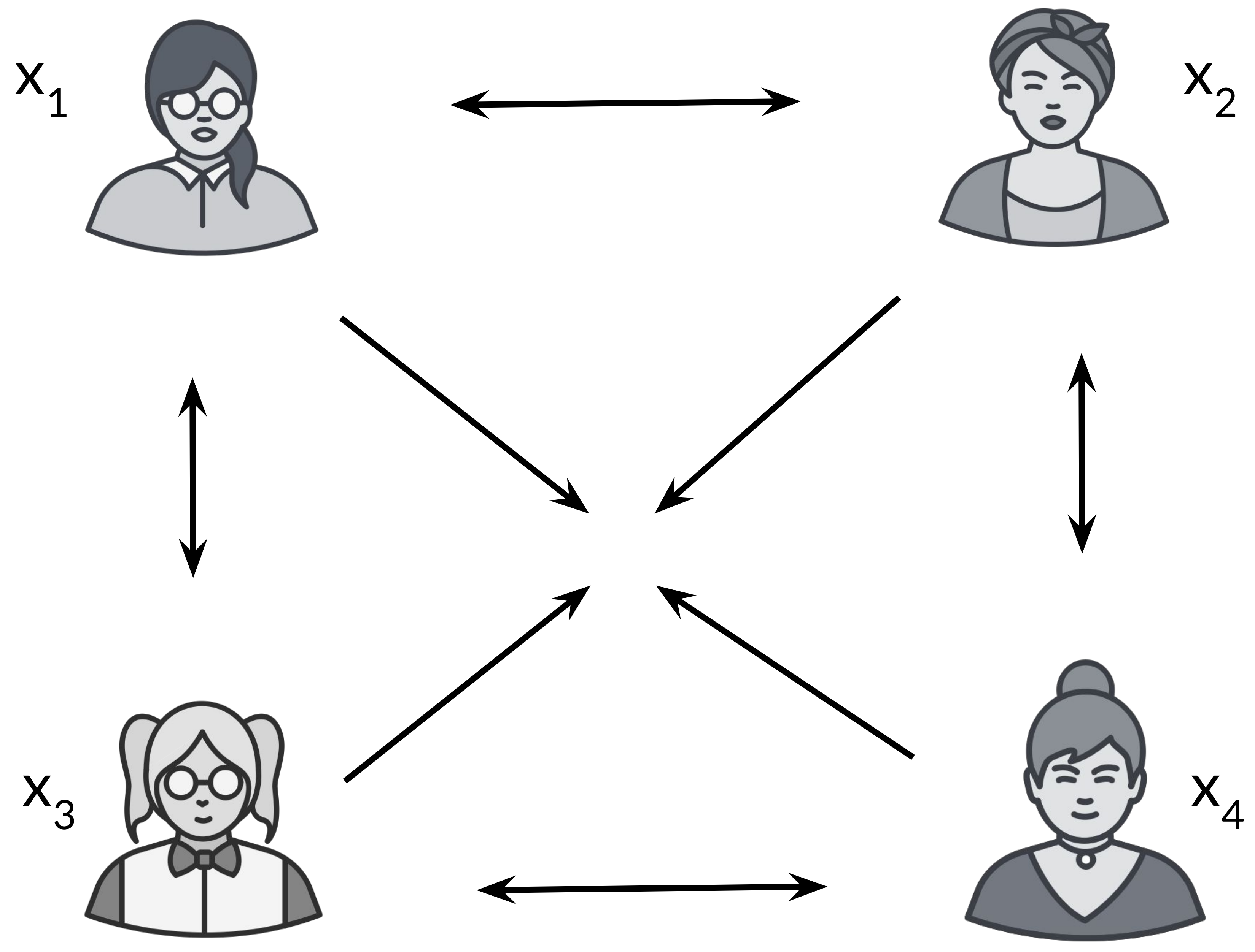
Multi-Party Computation

Framework for computation between parties who do not trust each other.

Examples:

- elections,
- auctions,
- distributed data mining,
- database privacy

$$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) = y$$



MPC - objectives

Multi-Party Computation

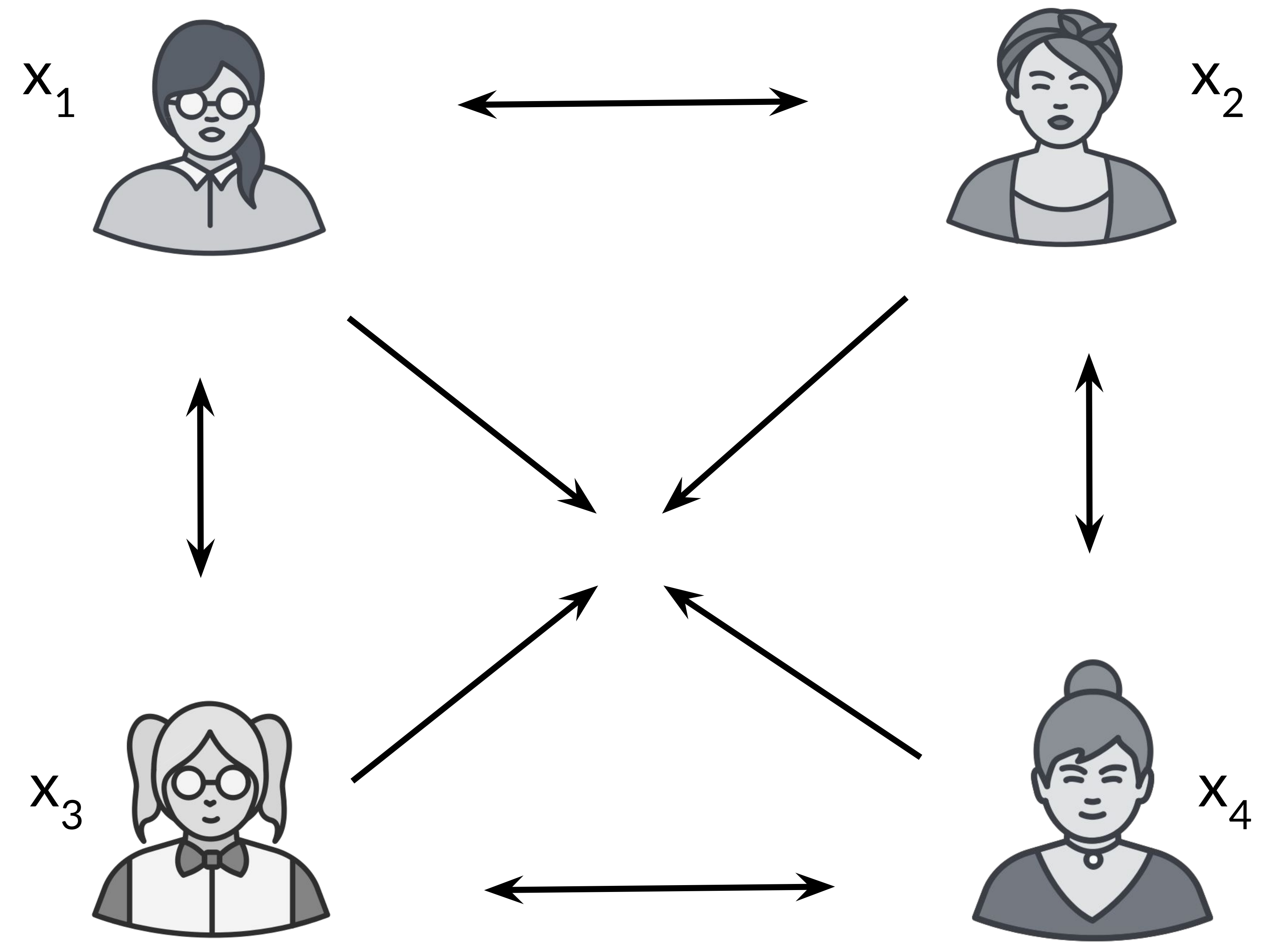
Framework for computation between parties who do not trust each other.

Examples:

- elections,
- auctions,
- distributed data mining,
- database privacy

Goals: preserve the ***privacy*** of each player's inputs and guarantee the ***correctness*** of the computation.

$$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) = y$$



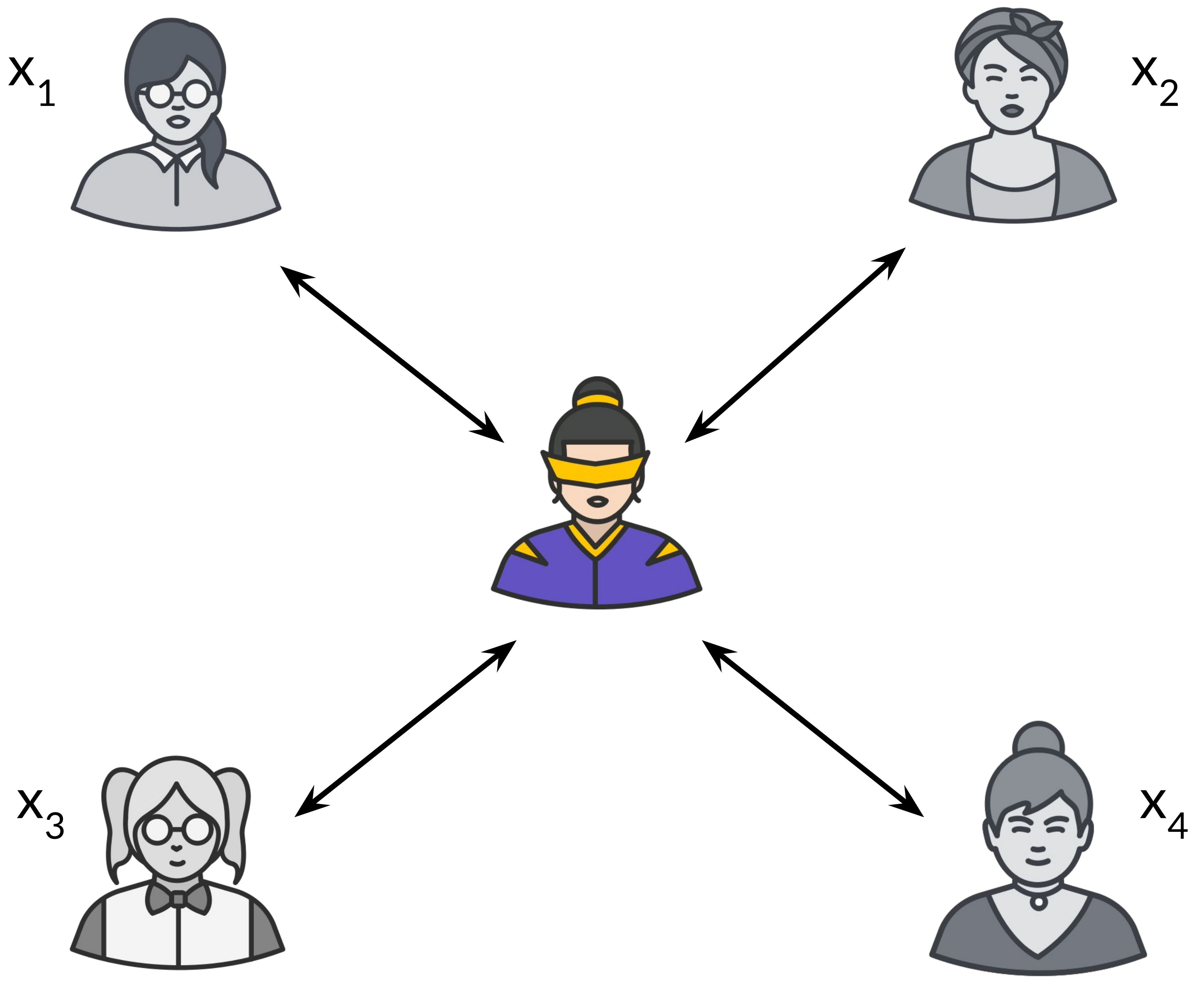
MPC - objectives

Multi-Party Computation

Trusted party:
All of these tasks can be easily computed by a trusted third party

Security :
- Same guarantees without involving a trusted third party
- Provide an exact problem definition -
  - Adversarial power
  - Network model
  - Meaning of security
- Prove that the protocol is secure - by reduction to an assumed hard problem

$$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) = y$$



MPC - Steps

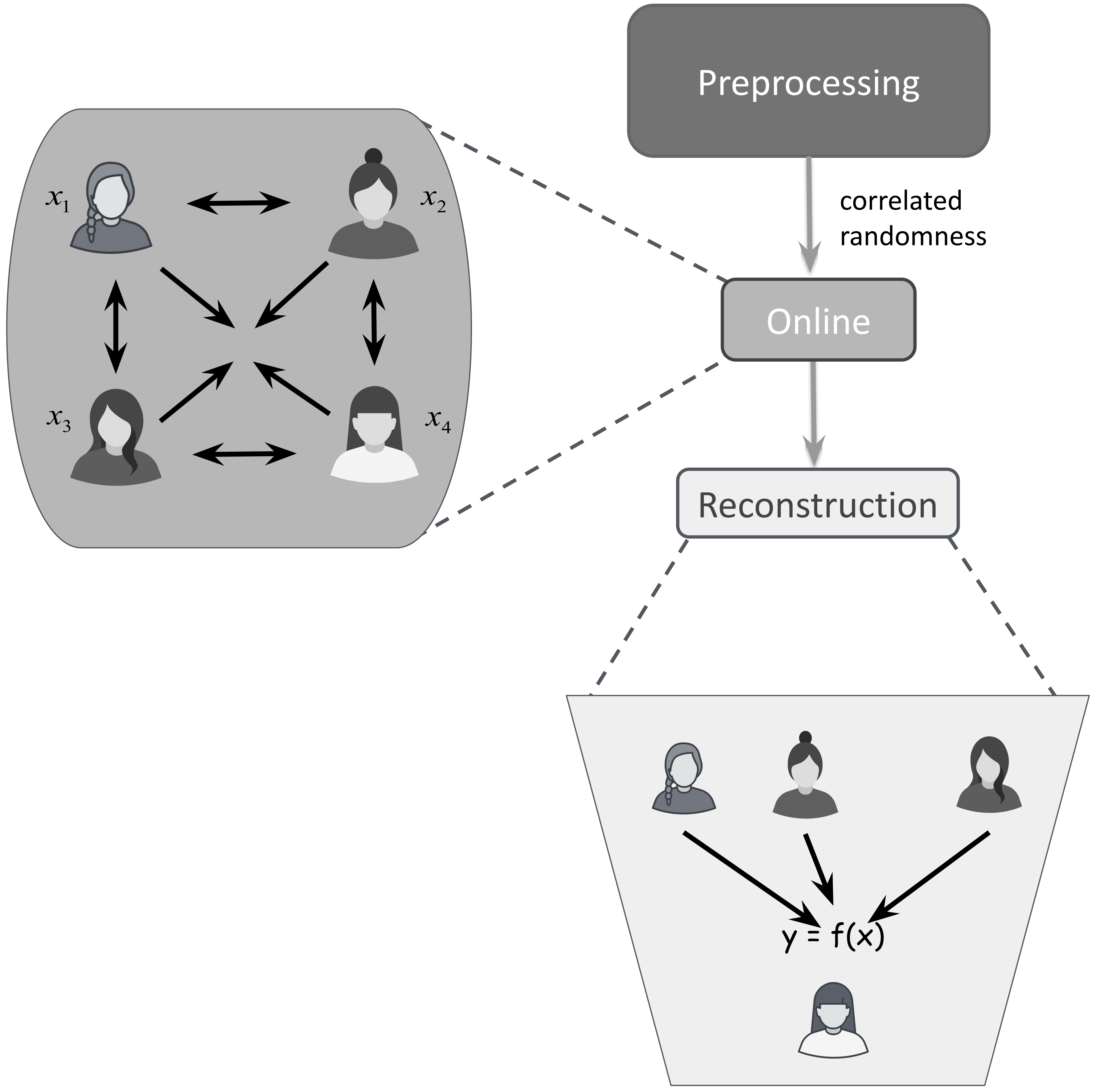
Multi-Party Computation

Preprocessing:
- Independent of x,y
- Typically only depends on size of f
- Uses public key crypto technology (slower)

Online:
- Uses only information theoretic tools
- Runs in order of magn. faster

Reconstruction:
- The intended parties learn the result of f

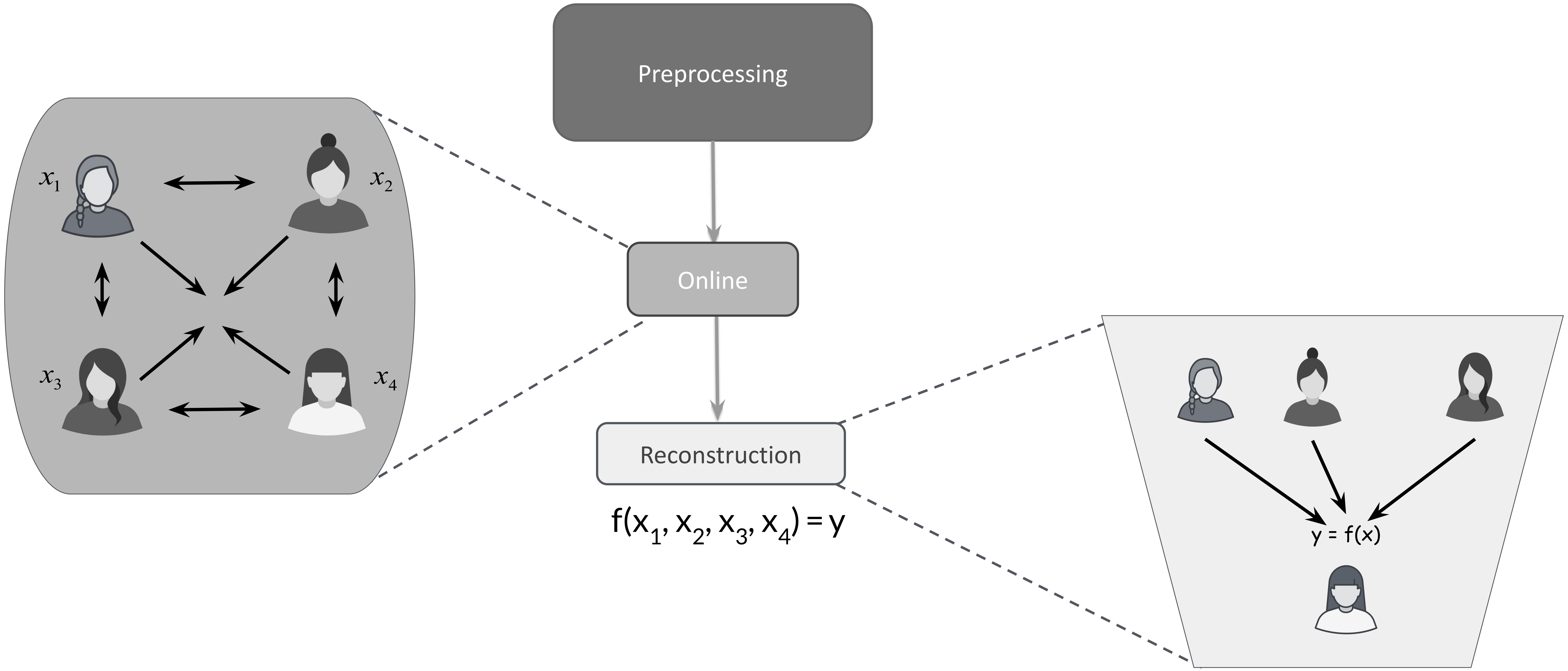
$$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) = y$$



MPC General View

Multi-Party Computation

$f(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4) = y$

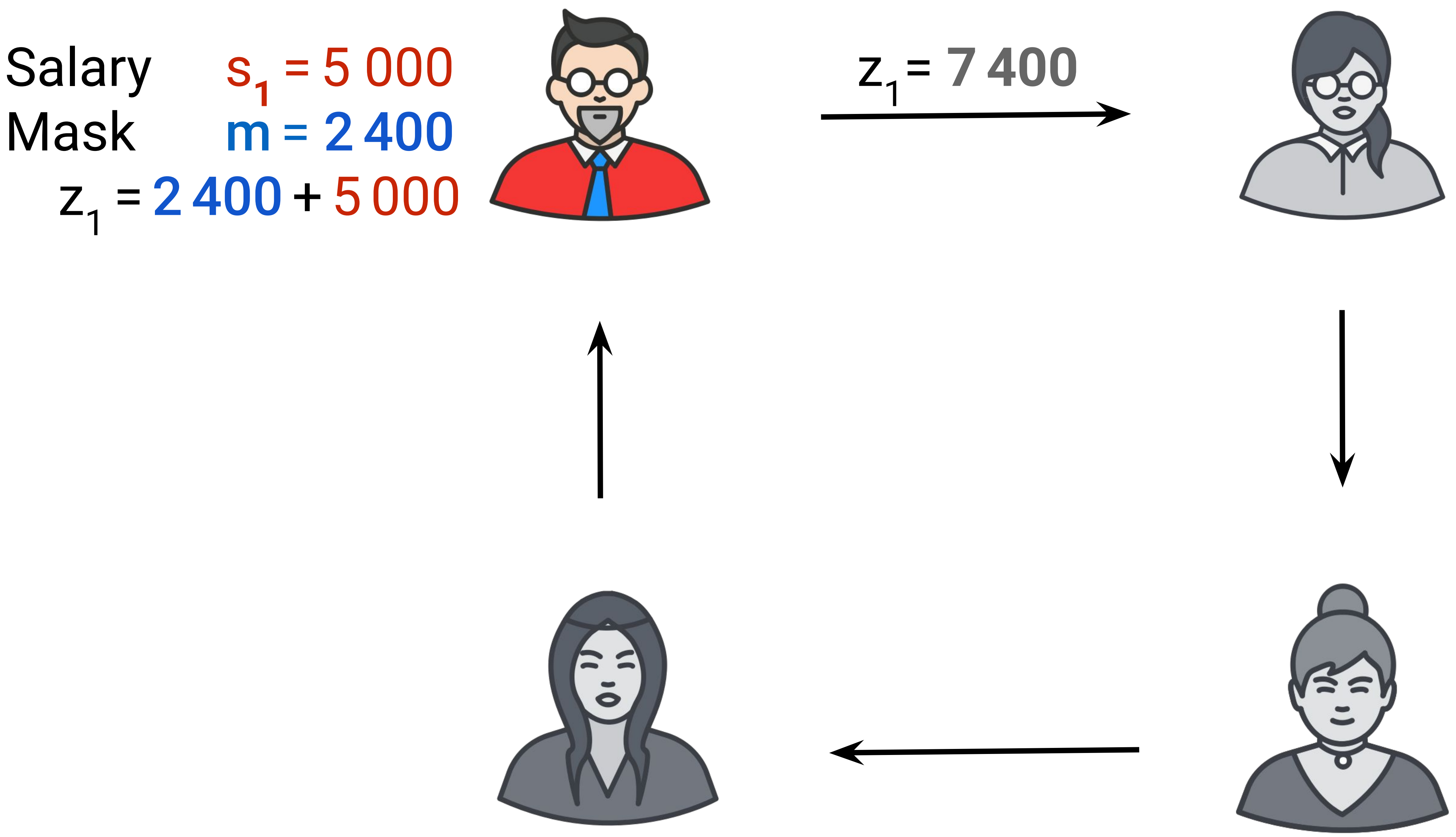


$f(x_1, x_2, x_3, x_4) = y$

MPC - Example

Exemple : Compute the average salary without revealing the salary of each employee.

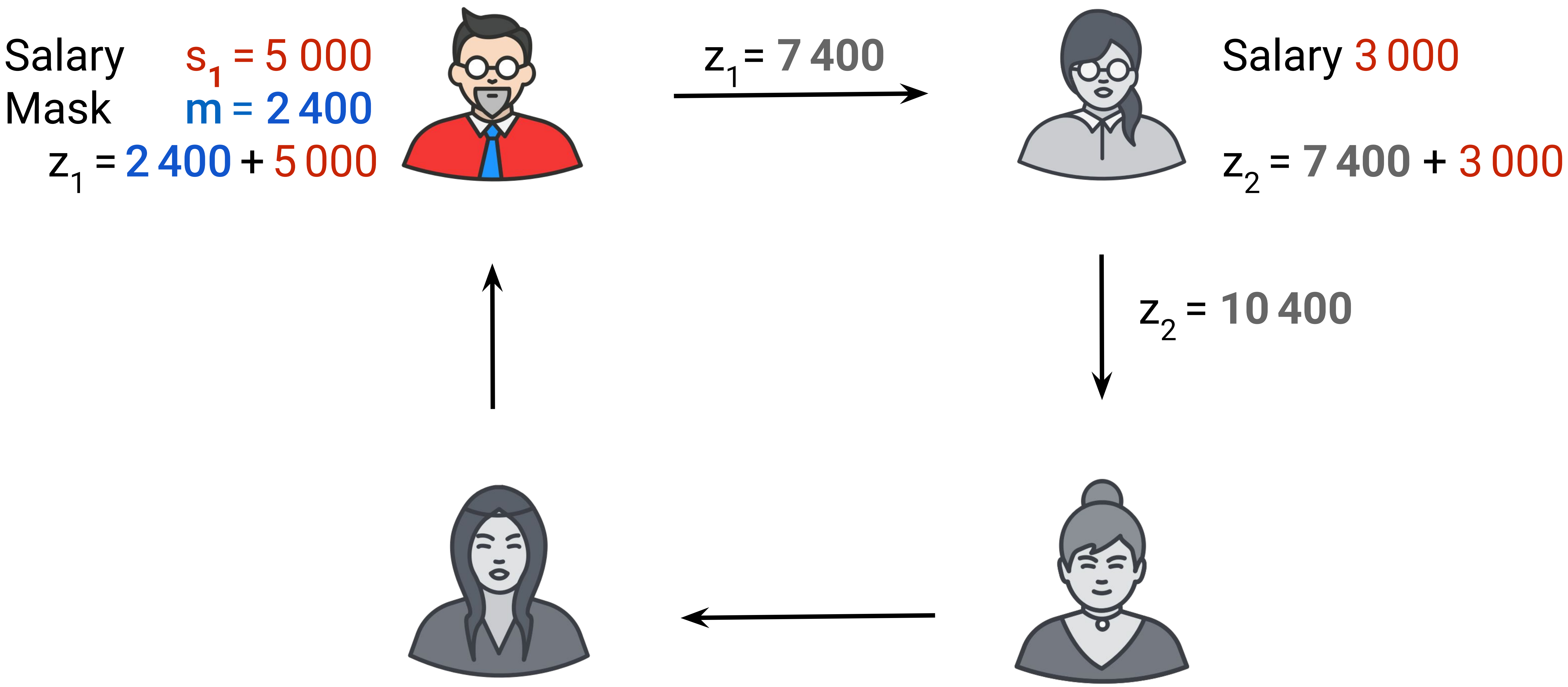
Here Bob runs a MPC protocol with his colleagues to learn the average salary.



MPC - Example

Exemple : Compute the average salary without revealing the salary of each employee.

Here Bob runs a MPC protocol with his colleagues to learn the average salary.

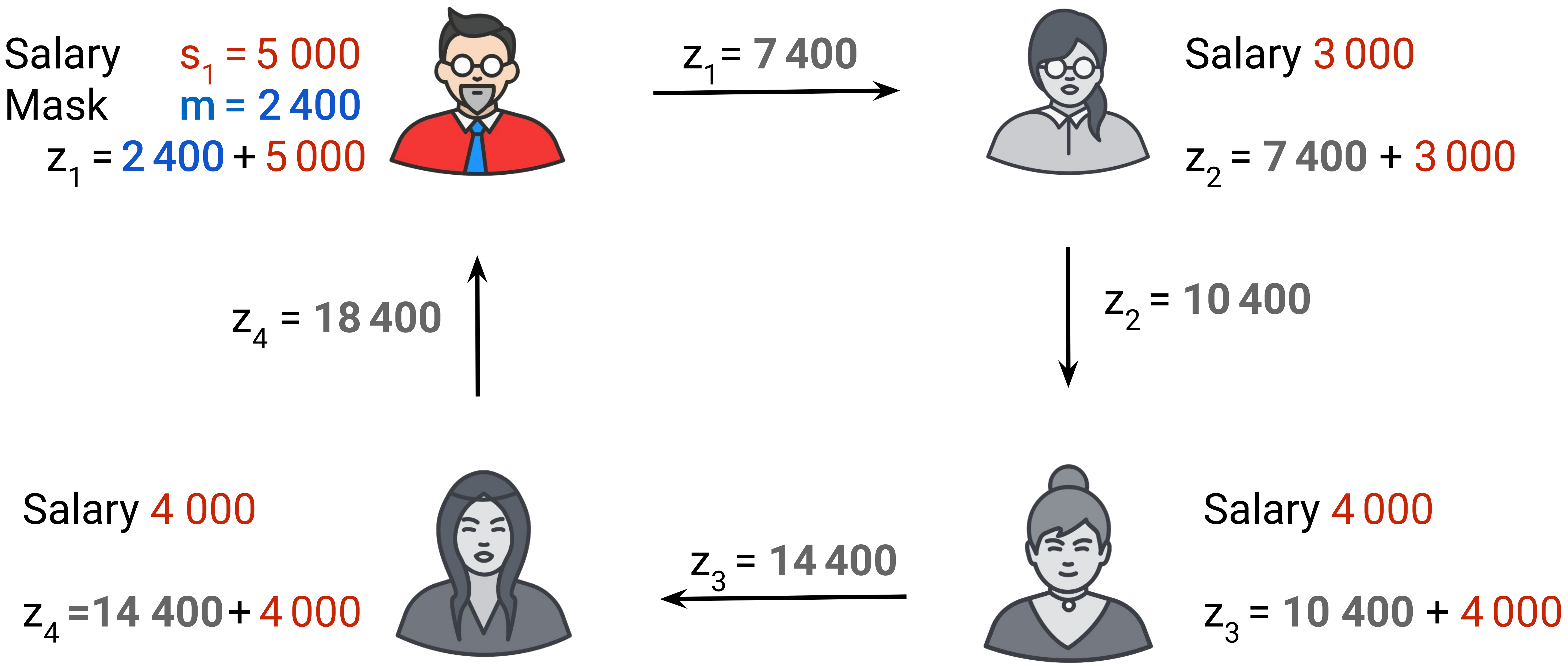


MPC - Example

Exemple : Compute the average salary without revealing the salary of each employee.

Here Bob runs a MPC protocol with his colleagues to learn the average salary.

What computation should Bob make in order to learn the average salary?



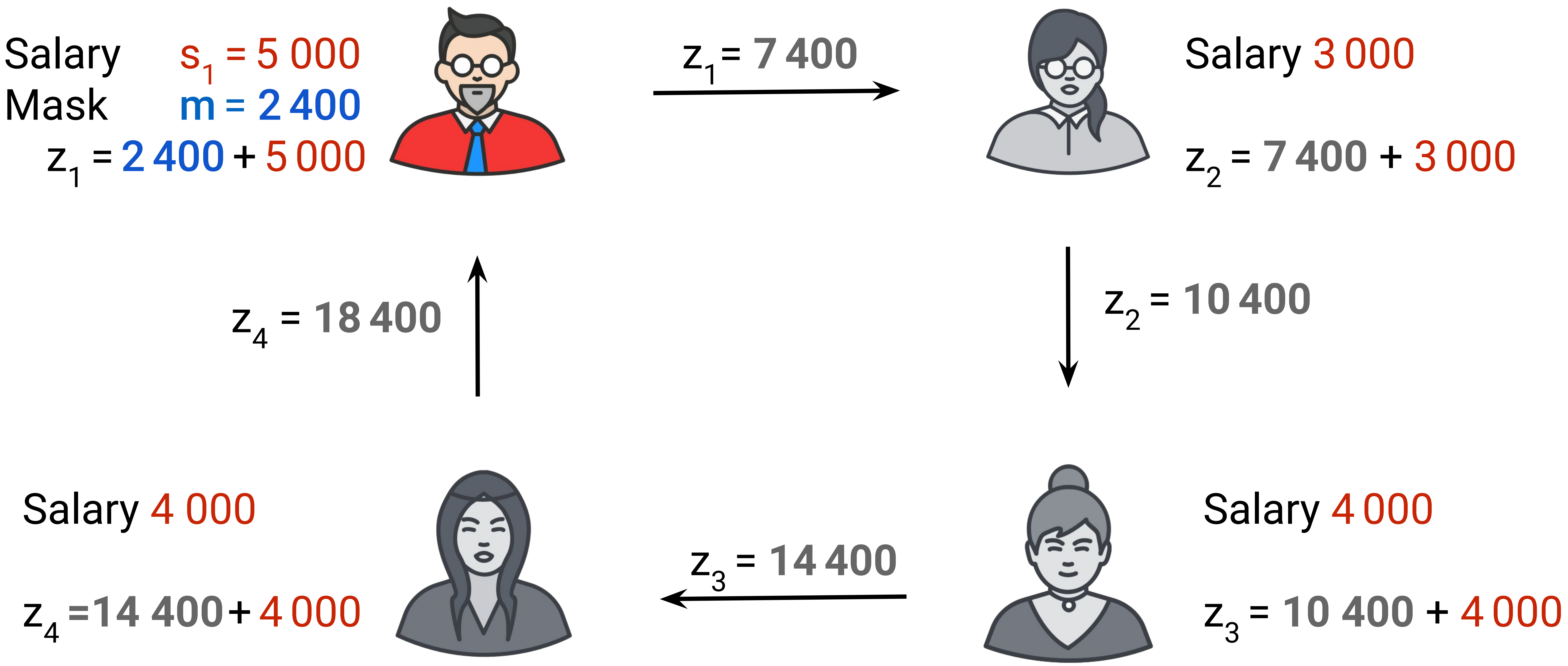
A. Average = $z_4/4$ = $18\,400/4$
B. Average = $z_4/4 - m$ = $18\,400/4 - 2\,400$
C. Average = $(z_4 - s_1)/4$ = $(18\,400 - 5\,000)/4$
D. Average = $(z_4 - m)/4$ = $(18\,400 - 2\,400)/4$

MPC - Example

Exemple : Compute the average salary without revealing the salary of each employee.

Here Bob runs a MPC protocol with his colleagues to learn the average salary.

What computation should Bob make in order to learn the average salary?



A. Average = $z_4/4 = 18\,400/4$

B. Average = $z_4/4 - m = 18\,400/4 - 2\,400$

C. Average = $(z_4 - s_1)/4 = (18\,400 - 5\,000)/4$

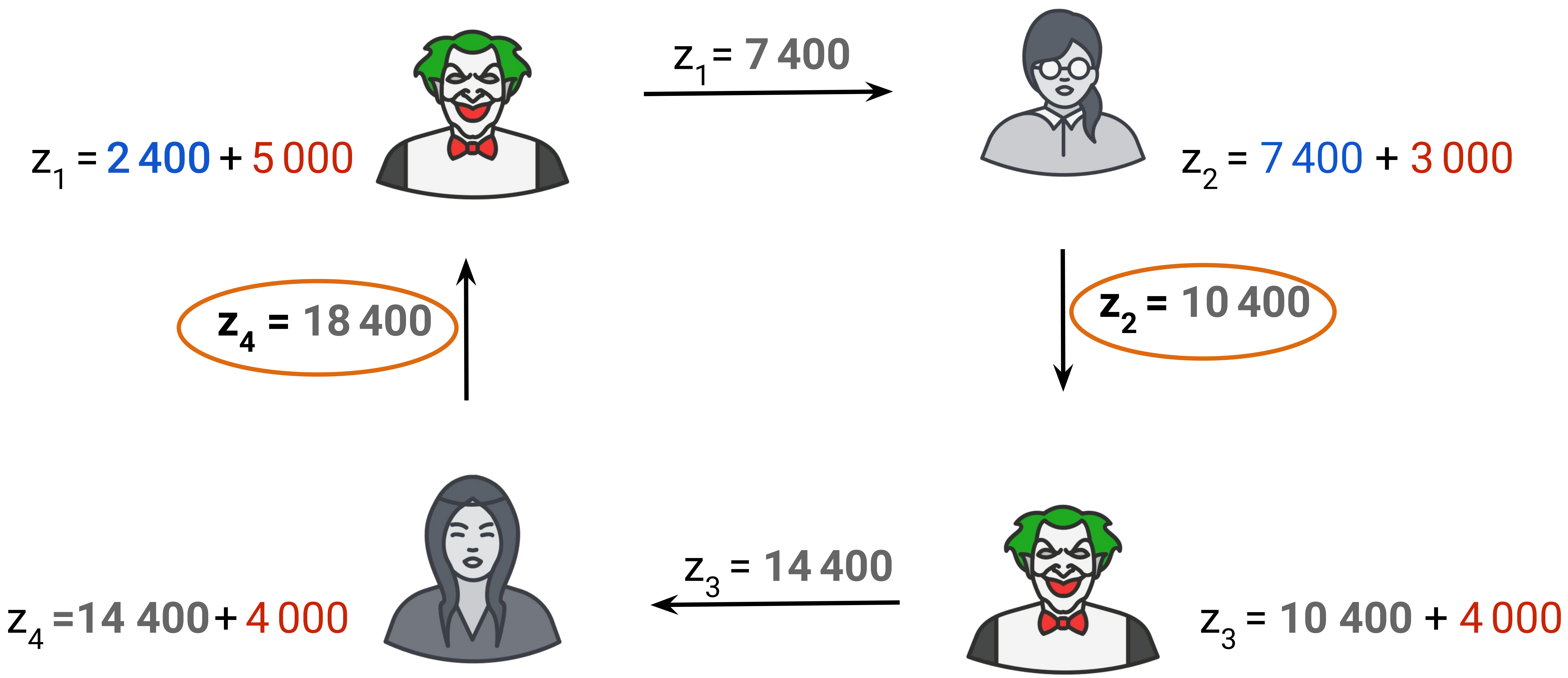
D. Average = $(z_4 - m)/4 = (18\,400 - 2\,400)/4$

Solution : D. Sum of salaries = **Masked Sum** – Random Mask = $z_4 - m$

MPC - Security

We need to make sure that an adversary cannot learn the private data of the honest parties.

We assume that **communication channels** are private and authenticated.



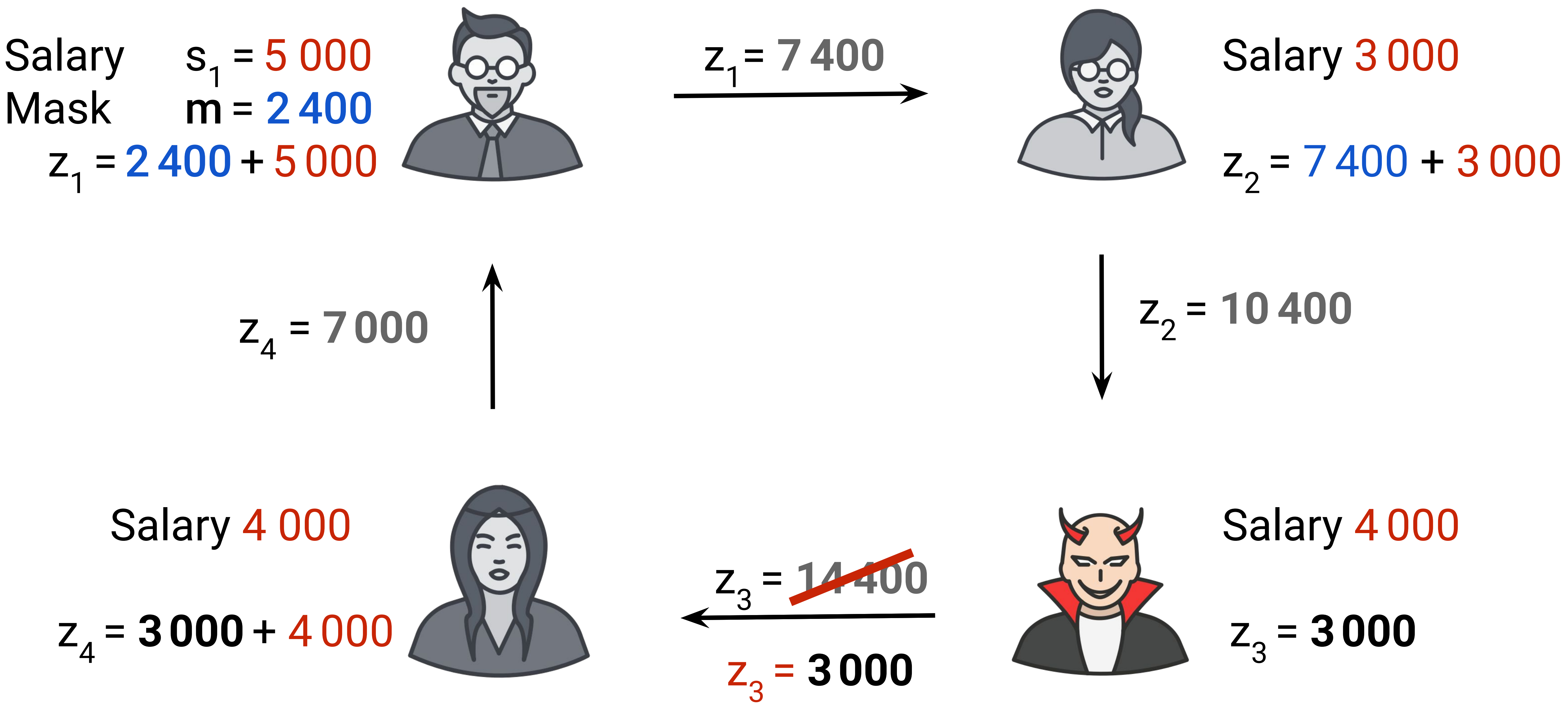
Security Model: The adversary is inside the system and corrupts some of the parties

- **Semi-honest (passive)** – Corrupted players follow the protocol but try to learn more → private computation

MPC - Security

We need to make sure that an adversary cannot learn the private data of the honest parties.

We assume that **communication channels** are private and authenticated.



Security Model: The adversary is inside the system and corrupts some of the parties

- **Semi-honest (passive)** – Corrupted players follow the protocol but try to learn more → private computation
- **Malicious (active)** – Corrupted players can collaborate in any way and misbehave arbitrarily → secure computation

MPC - Security

Security is defined by the types of attacks considered:

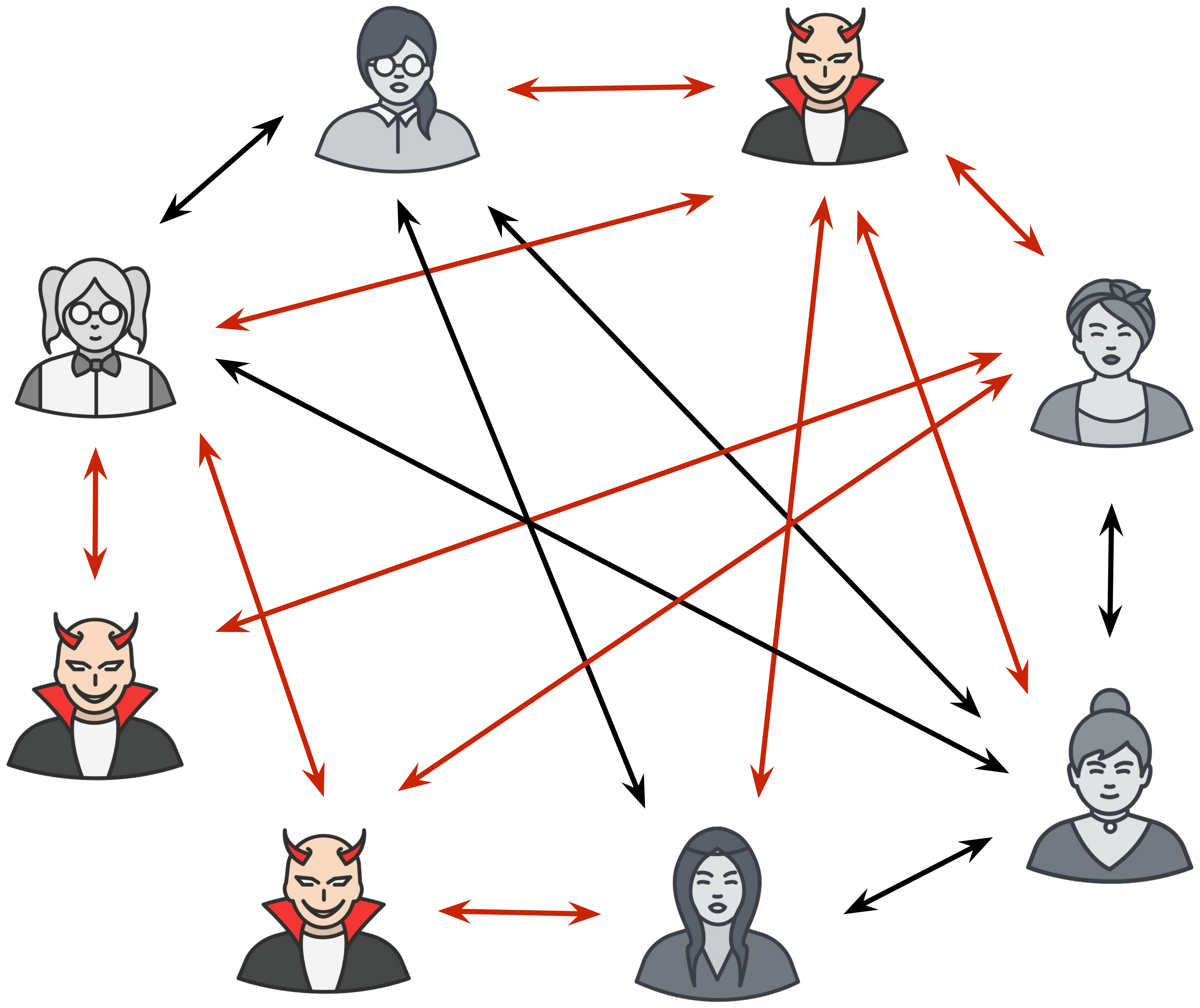
Adversary type:

- **Semi-honnête (passive)**

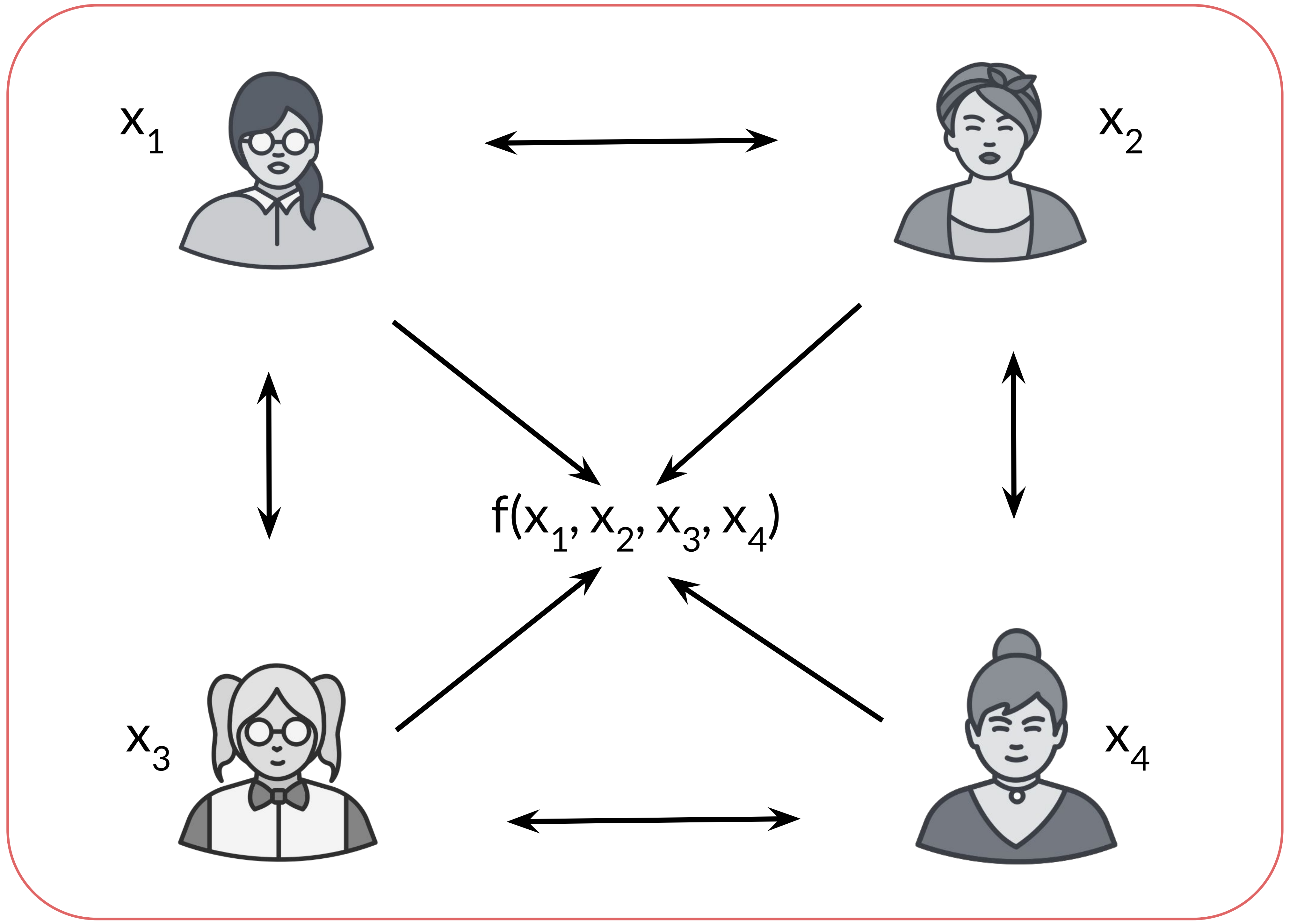
- **Malicieux (actif)**

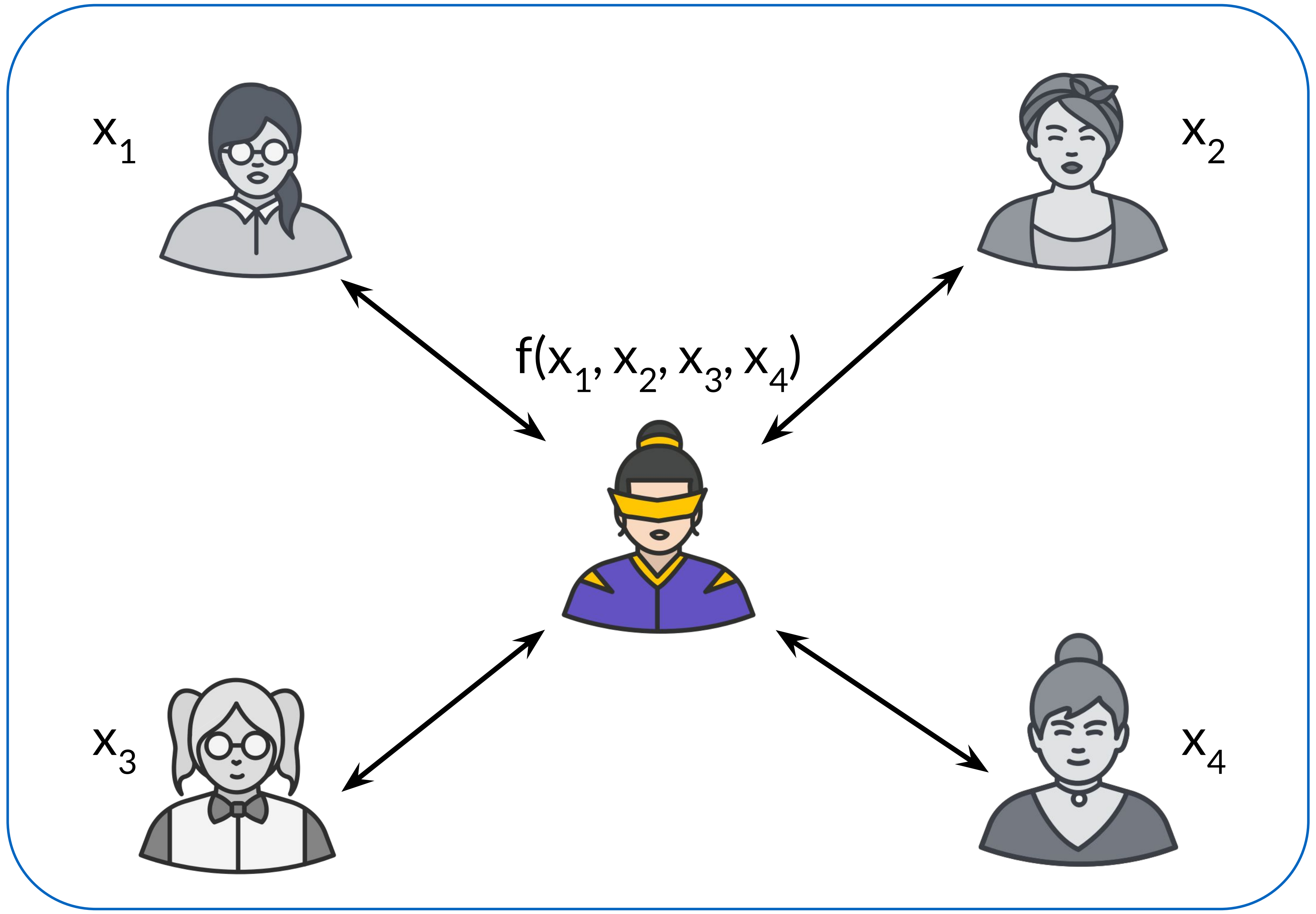
Number of corrupted parties t amoung n

- **Honest Majority** $t < n/2$
- **Dishonest Majority** $t \geq n/2$



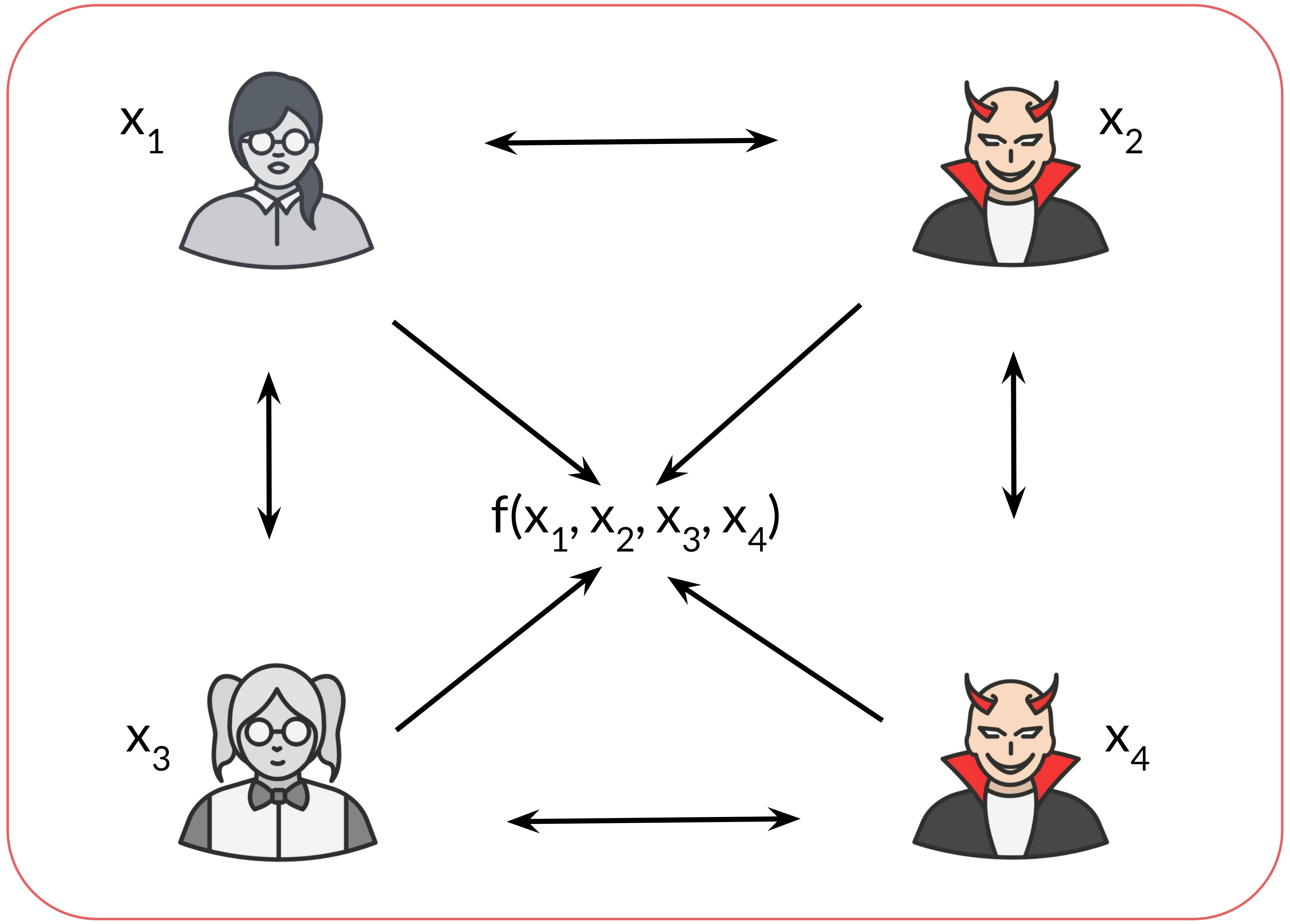
MPC - Real vs. Ideal Paradigm

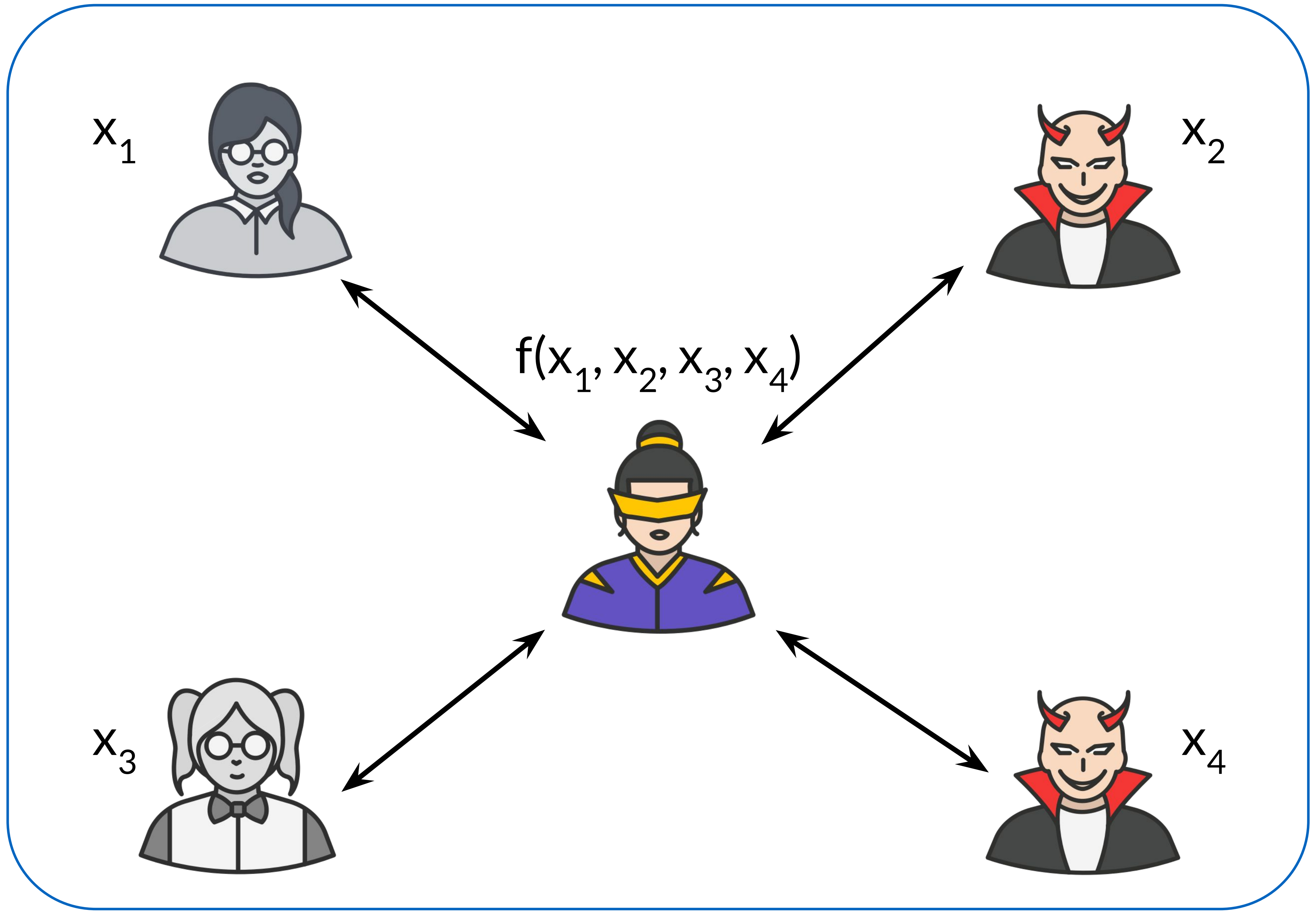




- **Real model:** - parties run a real protocol with no trusted help
- **Ideal model:** - parties send inputs to a trusted party T
  - T computes the function and sends the outputs

MPC - Paradigme monde réel $\leftrightarrow$ monde idéal





The adversary should not be able to do more damage in the real model than he is allowed in the ideal model

Semi-honest (passive)

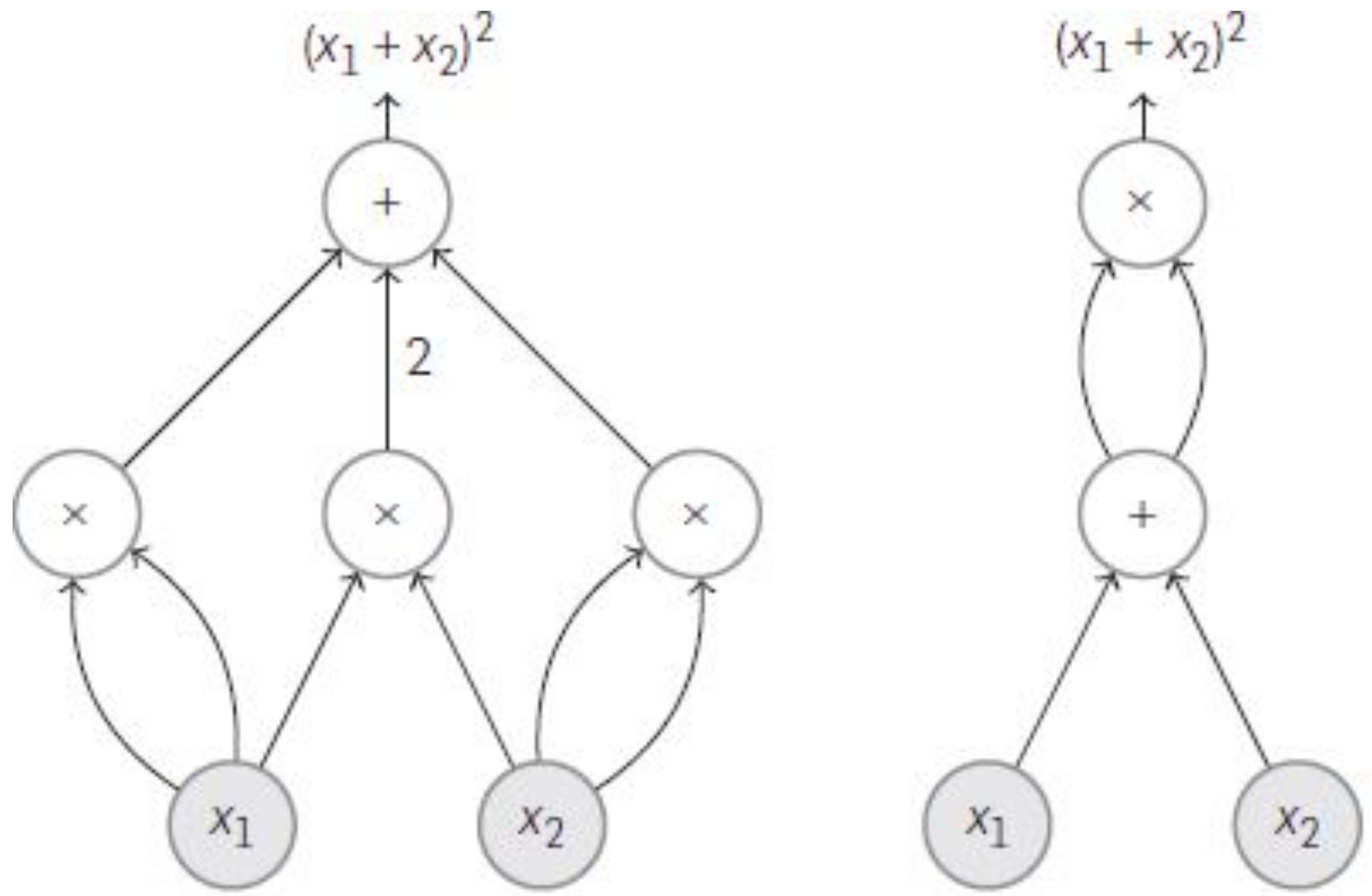
Malicious (active)

A MPC protocol is secure if any attack on a real protocol can be carried out (or simulated) in the ideal model.


MPC
from
secret
sharing

MPC - Arithmetic Circuits

Model of Computation: Represent the function as **Arithmetic**/Boolean circuit C
Limitting Factor: Number of multiplications

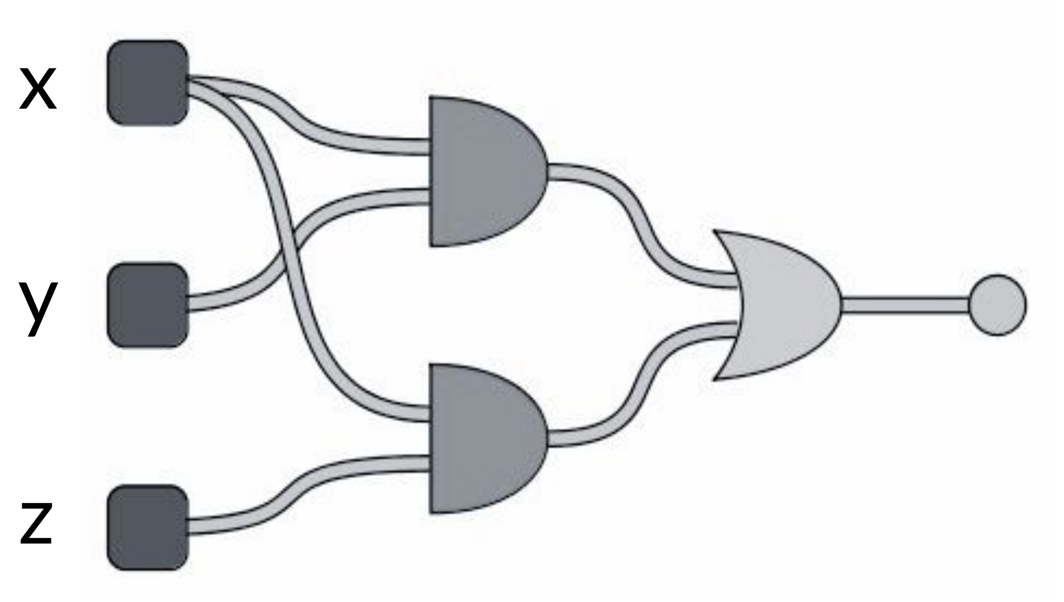


MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit

f(x,y,z)

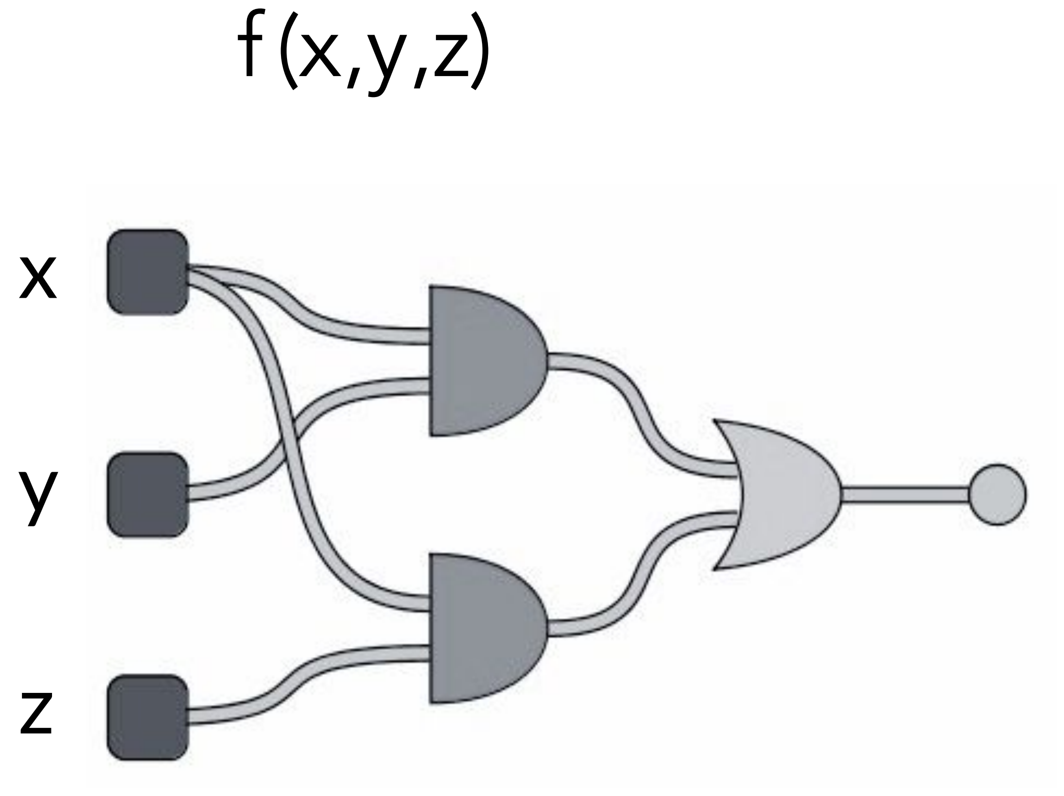


*[GMW87] Oded Goldreich, Silvio Micali, Avi Wigderson. How to prove all NP- statements in zero-knowledge, and a methodology of cryptographic protocol design.
[BGW88] M. Ben-Or, S. Goldwasser, A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation.

MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input



$x = x_1 + x_2 + x_3$

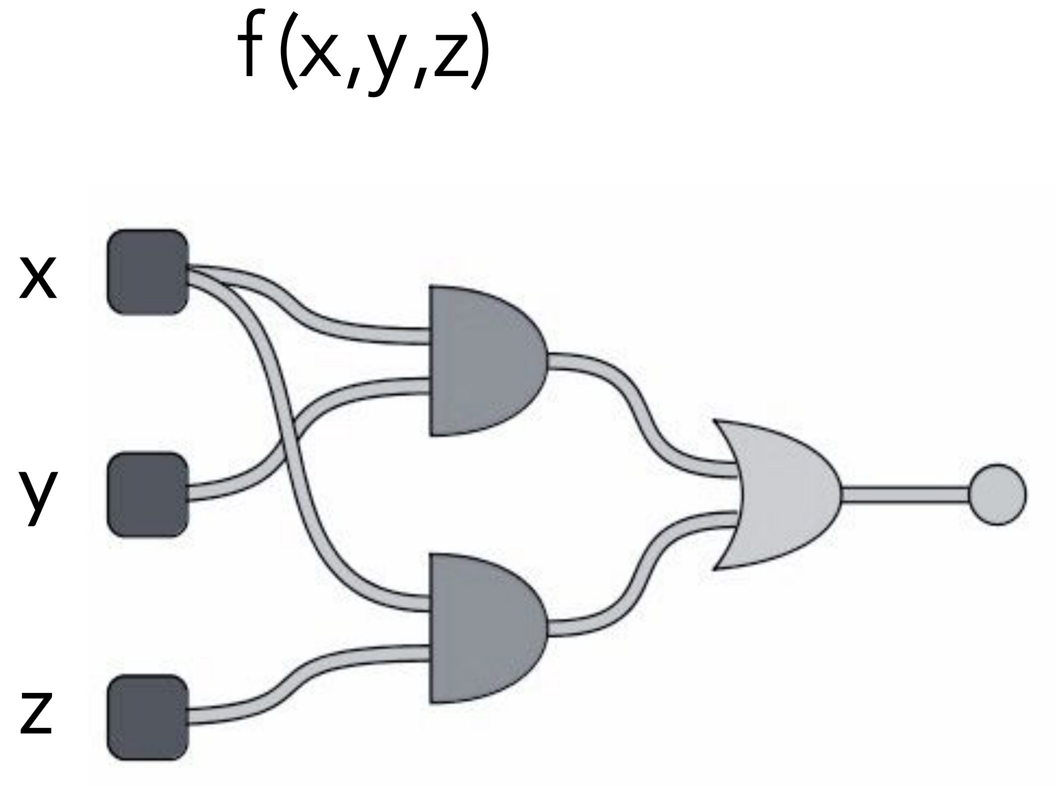
$z = z_1 + z_2 + z_3$

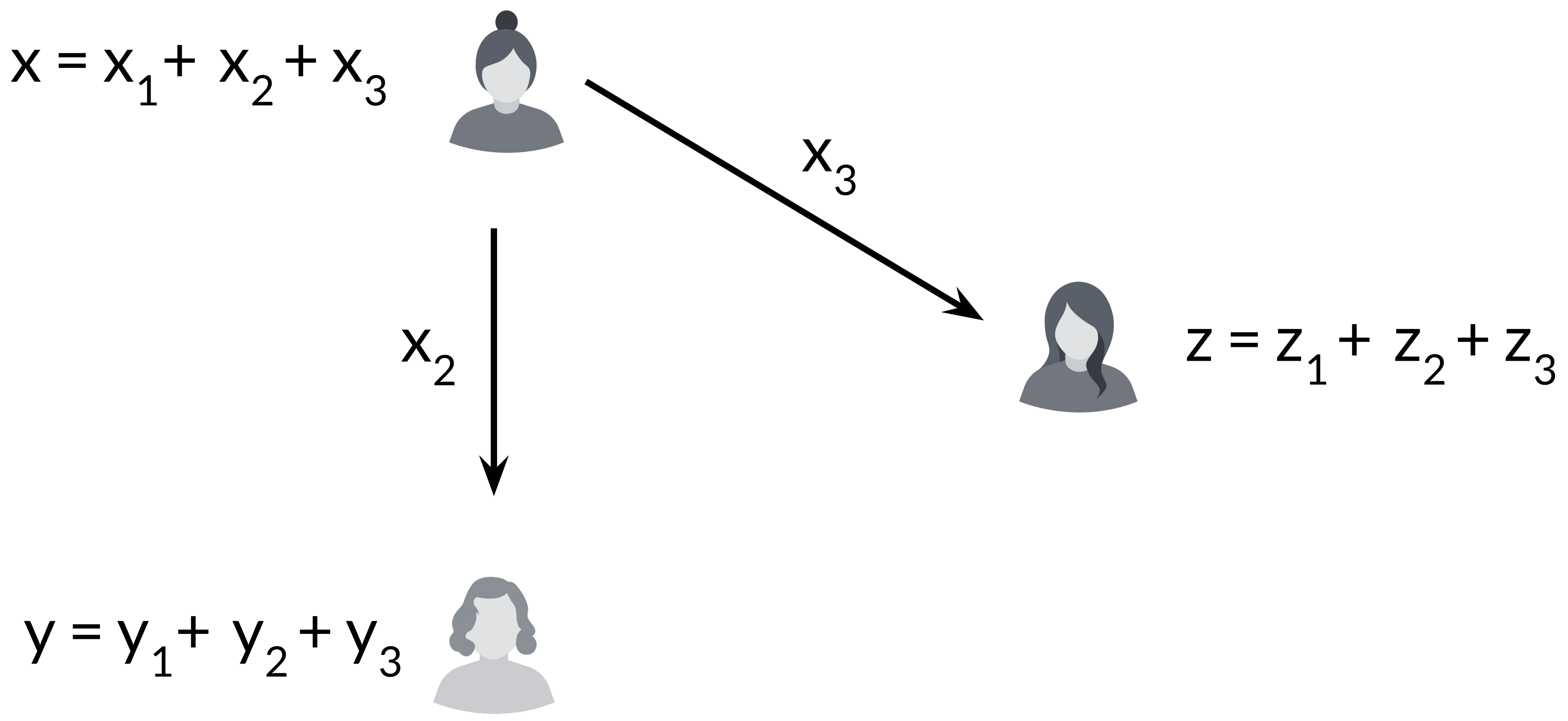
$y = y_1 + y_2 + y_3$

MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input

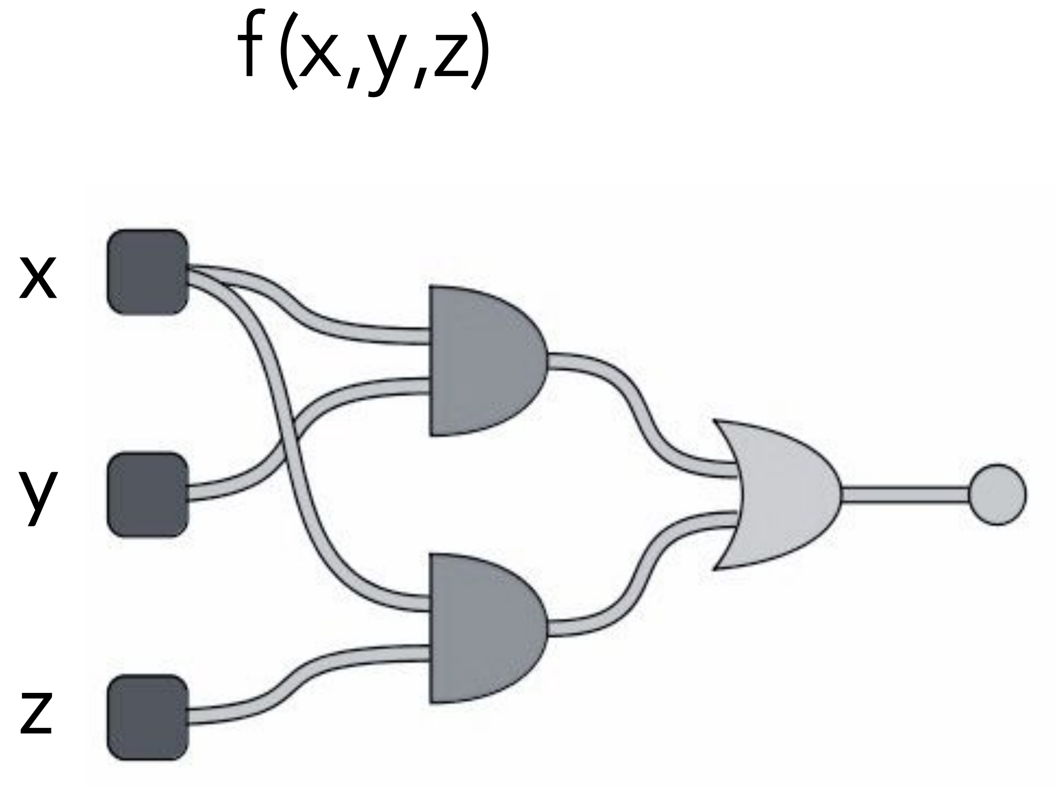


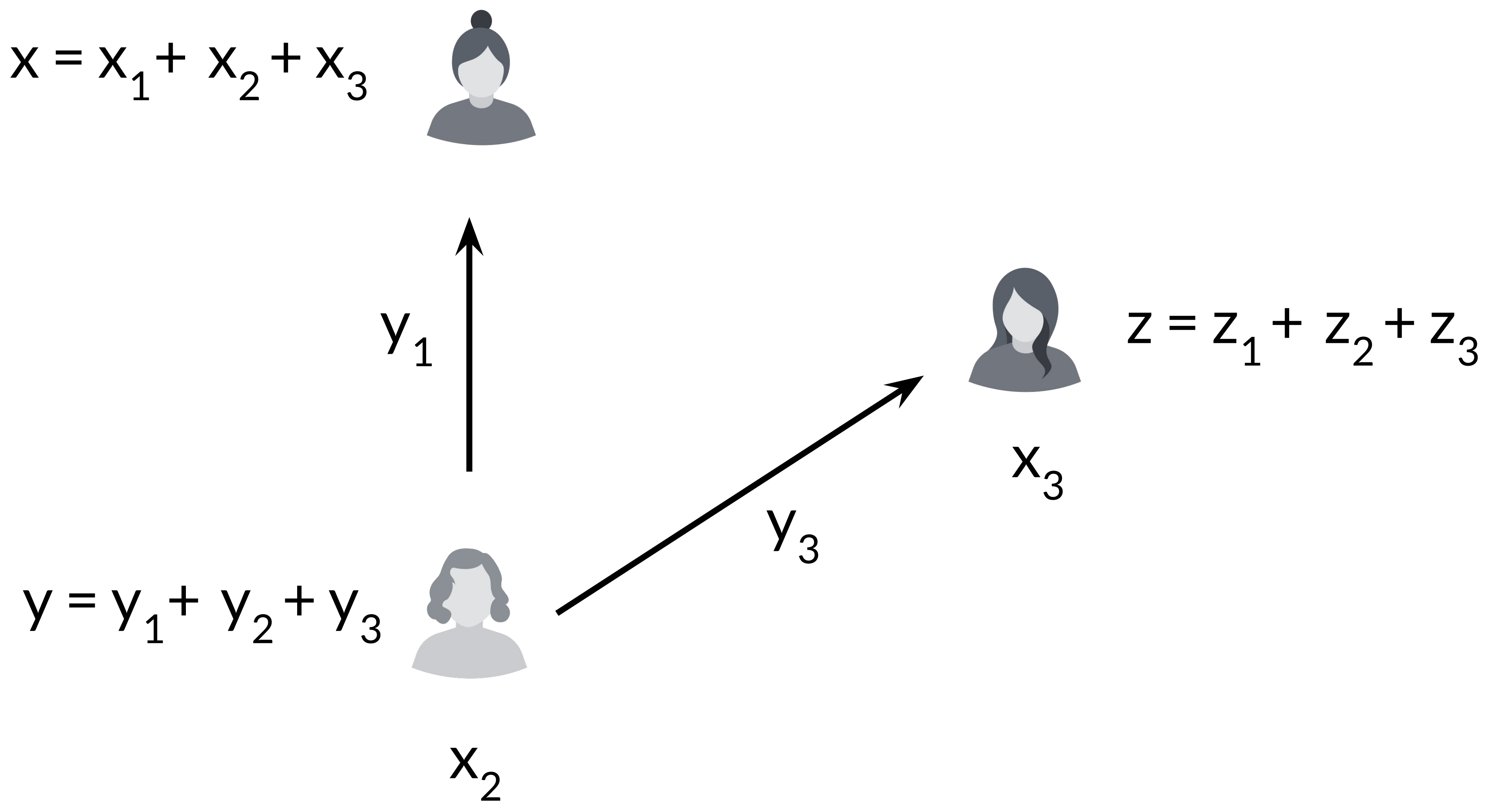


MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input

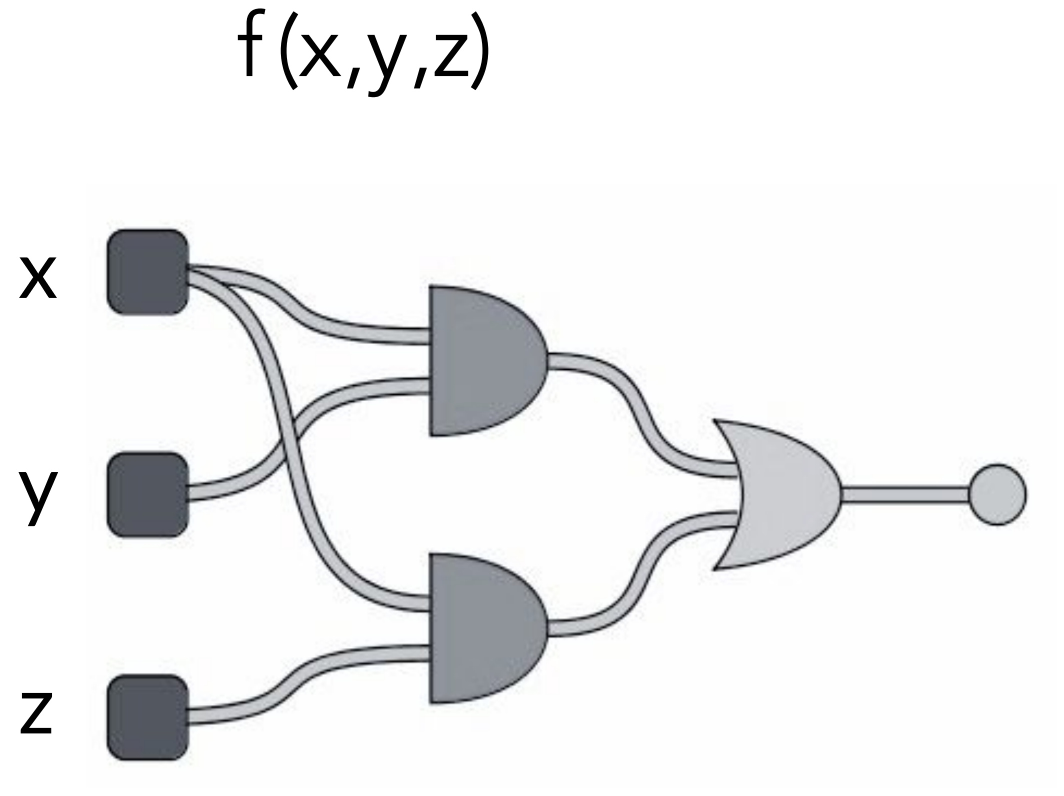


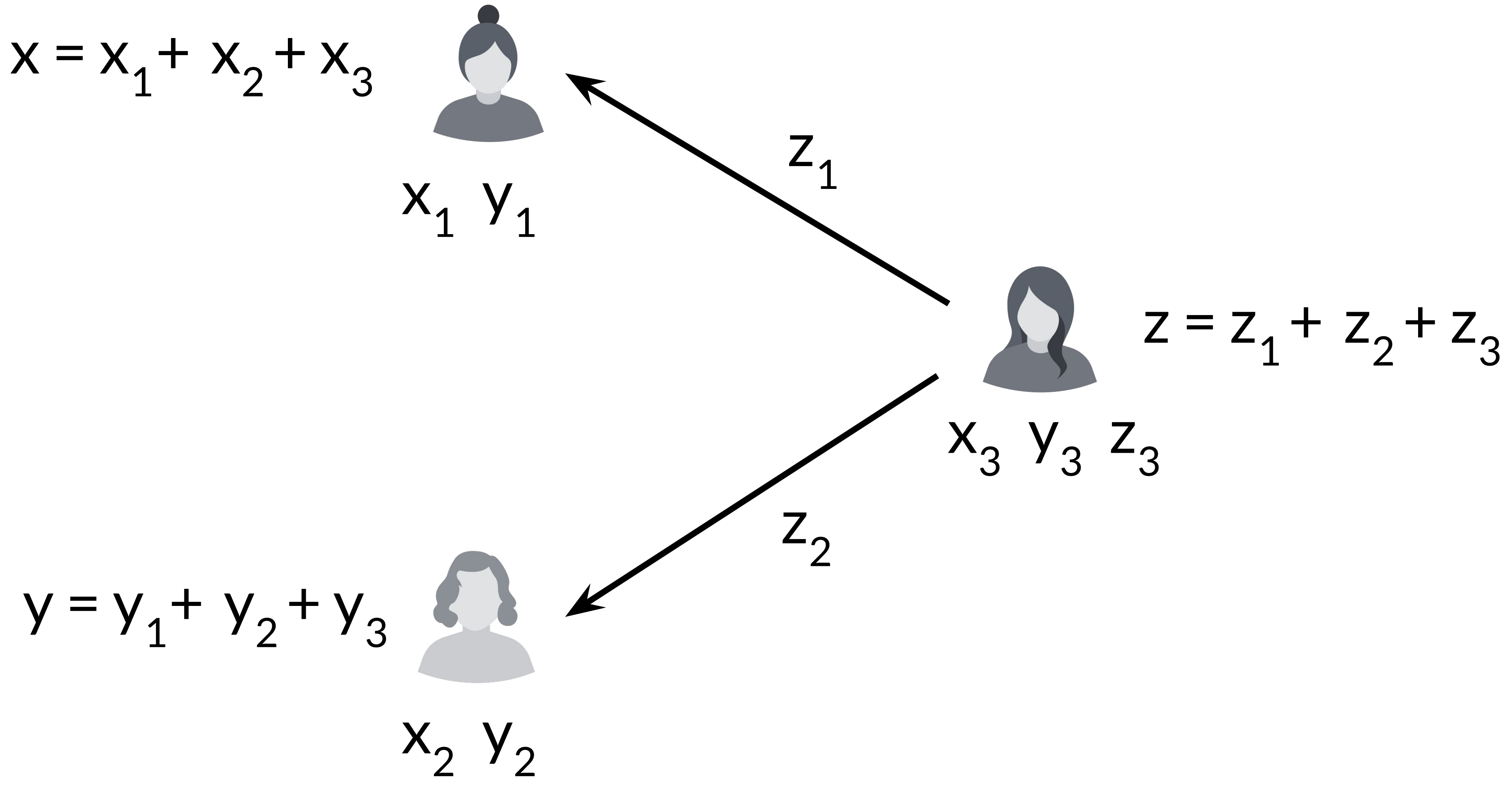


MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input

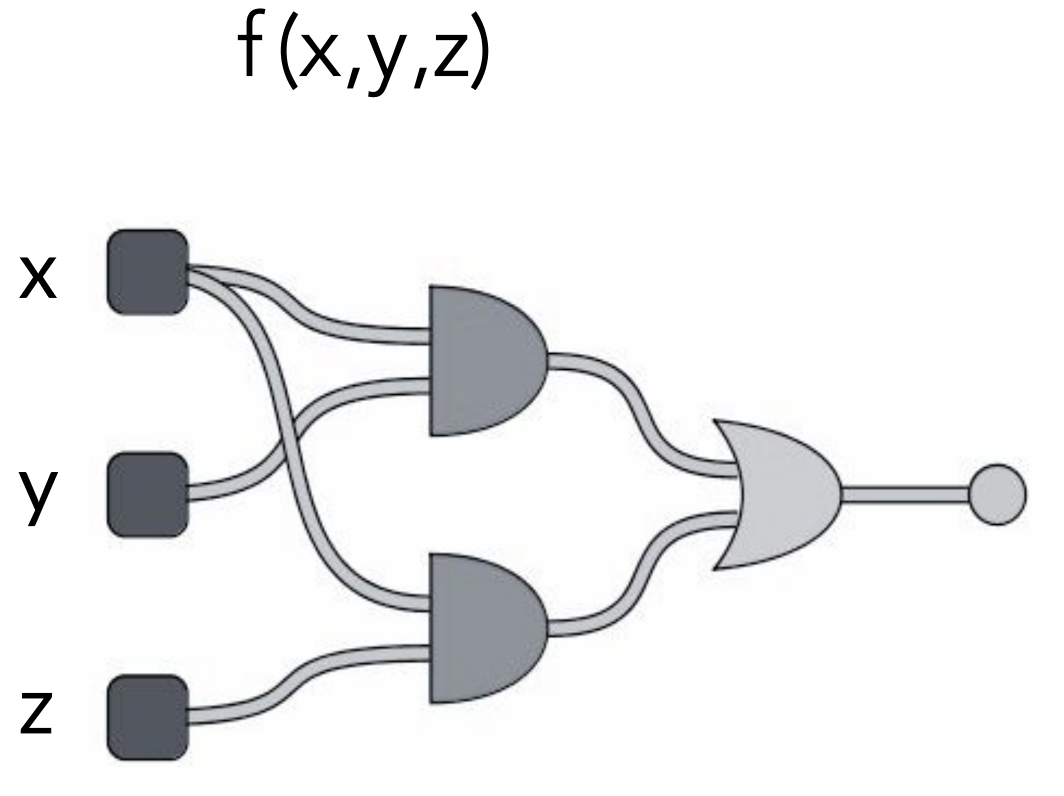


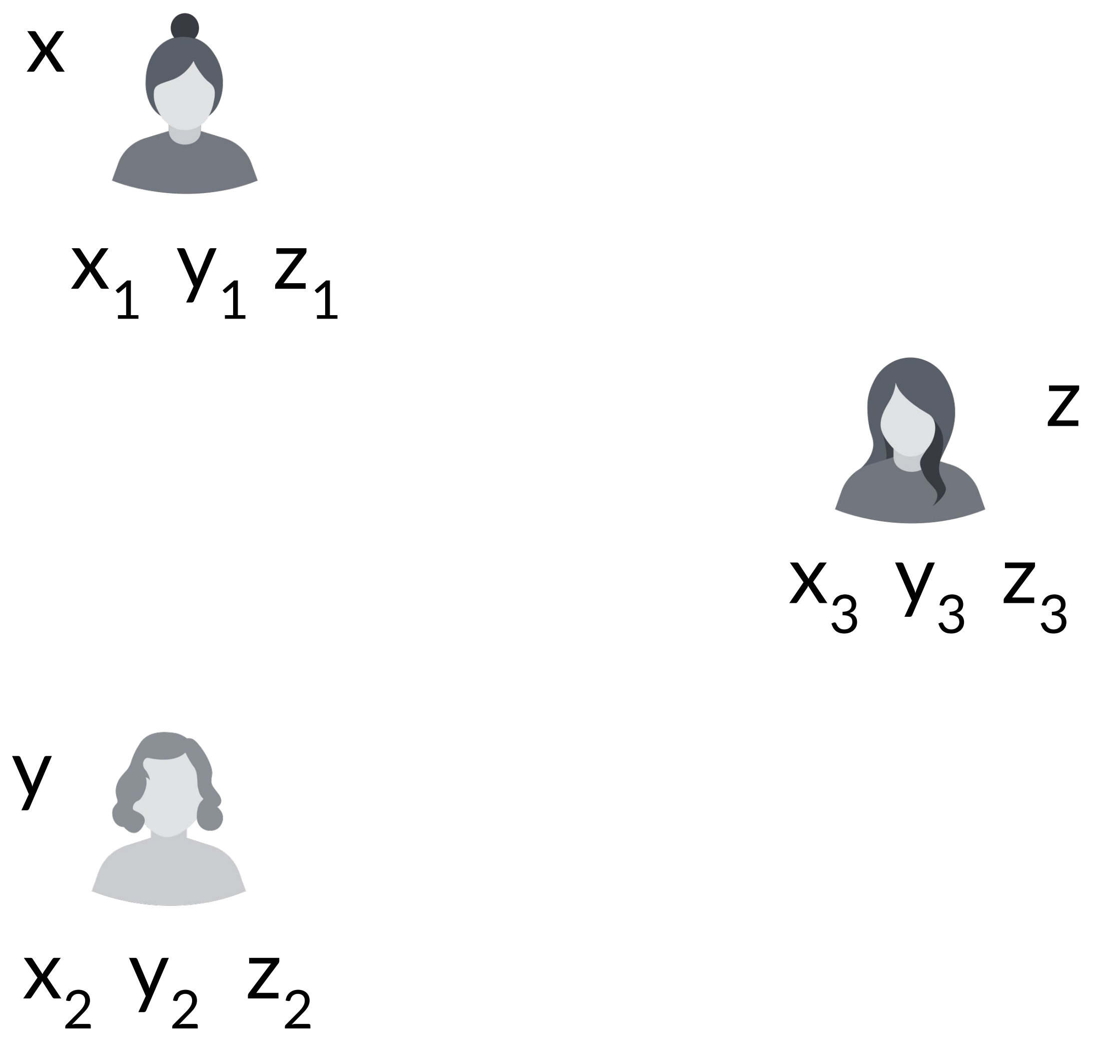


MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)

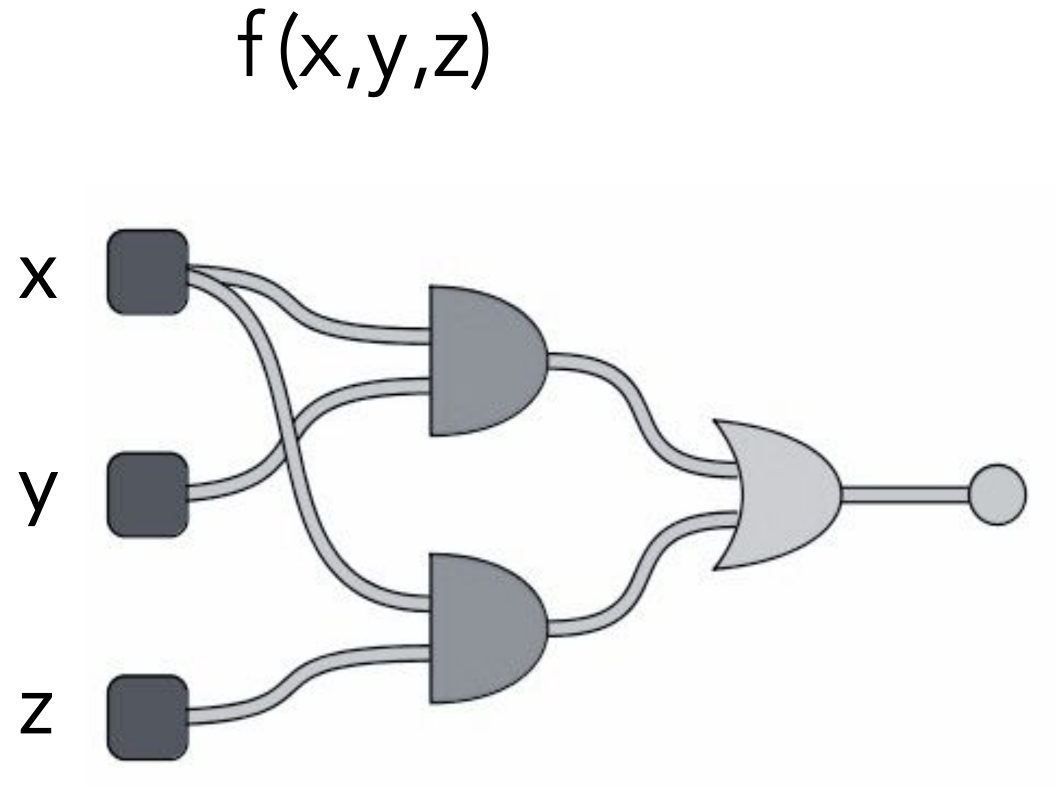


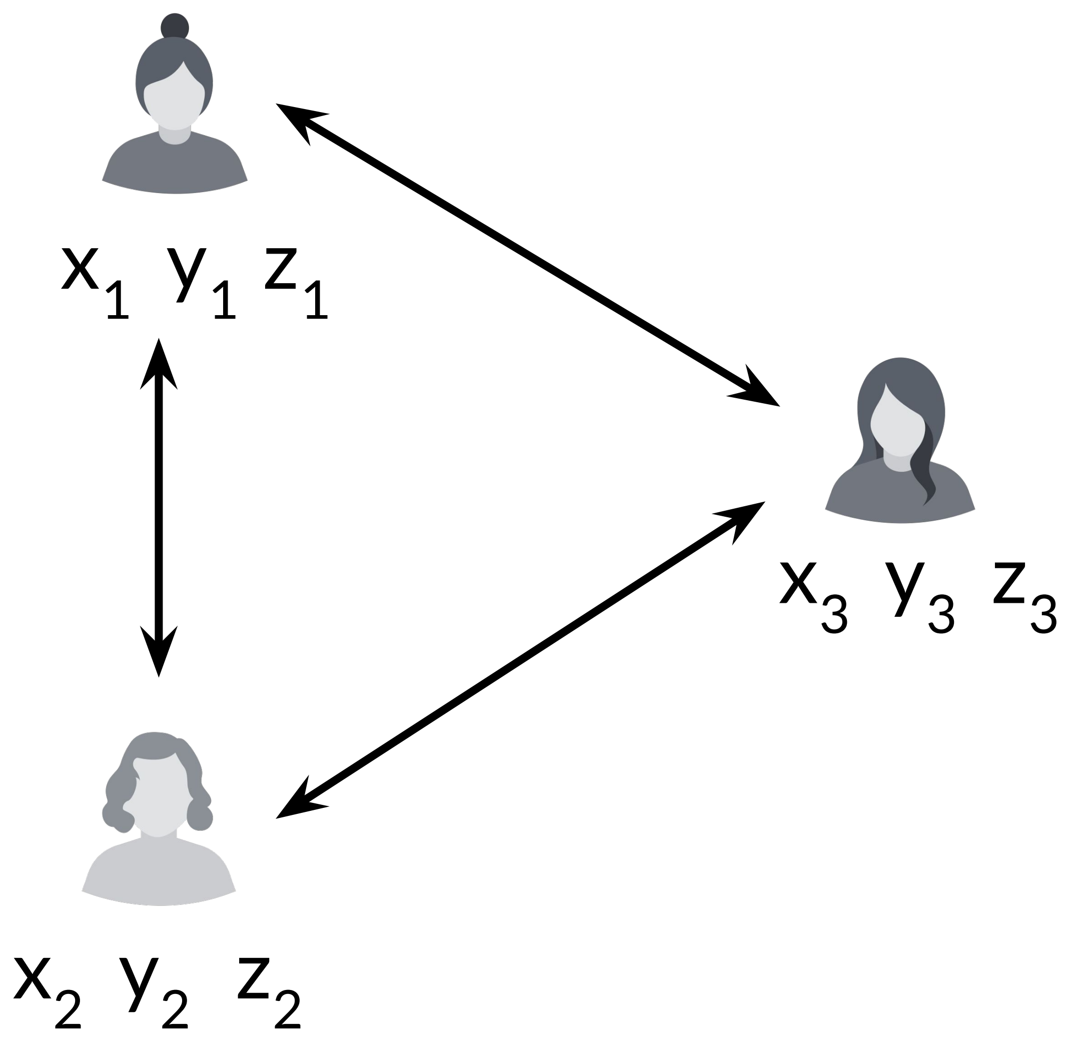


MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)





MPC from Secret Sharing

The GMW/BGW* Approach:

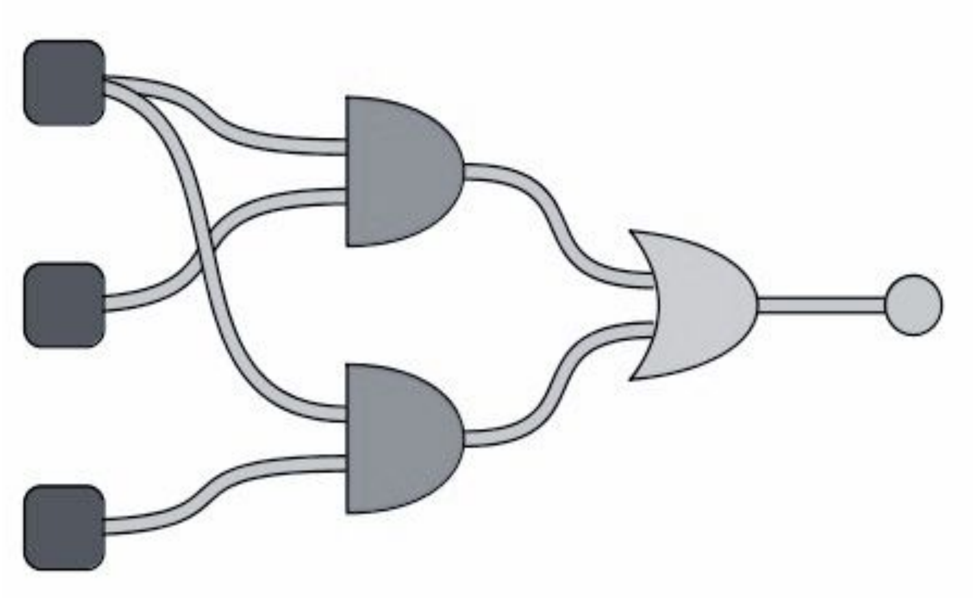
- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)
- Addition is possible without interaction → $S_i = x_i + y_i + z_i$

$f(x,y,z)$

$x_1 + x_2 + x_3 = x$

$y_1 + y_2 + y_3 = y$

$z_1 + z_2 + z_3 = z$



$S_1 = x_1 + y_1 + z_1$

x_1 y_1 z_1

$S_3 = x_3 + y_3 + z_3$

x_3 y_3 z_3

$S_2 = x_2 + y_2 + z_2$

x_2 y_2 z_2

MPC from Secret Sharing

$$\begin{array}{r l} & S_1 = x_1 + y_1 + z_1 \\ & S_2 = x_2 + y_2 + z_2 \\ & S_3 = x_3 + y_3 + z_3 \\ \hline + & S_1 + S_2 + S_3 = x + y + z \end{array}$$

The GMW/BGW* Approach:

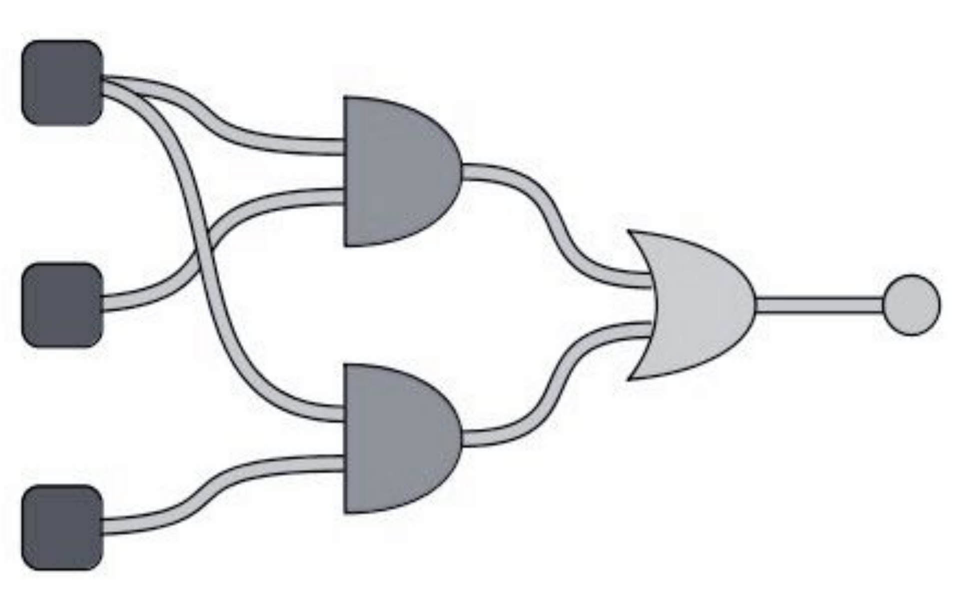
- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares
- Addition is possible without interaction → $S_i = x_i + y_i + z_i$ → $S = x + y + z = S_1 + S_2 + S_3$

$f(x,y,z)$

$x_1 + x_2 + x_3 = x$

$y_1 + y_2 + y_3 = y$

$z_1 + z_2 + z_3 = z$



$S_1 = x_1 + y_1 + z_1$

$x_1 \; y_1 \; z_1$

$S_3 = x_3 + y_3 + z_3$

$x_3 \; y_3 \; z_3$

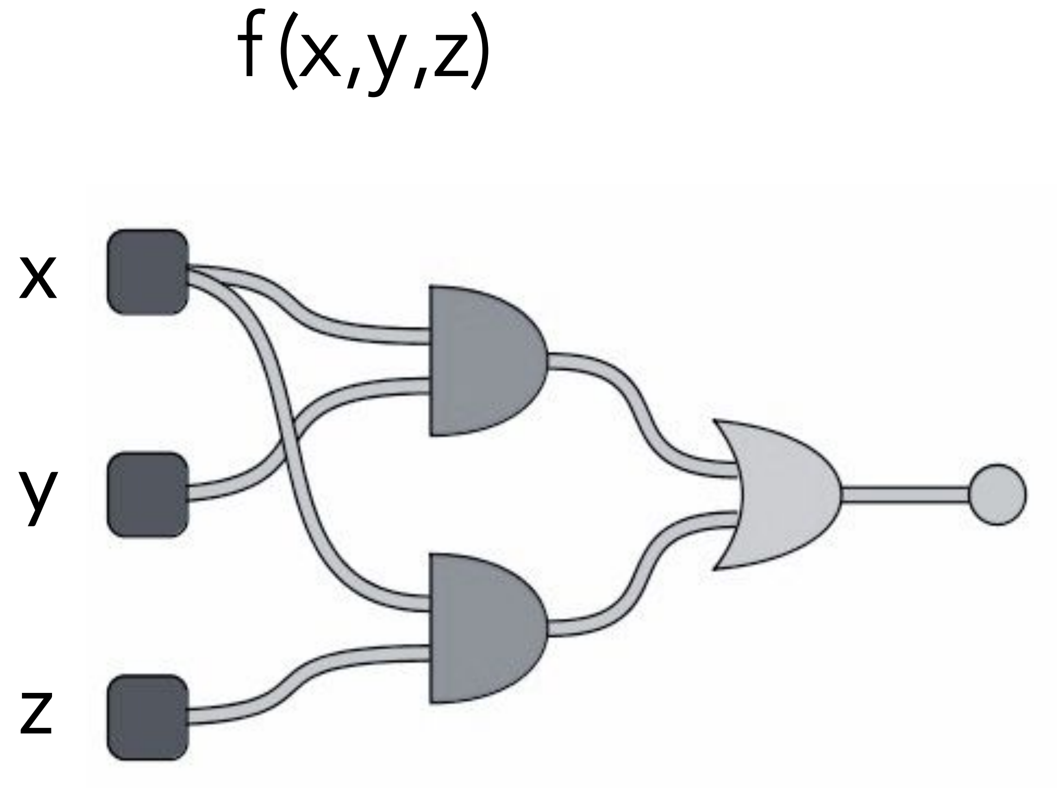
$S_2 = x_2 + y_2 + z_2$

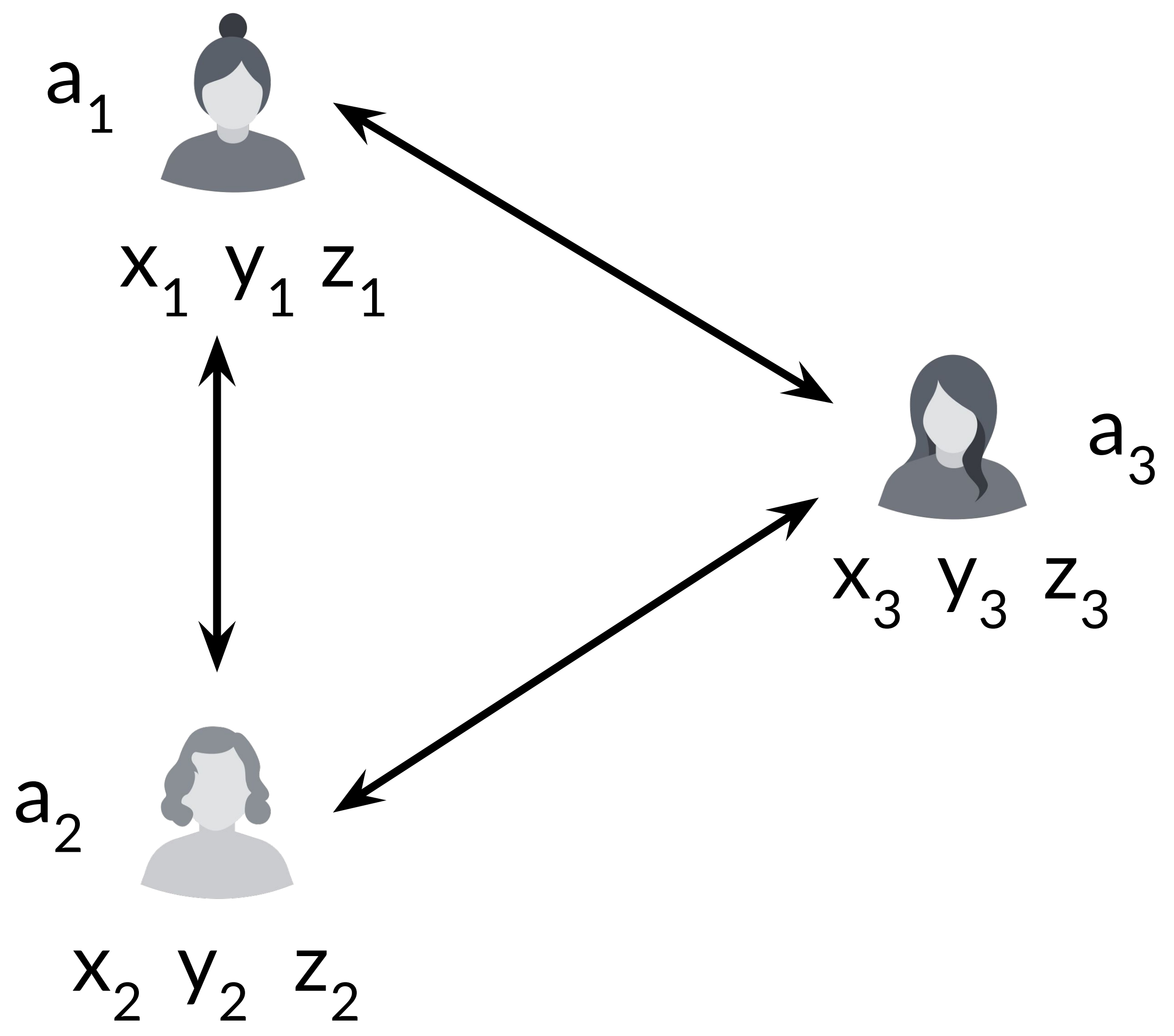
$x_2 \; y_2 \; z_2$

MPC from Secret Sharing

The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)
- Answer is reconstructed from final shares

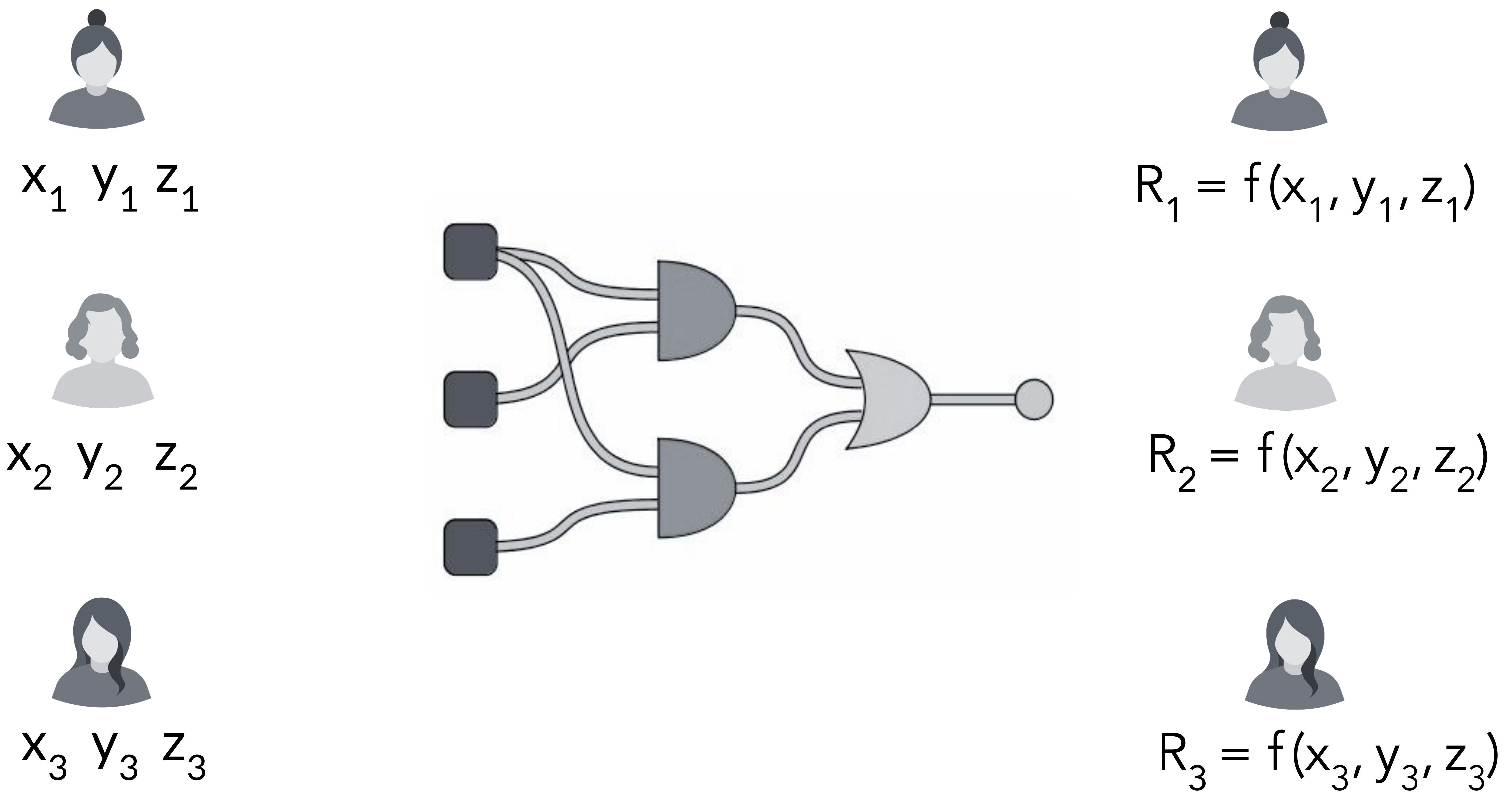




MPC from Secret Sharing

The GMW/BGW* Approach:

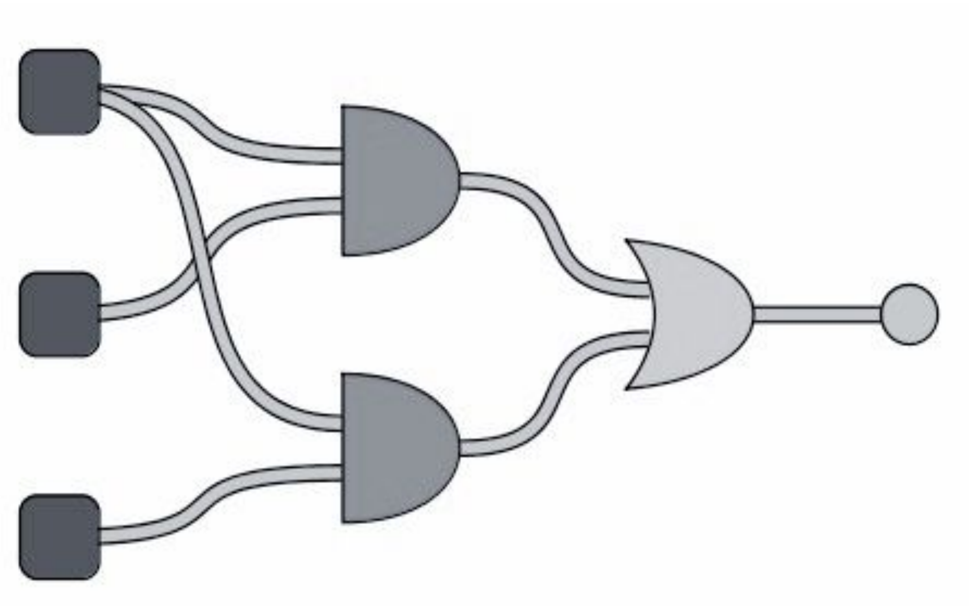
- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)
- Answer is reconstructed from final shares

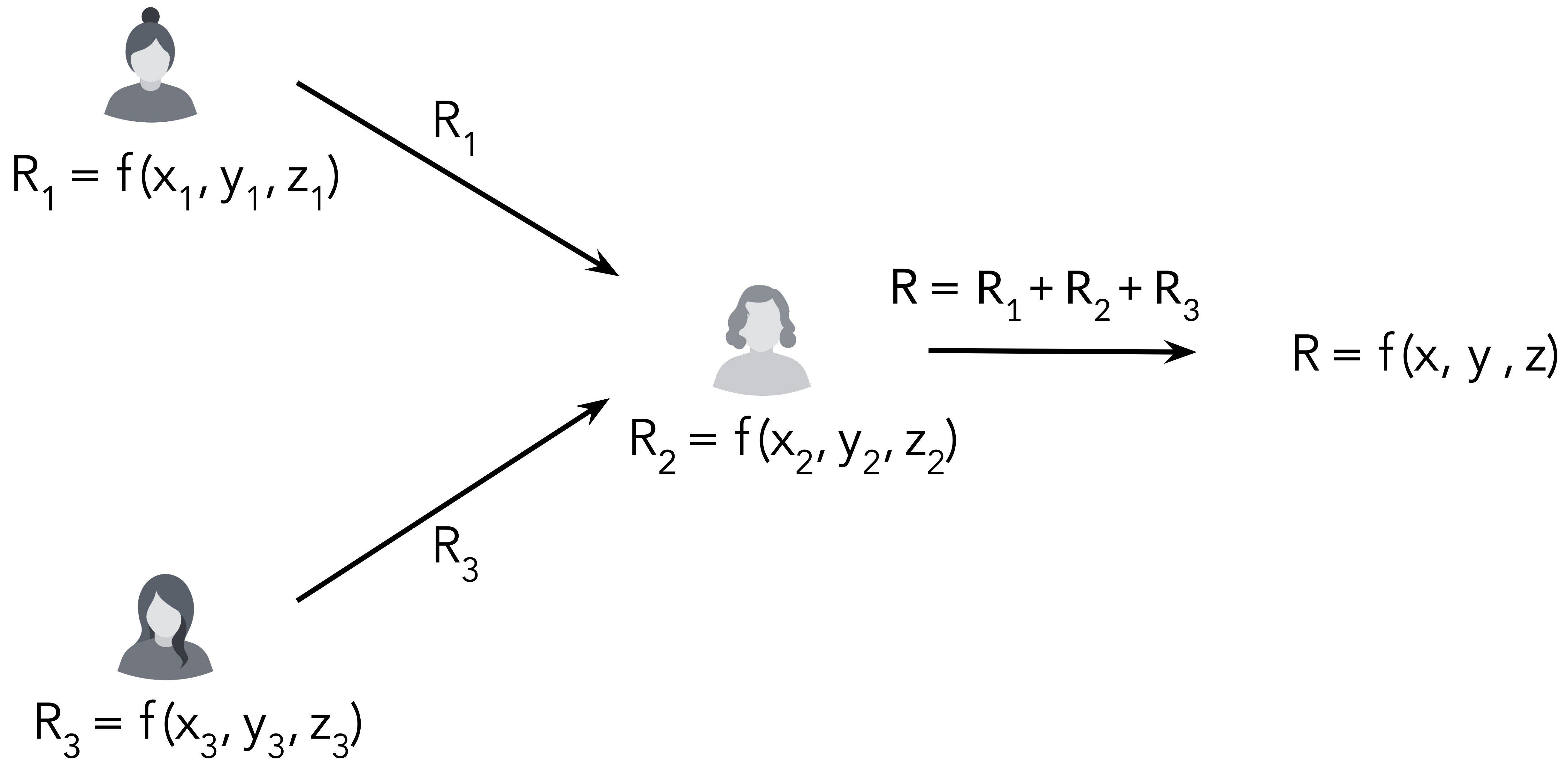


MPC from Secret Sharing

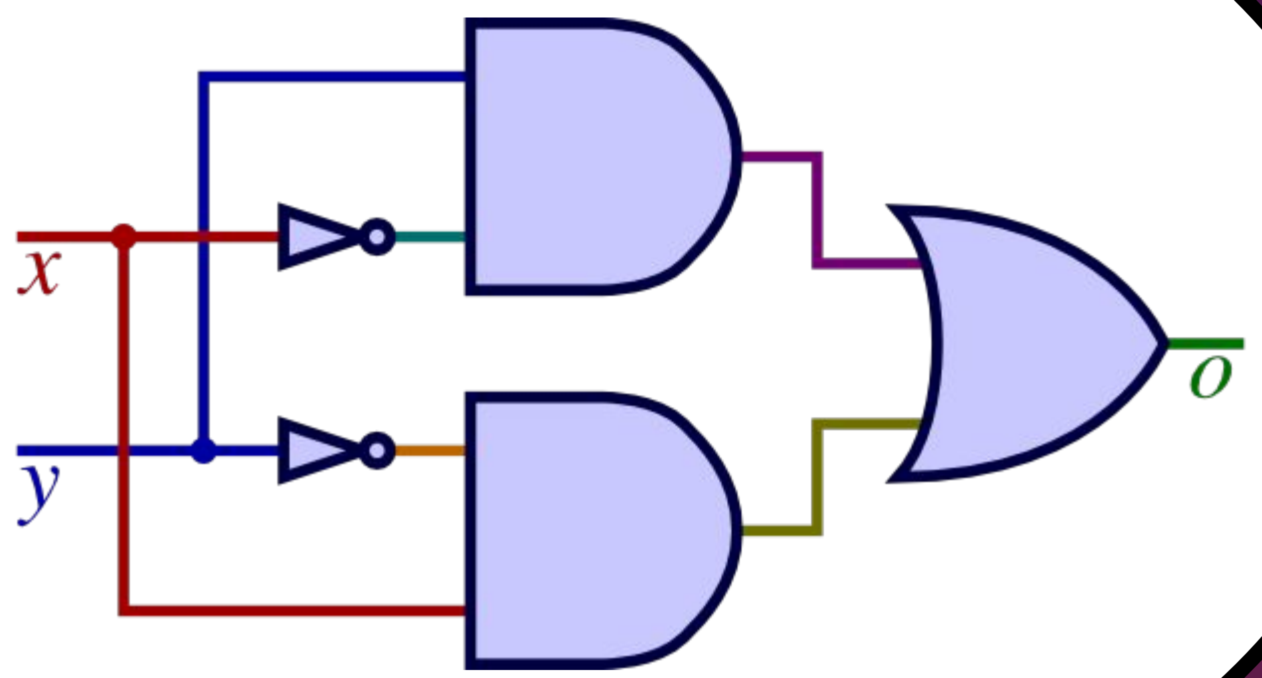
The GMW/BGW* Approach:

- The (public) function being computed is written as a circuit
- Each participant secret-shares their private input
- The circuit is evaluated gate-by-gate on the shares (this requires communication between participants)
- Answer is reconstructed from final shares





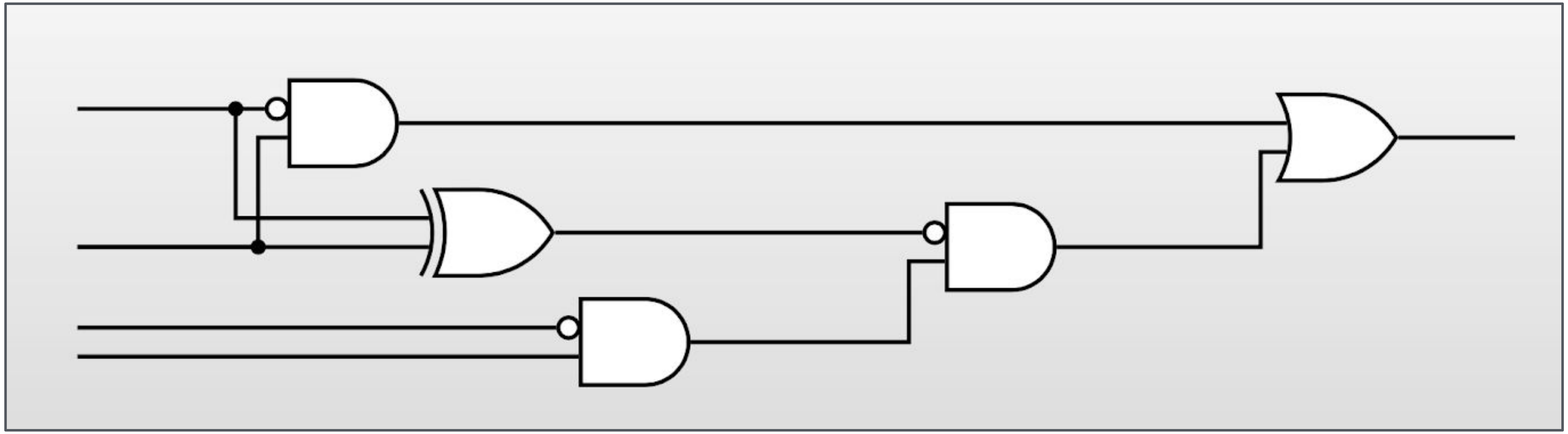
Yao Garbling Circuits



MPC - Boolean Circuits

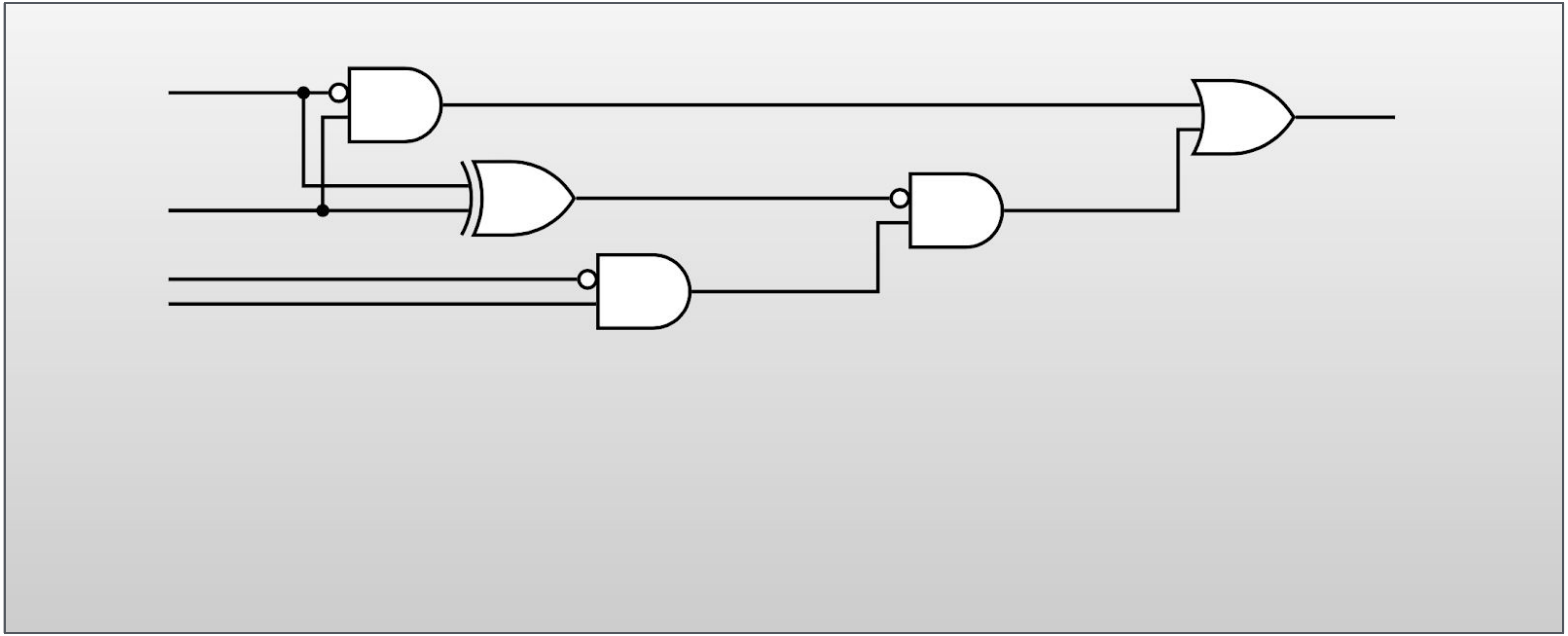
Model of Computation: Represent the function as Arithmetic/**Boolean** circuit C

Limitting Factor: Circuit Depth



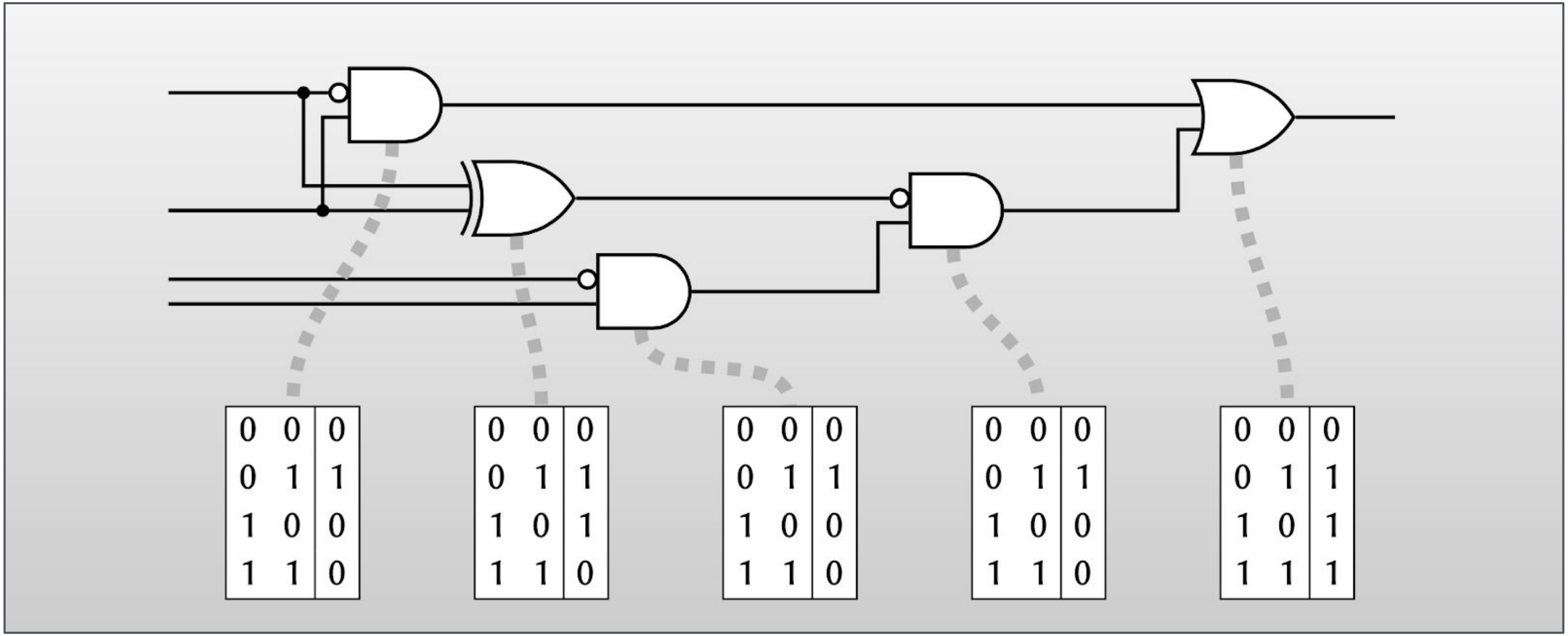
Depth 3

2PC - Yao Garbling Circuit 1982



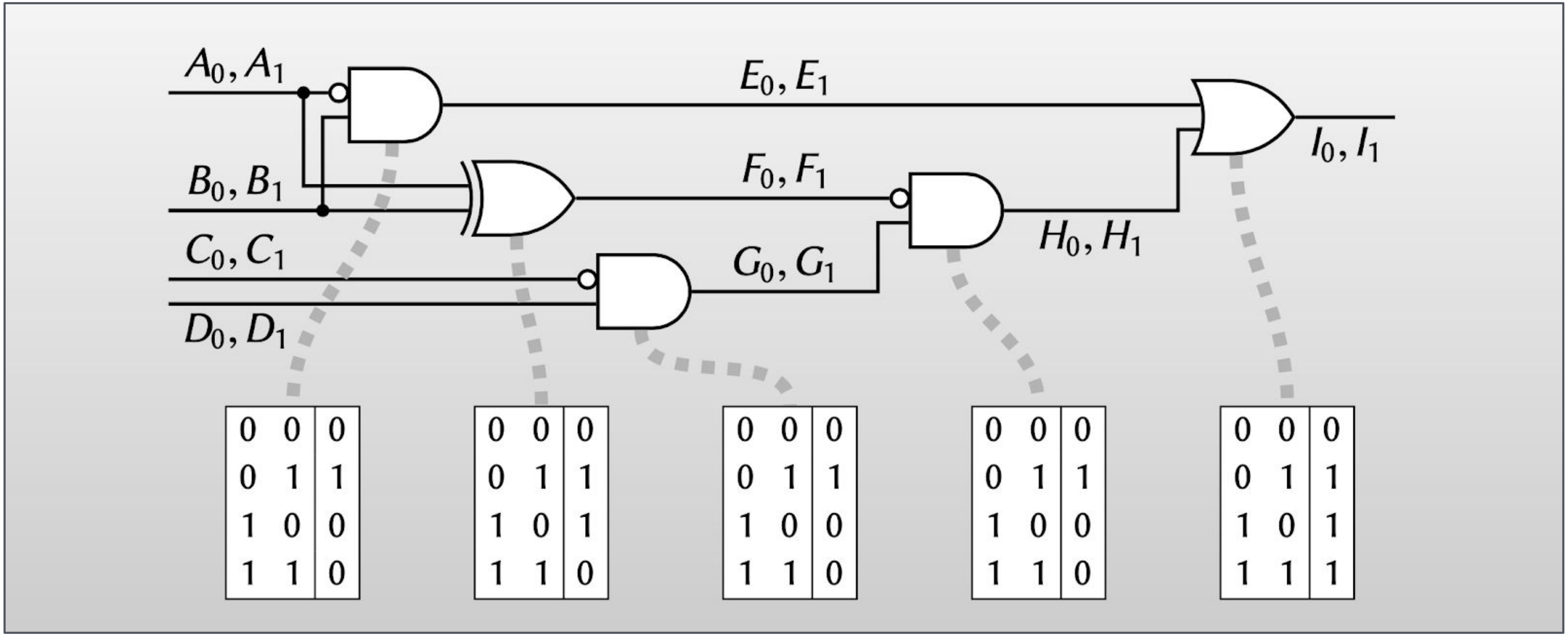
[Yao82] A. Yao, Protocols for secure computations. *In Proceedings of FOCS (1982)*

2PC - Yao Garbling Circuit



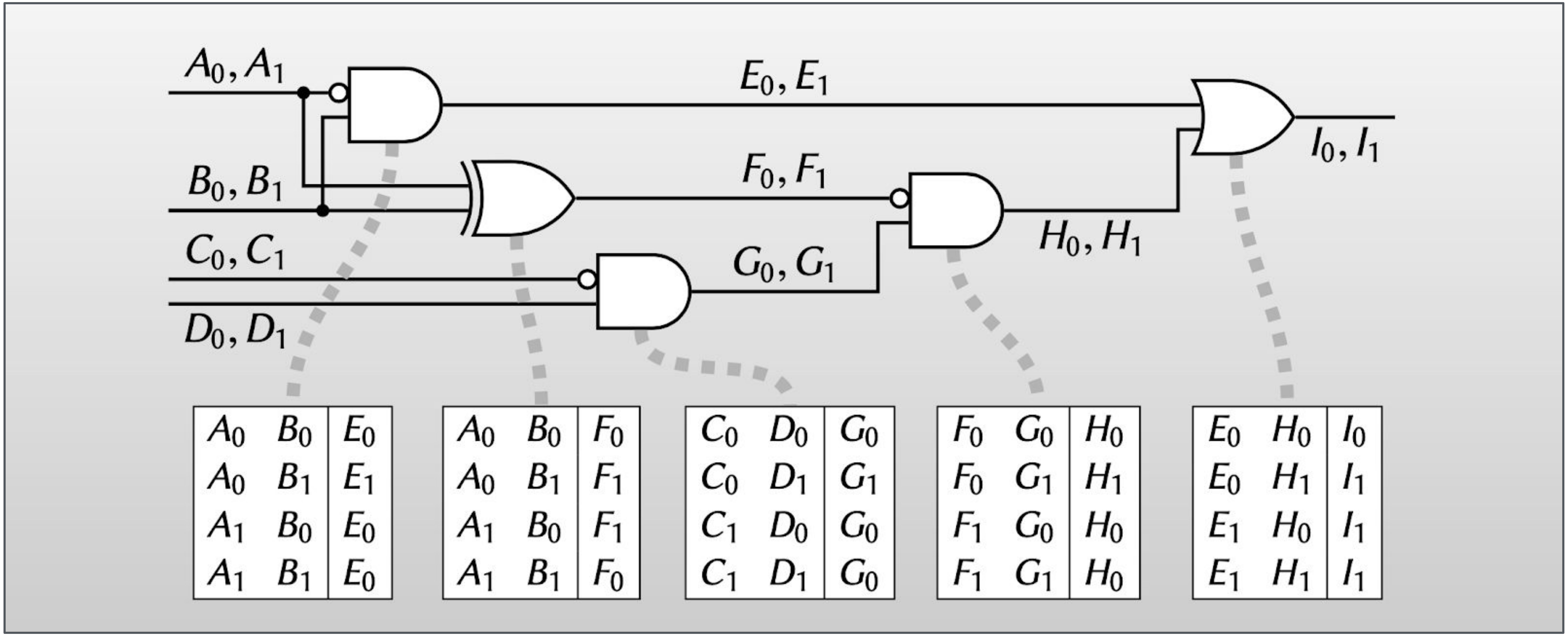
step 1 - Truth table for each logic gate

2PC - Yao Garbling Circuit



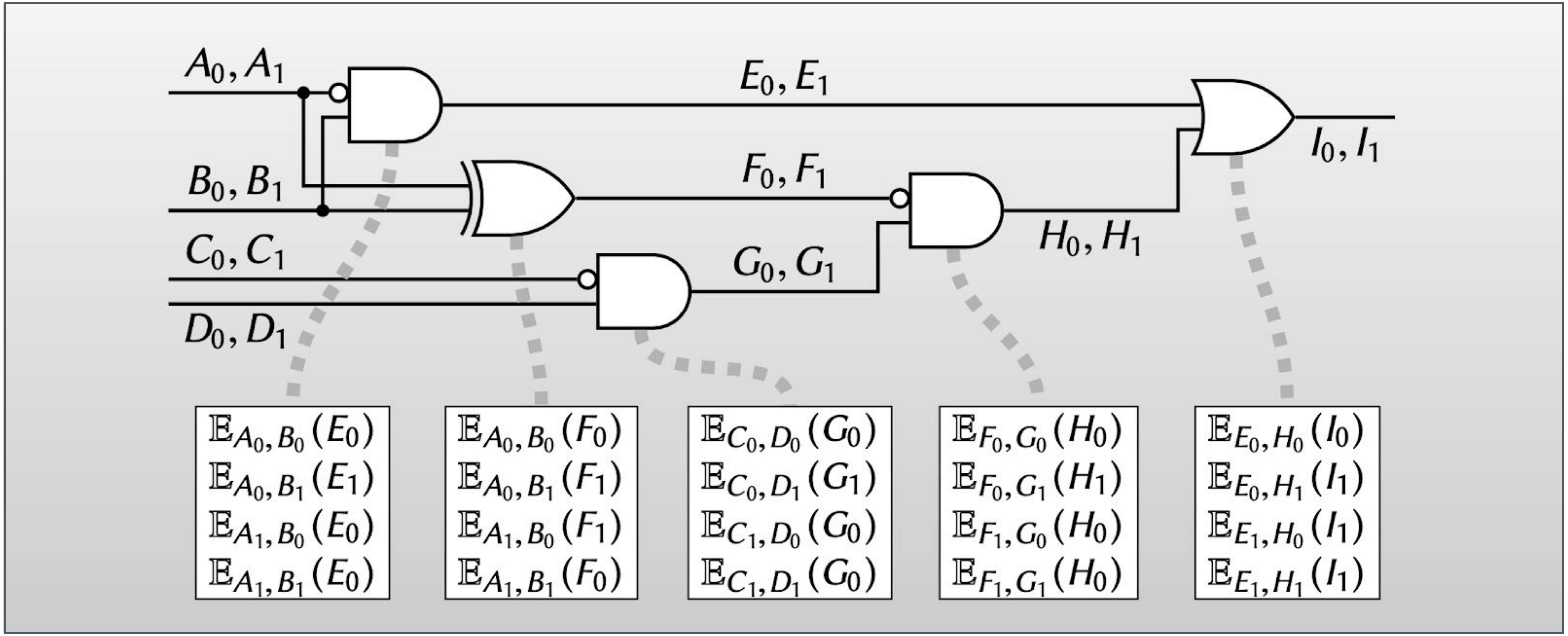
Step 2 - Generate 2 secret keys for each possible input (one for 0 and another one for 1)

Yao Garbling Circuit



Step 3 - Rewrite the Truth Table by replacing 0/1 with their corresponding keys

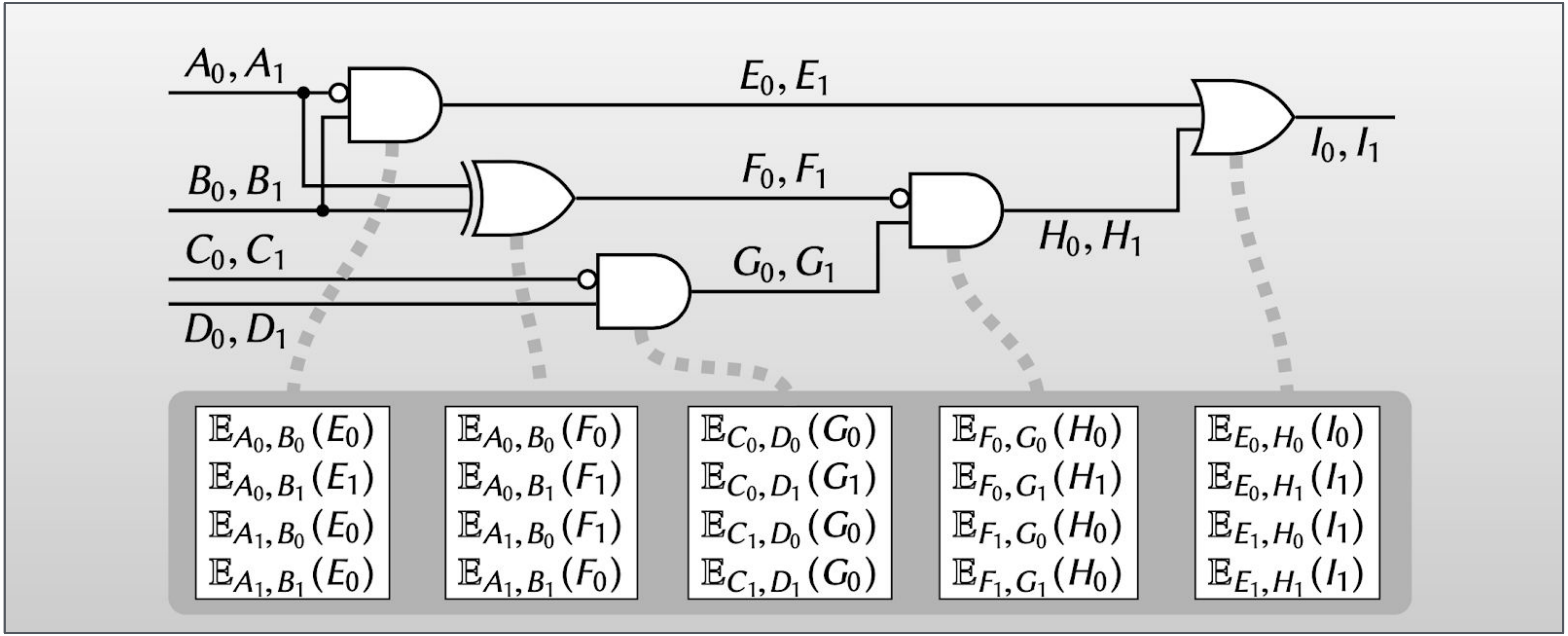
Yao Garbling Circuit



Step 4 - Encrypt with both keys the output of each gate

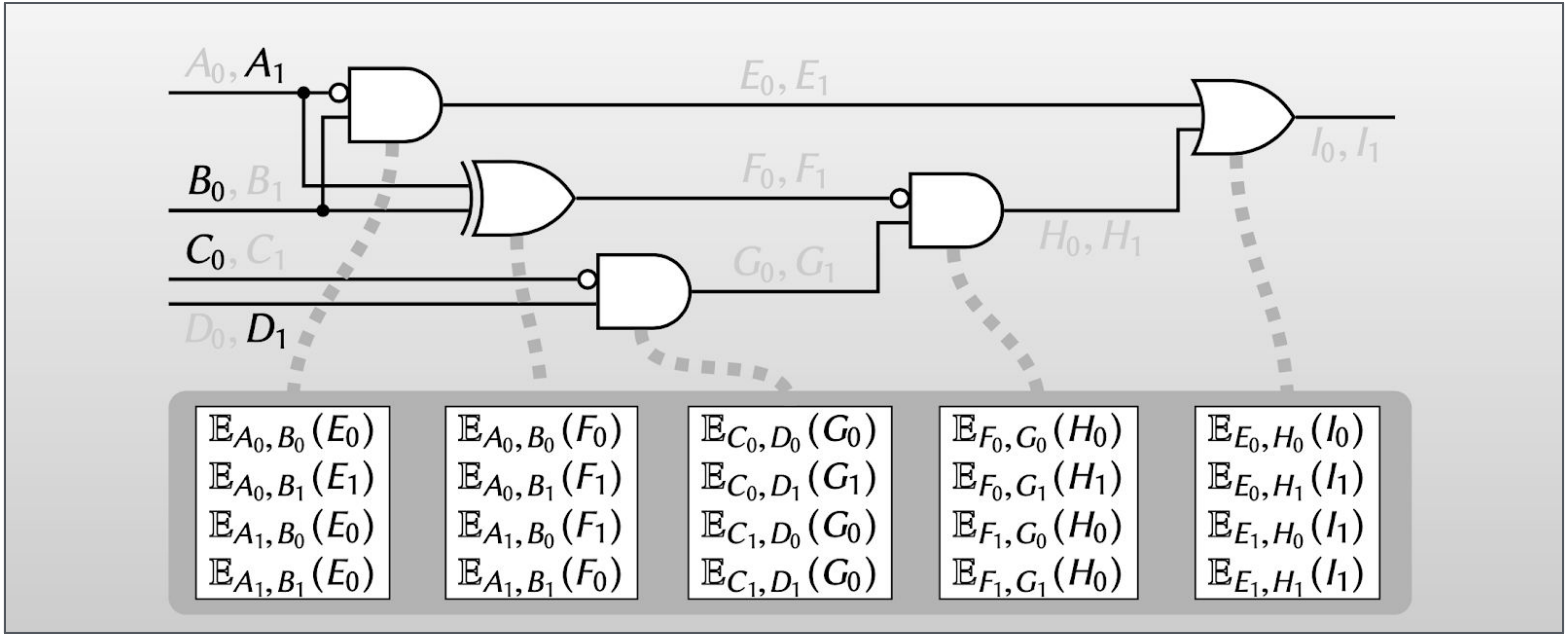
$$E_0 \xrightarrow{B_0} \mathbb{E}_{B_0}(E_0) \xrightarrow{A_0} \mathbb{E}_{A_0,B_0}(E_0)$$

Yao Garbling Circuit



Step 5 - Send this version of the circuit to the Evaluator

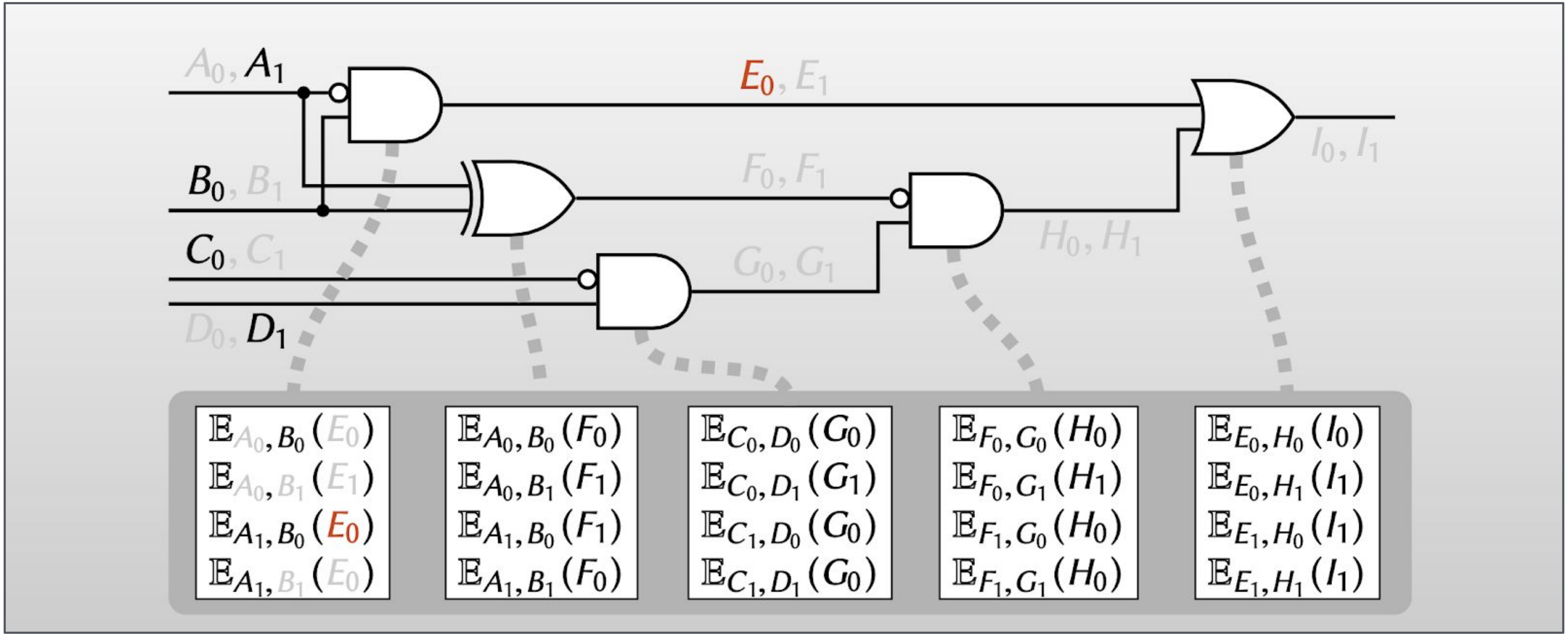
Yao Garbling Circuit - Evaluation



Step 1 - Use the known keys for your inputs to decrypt each gate value

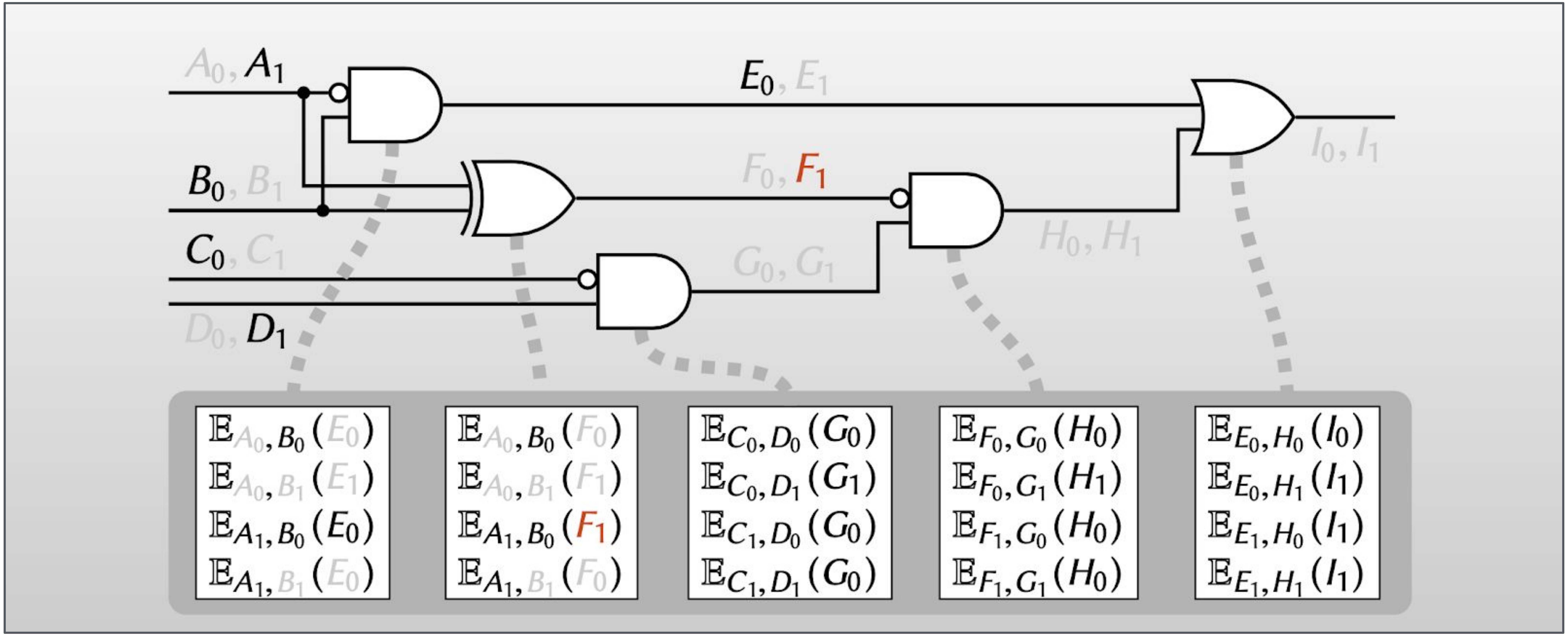
example - inputs (1, 0, 0, 1)

Yao Garbling Circuit - Evaluation



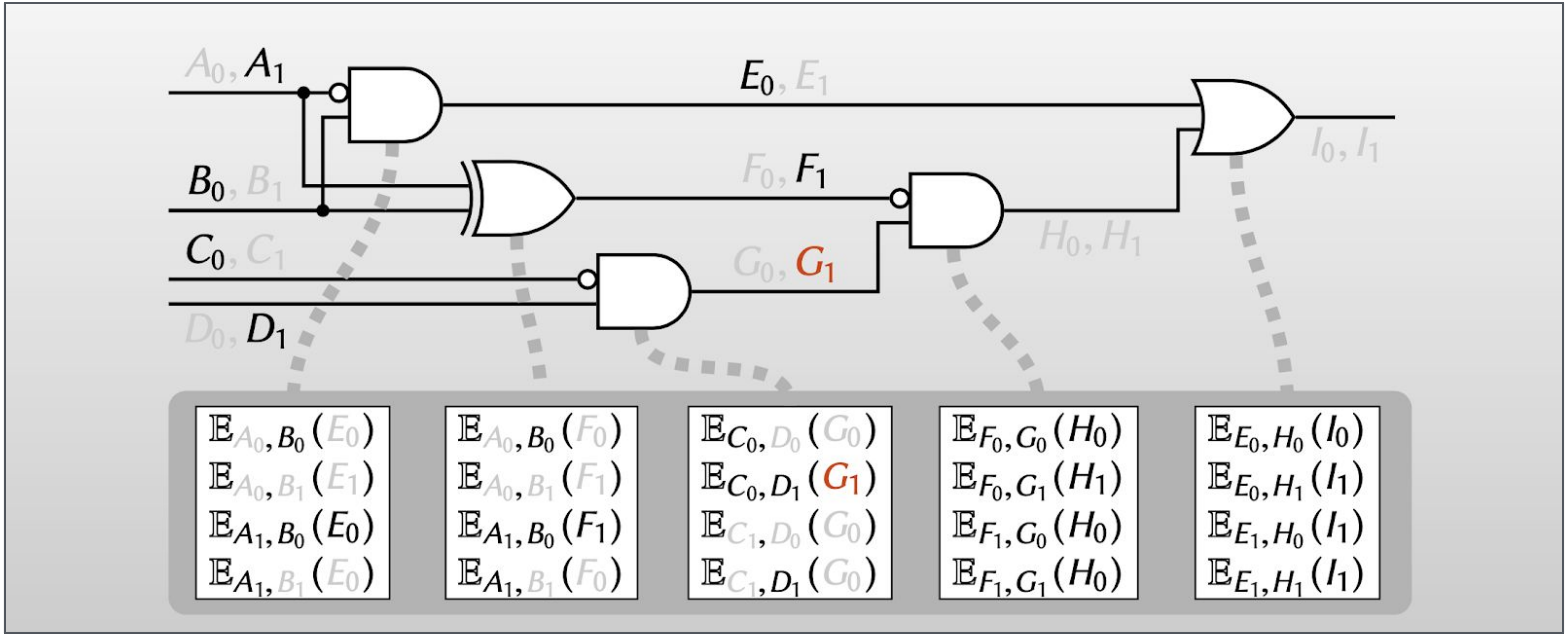
Step 2 - Continue the process for following gates

Yao Garbling Circuit - Evaluation



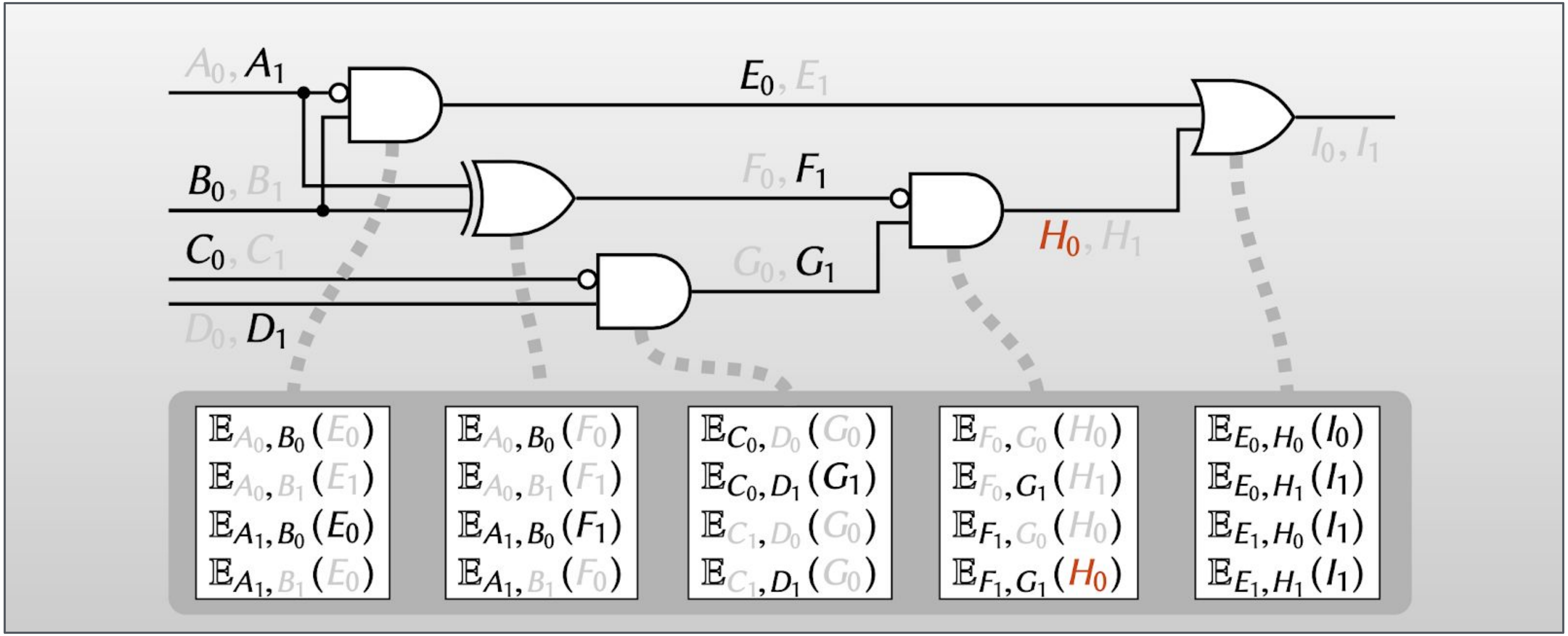
Step 2 - Continue the process for following gates

Yao Garbling Circuit - Evaluation



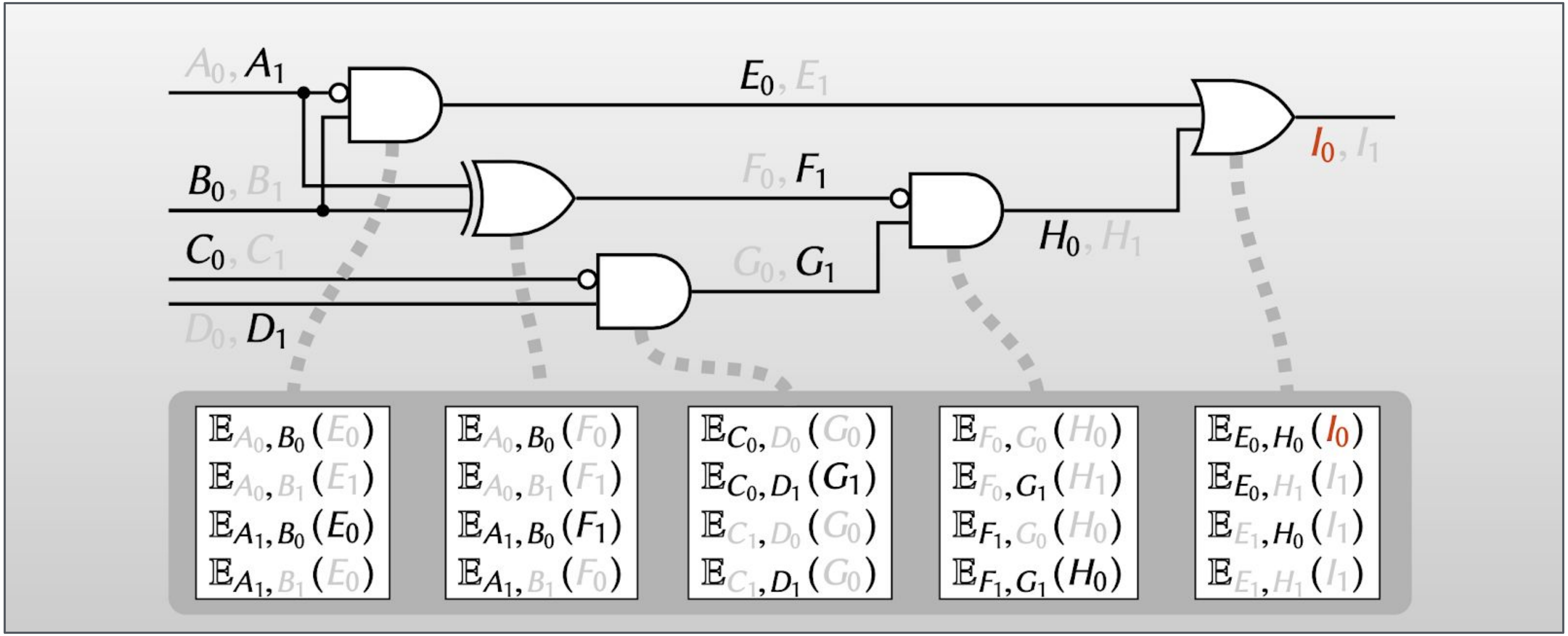
Step 2 - Continue the process for following gates

Yao Garbling Circuit - Evaluation



Step 2 - Continue the process for following gates

Yao Garbling Circuit - Evaluation



Step 4 - Obtain the expected result

Bibliography

[AOR+19] Abdelrahaman Aly, Emmanuela Orsini, Dragos Rotaru, Nigel P. Smart, Tim Wood. Zaphod: Efficiently Combining LSSS and Garbled Circuits in SCALE. *In Workshop on Encrypted Computing and Applied Homomorphic Cryptography 2019, 2019.*

[BDOZ11] Rikke Bendlin, Ivan Damgaard, Claudio Orlandi, Sarah Zakarias. BDOZ: Semi-homomorphic encryption and multiparty computation. *In Advances in Cryptology – EUROCRYPT 2011, 2011, Springer.*

[BGW88]M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. *In the 20th STOC, pages 1–10, 1988.*

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. *In 22nd ACM STOC, 1990.*

[DPSZ12] Ivan Damgaard, Valerio Pastro, Nigel P. Smart, Sarah Zakarias. SPDZ: Multiparty computation from somewhat homomorphic encryption. *In Advances in Cryptology – CRYPTO 2012, Springer.*

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP- statements in zero-knowledge, and a methodology of cryptographic protocol design. *In CRYPTO'86, volume 263 of LNCS, pages 171–185. Springer.*

[KRS18] M. Keller, D. Rotaru, N. Smart, and T. Wood, Reducing Communication Channels in MPC. *In Security and Cryptography for Networks 11, pp. 181-199, 2018.*

[LPSY19] Y. Lindell, B. Pinkas, N. Smart, A. Yanai. Efficient Constant Round Multi-Party Computation Combining BMR and SPDZ. *In Journal of Cryptology 32, 2019.*

[NNOB12] Jesper Nielsen, Peter Nordholt, Claudio Orlandi, Sai S. Burra. TinyOT: A new approach to practical active-secure two-party computation. *In CRYPTO 2012, Lecture Notes in Computer Science, pages 681–700, 2012, Springer*

[RT19] D. Rotaru, T. Wood, MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security. *In INDOCRYPT 2019, Lecture Notes in Computer Science, Springer-Verlag, 2019*

[Yao82] A. Yao, Protocols for secure computations. *In Proceedings of FOCS (1982)*