

Client-Server Homomorphic Secret Sharing in the CRS Model

EUROCRYPT 2026



Damiano Abram



Geoffroy Couteau



Lalita Devadas

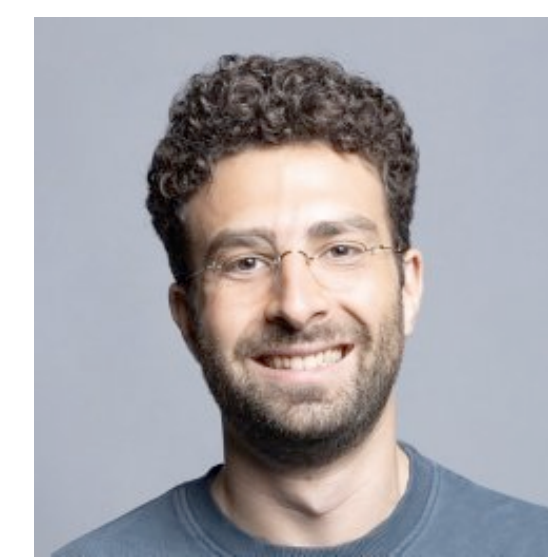
Aditya Hegde



Abhishek Jain



Lawrence Roy



Sacha Servan-Schreiber

Client-Server Secure Computation

x



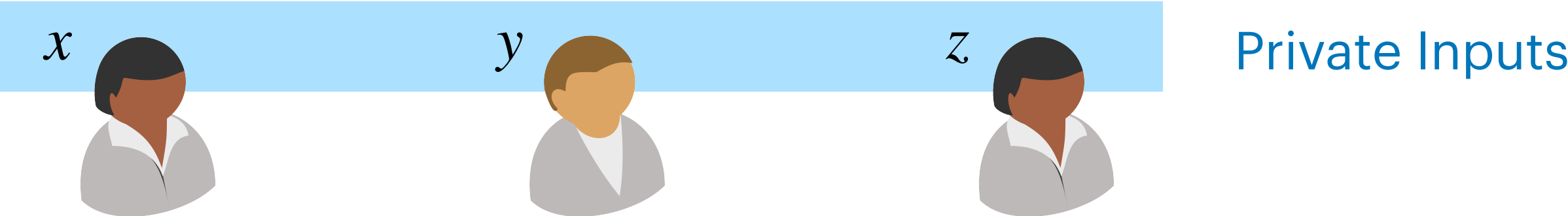
y



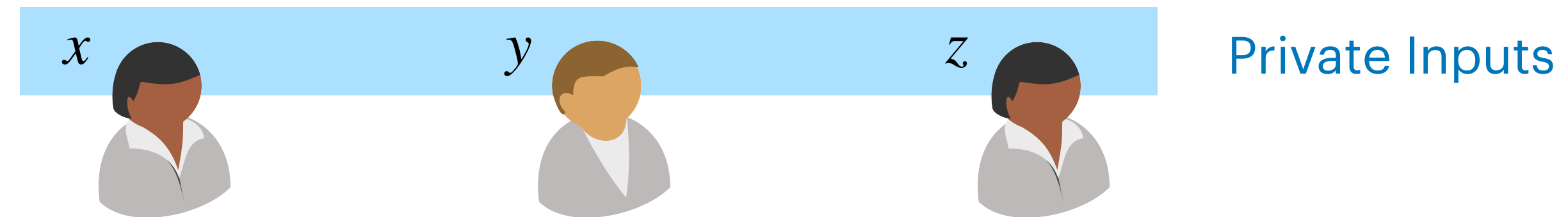
z



Client-Server Secure Computation

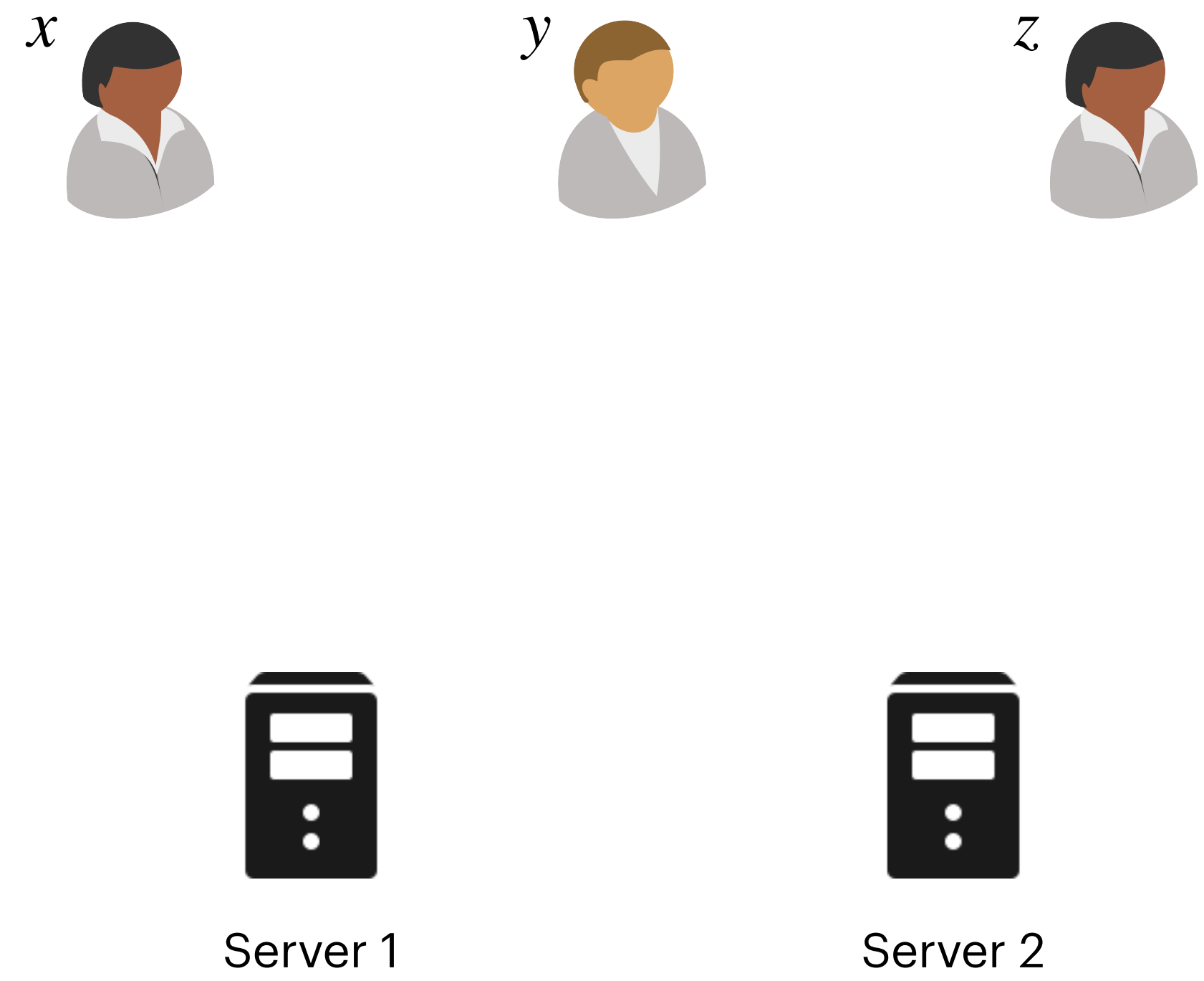


Client-Server Secure Computation

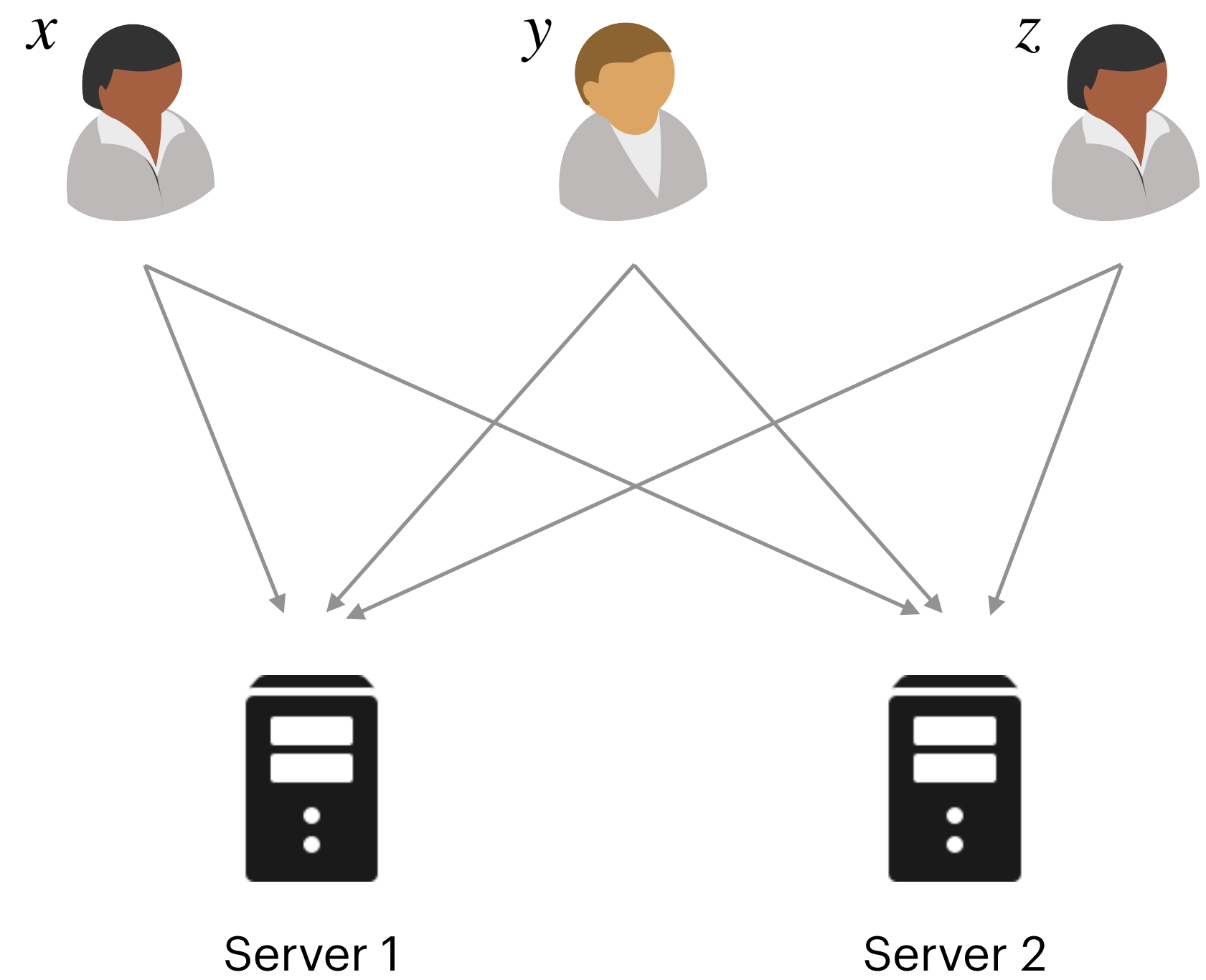


Want to delegate computation of $f(x, y, z)$

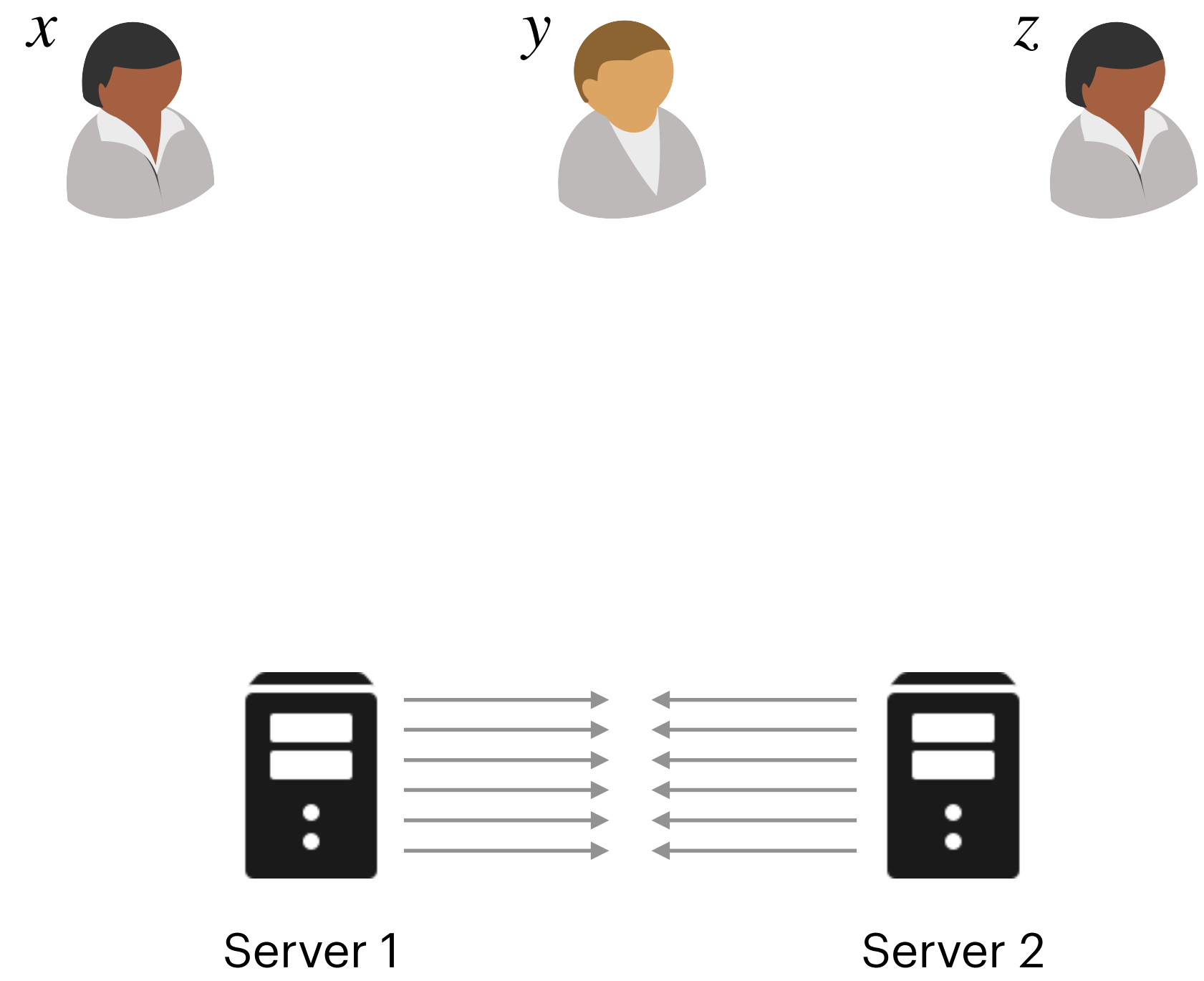
Client-Server Secure Computation



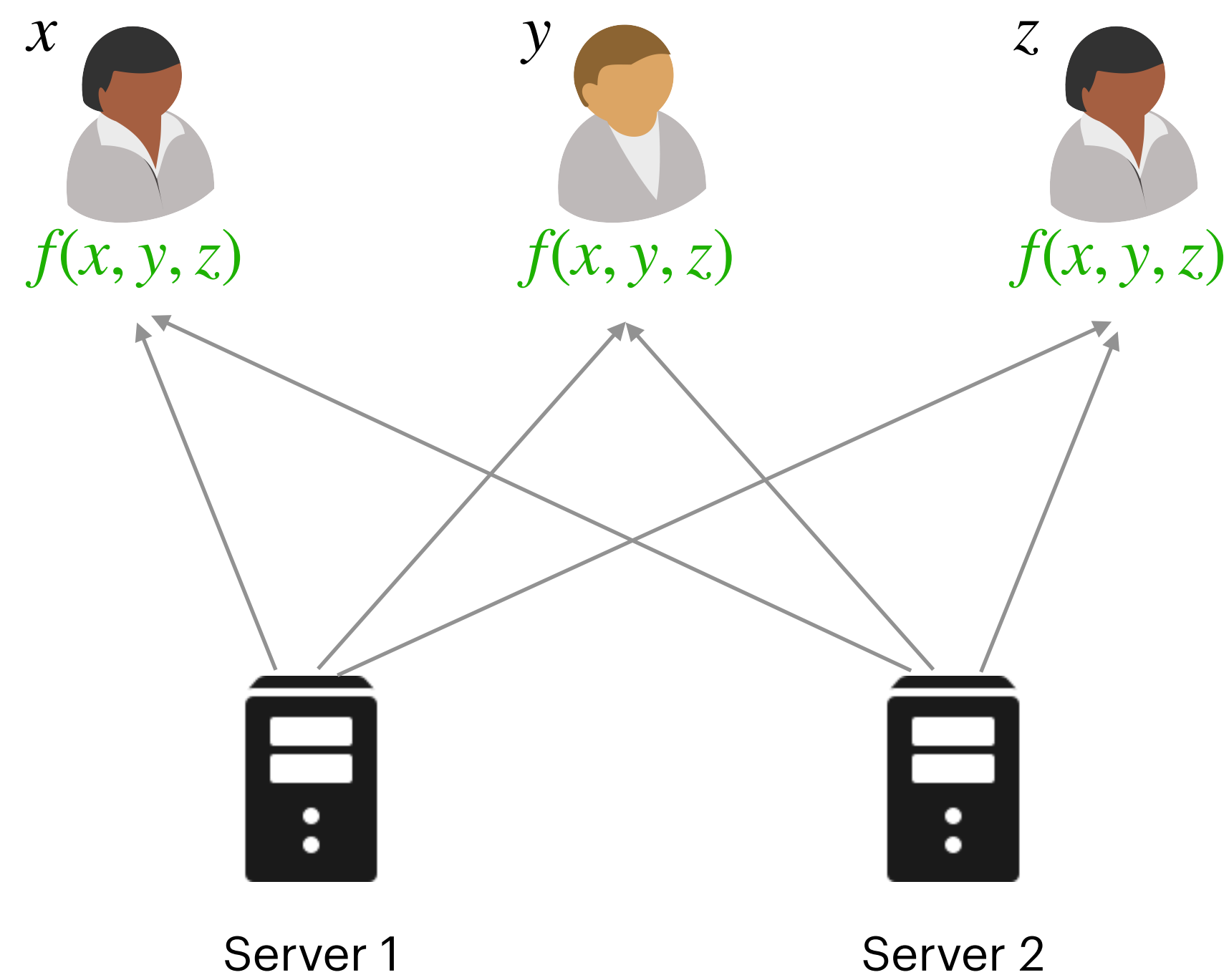
Client-Server Secure Computation



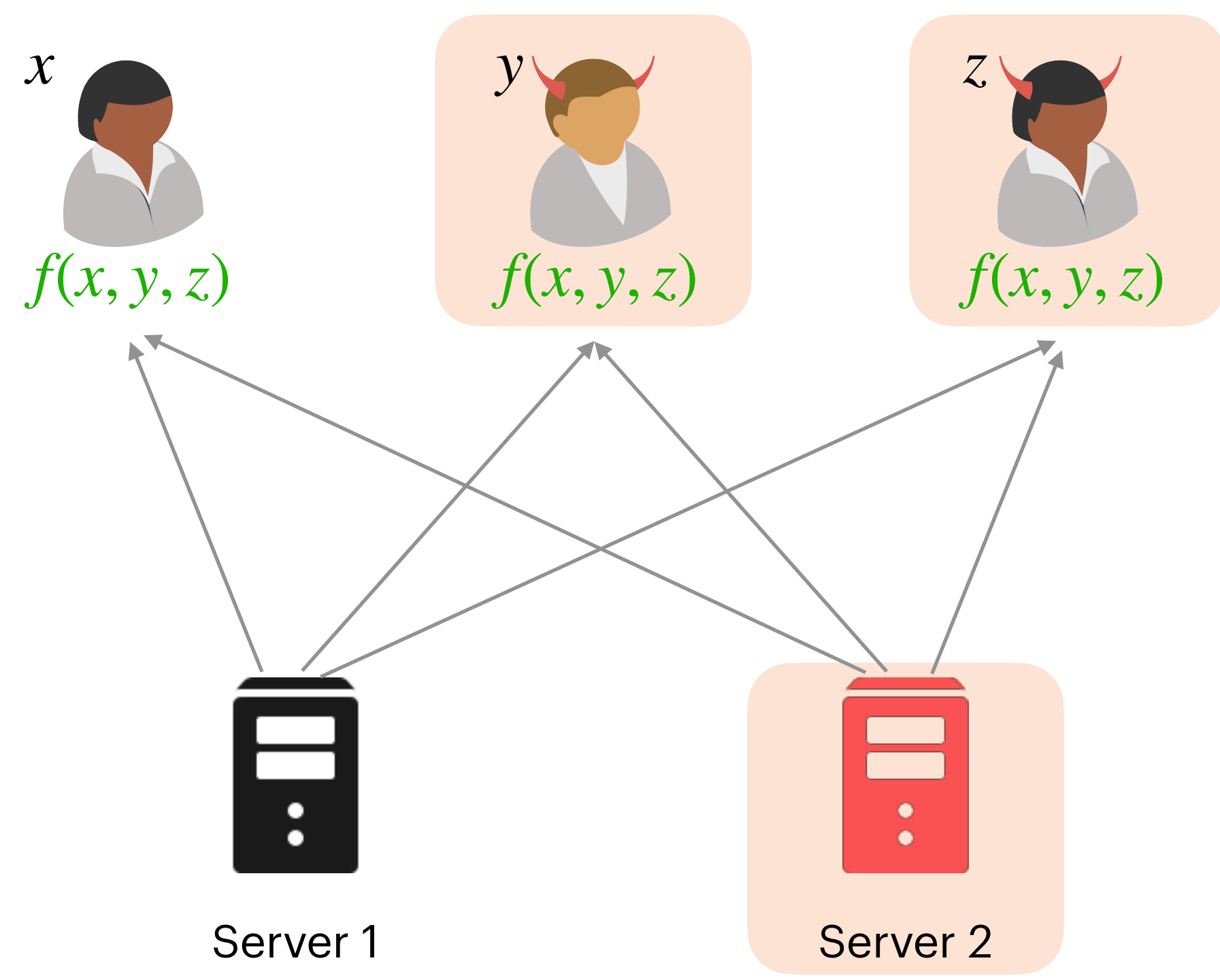
Client-Server Secure Computation



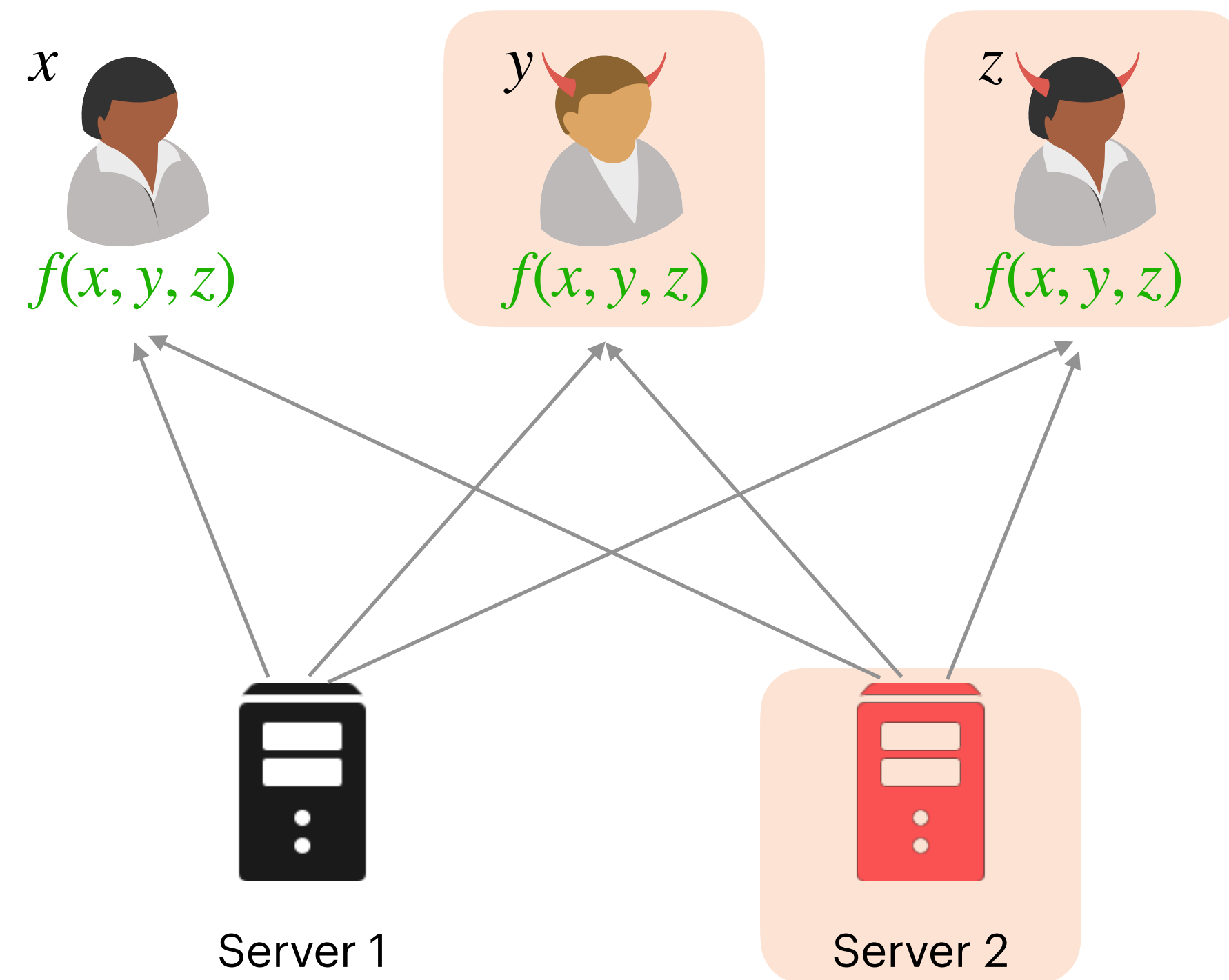
Client-Server Secure Computation



Client-Server Secure Computation

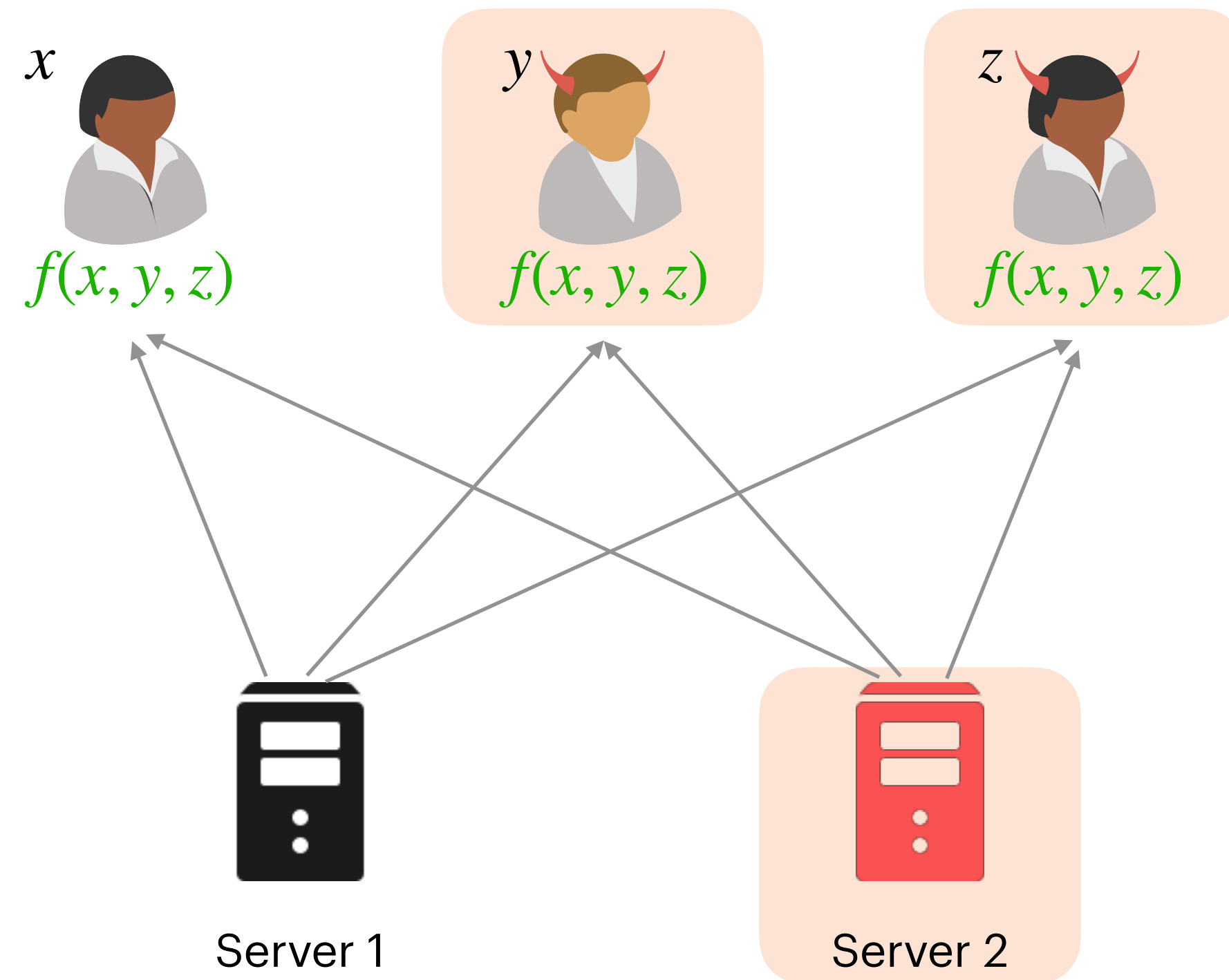


Client-Server Secure Computation



What is the **best possible** semi-honest protocol in the client-server setting?

Client-Server Secure Computation



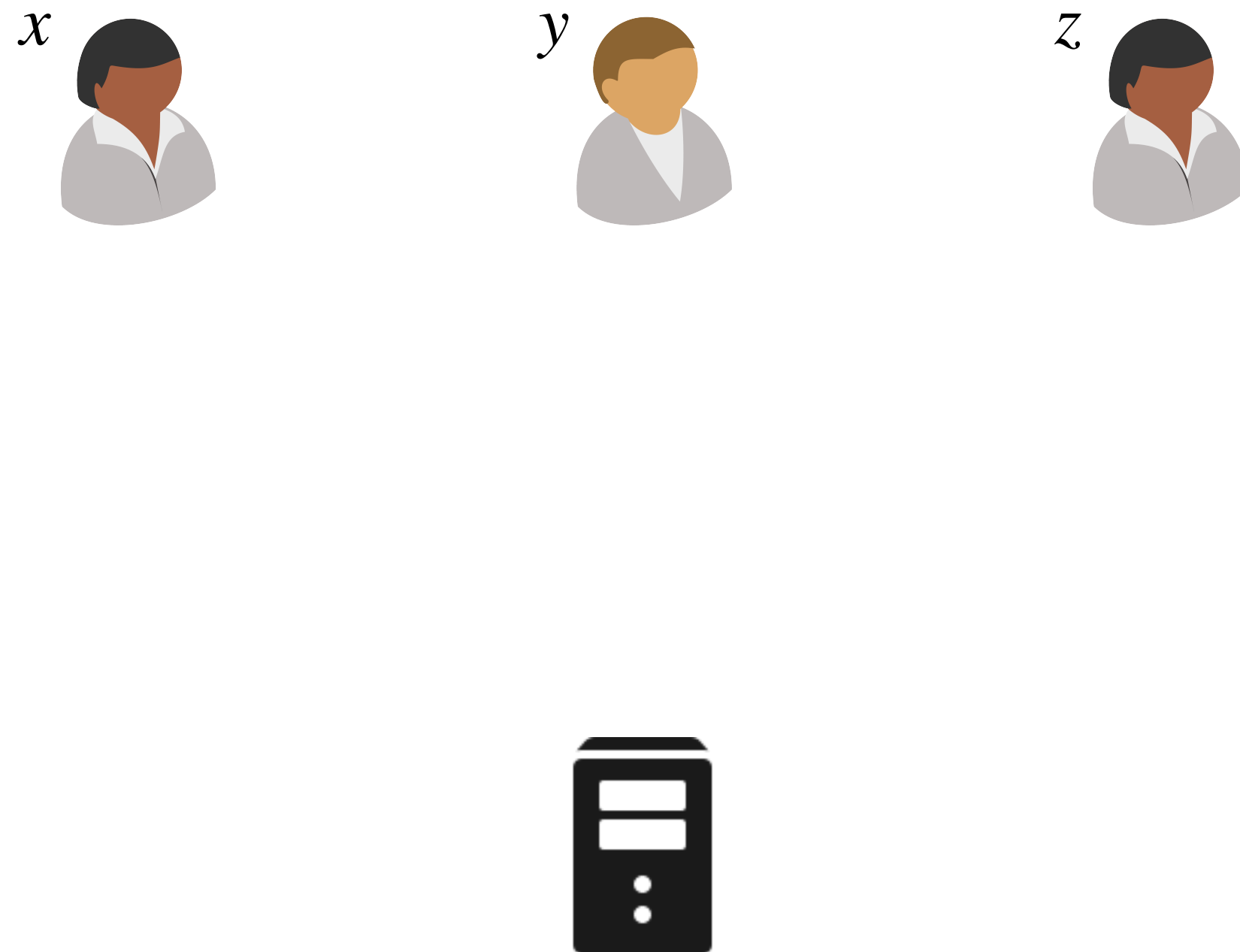
What is the **best possible** semi-honest protocol in the client-server setting?

Dream: Match **ideal world** efficiency

Ideal World Client-Server Computation

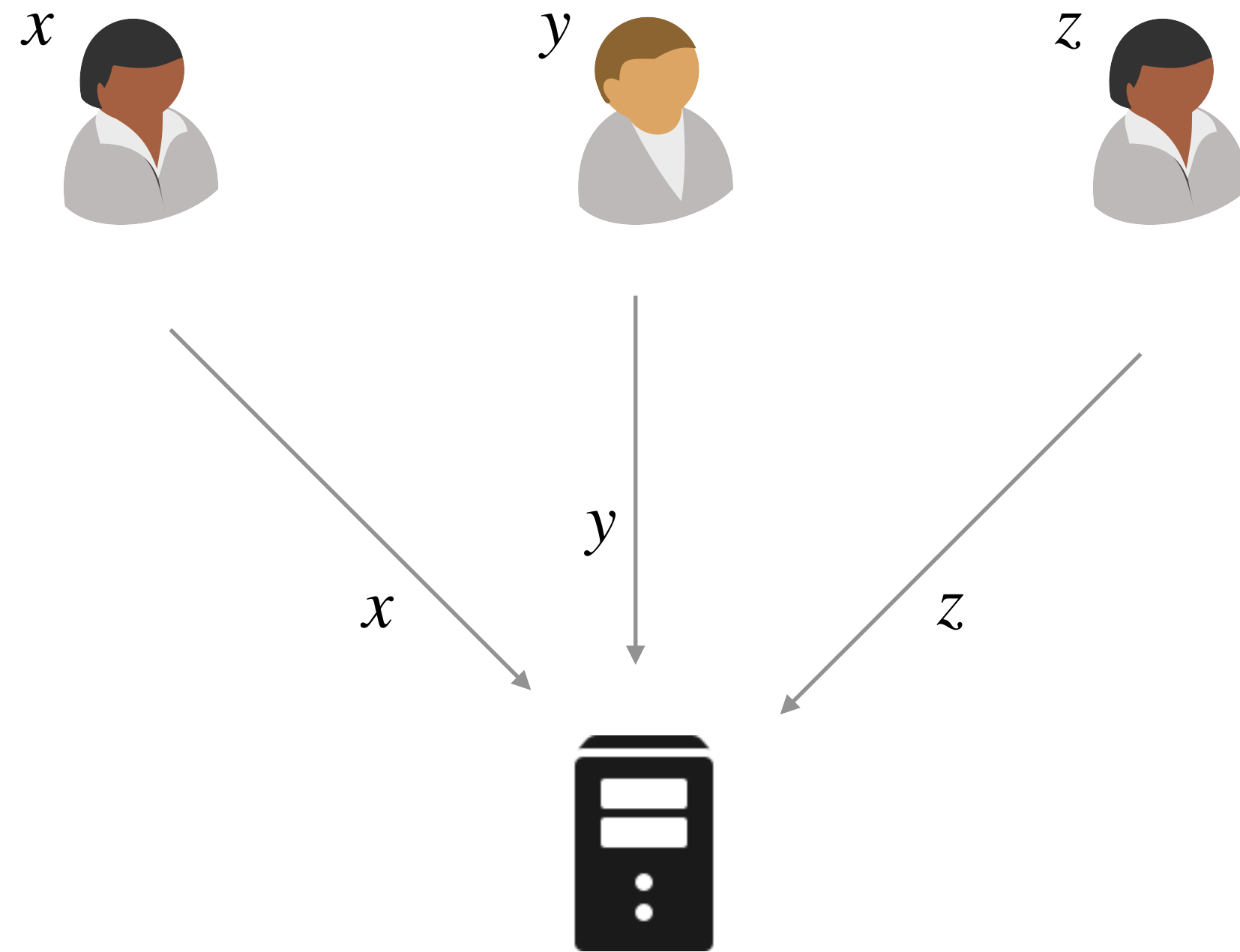
Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



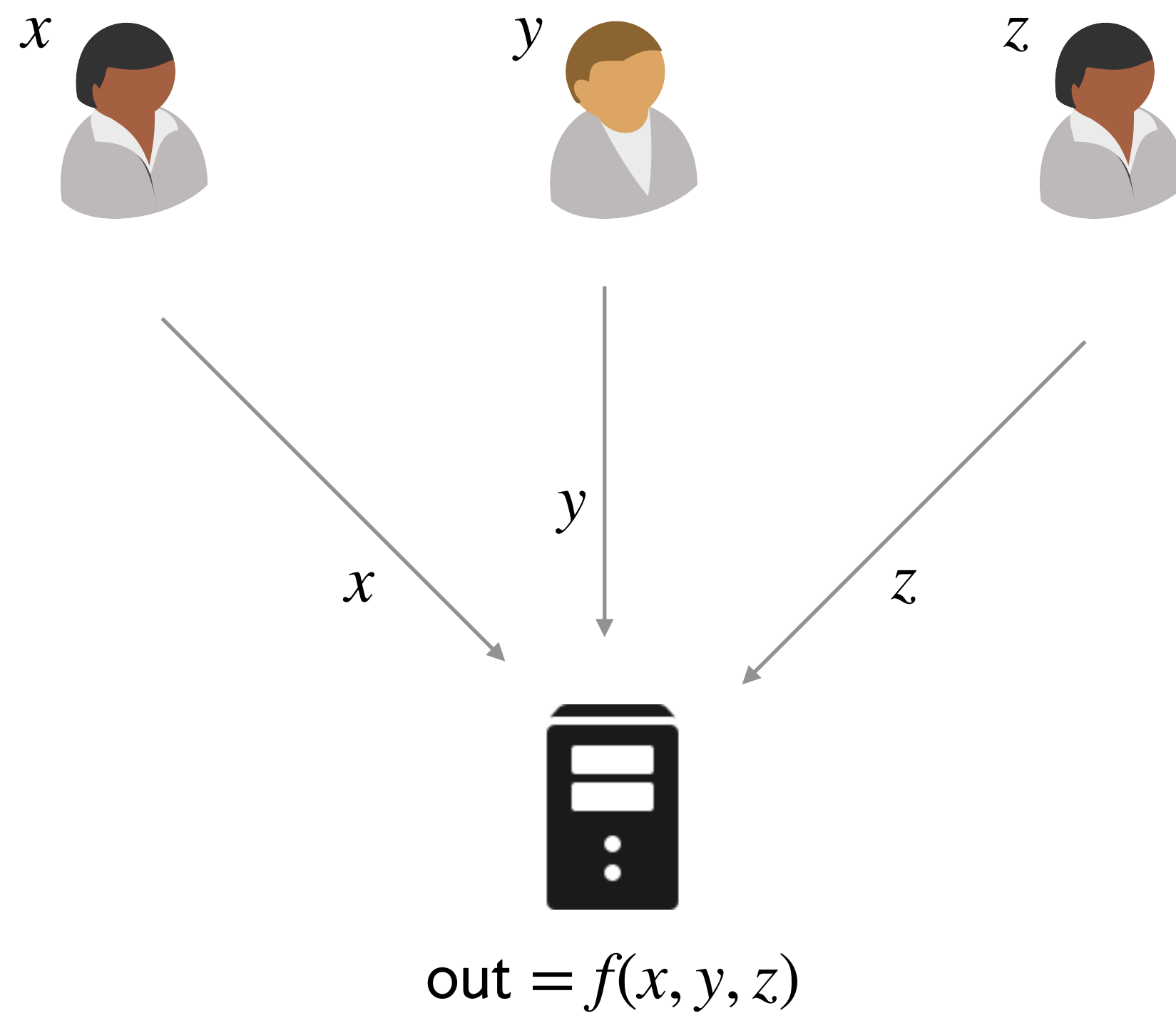
Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



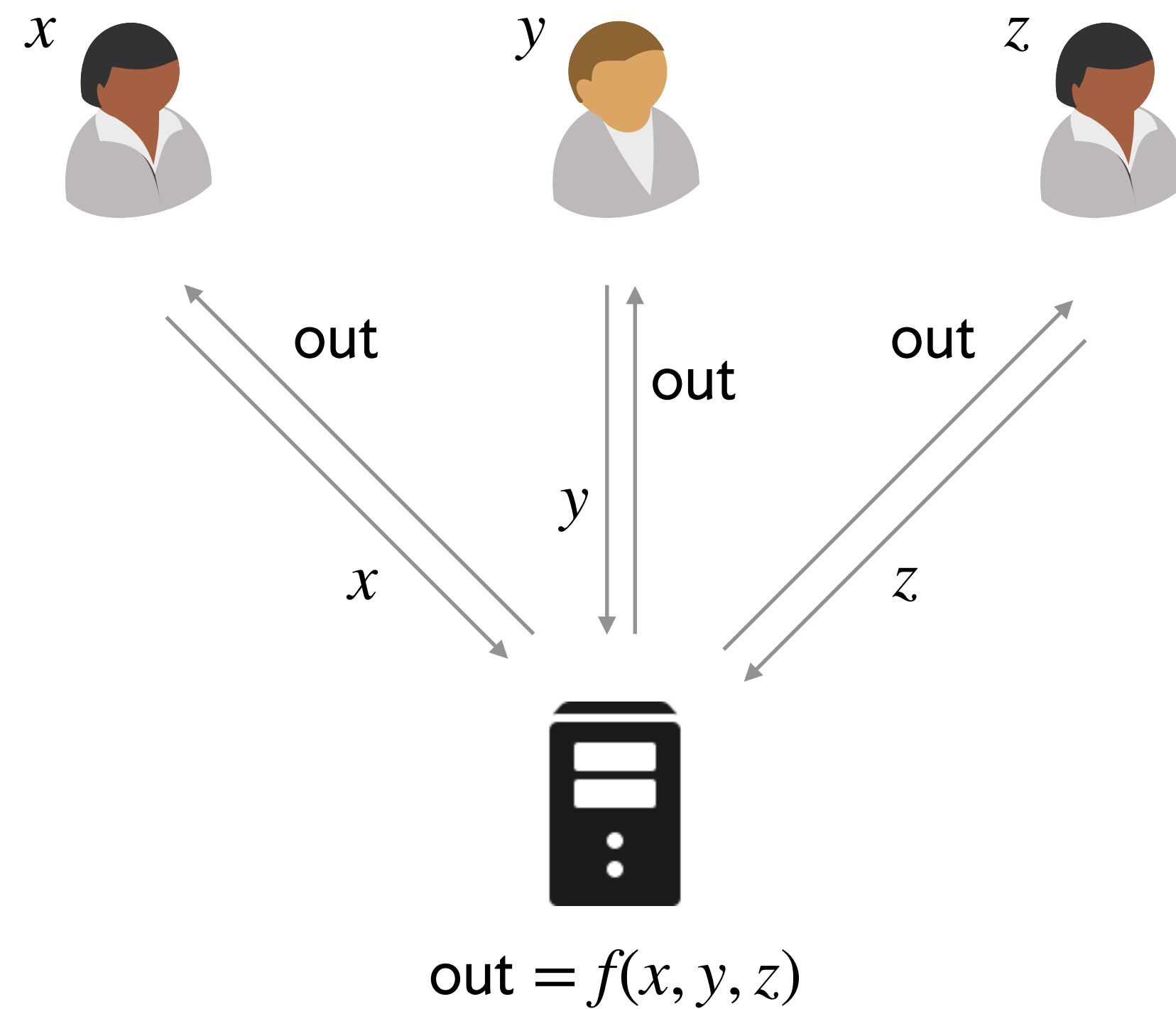
Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



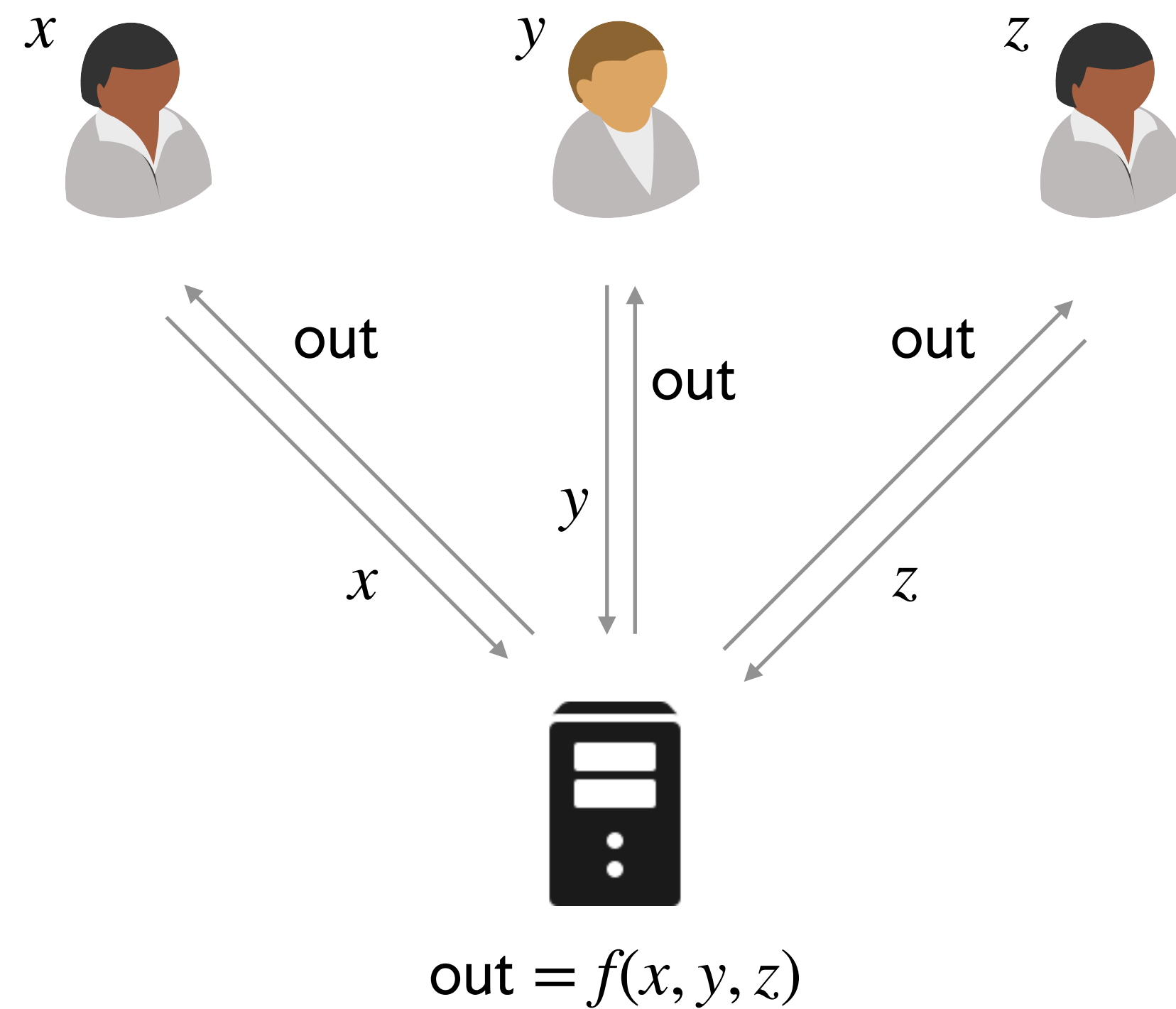
Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



Ideal World Client-Server Computation

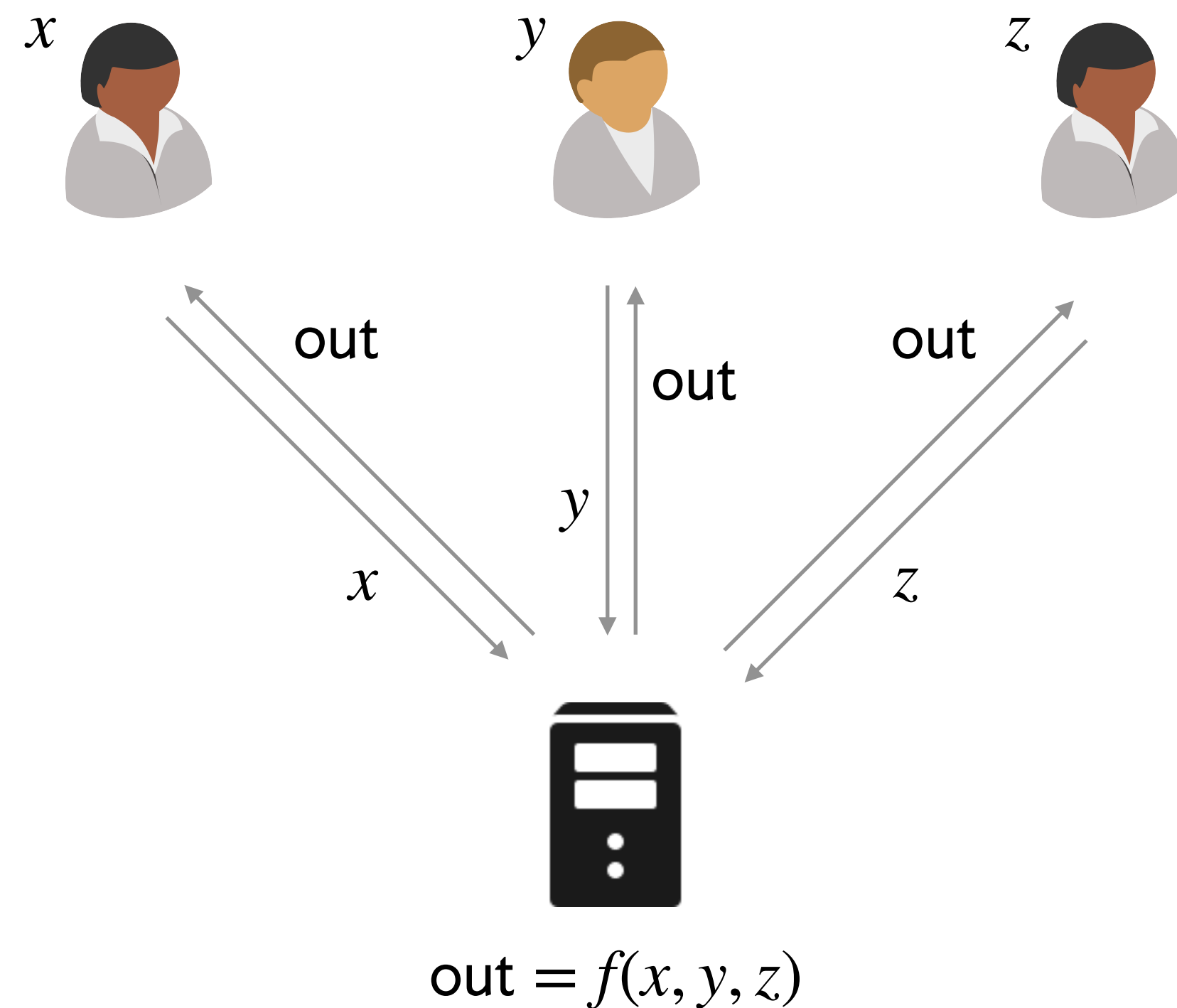
Ideal World: Clients delegate computation to a **trusted** server



Function independent
client runtime

Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server

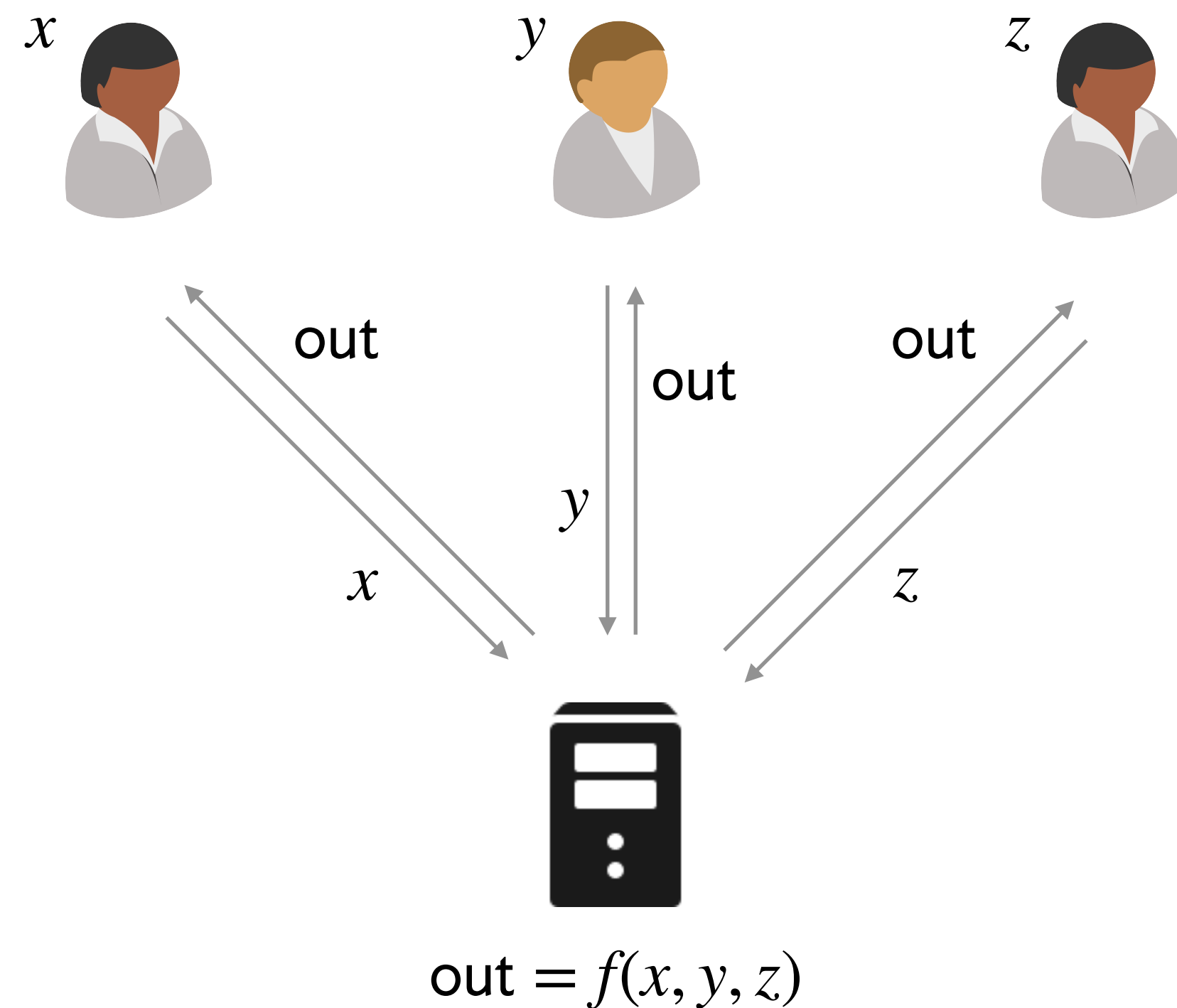


Function independent
client runtime

Function independent
communication cost

Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



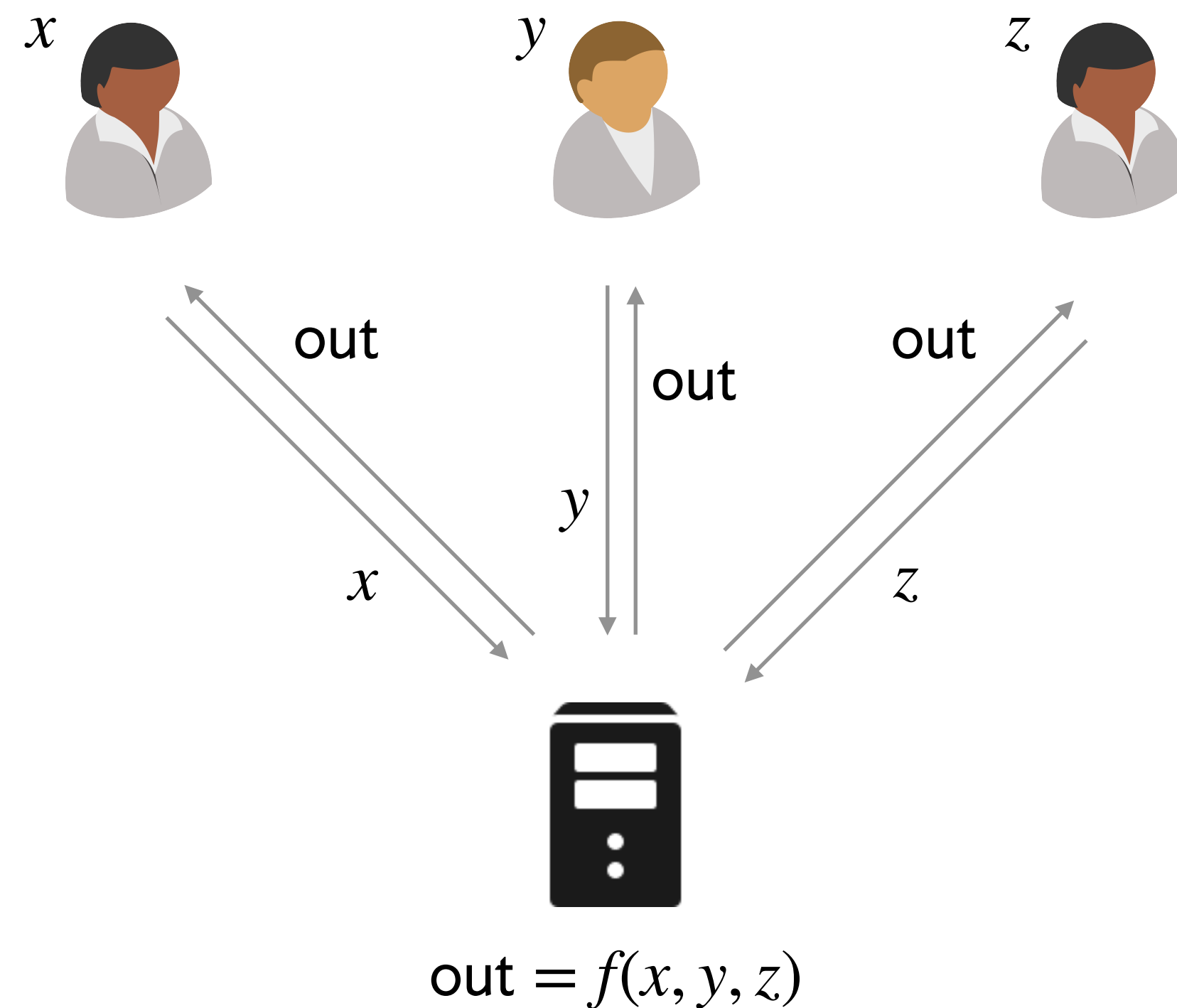
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Ideal World Client-Server Computation

Ideal World: Clients delegate computation to a **trusted** server



Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?

Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?

Goals

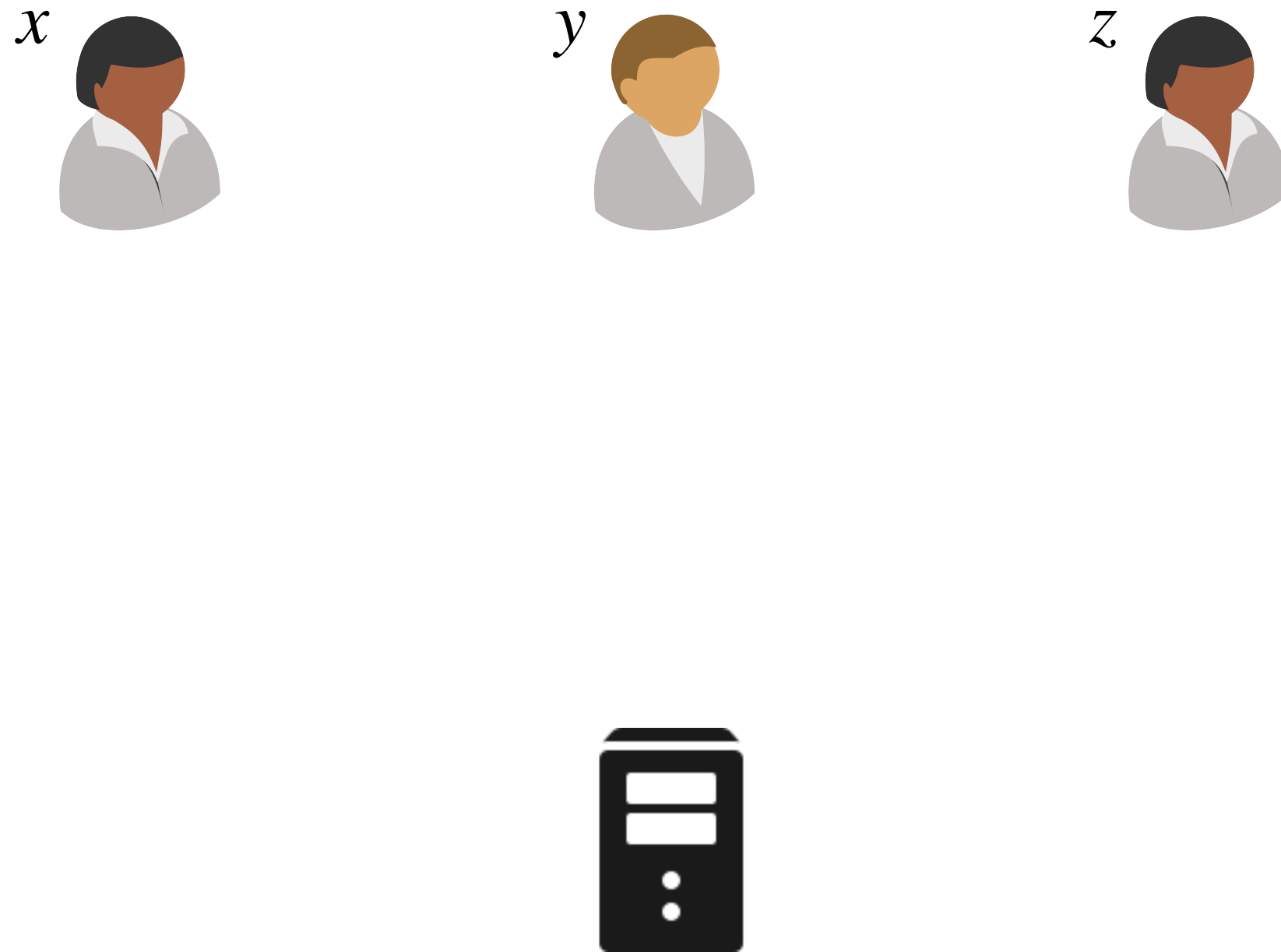
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

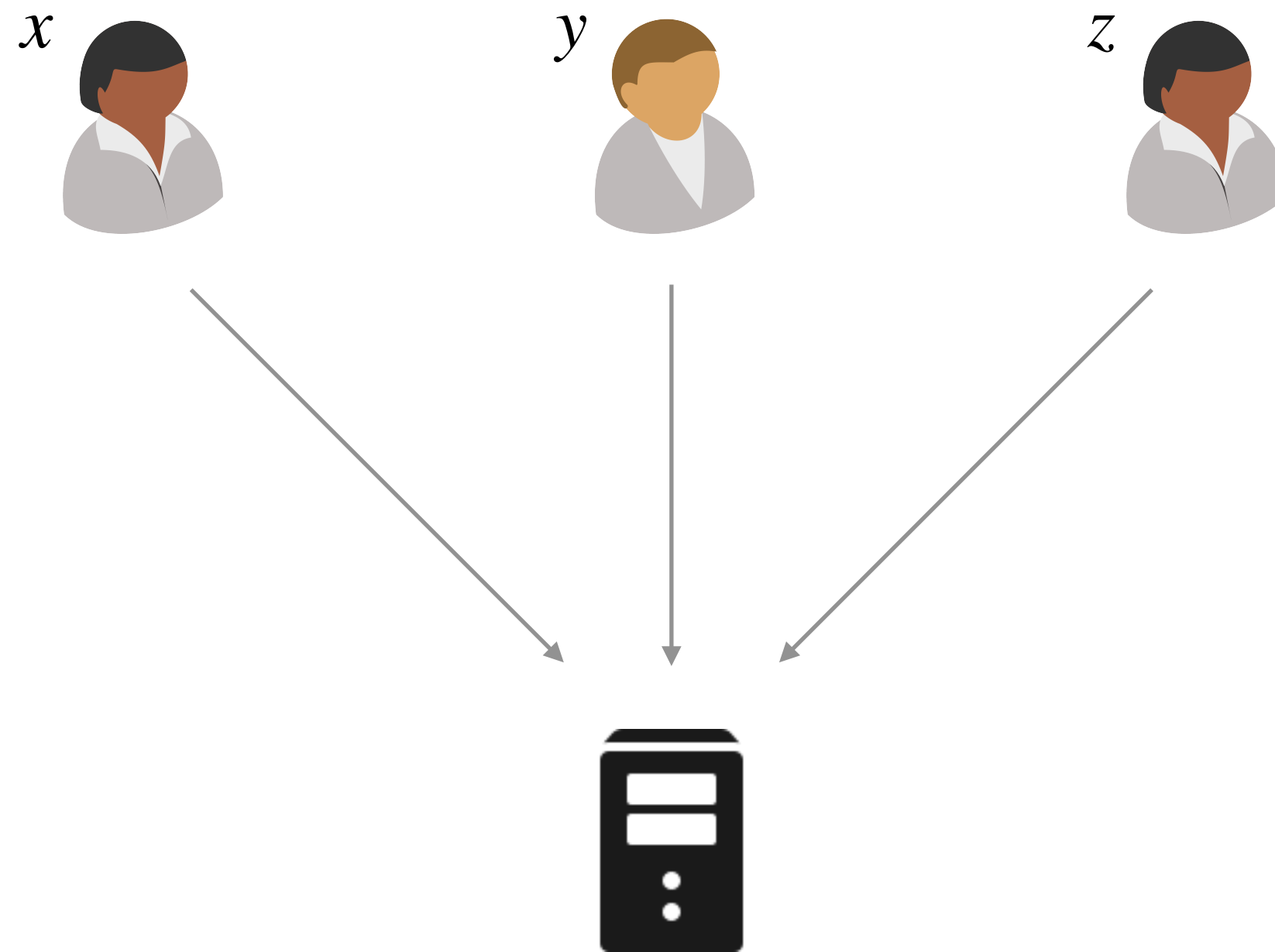
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

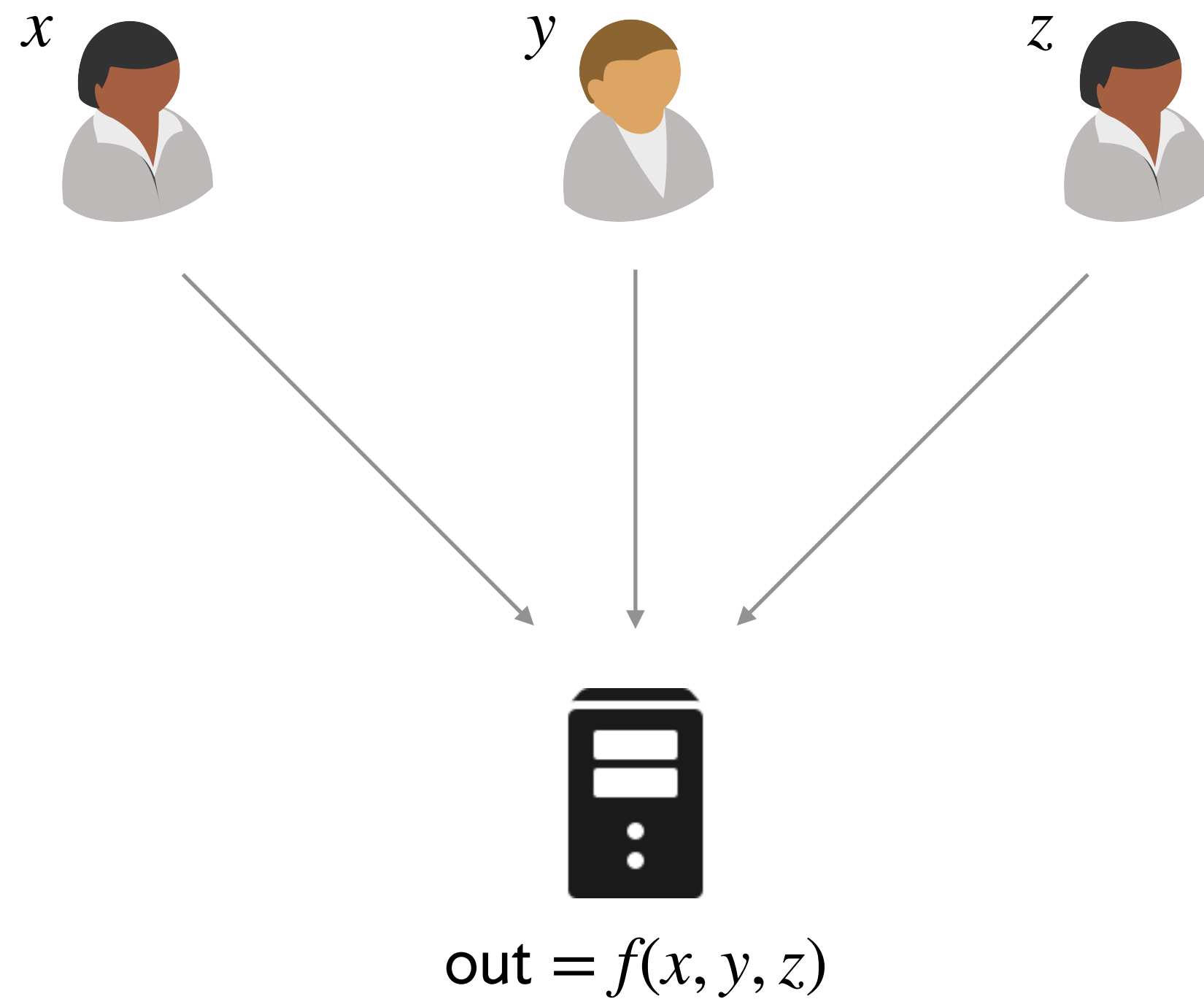
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

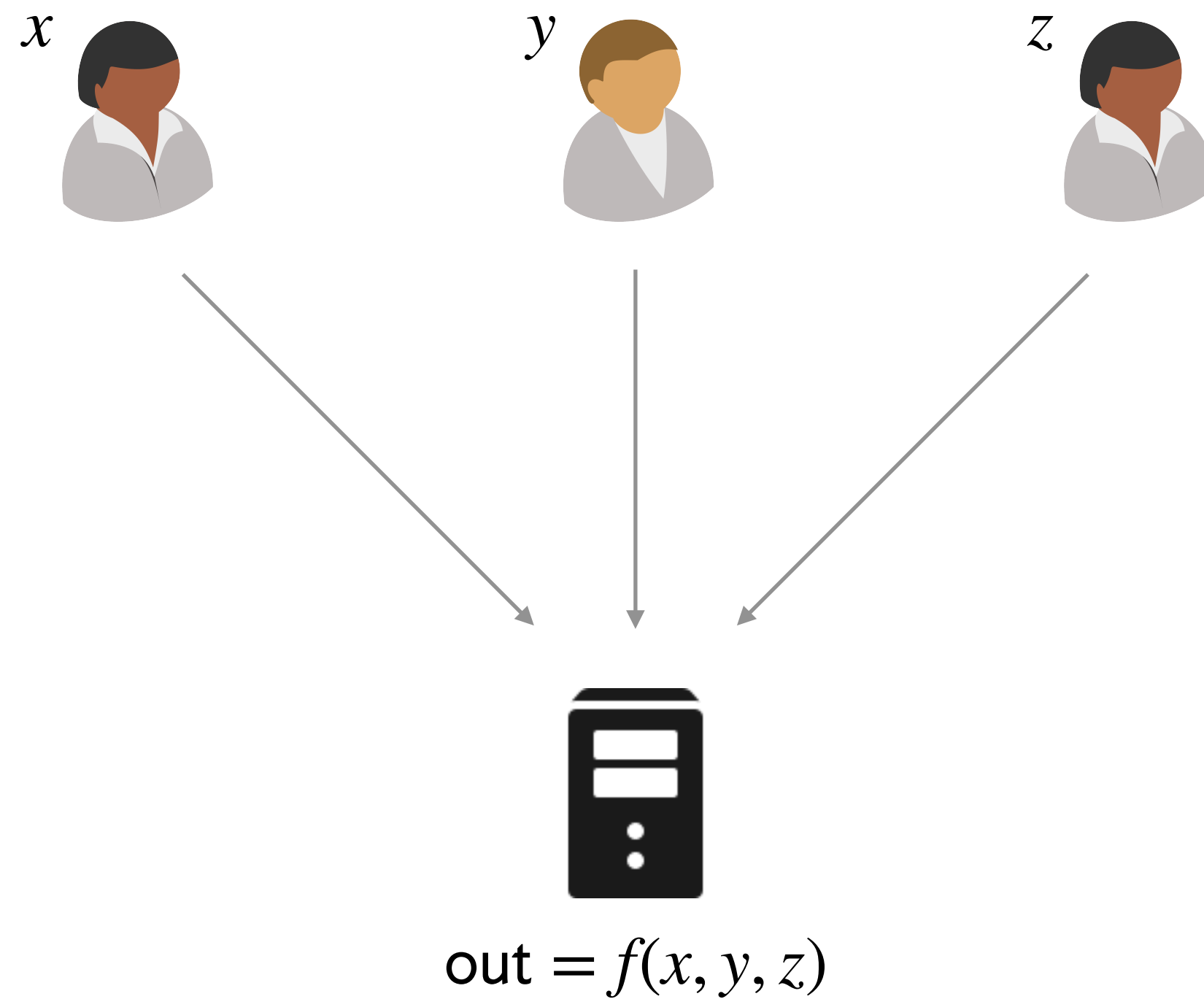
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

Function independent
client runtime

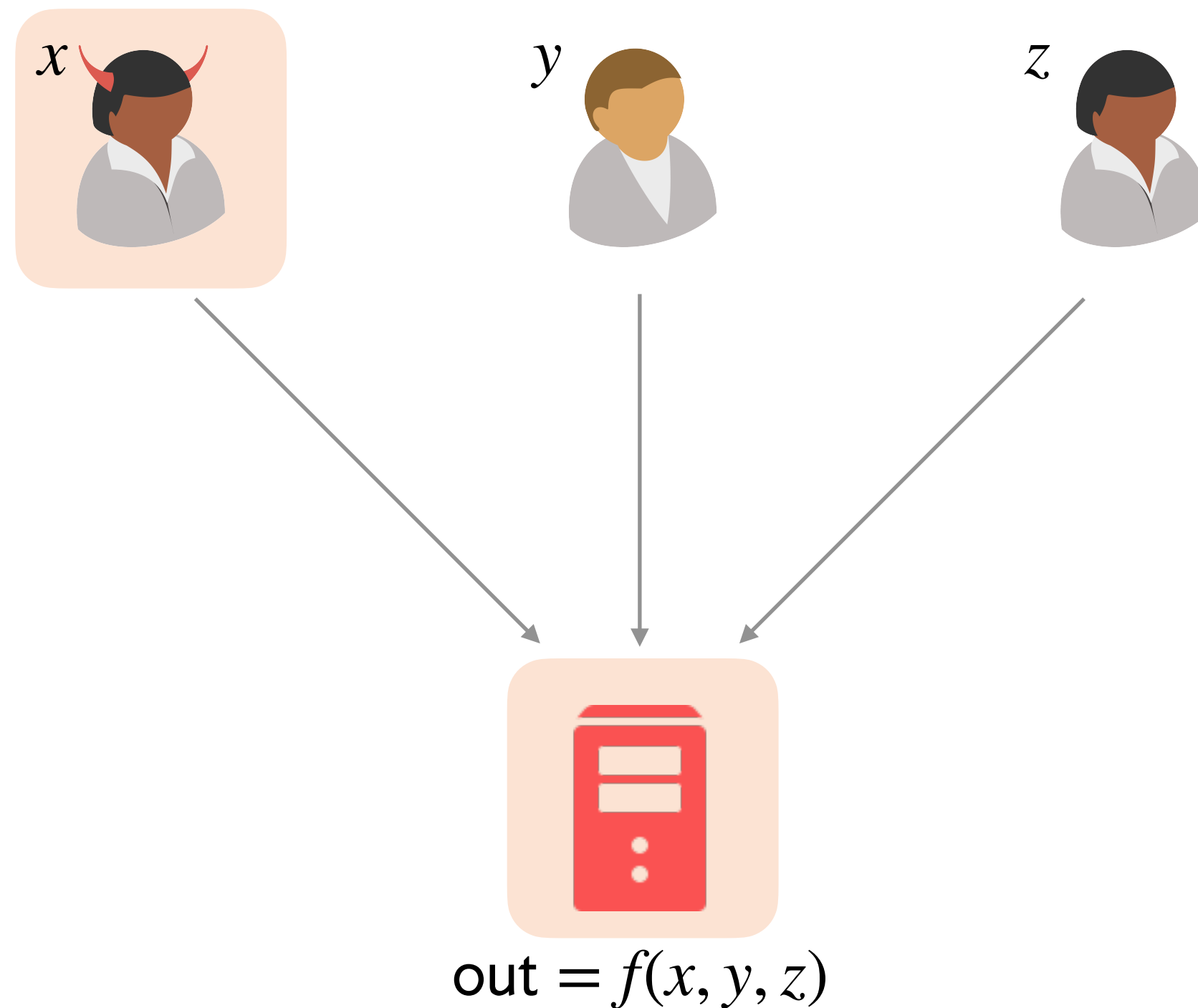
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Impossible for arbitrary functions

How Close Can We Get to the Ideal World?



Attack: Compute $f(x_1, y, z), f(x_2, y, z), \dots$

Impossible for arbitrary functions

Goals

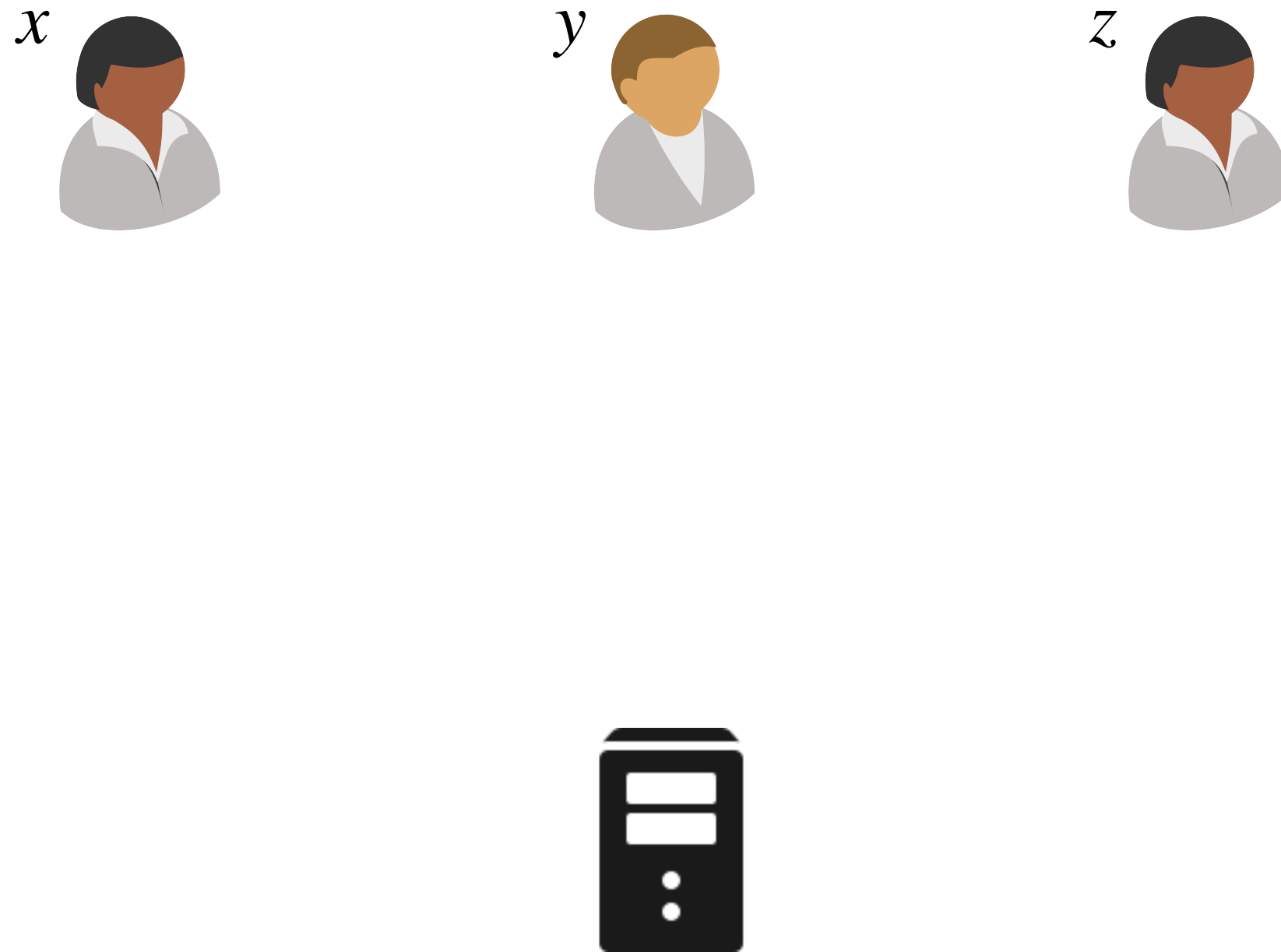
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

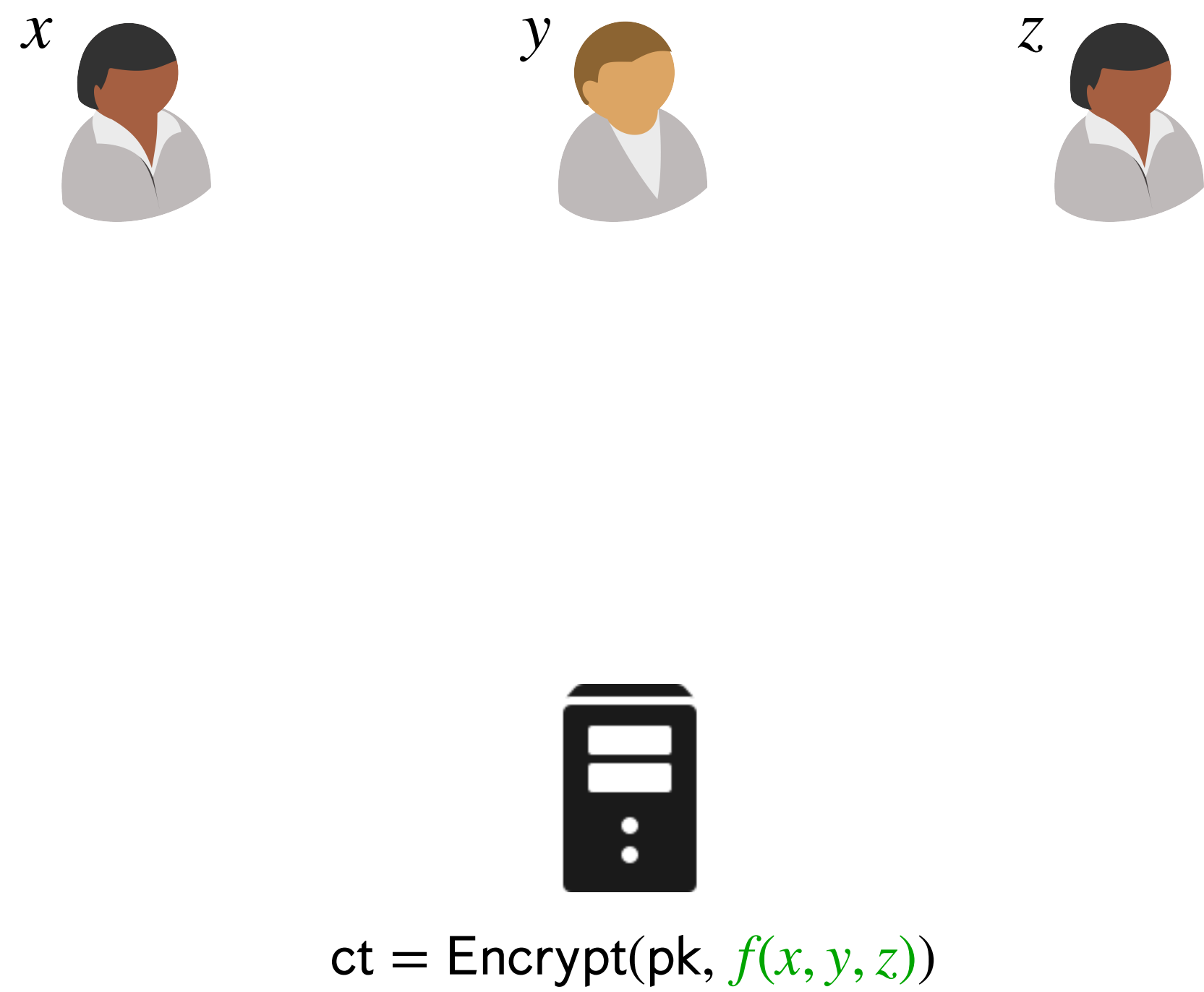
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Goals

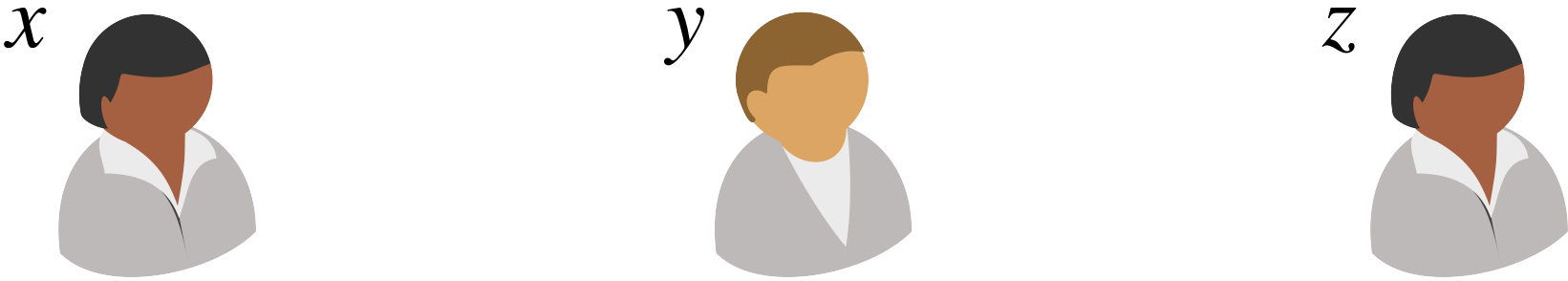
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



$$ct = \text{Encrypt}(pk, f(x, y, z))$$

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]
[Wichs-Mukherjee'16]

Goals

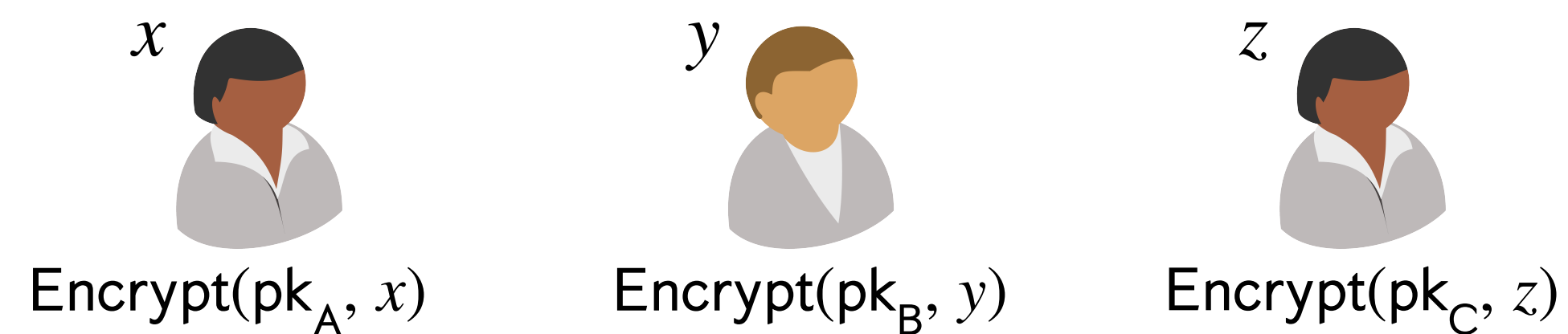
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]

[Wichs-Mukherjee'16]

Goals

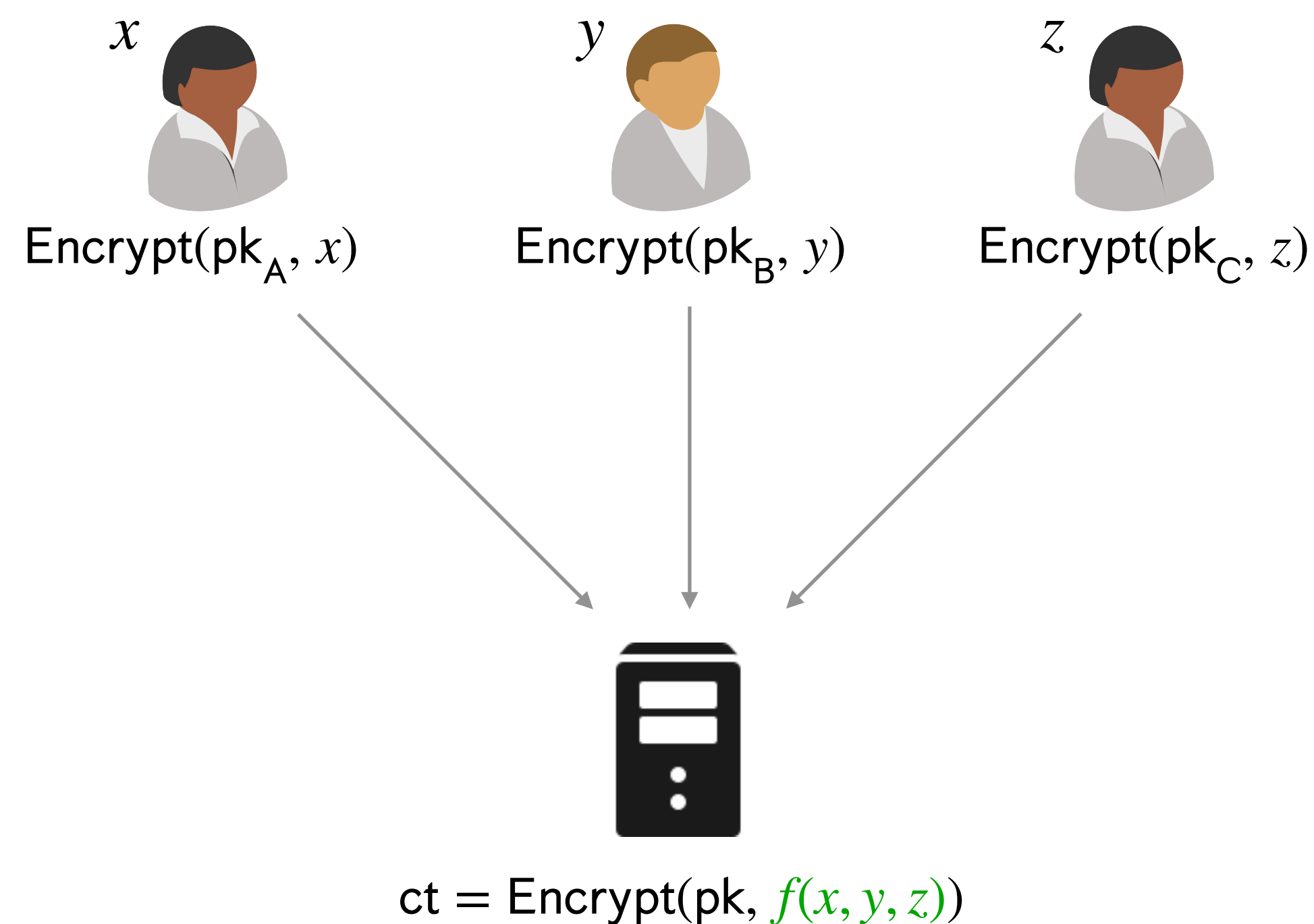
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]

[Wichs-Mukherjee'16]

Goals

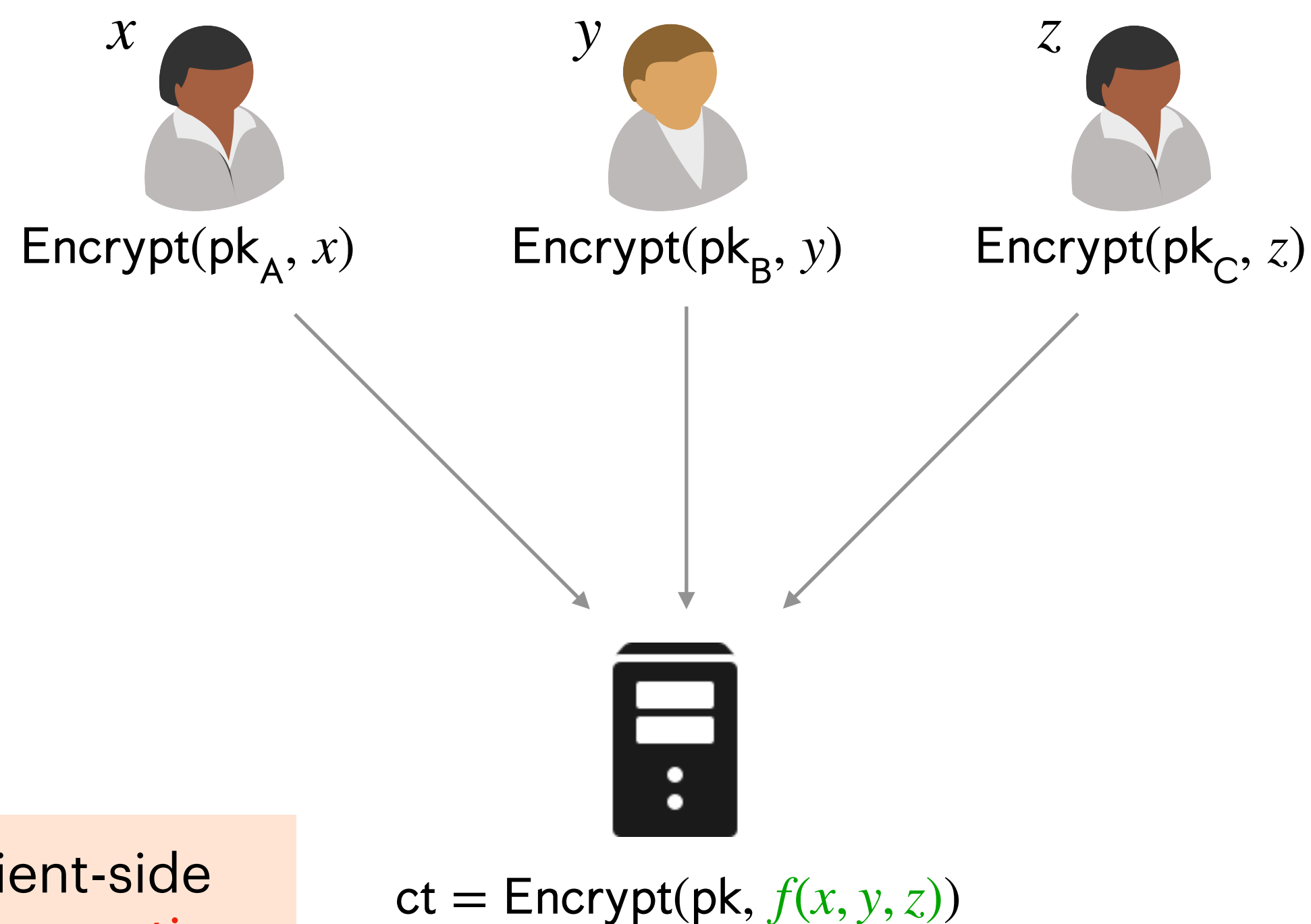
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Requires client-side
distributed decryption

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]

[Wichs-Mukherjee'16]

Goals

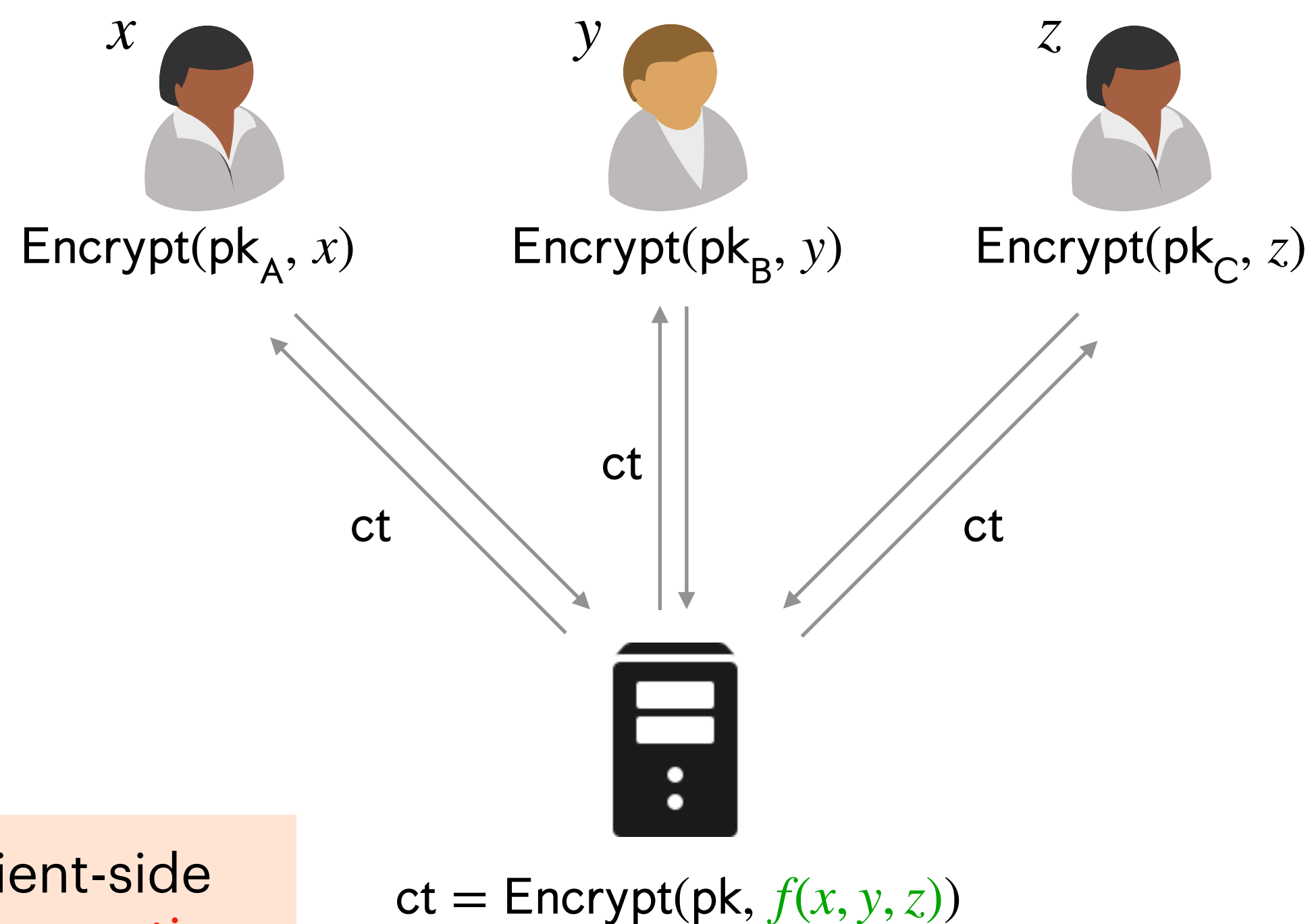
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Requires client-side
distributed decryption

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]

[Wichs-Mukherjee'16]

Goals

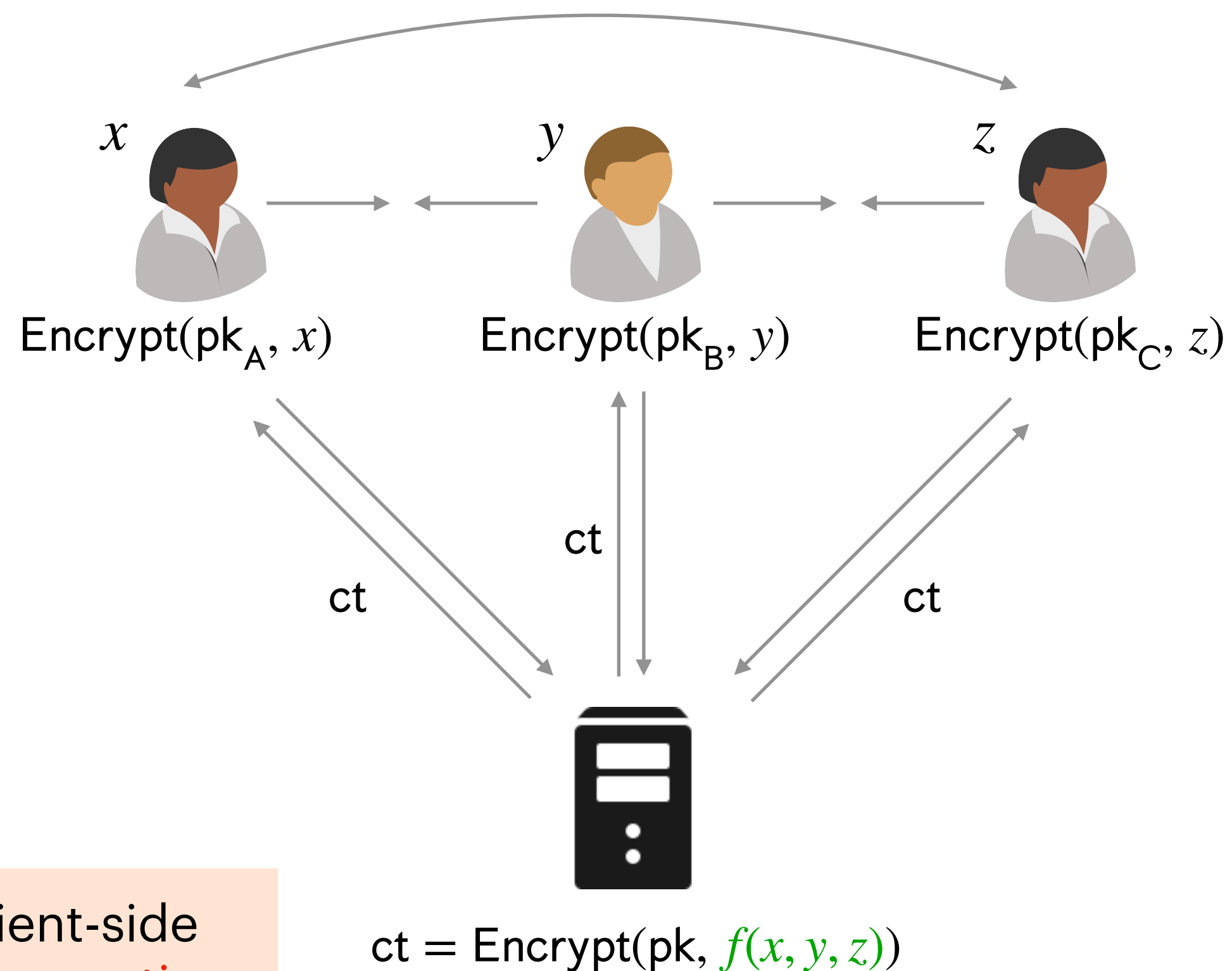
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Requires client-side
distributed decryption

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]
[Wichs-Mukherjee'16]

Goals

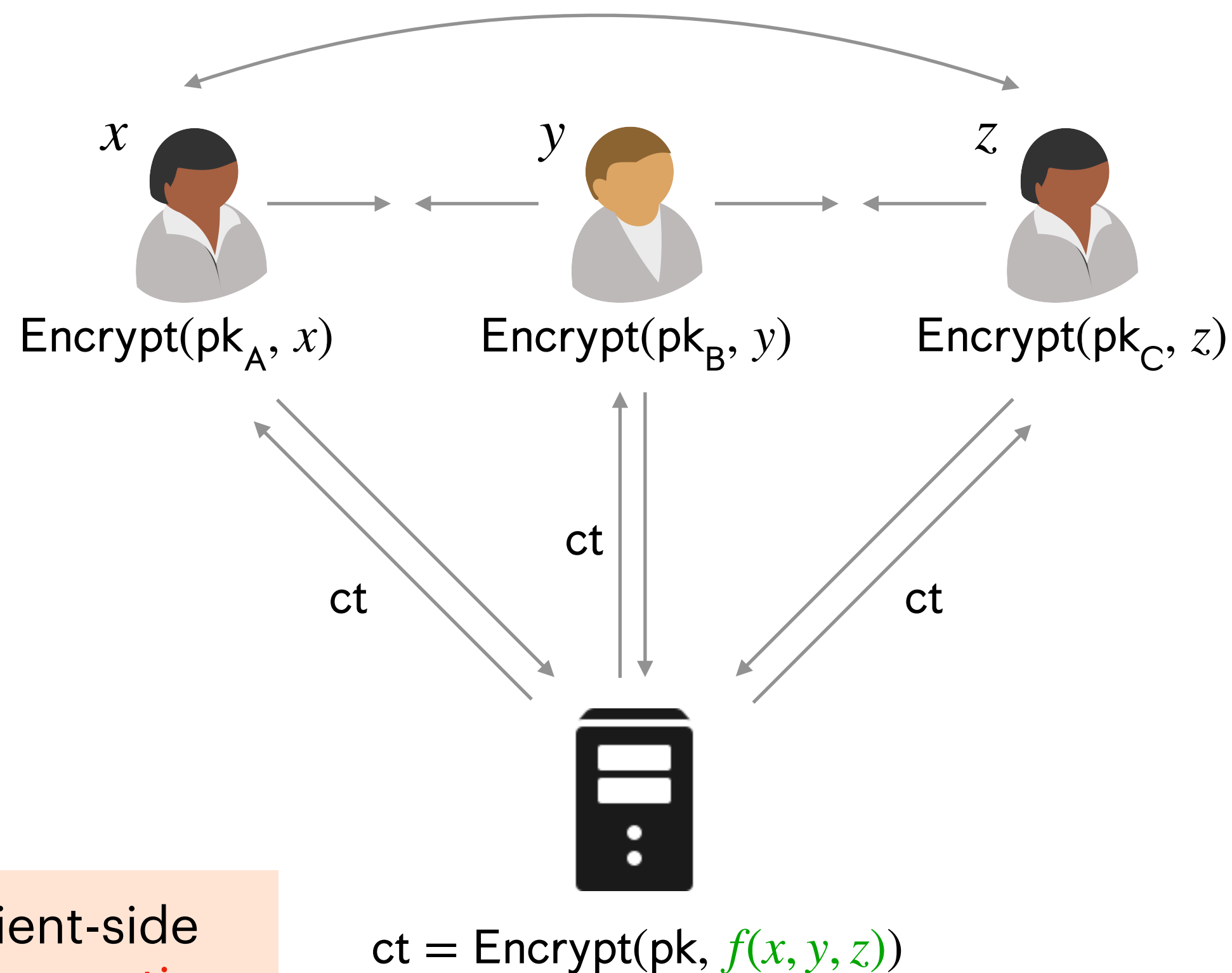
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Requires client-side
distributed decryption

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]
[Wichs-Mukherjee'16]

Goals

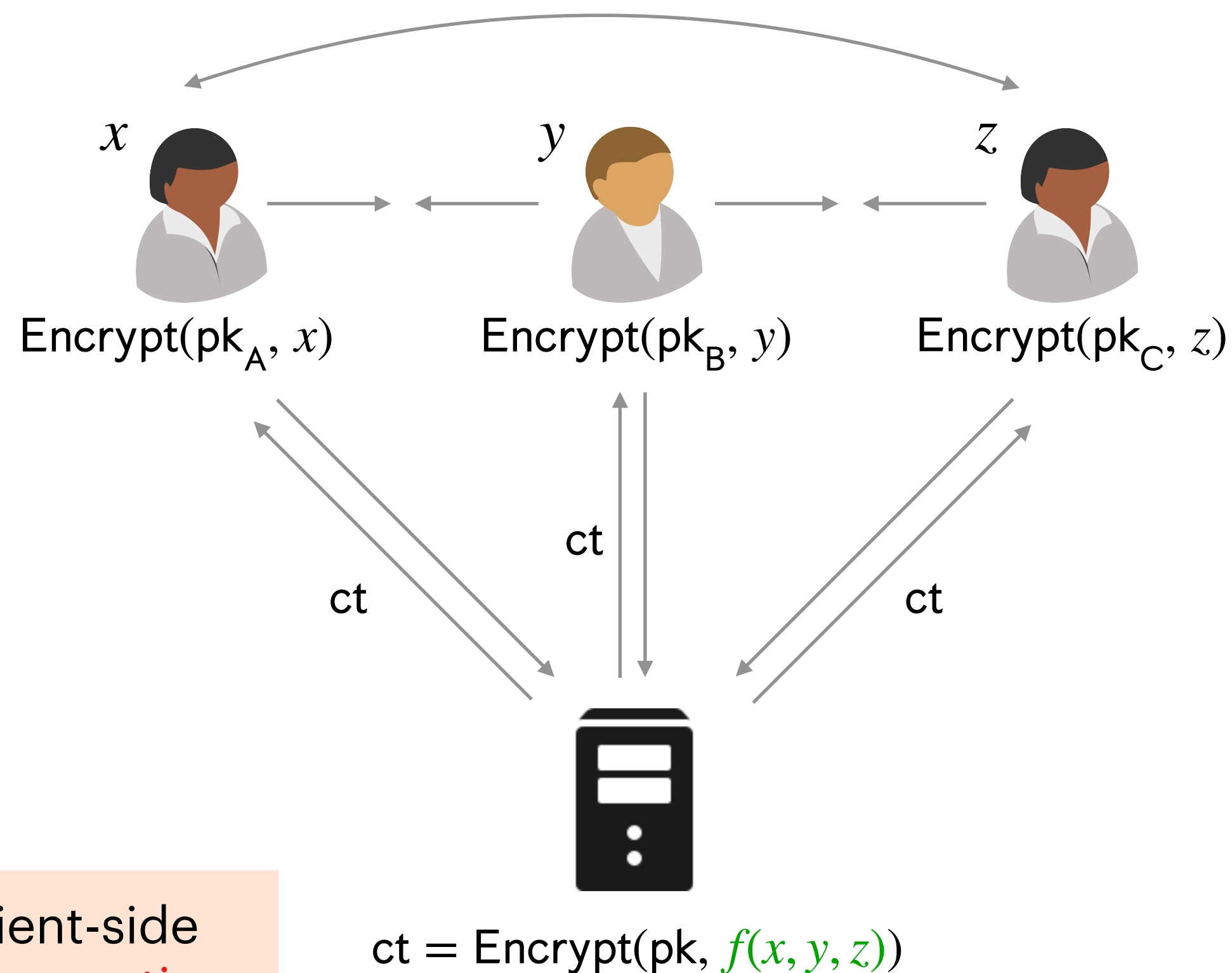
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Requires client-side
distributed decryption

Multi-key FHE

[López-Alt–Tromer–Vaikuntanathan'12]
[Wichs-Mukherjee'16]

Goals

Function independent
client runtime

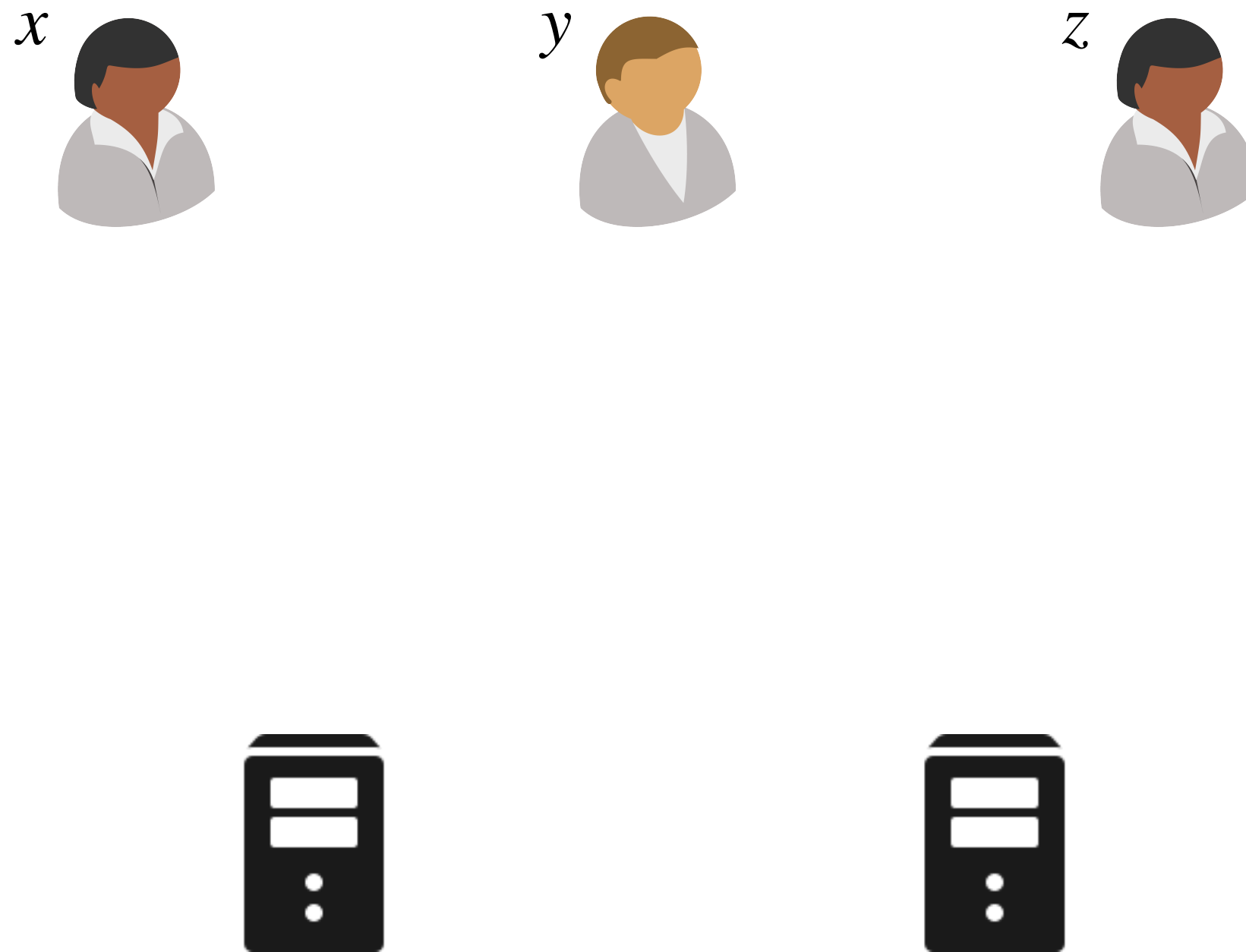
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Multi-key FHE achieves all the other goals

How Close Can We Get to the Ideal World?



Goals

Function independent
client runtime

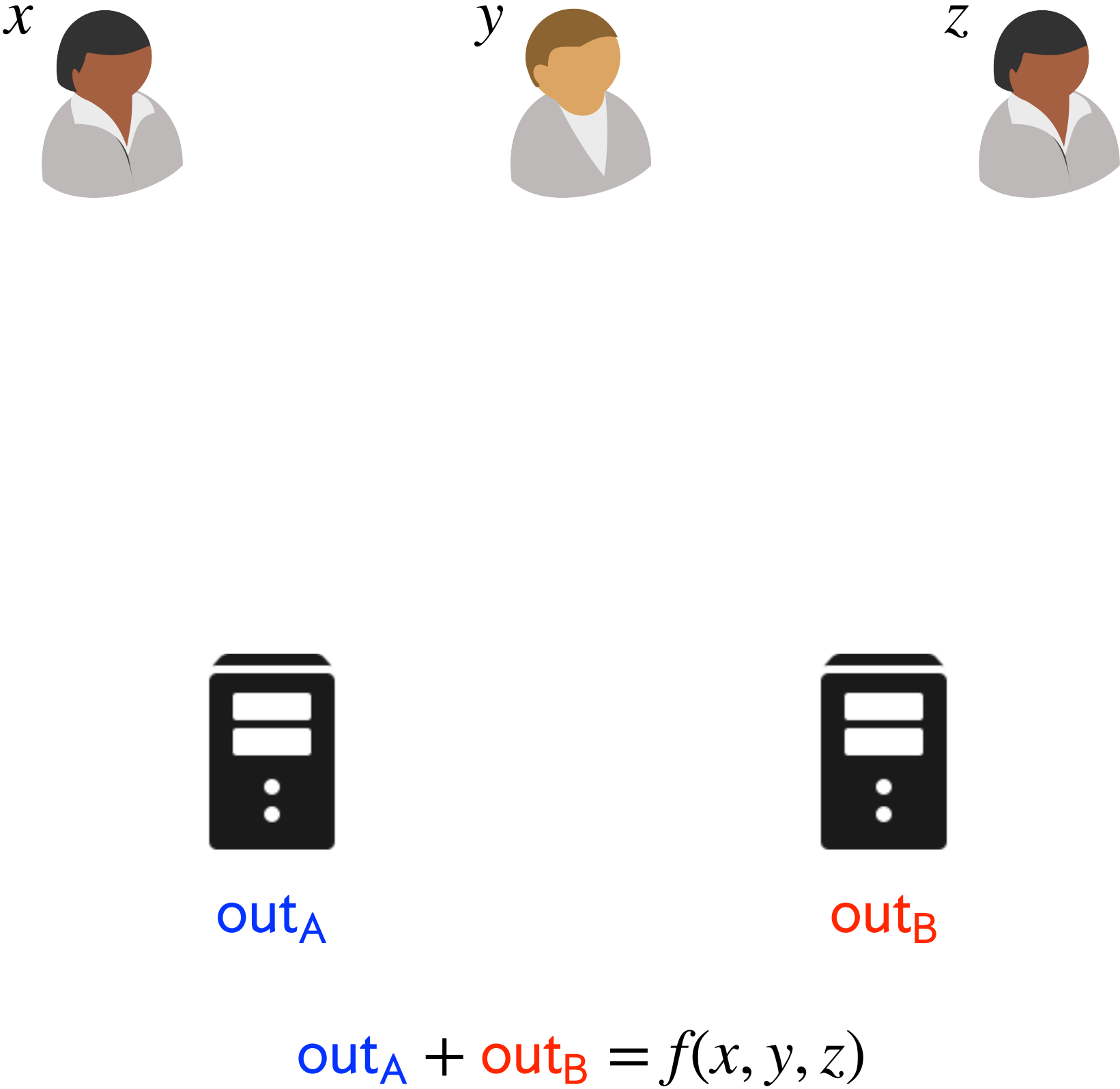
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Distribute the output across multiple servers

How Close Can We Get to the Ideal World?



Distribute the output across multiple servers

Goals

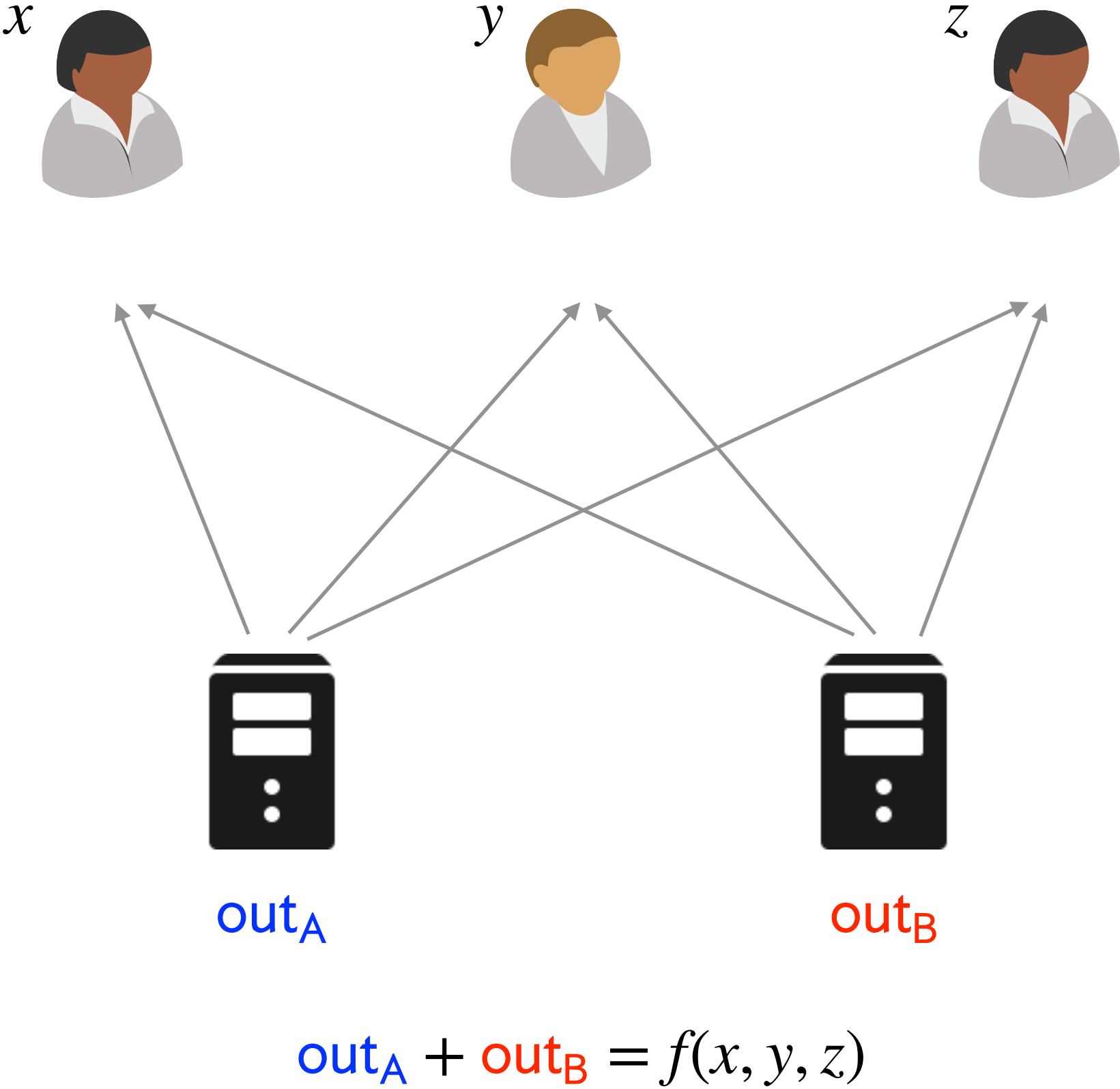
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

How Close Can We Get to the Ideal World?



Distribute the output across multiple servers

Goals

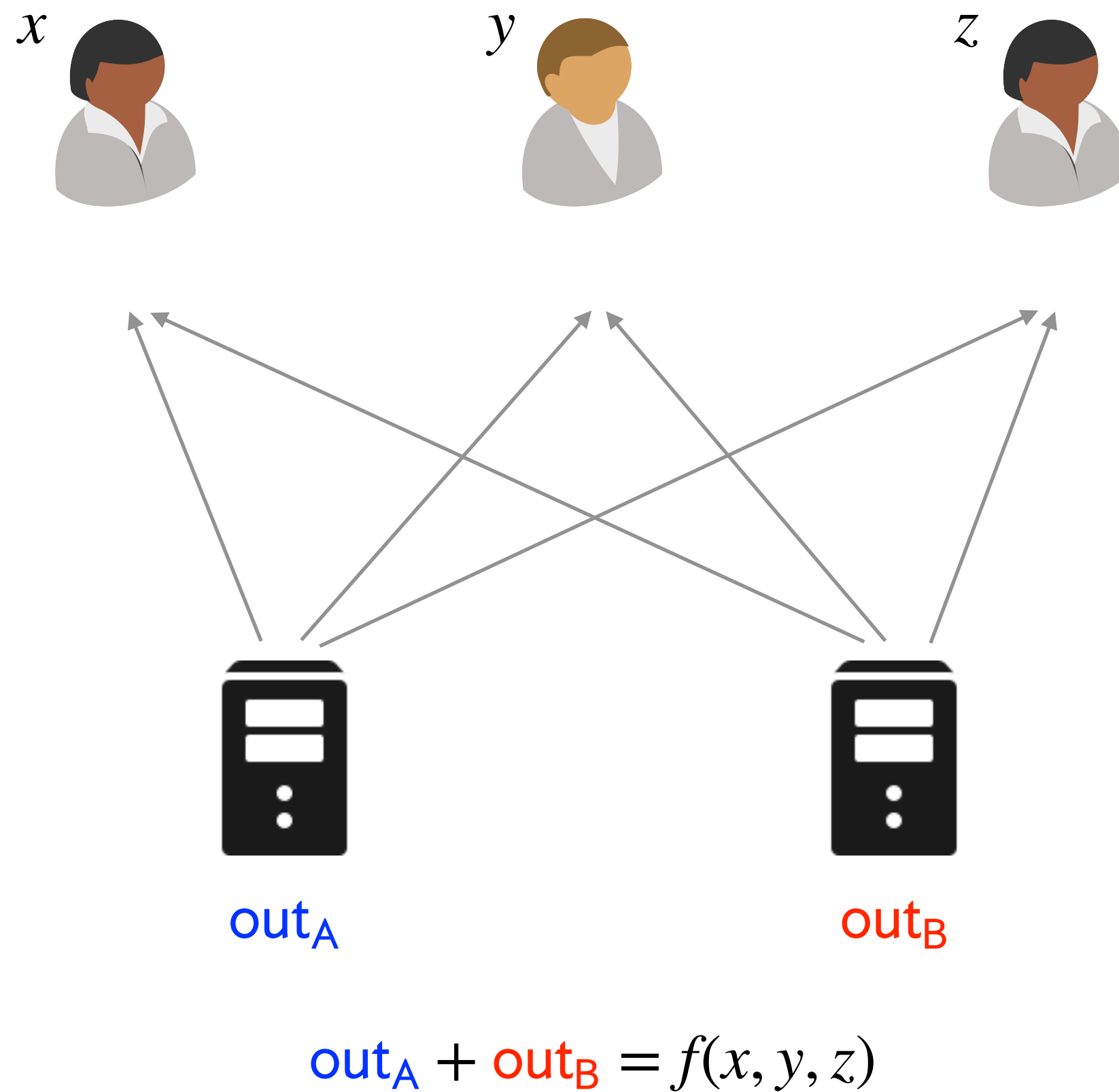
Function independent client runtime

Function independent communication cost

On-the-fly choice of clients, server, and function

Two rounds

How Close Can We Get to the Ideal World?



Distribute the output across multiple servers

Goals

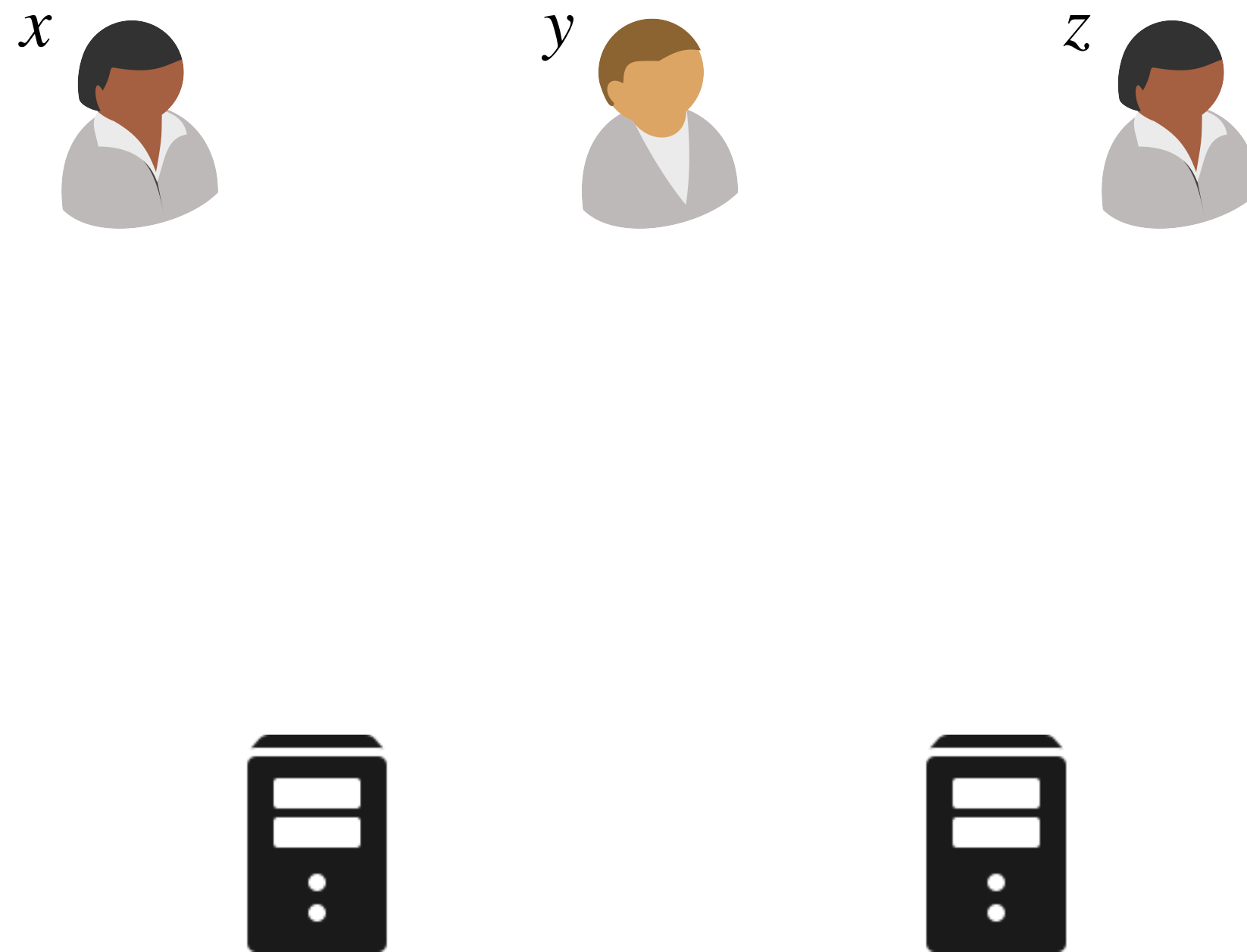
Function independent
client runtime

Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Client-Server HSS



Goals

Function independent
client runtime

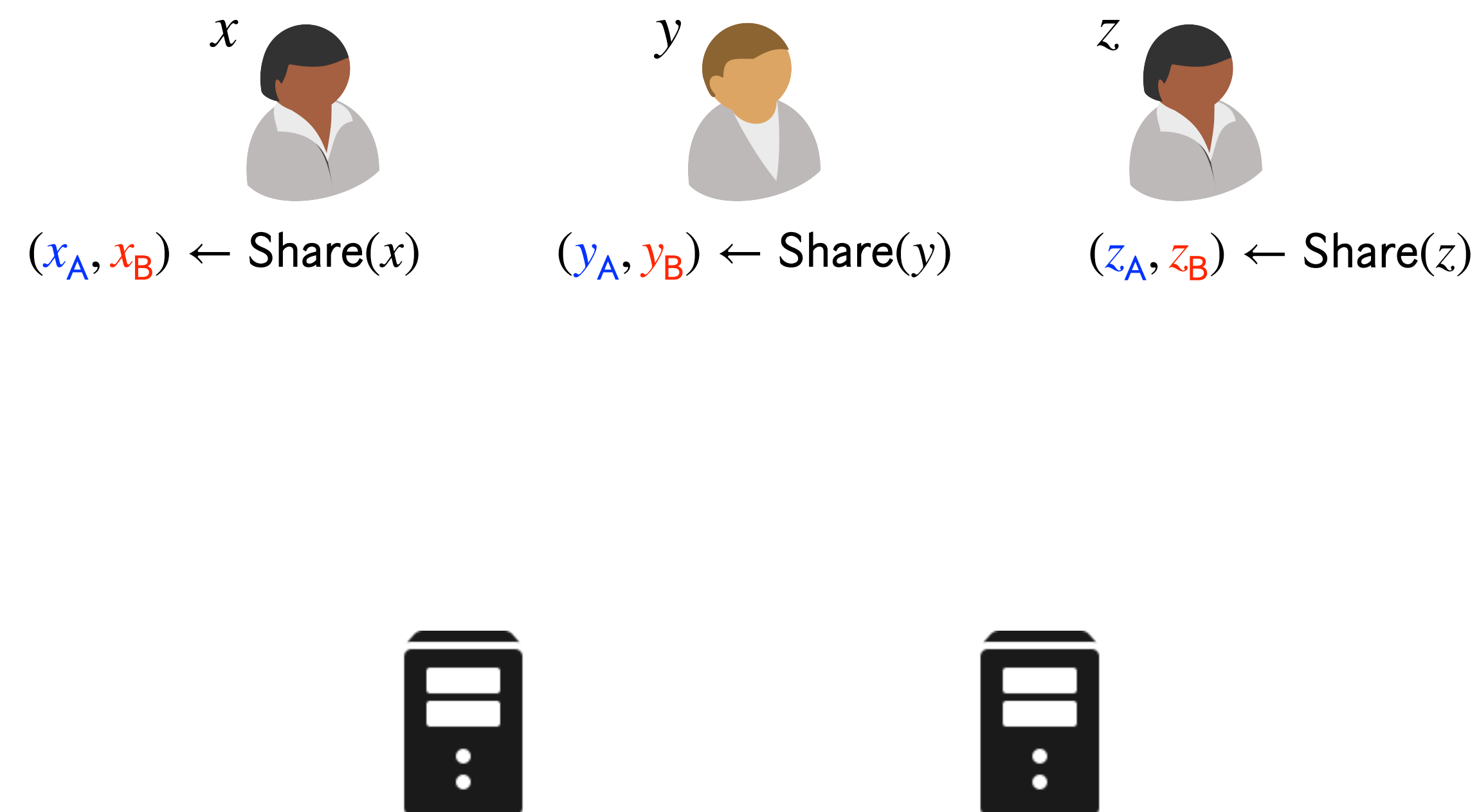
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Client-Server HSS: **Distributed variant** of multi-key FHE

Client-Server HSS

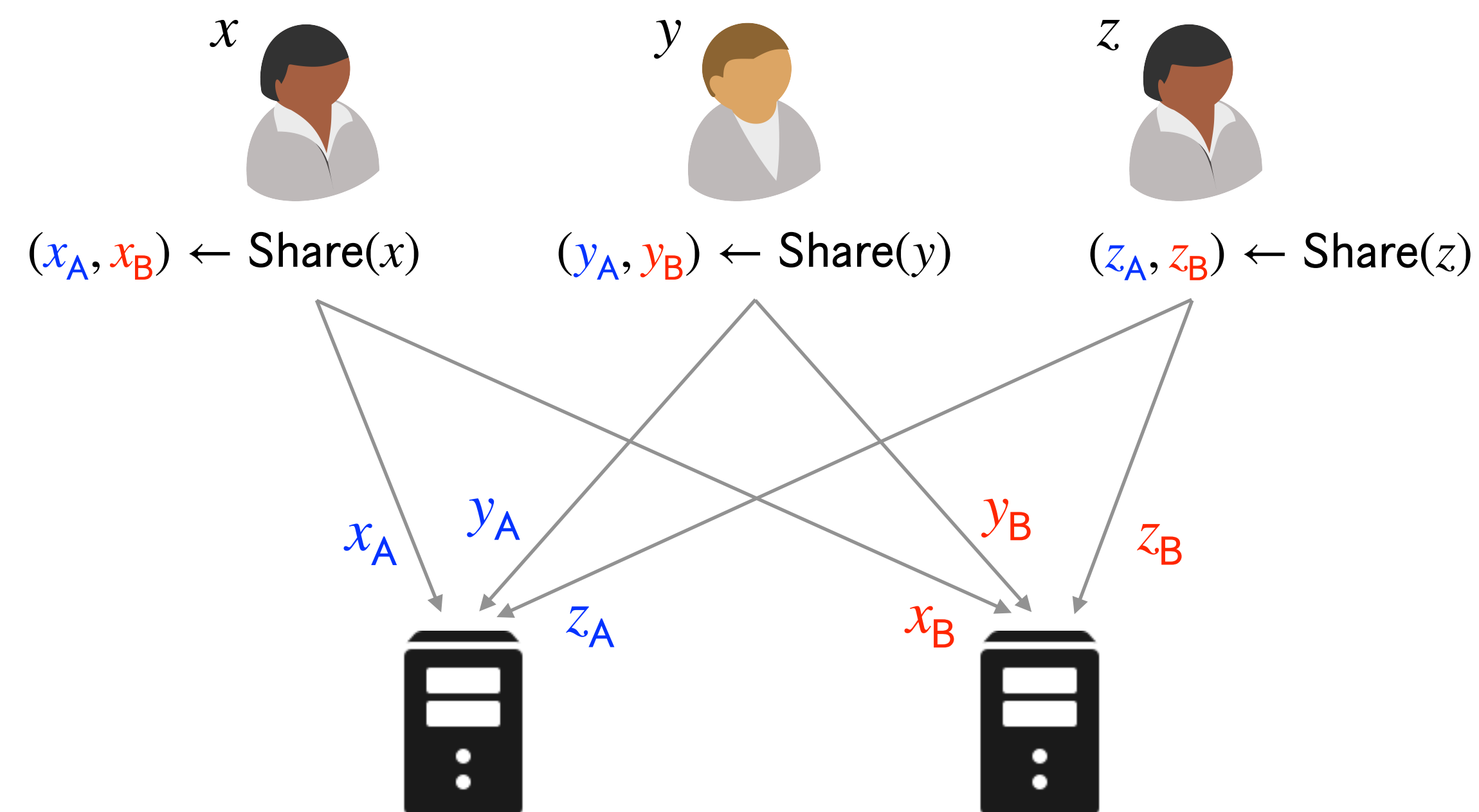


Goals

- Function independent client runtime
- Function independent communication cost
- On-the-fly choice of clients, server, and function
- Two rounds

Client-Server HSS: Distributed variant of multi-key FHE

Client-Server HSS



Goals

Function independent
client runtime

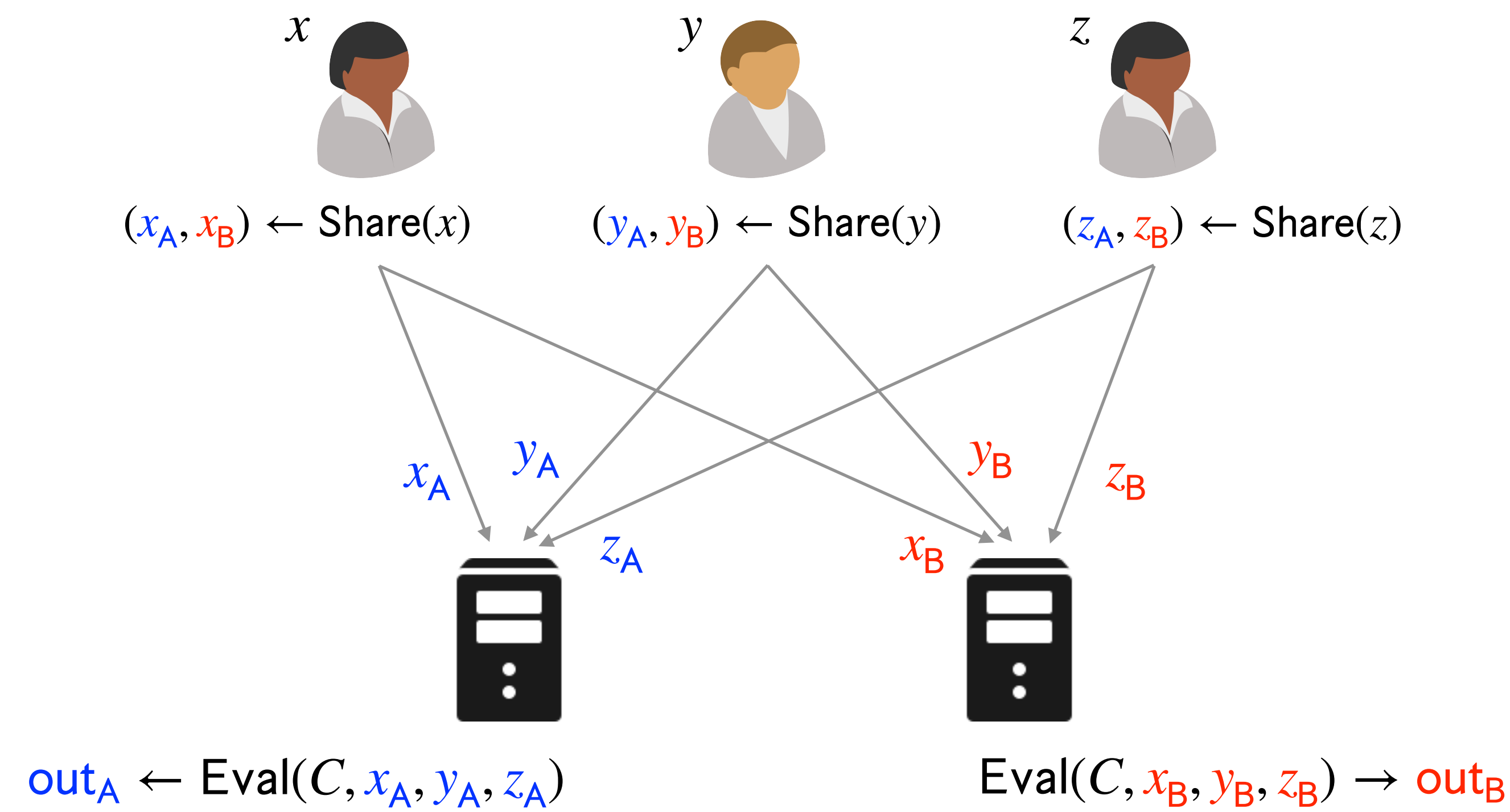
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Client-Server HSS: **Distributed variant** of multi-key FHE

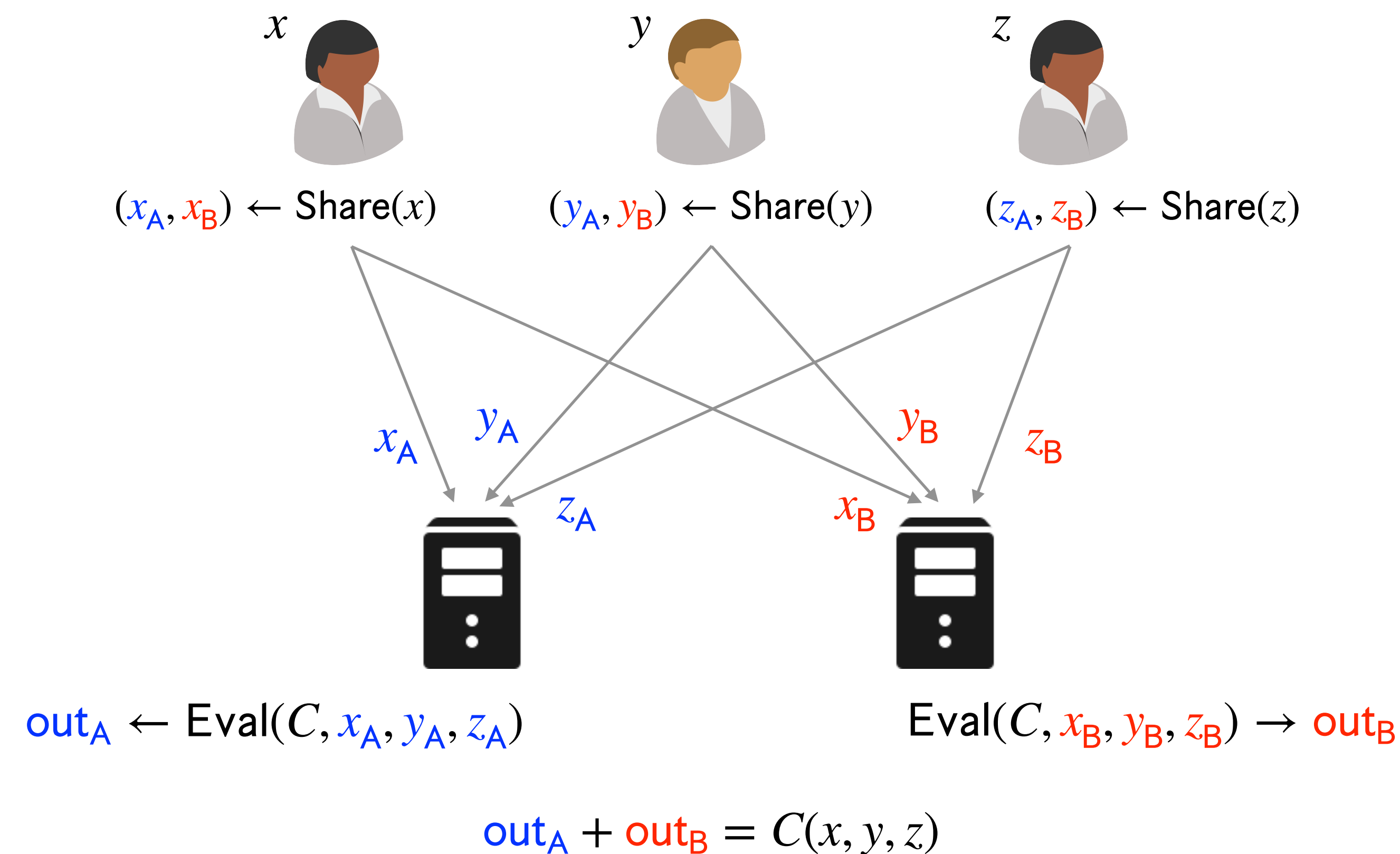
Client-Server HSS



Goals
Function independent client runtime
Function independent communication cost
On-the-fly choice of clients, server, and function
Two rounds

Client-Server HSS: Distributed variant of multi-key FHE

Client-Server HSS



Goals

Function independent
client runtime

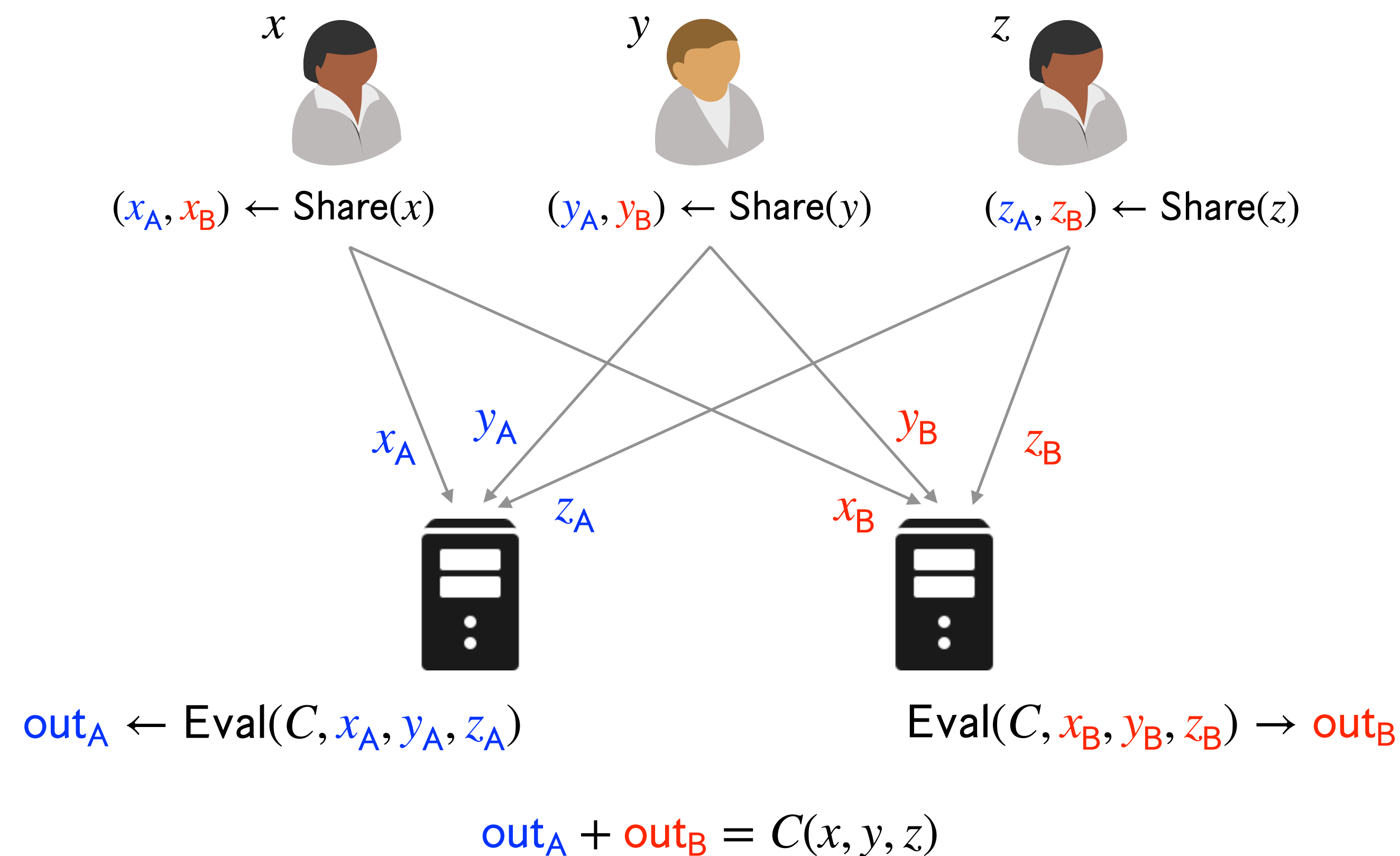
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

Client-Server HSS: **Distributed variant** of multi-key FHE

Client-Server HSS



Client-Server HSS: **Distributed variant** of multi-key FHE

Goals

Function independent
client runtime

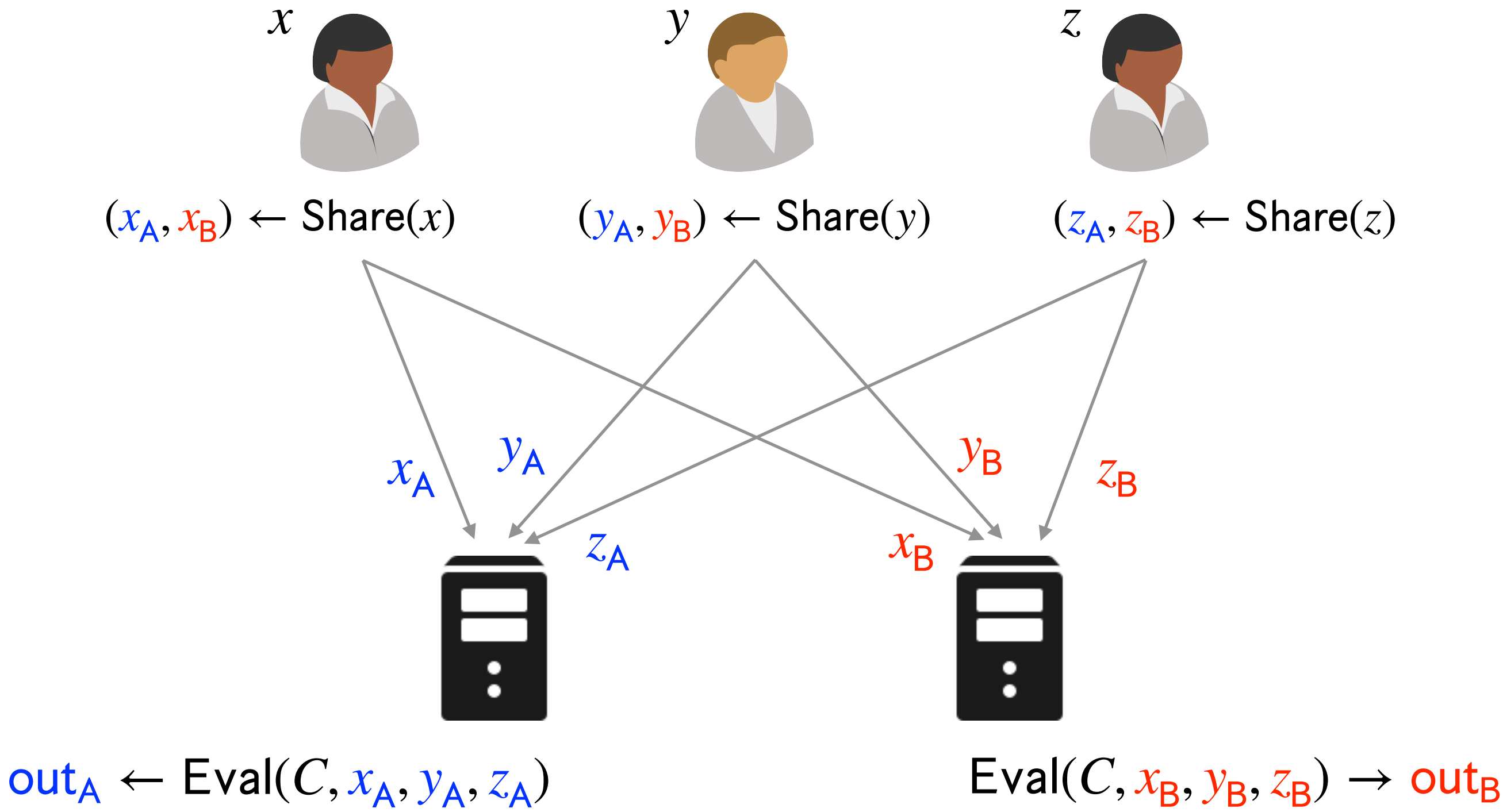
Function independent
communication cost

On-the-fly choice of clients,
server, and function

Two rounds

"Best possible" semi-honest
client-server secure computation

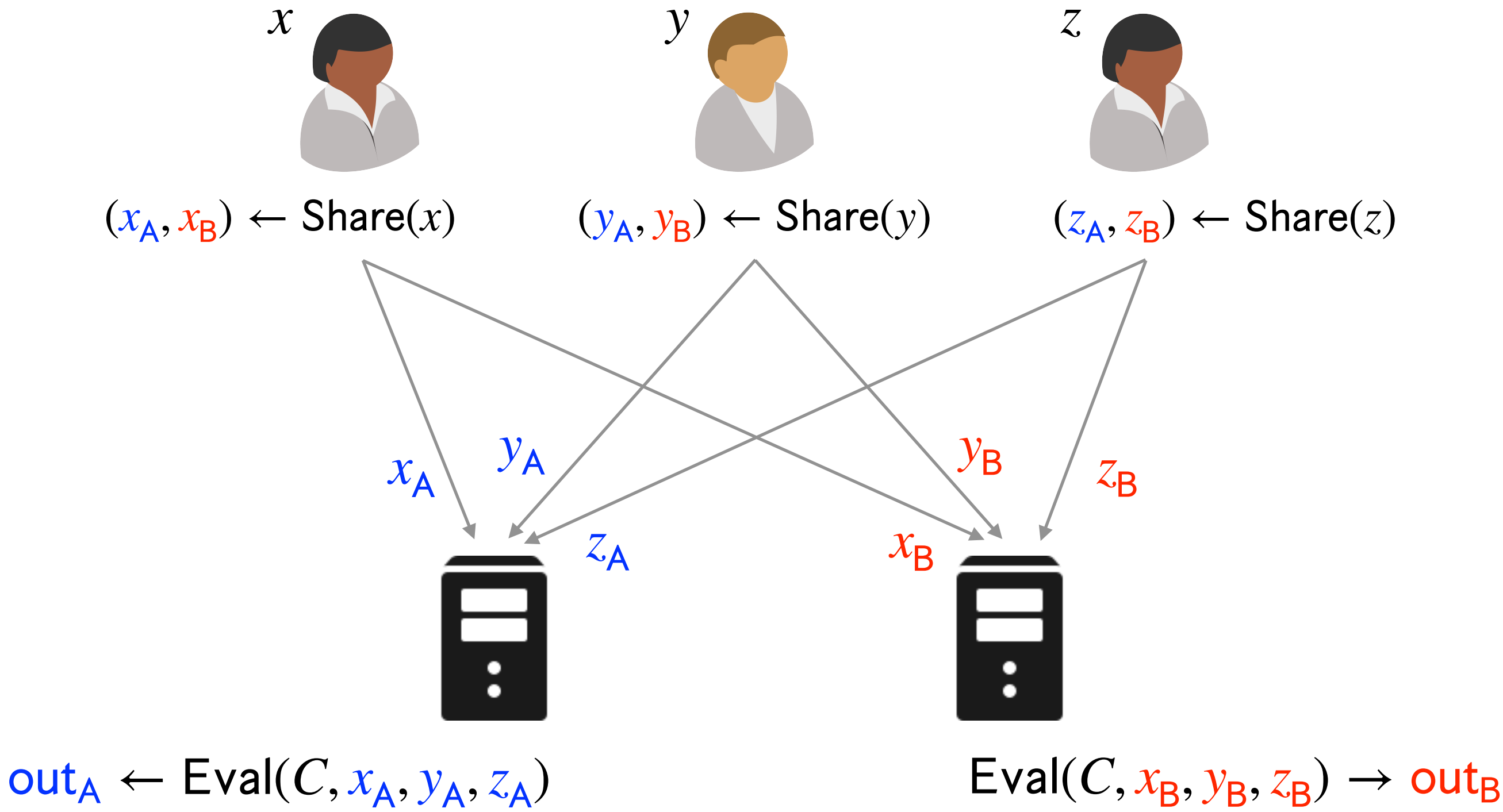
Client-Server HSS



Correctness

$$\text{out}_A + \text{out}_B = C(x, y, z)$$

Client-Server HSS



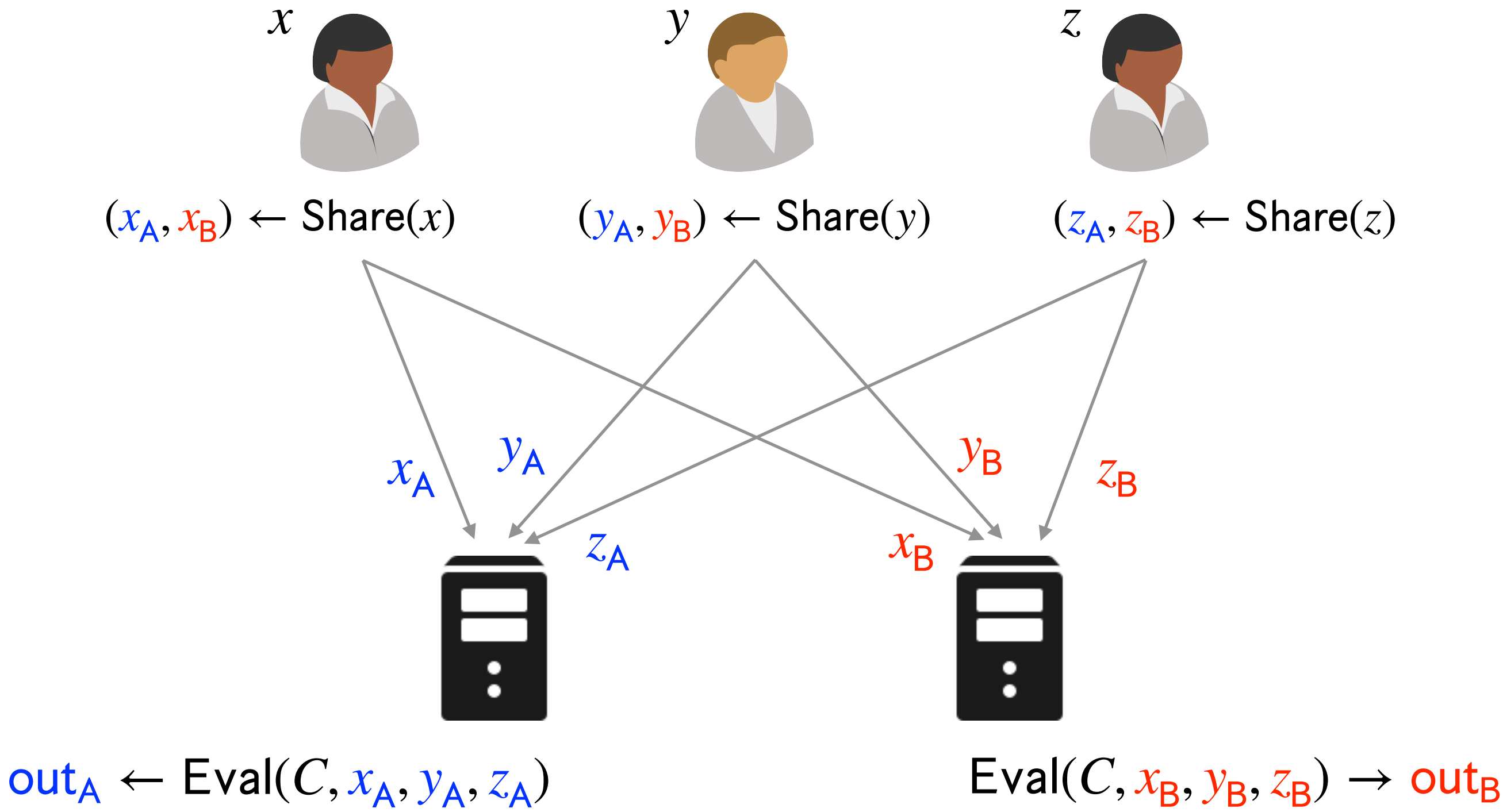
Correctness

$$\text{out}_A + \text{out}_B = C(x, y, z)$$

Security

Each server's share preserves input privacy

Client-Server HSS



Correctness

$$\text{out}_A + \text{out}_B = C(x, y, z)$$

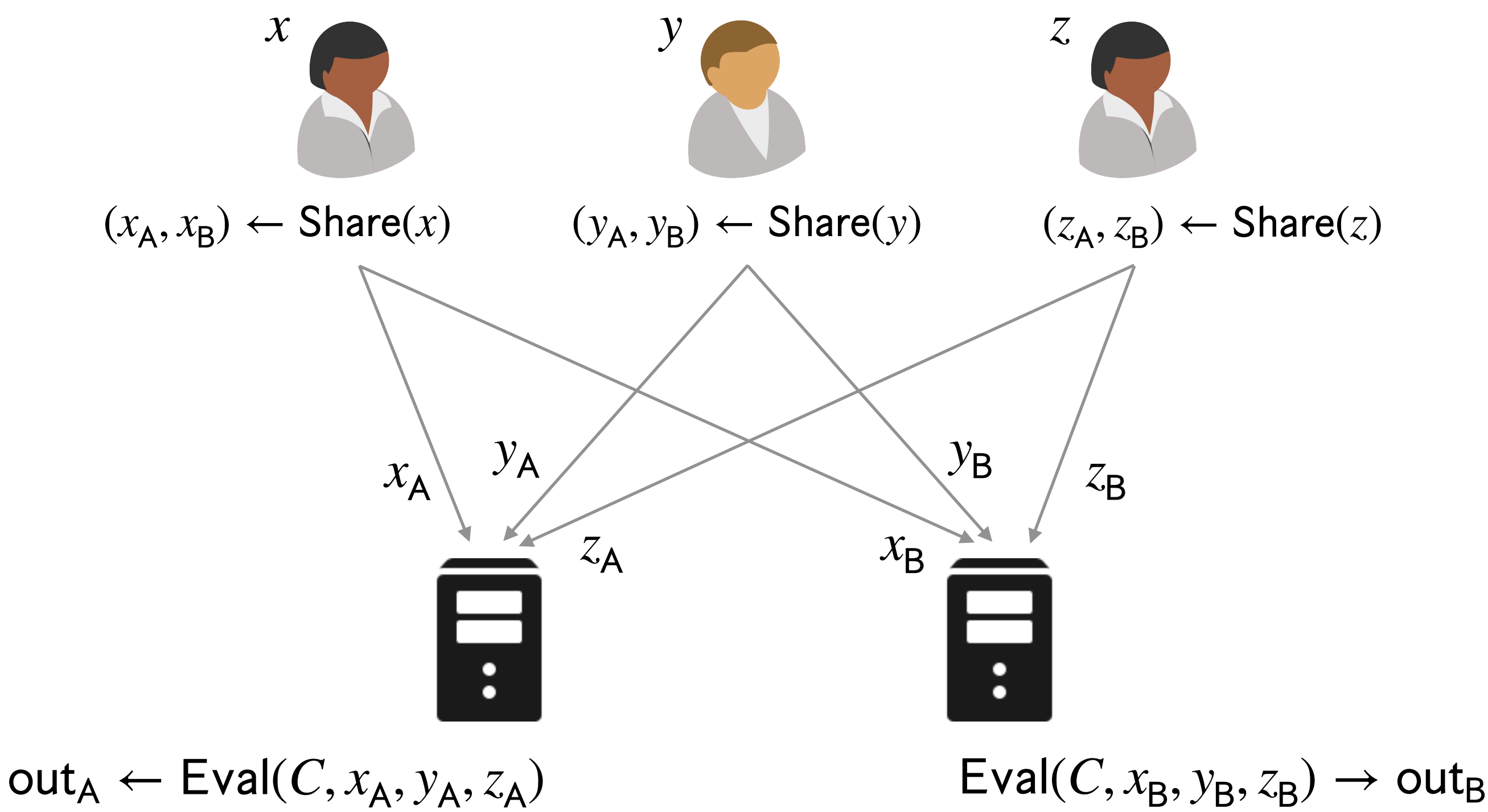
Security

Each server's share preserves input privacy

Succinctness

Input share size is circuit independent

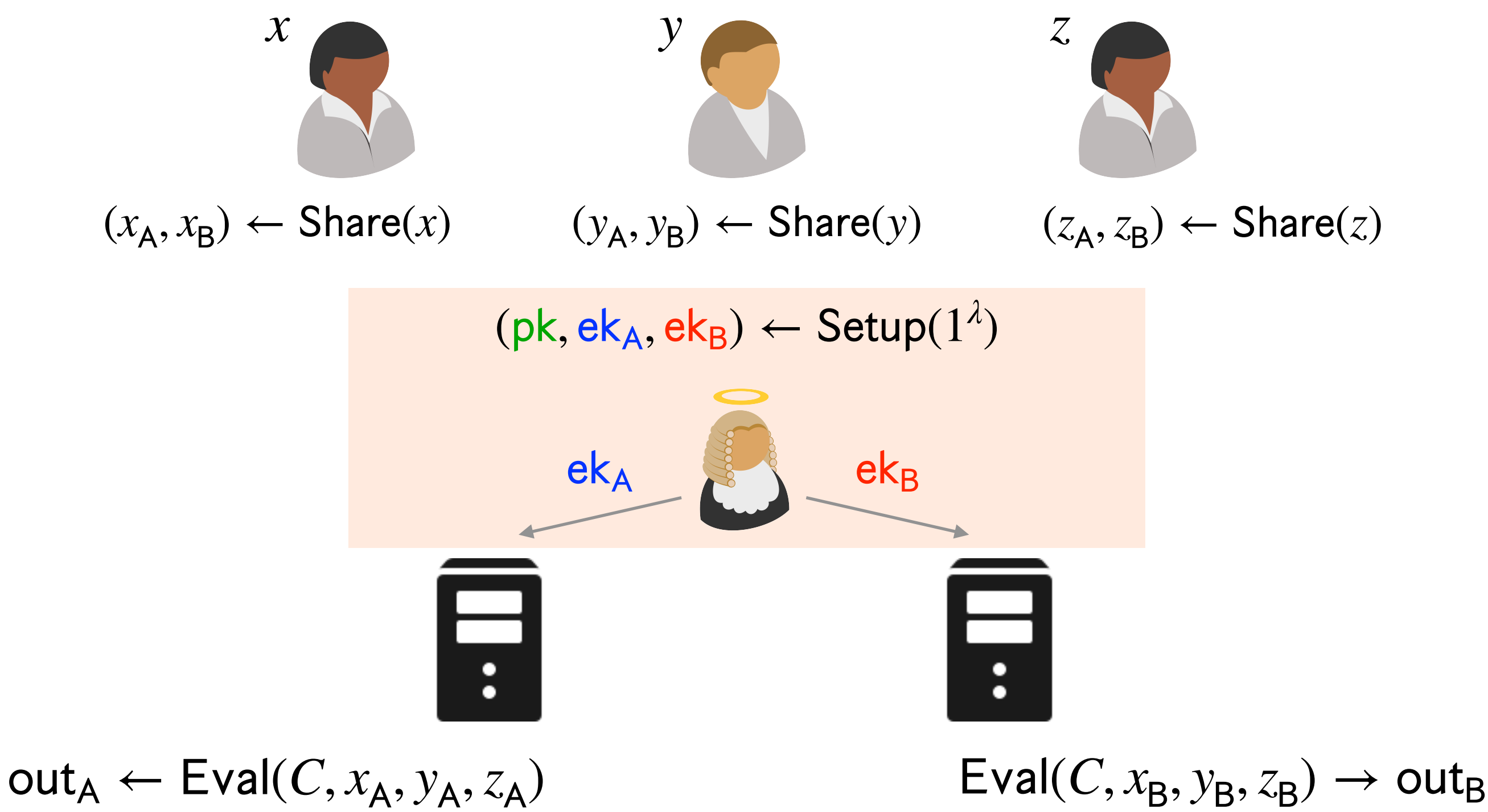
Client-Server HSS



Most existing client-server
HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

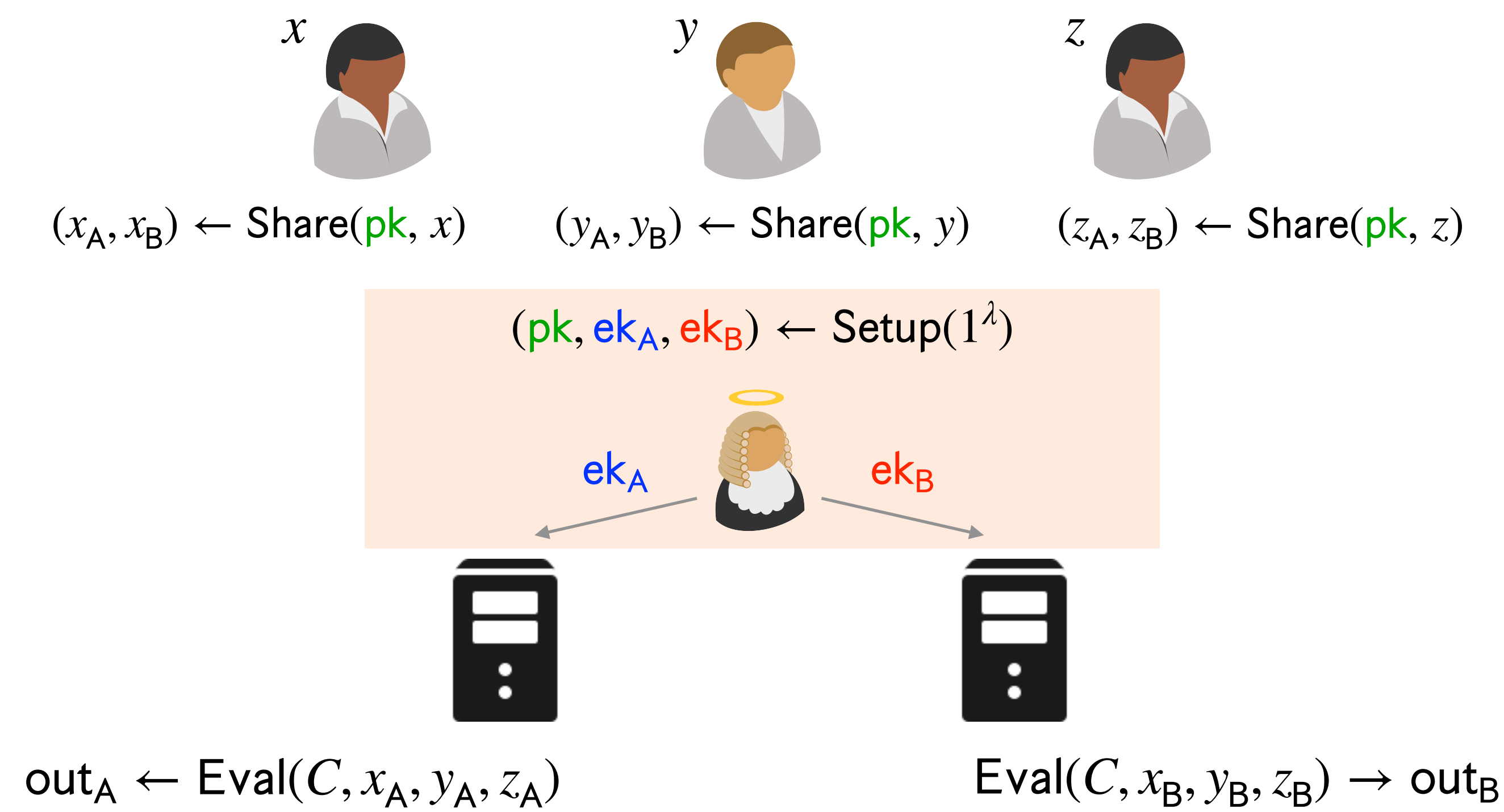
Client-Server HSS



Most existing client-server
HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

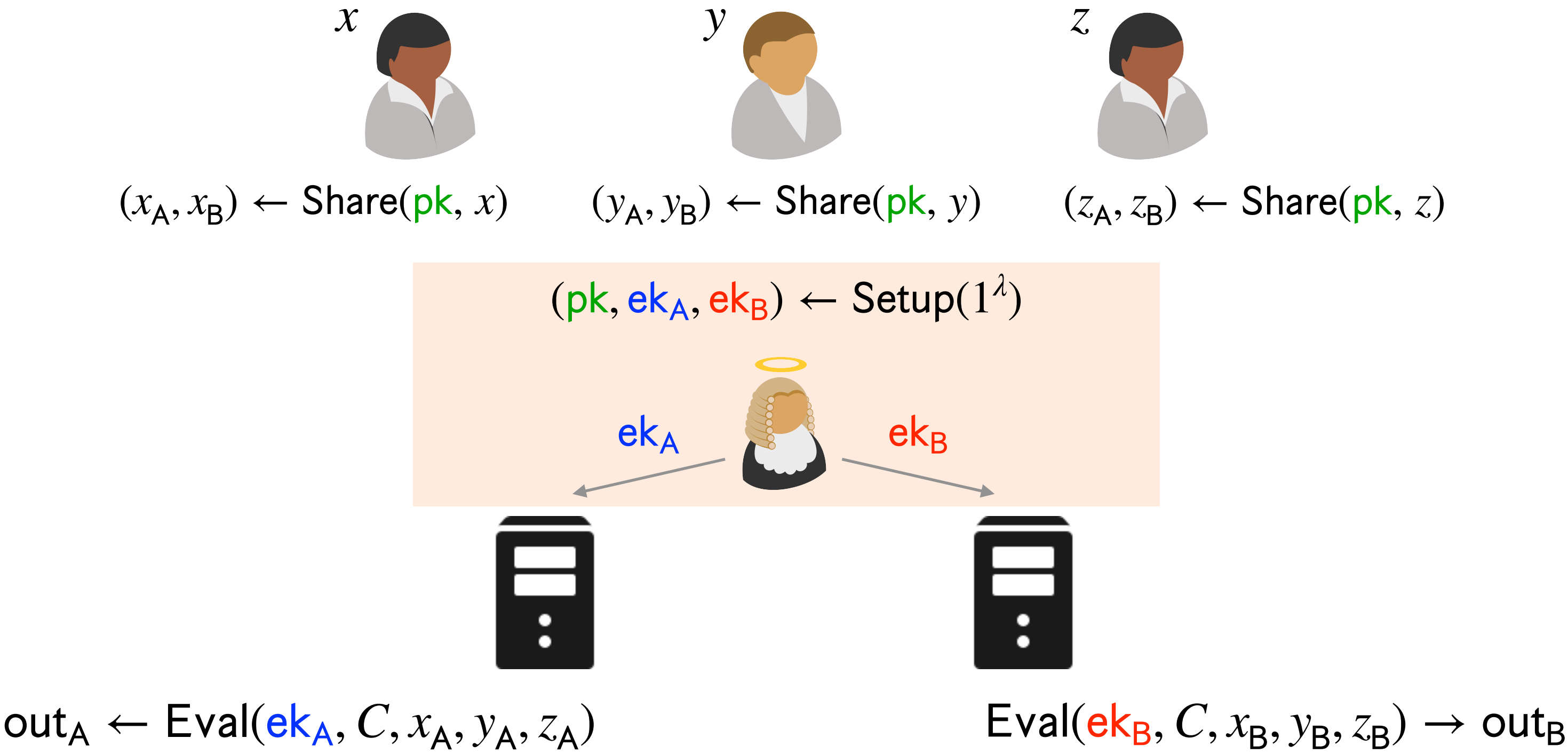
Client-Server HSS



Most existing client-server
HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

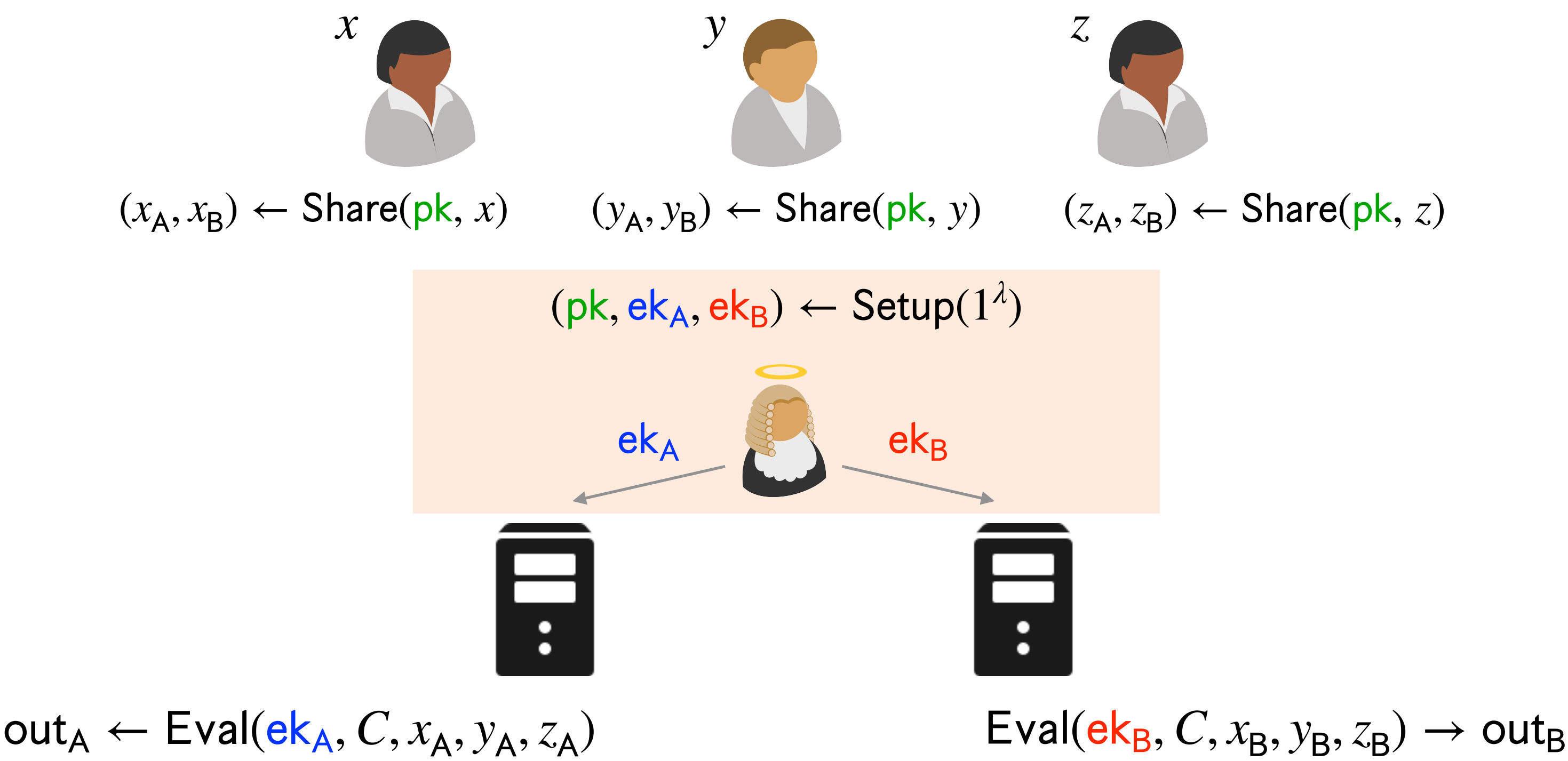
Client-Server HSS



Most existing client-server
HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Client-Server HSS

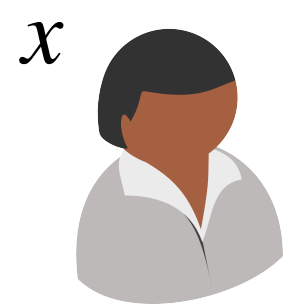


Most existing client-server HSS require **correlated setup**

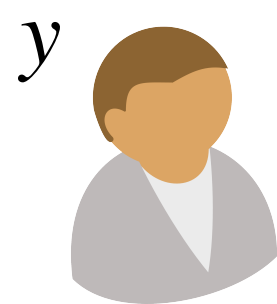
[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Correlated setup does not allow on-the-fly selection of servers!

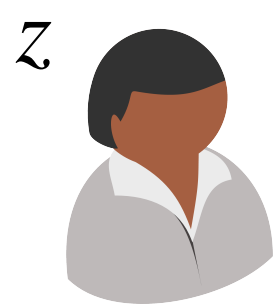
Client-Server HSS



$$(x_A, x_B) \leftarrow \text{Share}(\text{crs}, x)$$



$$(y_A, y_B) \leftarrow \text{Share}(\text{crs}, y)$$



$$(z_A, z_B) \leftarrow \text{Share}(\text{crs}, z)$$

Common Reference String



$$\text{out}_A \leftarrow \text{Eval}(\text{crs}, C, x_A, y_A, z_A)$$



$$\text{Eval}(\text{crs}, C, x_B, y_B, z_B) \rightarrow \text{out}_B$$

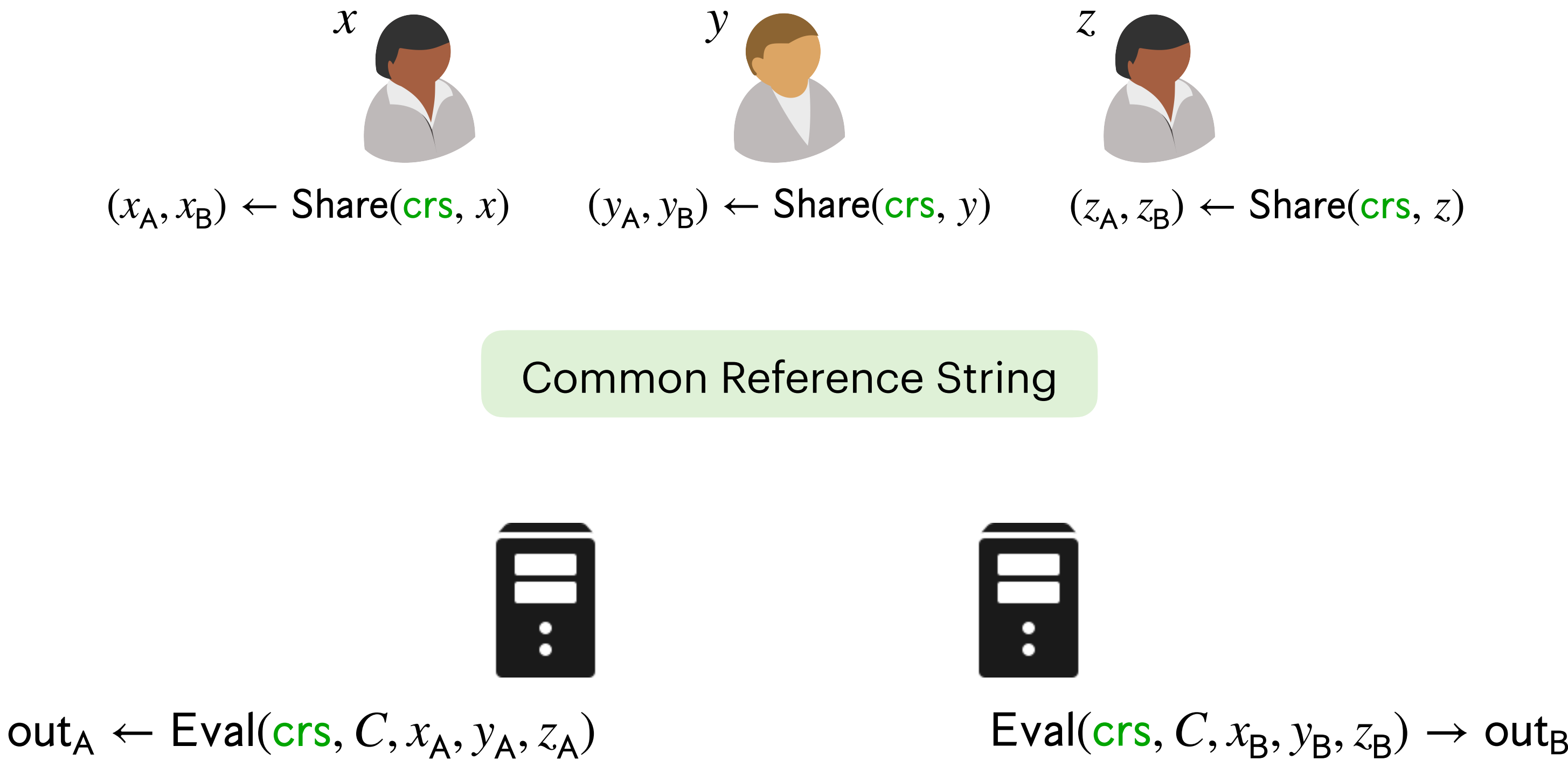
Most existing client-server HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Correlated setup does not allow on-the-fly selection of servers!

Client-Server HSS in the **CRS model**

Client-Server HSS



Most existing client-server HSS require **correlated setup**

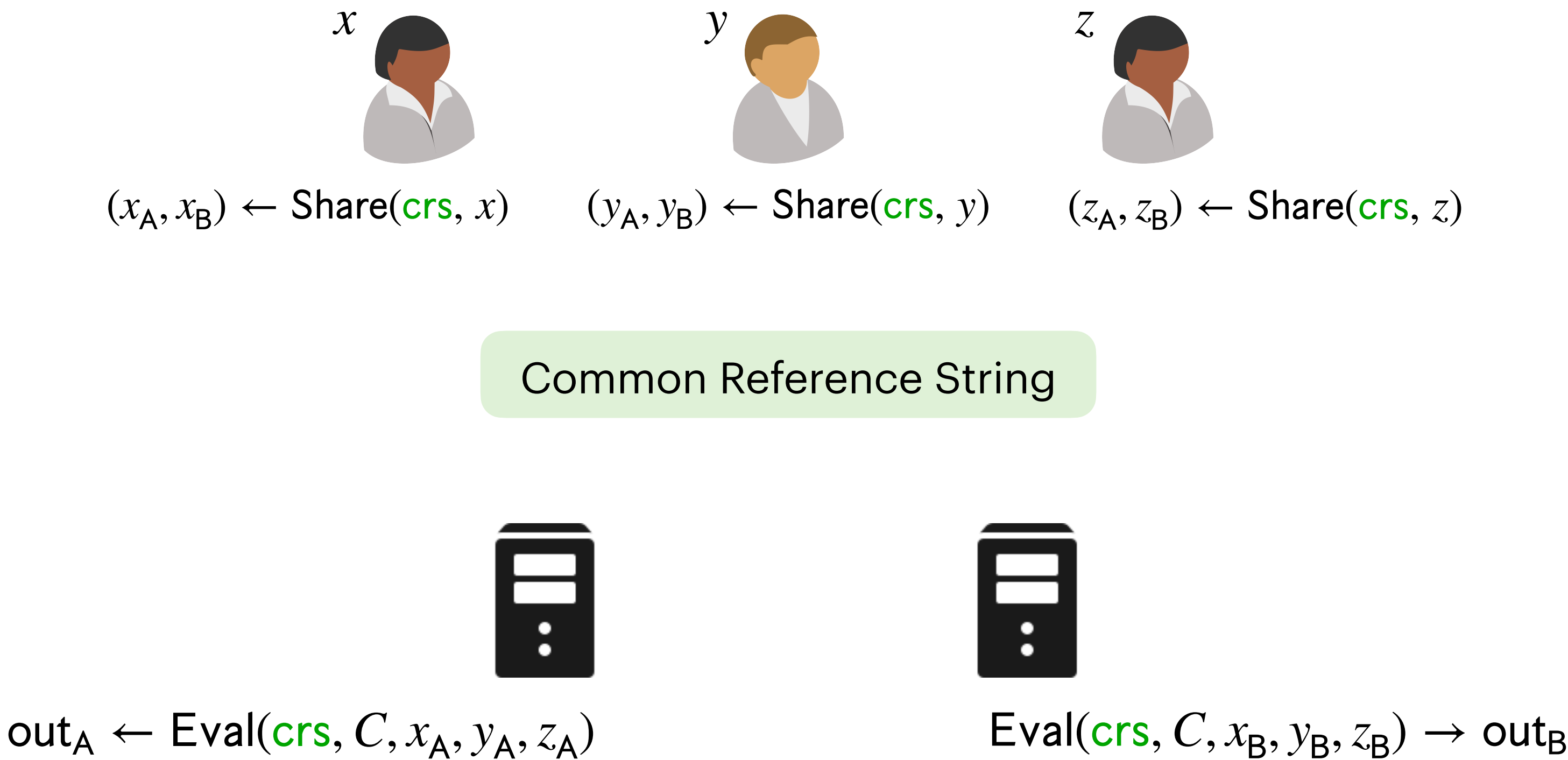
[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Correlated setup does not allow on-the-fly selection of servers!

Client-Server HSS in the **CRS model**

Recover on-the-fly server selection!

Client-Server HSS



Most existing client-server HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Correlated setup does not allow on-the-fly selection of servers!

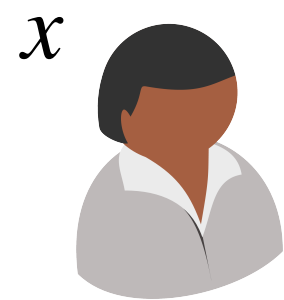
Client-Server HSS in the **CRS model**

Recover on-the-fly server selection!

Only known from assumptions that imply multi-key FHE

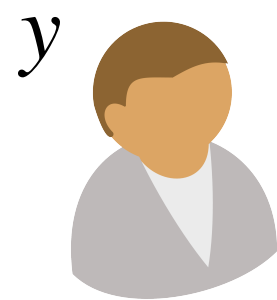
[Dodis-Halevi-Rothblum-Wichs'16]

Client-Server HSS



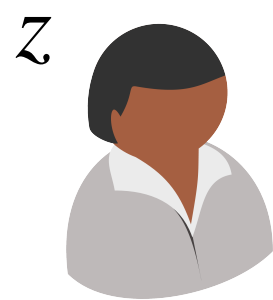
x

$$(x_A, x_B) \leftarrow \text{Share}(\text{crs}, x)$$



y

$$(y_A, y_B) \leftarrow \text{Share}(\text{crs}, y)$$



z

$$(z_A, z_B) \leftarrow \text{Share}(\text{crs}, z)$$

Common Reference String



$$\text{out}_A \leftarrow \text{Eval}(\text{crs}, C, x_A, y_A, z_A)$$



$$\text{Eval}(\text{crs}, C, x_B, y_B, z_B) \rightarrow \text{out}_B$$

Most existing client-server HSS require **correlated setup**

[Boyle-Gilboa-Ishai'16] [Boyle-Kohl-Scholl'19]
[Roy-Singh'21] [Orlandi-Scholl-Yakoubov'21]
[Abram-Damgård-Orlandi-Scholl'22]

Correlated setup does not allow on-the-fly selection of servers!

Client-Server HSS in the **CRS model**

Recover on-the-fly server selection!

Only known from assumptions that imply multi-key FHE

[Dodis-Halevi-Rothblum-Wichs'16]

Goal: Construct client-server HSS in the **CRS model** from **group-based** assumptions

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Our Results

Unbounded polynomial number of clients

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Our Results

Contains NC¹

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

DDH

DCR

Class groups

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

DDH

DCR

Class groups

Previously known only from **LWE** or **$i\mathcal{O}$ + DDH** [Dodis-Halevi-Rothblum-Wichs'16]

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Client-Server HSS from Prior Works
(Require Correlated Setup)

DDH

[Boyle-Gilboa-Ishai'16]

DCR

[Orlandi-Scholl-Yakoubov'21]
[Roy-Singh'21]

Class groups

[Abram-Damgård-Orlandi-Scholl'22]

Previously known only from **LWE** or **$i\mathcal{O}$ + DDH** [Dodis-Halevi-Rothblum-Wichs'16]

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Client-Server HSS from Prior Works
(Require Correlated Setup)

Inverse polynomial
correctness error

DDH

[Boyle-Gilboa-Ishai'16]

DCR

[Orlandi-Scholl-Yakoubov'21]
[Roy-Singh'21]

Class groups

[Abram-Damgård-Orlandi-Scholl'22]

Previously known only from **LWE** or **$i\mathcal{O}$ + DDH** [Dodis-Halevi-Rothblum-Wichs'16]

Our Results

Multi-client **two**-server HSS in the **CRS model** for evaluating **RMS Programs**

Client-Server HSS from Prior Works
(Require Correlated Setup)

Transparent setup

DDH

[Boyle-Gilboa-Ishai'16]

DCR

[Orlandi-Scholl-Yakoubov'21]
[Roy-Singh'21]

Transparent setup

Class groups

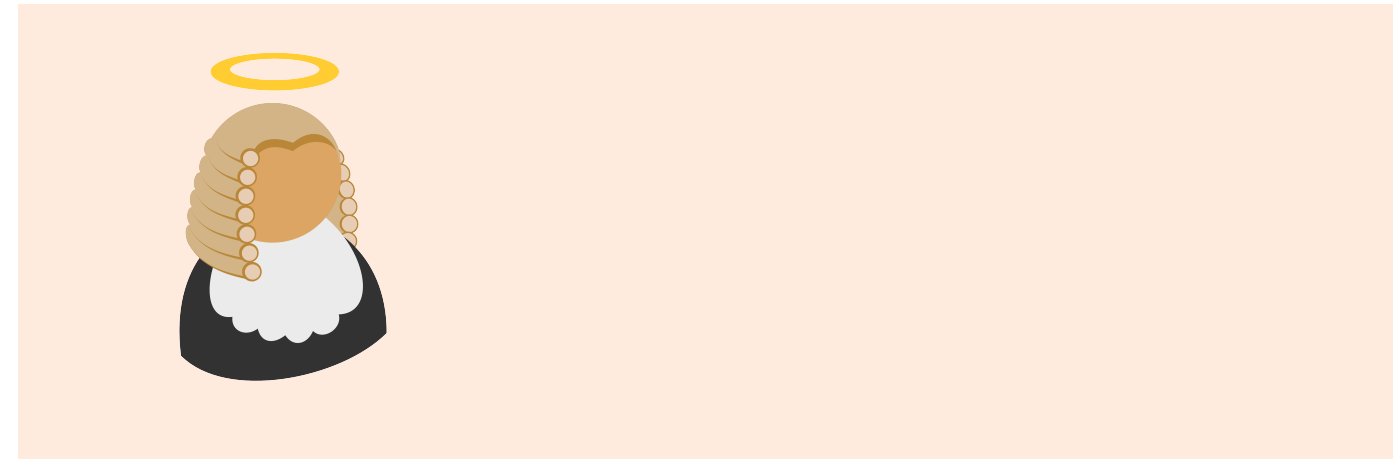
[Abram-Damgård-Orlandi-Scholl'22]

Previously known only from **LWE** or **$i\mathcal{O}$ + DDH** [Dodis-Halevi-Rothblum-Wichs'16]

Constructing Client-Server HSS in the CRS Model

Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$



Client-Server HSS Requiring Correlated Setup

[Boyle-Gilboa-Ishai'16]



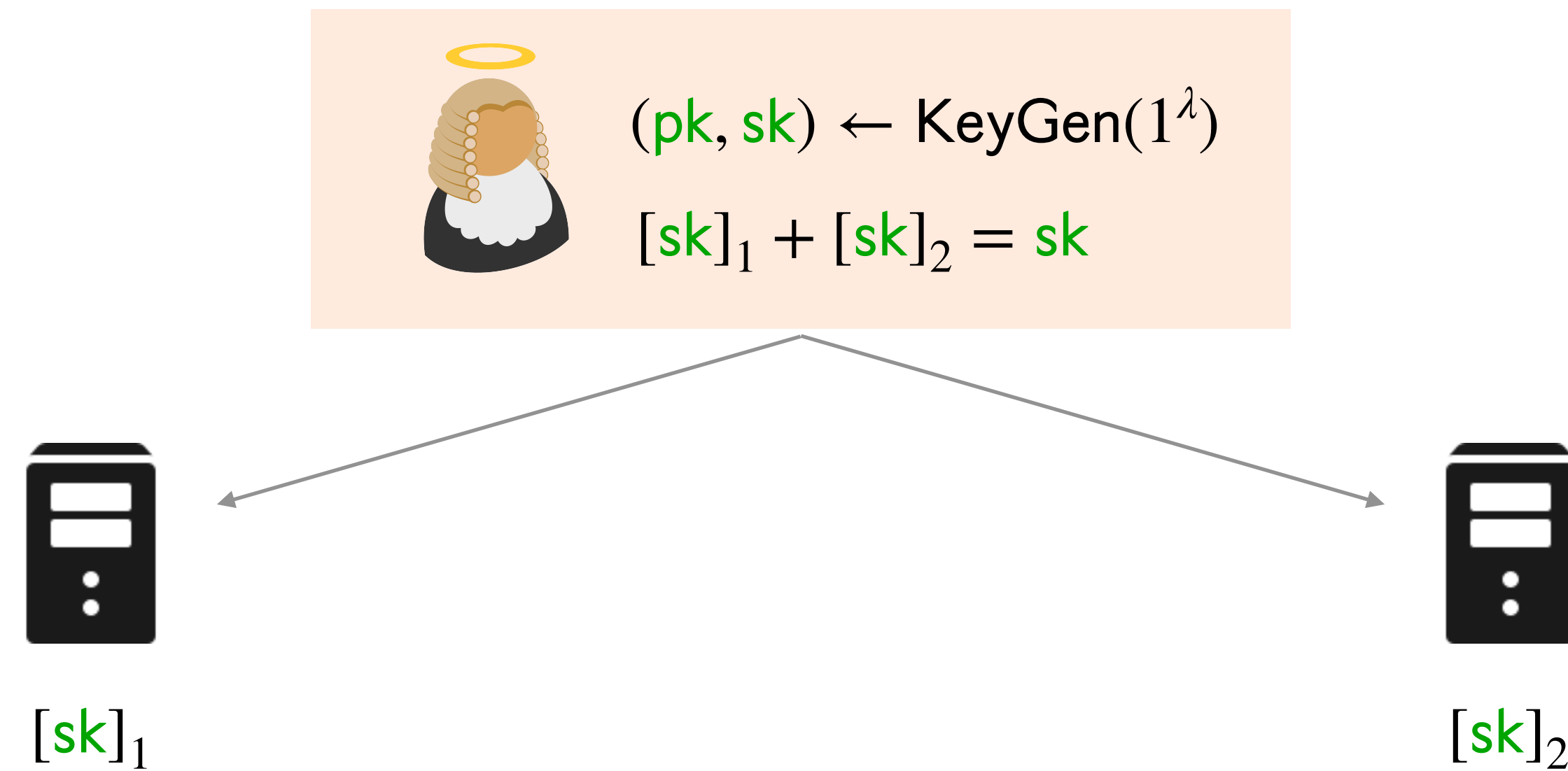
$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$

$[sk]_1 + [sk]_2 = sk$



Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



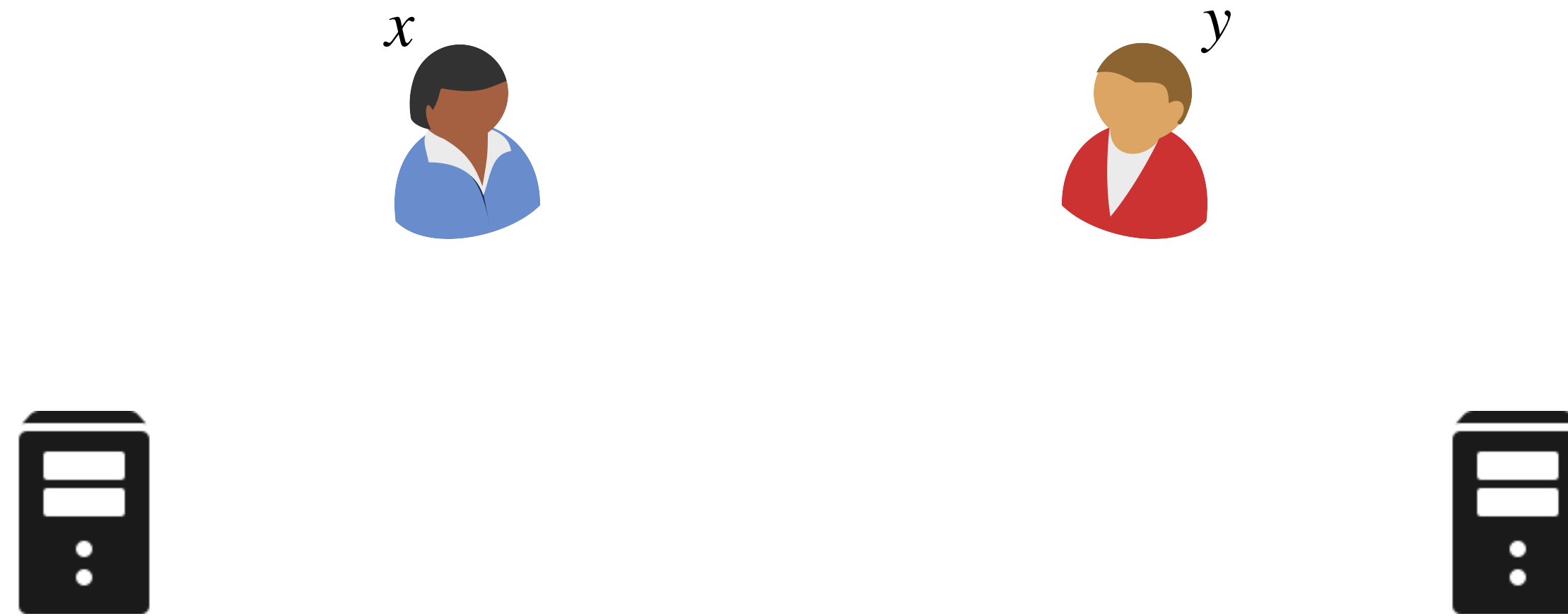
Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



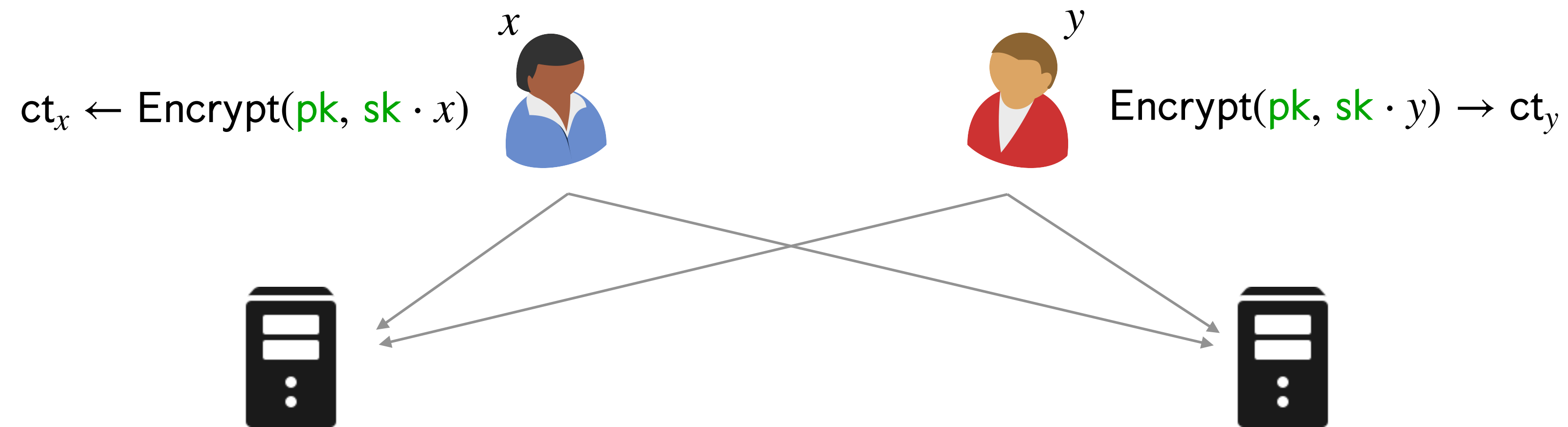
Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



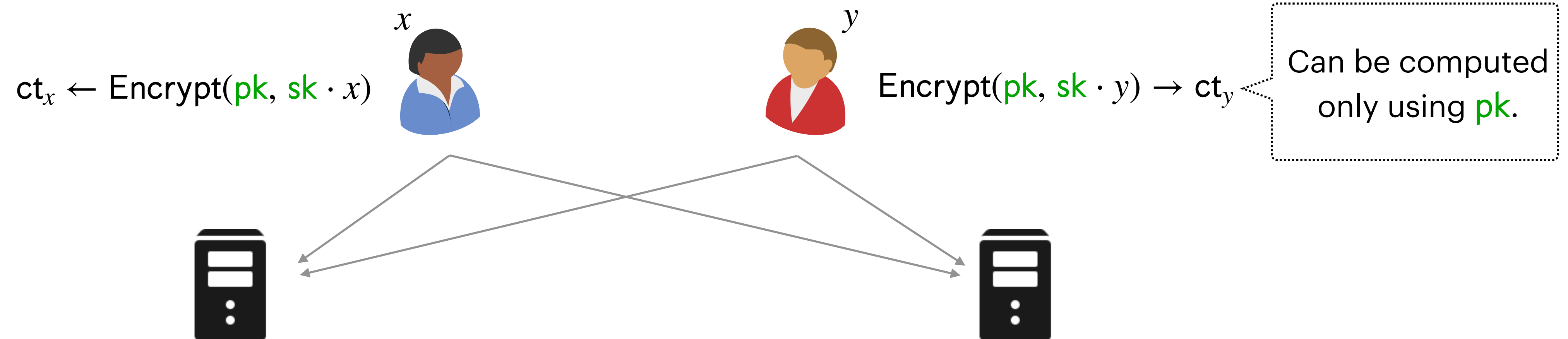
Client-Server HSS Requiring Correlated Setup

[Boyle-Gilboa-Ishai'16]



Client-Server HSS Requiring **Correlated Setup**

[Boyle-Gilboa-Ishai'16]



Crux of HSS Evaluation Using Correlated Setup

[Boyle-Gilboa-Ishai'16]

Input



$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$



$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

Crux of HSS Evaluation Using **Correlated Setup**

[Boyle-Gilboa-Ishai'16]

Input

$$ct_x \leftarrow \text{Encrypt}(\text{pk}, \text{sk} \cdot x)$$

$$[\text{sk} \cdot y]_1$$

Memory Share

$$ct_x \leftarrow \text{Encrypt}(\text{pk}, \text{sk} \cdot x)$$

$$[\text{sk} \cdot y]_2$$

y is an arbitrary intermediate value

Crux of HSS Evaluation Using **Correlated Setup**

[Boyle-Gilboa-Ishai'16]

Input
Memory Share



$$ct_x \leftarrow \text{Encrypt}(\text{pk}, \text{sk} \cdot x)$$

$$[\text{sk} \cdot y]_1$$



$$ct_x \leftarrow \text{Encrypt}(\text{pk}, \text{sk} \cdot x)$$

$$[\text{sk} \cdot y]_2$$

y is an arbitrary intermediate value

RMS Program: 1) Input share + Memory share $\rightarrow$ Memory share of product 2) Add memory shares

Crux of HSS Evaluation Using Correlated Setup

[Boyle-Gilboa-Ishai'16]

Input


$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$


$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

$$[sk \cdot y]_2$$

y is an arbitrary intermediate value

RMS Program:

1) Input share + Memory share $\rightarrow$ Memory share of product

2) Add memory shares

Crux of HSS Evaluation Using Correlated Setup

[Boyle-Gilboa-Ishai'16]

Input

$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$

Memory Share
of Product



$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

$$[sk \cdot y]_2$$



Crux of HSS Evaluation Using **Correlated Setup**

[Boyle-Gilboa-Ishai'16]

Encryption with **Linear** Decryption

$$ct \leftarrow \text{Encrypt}(\mathbf{pk}, \alpha)$$



Input

$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

Memory Share

$$[\mathbf{sk} \cdot y]_1$$

Memory Share
of Product



$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

$$[\mathbf{sk} \cdot y]_2$$

Crux of HSS Evaluation Using Correlated Setup

[Boyle-Gilboa-Ishai'16]

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(pk, \alpha)$$

$$\text{Decrypt}([sk], ct) = [\alpha]$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$

Memory Share
of Product



$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

$$[sk \cdot y]_2$$

Crux of HSS Evaluation Using **Correlated Setup**

[Boyle-Gilboa-Ishai'16]

Encryption with **Linear** Decryption

$$ct \leftarrow \text{Encrypt}(\mathbf{pk}, \alpha)$$

$$\text{Decrypt}([\mathbf{sk}], ct) = [\alpha]$$

$$\text{Decrypt}([\mathbf{sk} \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

Memory Share

$$[\mathbf{sk} \cdot y]_1$$

Memory Share
of Product



$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

$$[\mathbf{sk} \cdot y]_2$$

Crux of HSS Evaluation Using **Correlated Setup**

[Boyle-Gilboa-Ishai'16]

Encryption with **Linear** Decryption

$$ct \leftarrow \text{Encrypt}(\mathbf{pk}, \alpha)$$

$$\text{Decrypt}([\mathbf{sk}], ct) = [\alpha]$$

$$\text{Decrypt}([\mathbf{sk} \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

Memory Share

$$[\mathbf{sk} \cdot y]_1$$

Memory Share
of Product

$$[\mathbf{sk} \cdot xy]_1 = \text{Decrypt}([\mathbf{sk} \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(\mathbf{pk}, \mathbf{sk} \cdot x)$$

$$[\mathbf{sk} \cdot y]_2$$

$$[\mathbf{sk} \cdot xy]_2 = \text{Decrypt}([\mathbf{sk} \cdot y]_2, ct_x)$$

Crux of HSS Evaluation Using Correlated Setup

[Boyle-Gilboa-Ishai'16]

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(\text{pk}, \alpha)$$

$$\text{Decrypt}([sk], ct) = [\alpha]$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(\text{pk}, sk \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$

Memory Share
of Product

$$[sk \cdot xy]_1 = \text{Decrypt}([sk \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(\text{pk}, sk \cdot x)$$

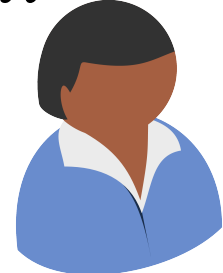
$$[sk \cdot y]_2$$

$$[sk \cdot xy]_2 = \text{Decrypt}([sk \cdot y]_2, ct_x)$$

$$[sk \cdot xy]_1 + [sk \cdot xy]_2 = sk \cdot xy$$

Towards Removing Correlated Setup

$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$





$\text{Encrypt}(pk, sk \cdot y) \rightarrow ct_y$



Input

$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$

Memory Share

$[sk \cdot y]_1$

Memory Share of Product

$[sk \cdot xy]_1 = \text{Decrypt}([sk \cdot y]_1, ct_x)$



Input

$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$

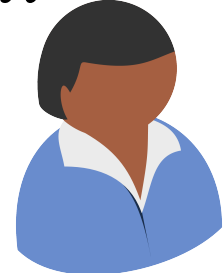
Memory Share

$[sk \cdot y]_2$

Memory Share of Product

$[sk \cdot xy]_2 = \text{Decrypt}([sk \cdot y]_2, ct_x)$

Towards Removing Correlated Setup

$(pk_A, sk_A) \leftarrow \text{KeyGen}(1^\lambda)$  ^x

 ^y $\text{KeyGen}(1^\lambda) \rightarrow (pk_B, sk_B)$



Input

$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$

Memory Share of Product

$$[sk \cdot xy]_1 = \text{Decrypt}([sk \cdot y]_1, ct_x)$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk, sk \cdot x)$$

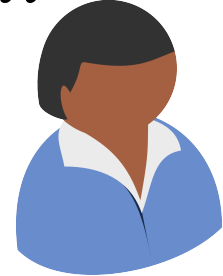
Memory Share

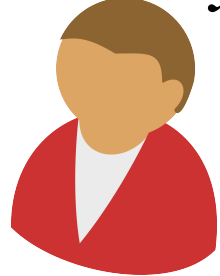
$$[sk \cdot y]_2$$

Memory Share of Product

$$[sk \cdot xy]_2 = \text{Decrypt}([sk \cdot y]_2, ct_x)$$

Towards Removing Correlated Setup

$$\begin{aligned} (\textcolor{blue}{pk}_A, \textcolor{blue}{sk}_A) &\leftarrow \text{KeyGen}(1^\lambda) \\ \textcolor{blue}{ct}_x &\leftarrow \text{Encrypt}(\textcolor{blue}{pk}_A, \textcolor{blue}{sk}_A \cdot x) \end{aligned}$$


$$\begin{aligned} \text{KeyGen}(1^\lambda) &\rightarrow (\textcolor{red}{pk}_B, \textcolor{red}{sk}_B) \\ \text{Encrypt}(\textcolor{red}{pk}_B, \textcolor{red}{sk}_B \cdot y) &\rightarrow \textcolor{red}{ct}_y \end{aligned}$$


Input


$$\textcolor{gray}{ct}_x \leftarrow \text{Encrypt}(\textcolor{green}{pk}, \textcolor{green}{sk} \cdot x)$$

Memory Share

$$\textcolor{green}{[sk \cdot y]}_1$$

Memory Share
of Product

$$\textcolor{green}{[sk \cdot xy]}_1 = \text{Decrypt}(\textcolor{green}{[sk \cdot y]}_1, \textcolor{gray}{ct}_x)$$

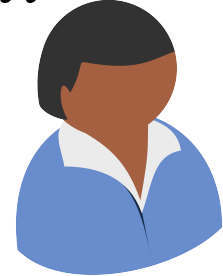

$$\textcolor{gray}{ct}_x \leftarrow \text{Encrypt}(\textcolor{green}{pk}, \textcolor{green}{sk} \cdot x)$$

$$\textcolor{green}{[sk \cdot y]}_2$$

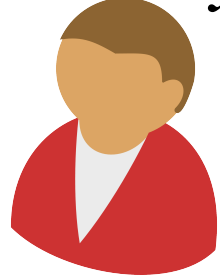
$$\textcolor{green}{[sk \cdot xy]}_2 = \text{Decrypt}(\textcolor{green}{[sk \cdot y]}_2, \textcolor{gray}{ct}_x)$$

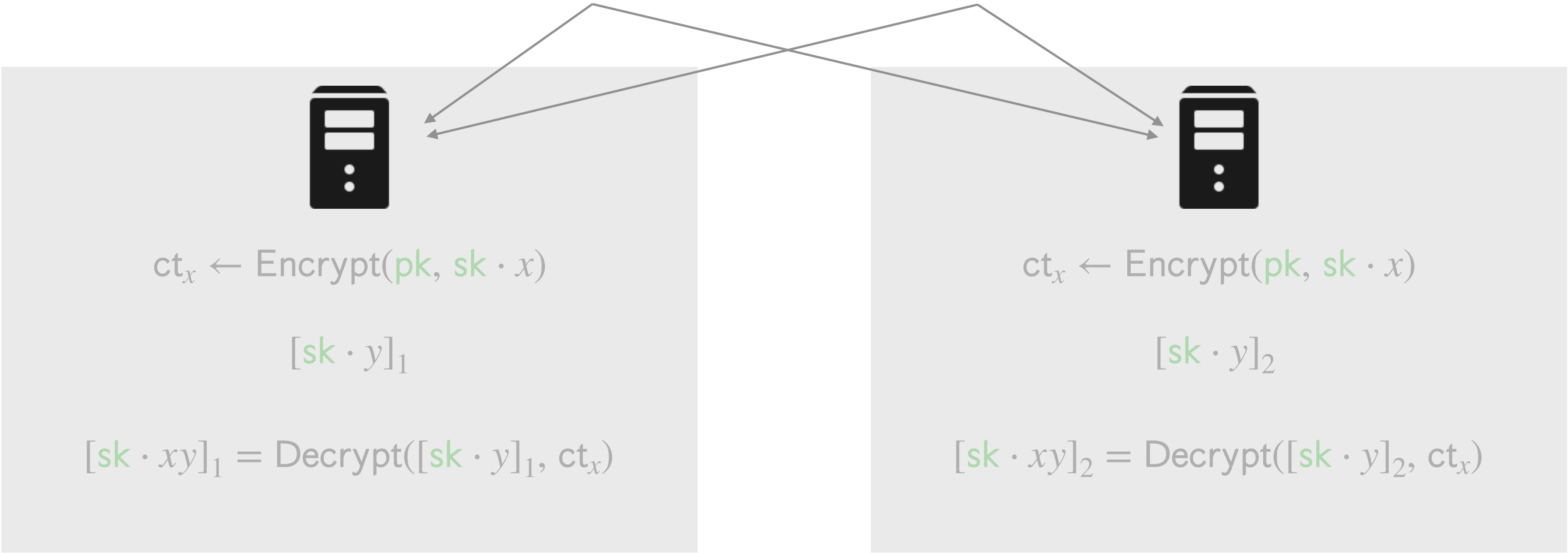
Towards Removing Correlated Setup

$(pk_A, sk_A) \leftarrow \text{KeyGen}(1^\lambda)$
 $ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$

x


$\text{KeyGen}(1^\lambda) \rightarrow (pk_B, sk_B)$
 $\text{Encrypt}(pk_B, sk_B \cdot y) \rightarrow ct_y$

y




Towards Removing Correlated Setup

Input


$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk \cdot y]_1$$

Memory Share
of Product

$$[sk \cdot xy]_1 = \text{Decrypt}([sk \cdot y]_1, ct_x)$$


$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk \cdot y]_2$$

$$[sk \cdot xy]_2 = \text{Decrypt}([sk \cdot y]_2, ct_x)$$

Towards Removing Correlated Setup



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk \cdot xy]_1 = \text{Decrypt}([sk \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk \cdot xy]_2 = \text{Decrypt}([sk \cdot y]_2, ct_x)$$

Towards Removing Correlated Setup

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(pk, \alpha)$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

Towards Removing Correlated Setup

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(pk, \alpha)$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$

$$[sk_B \cdot xy]_1$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

$$[sk_B \cdot xy]_2$$

Towards Removing Correlated Setup

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(pk, \alpha)$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

$$[sk_B \cdot xy]_1$$

$$[sk_B \cdot xy]_2$$

x is Alice's input, encrypted under sk_A .

Towards Removing Correlated Setup

Encryption with Linear Decryption

$$ct \leftarrow \text{Encrypt}(pk, \alpha)$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

$$[sk_B \cdot xy]_1$$

$$[sk_B \cdot xy]_2$$

x is Alice's input, encrypted under sk_A .

sk_B is Bob's secret key.

Towards Removing Correlated Setup

Encryption with **Linear** Decryption

$$ct \leftarrow \text{Encrypt}(\text{pk}, \alpha)$$

$$\text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$$



Input

$$ct_x \leftarrow \text{Encrypt}(\text{pk}_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(\text{pk}_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

$$[sk_B \cdot xy]_1$$

$$[sk_B \cdot xy]_2$$

Barrier to removing correlated setup: Servers need to **non-interactively multiply** inputs from **different** clients.

Non-Interactive Multiplication Gadget

Input

Memory Share

Memory Share
of Product



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

Non-Interactive Multiplication Gadget

Non-Interactive Multiplication Gadget



Input

$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

Memory Share

$$[sk_A \cdot y]_1 \quad [sk_B \cdot y]_1$$

Memory Share
of Product

$$[sk_A \cdot xy]_1 = \text{Decrypt}([sk_A \cdot y]_1, ct_x)$$



$$ct_x \leftarrow \text{Encrypt}(pk_A, sk_A \cdot x)$$

$$[sk_A \cdot y]_2 \quad [sk_B \cdot y]_2$$

$$[sk_A \cdot xy]_2 = \text{Decrypt}([sk_A \cdot y]_2, ct_x)$$

Non-Interactive Multiplication Gadget

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Non-Interactive Multiplication Gadget

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], \text{enc}) = [\text{sk}_B \cdot \alpha]$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Non-Interactive Multiplication Gadget

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Non-Interactive Multiplication Gadget

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{ct}_x \leftarrow \text{Encrypt}(\text{pk}_A, \text{sk}_A \cdot x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha \beta]$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

$$[\text{sk}_A \cdot xy]_1 = \text{Decrypt}([\text{sk}_A \cdot y]_1, \text{ct}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decrypt}([\text{sk}_A \cdot y]_2, \text{ct}_x)$$

Input

Memory Share

Memory Share
of Product

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



Input

$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1 = \text{Decode}([\text{sk}_A \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decode}([\text{sk}_A \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



Input

$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1 = \text{Decode}([\text{sk}_A \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_1 = \text{Decode}([\text{sk}_B \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decode}([\text{sk}_A \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_2 = \text{Decode}([\text{sk}_B \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



Input

$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1 \quad [\text{sk}_C \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1 = \text{Decode}([\text{sk}_A \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_1 = \text{Decode}([\text{sk}_B \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2 \quad [\text{sk}_C \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decode}([\text{sk}_A \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_2 = \text{Decode}([\text{sk}_B \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



Input

$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1 \quad [\text{sk}_C \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1 = \text{Decode}([\text{sk}_A \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_1 = \text{Decode}([\text{sk}_B \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$

$$[\text{sk}_C \cdot xy]_1 = \text{Decode}([\text{sk}_C \cdot y]_1, [\text{sk}_A \cdot y]_1, \text{enc}_x)$$



$$\text{enc}_x \leftarrow \text{Encode}(\text{pk}_A, x)$$

$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2 \quad [\text{sk}_C \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2 = \text{Decode}([\text{sk}_A \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

$$[\text{sk}_B \cdot xy]_2 = \text{Decode}([\text{sk}_B \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

$$[\text{sk}_C \cdot xy]_2 = \text{Decode}([\text{sk}_C \cdot y]_2, [\text{sk}_A \cdot y]_2, \text{enc}_x)$$

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$



Input

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1 \quad [\text{sk}_C \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1$$

$$[\text{sk}_B \cdot xy]_1$$

$$[\text{sk}_C \cdot xy]_1$$



$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2 \quad [\text{sk}_C \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2$$

$$[\text{sk}_B \cdot xy]_2$$

$$[\text{sk}_C \cdot xy]_2$$

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha \beta]$$



Input

Memory Share

$$[\text{sk}_A \cdot y]_1 \quad [\text{sk}_B \cdot y]_1 \quad [\text{sk}_C \cdot y]_1$$

Memory Share
of Product

$$[\text{sk}_A \cdot xy]_1$$

$$[\text{sk}_B \cdot xy]_1$$

$$[\text{sk}_C \cdot xy]_1$$



$$[\text{sk}_A \cdot y]_2 \quad [\text{sk}_B \cdot y]_2 \quad [\text{sk}_C \cdot y]_2$$

$$[\text{sk}_A \cdot xy]_2$$

$$[\text{sk}_B \cdot xy]_2$$

$$[\text{sk}_C \cdot xy]_2$$

RMS Program: 1) Input share + Memory share $\rightarrow$ Memory share of product 2) Add memory shares

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha \beta]$$

Sufficient to construct client-server HSS in the CRS model!

Client-Server HSS in the CRS Model

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B], [\text{sk}_A], \text{enc}) = [\text{sk}_B \cdot \alpha]$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha \beta]$$

Sufficient to construct client-server HSS in the CRS model!

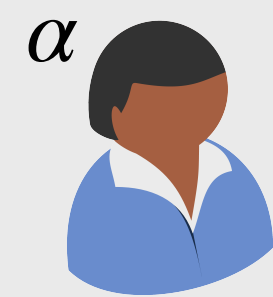
How can we construct this gadget?

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

Non-Interactive Multiplication

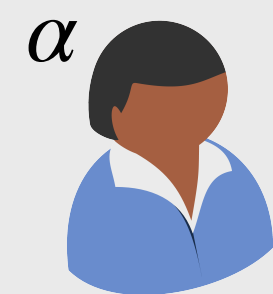
[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]



Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

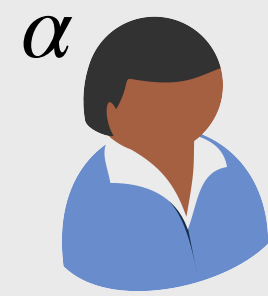
$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



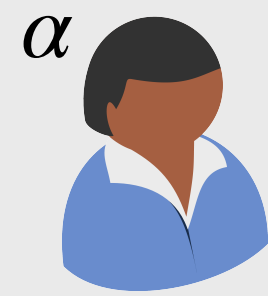
$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



Goal: Non-interactively
compute $[sk_B \cdot \alpha]$



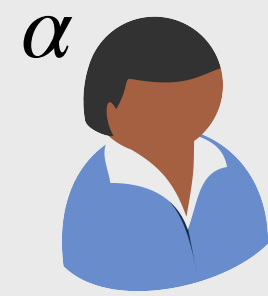
$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



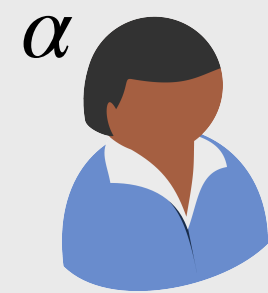
$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



pk_B

A horizontal arrow pointing from right to left, indicating the transmission of the public key pk_B from user B to user A.

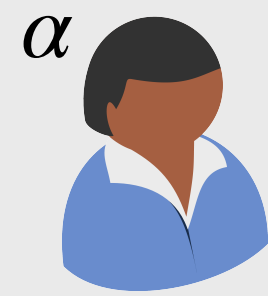
$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$pk_B$$



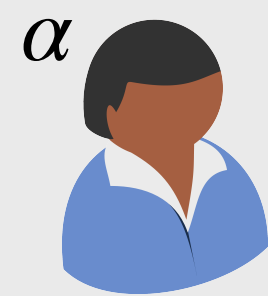
$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

enc

pk_B



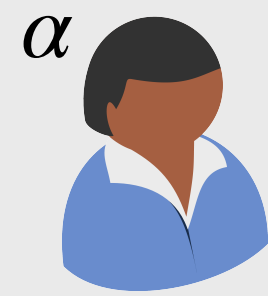
$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

enc

pk_B



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

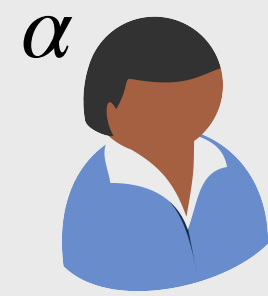
$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

enc

pk_B

$$\frac{(\text{enc})^{\text{sk}_B}}{(\text{pk}_B)^r} = \frac{g^{r \cdot \text{sk}_B} \cdot g^{\alpha \cdot \text{sk}_B}}{g^{\text{sk}_B \cdot r}} = g^{\text{sk}_B \cdot \alpha}$$



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

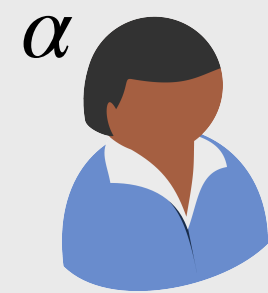
$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

 α

$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

 enc pk_B

$$\frac{(\text{enc})^{\text{sk}_B}}{(\text{pk}_B)^r} = \frac{g^{r \cdot \text{sk}_B} \cdot g^{\alpha \cdot \text{sk}_B}}{g^{\text{sk}_B \cdot r}} = g^{\text{sk}_B \cdot \alpha}$$



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

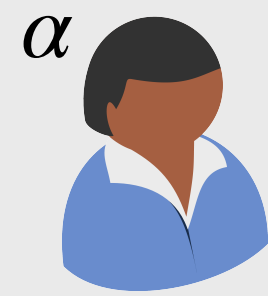
Distributed Discrete Log (DDLog): Non-interactively convert divisive shares into additive shares

[Boyle-Gilboa-Ishai'16]

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

DDLog $\downarrow$

$$[\text{sk}_B \cdot \alpha]_1$$

enc

pk_B

$$\frac{(\text{enc})^{\text{sk}_B}}{(\text{pk}_B)^r} = \frac{g^{r \cdot \text{sk}_B} \cdot g^{\alpha \cdot \text{sk}_B}}{g^{\text{sk}_B \cdot r}} = g^{\text{sk}_B \cdot \alpha}$$



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

DDLog $\downarrow$

$$[\text{sk}_B \cdot \alpha]_2$$

Distributed Discrete Log (DDLog): Non-interactively convert divisive shares into additive shares

[Boyle-Gilboa-Ishai'16]

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_1$$

enc →

← pk_B



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_2$$

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_1$$

enc →

← pk_B



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_2$$

Linear computation over public encodings

Non-Interactive Multiplication

[Döttling-Garg-Ishai-Malavolta-Mour-Ostrovsky'19] [Abram-Roy-Scholl'24]

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



$$r \leftarrow \mathbb{Z}_p$$

$$\text{enc} = g^r \cdot g^\alpha$$

$$(\text{pk}_B)^r$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_1$$

enc →

← pk_B



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$(\text{enc})^{\text{sk}_B}$$

DDLog ↓

$$[\text{sk}_B \cdot \alpha]_2$$

Linear computation over public encodings

Privately delegate computation to servers using
additive shares $[r]$ and $[\text{sk}_B]$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

α



Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

α

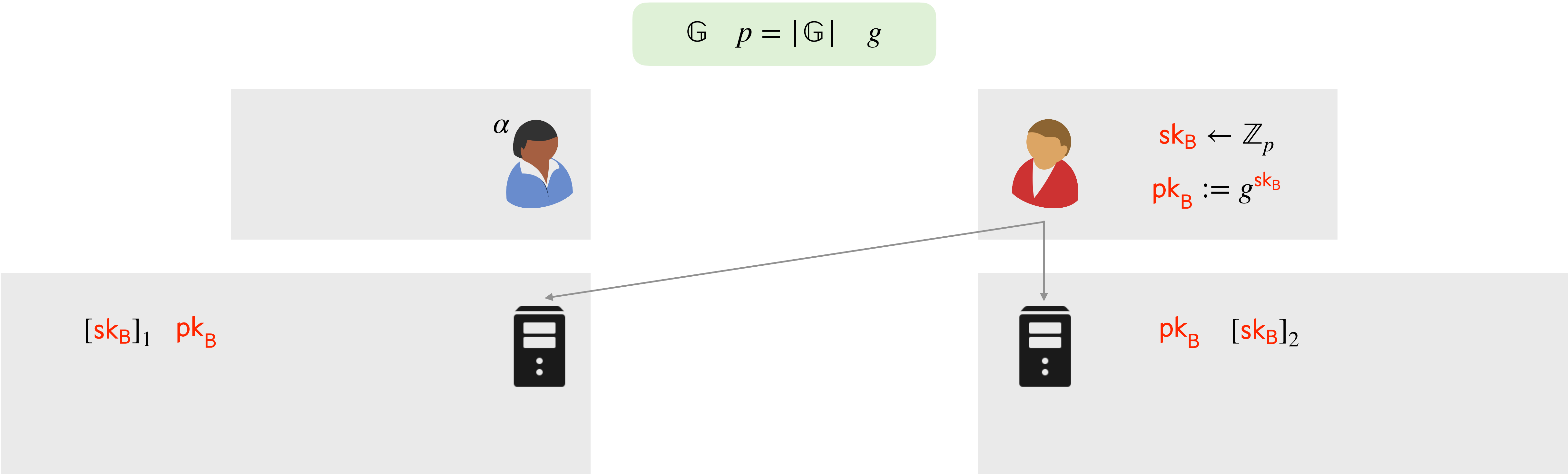


$$sk_B \leftarrow \mathbb{Z}_p$$

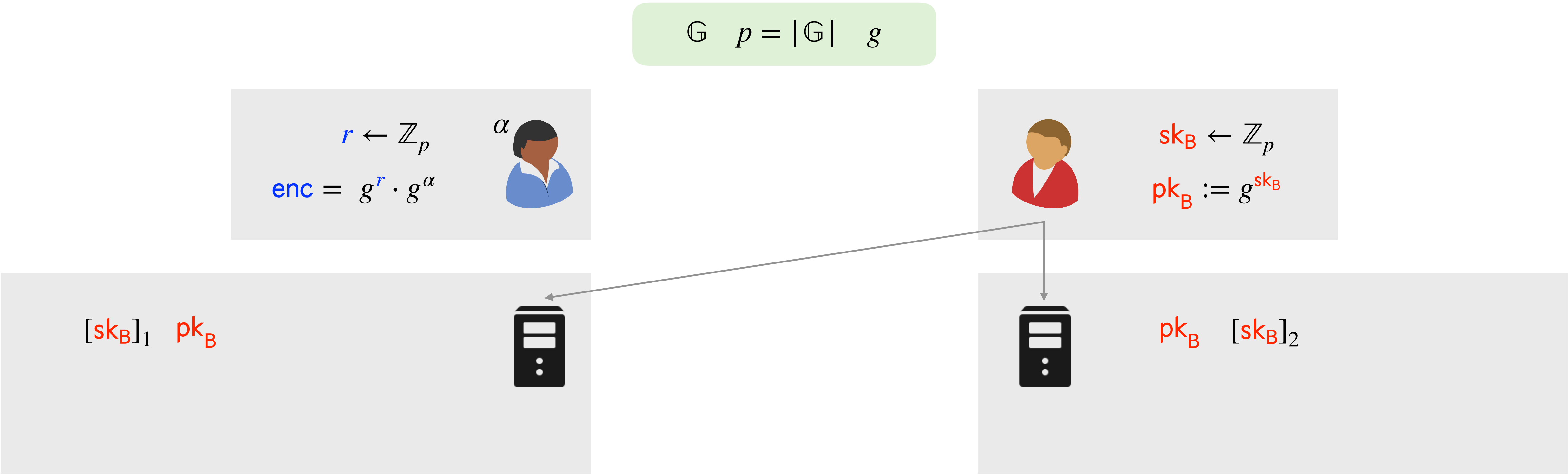
$$pk_B := g^{sk_B}$$



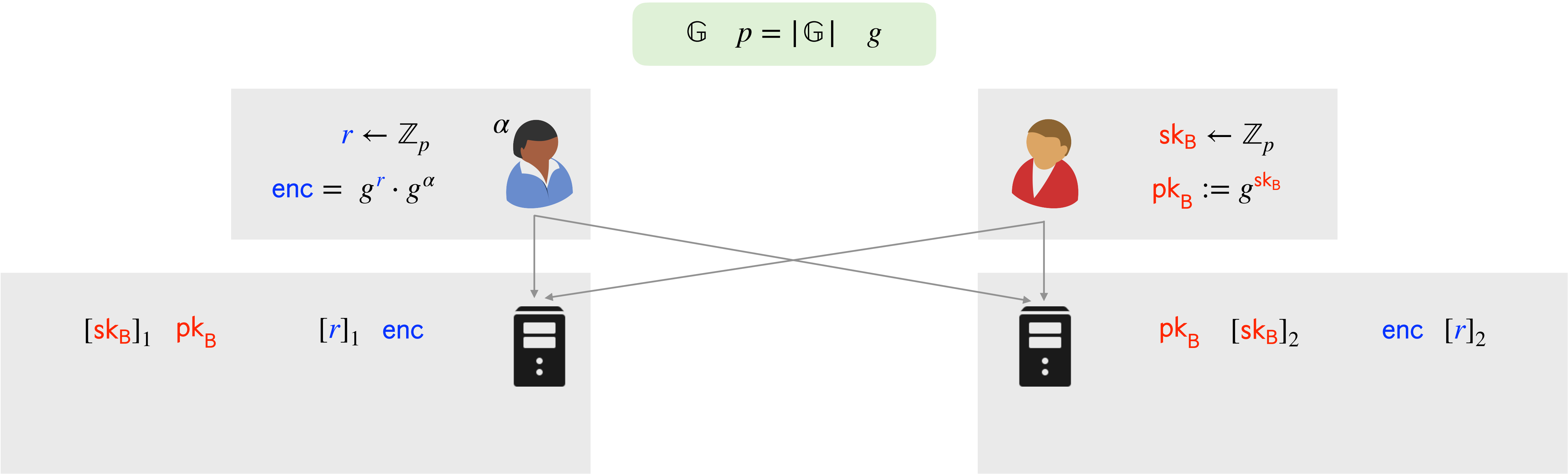
Constructing the Non-Interactive Multiplication Gadget



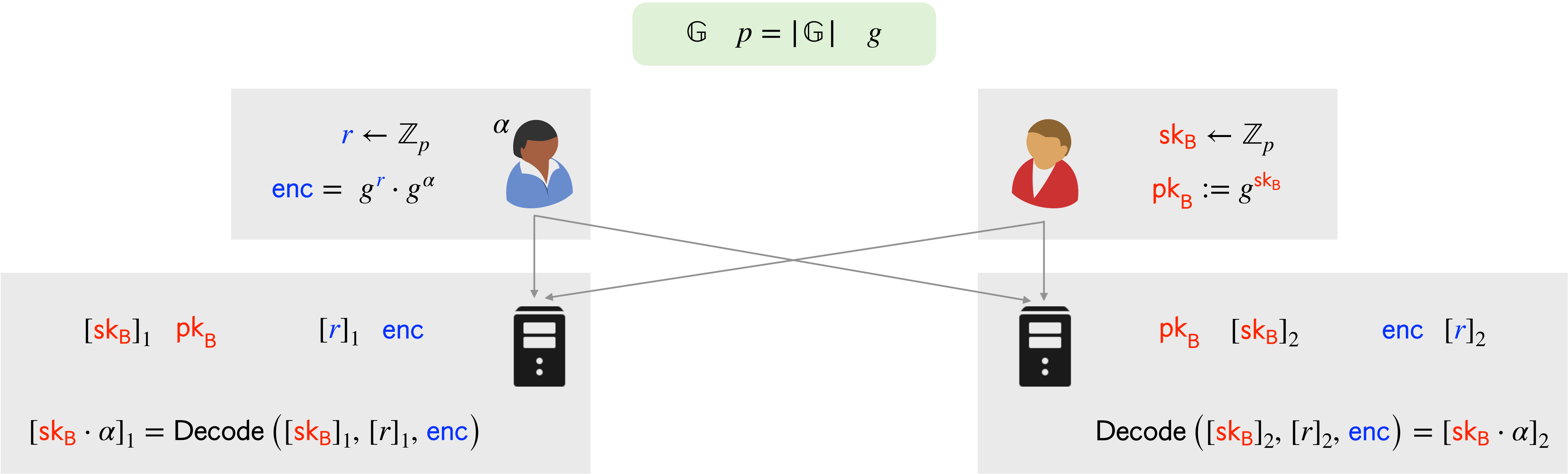
Constructing the Non-Interactive Multiplication Gadget



Constructing the Non-Interactive Multiplication Gadget



Constructing the Non-Interactive Multiplication Gadget



Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

$$r \leftarrow \mathbb{Z}_p \quad \alpha$$

$$\text{enc} = g^r \cdot g^\alpha$$




$$sk_B \leftarrow \mathbb{Z}_p$$

$$pk_B := g^{sk_B}$$

$$[sk_B \cdot \beta]_1 \quad pk_B \quad [r \cdot \beta]_1 \quad \text{enc}$$



$$[sk_B \cdot \alpha \beta]_1 = \text{Decode}([sk_B \cdot \beta]_1, [r \cdot \beta]_1, \text{enc})$$



$$pk_B \quad [sk_B \cdot \beta]_2 \quad \text{enc} \quad [r \cdot \beta]_2$$

$$\text{Decode}([sk_B \cdot \beta]_2, [r \cdot \beta]_2, \text{enc}) = [sk_B \cdot \alpha \beta]_2$$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

$$r \leftarrow \mathbb{Z}_p \quad \alpha$$

$$\text{enc} = g^r \cdot g^\alpha$$




$$\text{sk}_B \leftarrow \mathbb{Z}_p$$

$$\text{pk}_B := g^{\text{sk}_B}$$

$$[\text{sk}_B \cdot \beta]_1 \quad \text{pk}_B \quad [r \cdot \beta]_1 \quad \text{enc}$$



$$[\text{sk}_B \cdot \alpha\beta]_1 = \text{Decode}([\text{sk}_B \cdot \beta]_1, [r \cdot \beta]_1, \text{enc})$$



$$\text{pk}_B \quad [\text{sk}_B \cdot \beta]_2 \quad \text{enc} \quad [r \cdot \beta]_2$$

$$\text{Decode}([\text{sk}_B \cdot \beta]_2, [r \cdot \beta]_2, \text{enc}) = [\text{sk}_B \cdot \alpha\beta]_2$$

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

$$r \leftarrow \mathbb{Z}_p \quad \alpha$$

$$\text{enc} = g^r \cdot g^\alpha$$



$$\text{sk}_B \leftarrow \mathbb{Z}_p$$
$$\text{pk}_B := g^{\text{sk}_B}$$

$$[\text{sk}_B \cdot \beta]_1 \quad \text{pk}_B \quad [r \cdot \beta]_1 \quad \text{enc}$$



$$[\text{sk}_B \cdot \alpha\beta]_1 = \text{Decode}([\text{sk}_B \cdot \beta]_1, [r \cdot \beta]_1, \text{enc})$$



$$\text{pk}_B \quad [\text{sk}_B \cdot \beta]_2 \quad \text{enc} \quad [r \cdot \beta]_2$$

$$\text{Decode}([\text{sk}_B \cdot \beta]_2, [r \cdot \beta]_2, \text{enc}) = [\text{sk}_B \cdot \alpha\beta]_2$$

Non-Interactive Multiplication Gadget

$$\text{enc} \leftarrow \text{Encode}(\text{pk}_A, \alpha)$$

$$\text{Decode}([\text{sk}_B \cdot \beta], [\text{sk}_A \cdot \beta], \text{enc}) = [\text{sk}_B \cdot \alpha\beta]$$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

HSS evaluation requires decoding using $[sk_A \cdot \beta]$, but construction requires $[r \cdot \beta]$

$$[sk_B \cdot \beta]_1 \quad pk_B \quad [r \cdot \beta]_1 \quad enc$$



$$[sk_B \cdot \alpha\beta]_1 = \text{Decode}([sk_B \cdot \beta]_1, [r \cdot \beta]_1, enc)$$

$$pk_B \quad [sk_B \cdot \beta]_2 \quad enc \quad [r \cdot \beta]_2$$



$$\text{Decode}([sk_B \cdot \beta]_2, [r \cdot \beta]_2, enc) = [sk_B \cdot \alpha\beta]_2$$

Non-Interactive Multiplication Gadget

$$enc \leftarrow \text{Encode}(pk_A, \alpha)$$

$$\text{Decode}([sk_B \cdot \beta], [sk_A \cdot \beta], enc) = [sk_B \cdot \alpha\beta]$$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

HSS evaluation requires decoding using $[sk_A \cdot \beta]$, but construction requires $[r \cdot \beta]$

We can compute $[r \cdot \beta]$ from $[sk_A \cdot \beta]$ using
encryption with linear decryption

$$[sk_B \cdot \beta]_1 \quad pk_B \quad [r \cdot \beta]_1 \quad enc$$

$$[sk_B \cdot \alpha\beta]_1 = \text{Decode}([sk_B \cdot \beta]_1, [r \cdot \beta]_1, enc)$$

$$pk_B \quad [sk_B \cdot \beta]_2 \quad enc \quad [r \cdot \beta]_2$$

$$\text{Decode}([sk_B \cdot \beta]_2, [r \cdot \beta]_2, enc) = [sk_B \cdot \alpha\beta]_2$$

Non-Interactive Multiplication Gadget

$$enc \leftarrow \text{Encode}(pk_A, \alpha)$$

$$\text{Decode}([sk_B \cdot \beta], [sk_A \cdot \beta], enc) = [sk_B \cdot \alpha\beta]$$

Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$



Has $[sk_A \cdot \beta]_1$

Needs $[r \cdot \beta]_1$



Has $[sk_A \cdot \beta]_2$

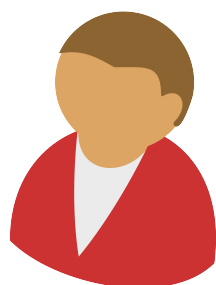
Needs $[r \cdot \beta]_2$



Constructing the Non-Interactive Multiplication Gadget

$$\mathbb{G} \quad p = |\mathbb{G}| \quad g$$

$ct \leftarrow \text{Encrypt}(pk_A, r)$



Has $[sk_A \cdot \beta]_1$

Needs $[r \cdot \beta]_1$

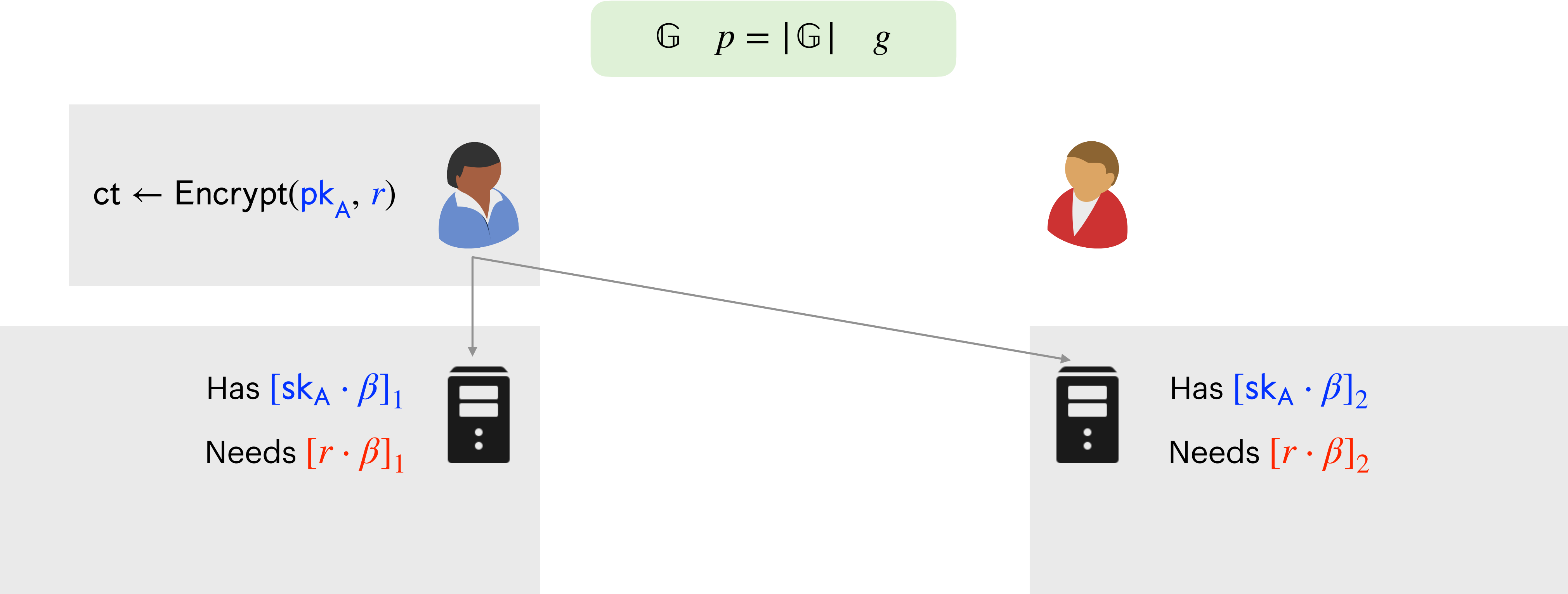


Has $[sk_A \cdot \beta]_2$

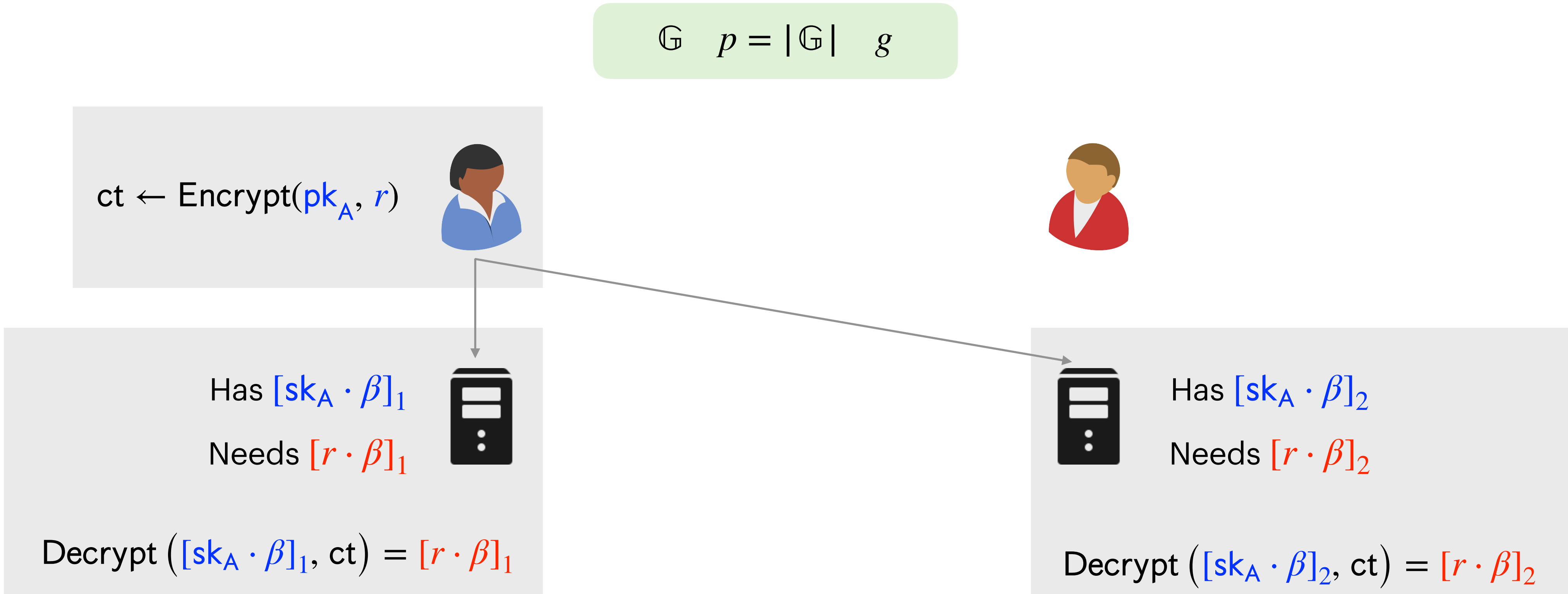
Needs $[r \cdot \beta]_2$



Constructing the Non-Interactive Multiplication Gadget



Constructing the Non-Interactive Multiplication Gadget



Encryption with **Linear** Decryption

$ct \leftarrow \text{Encrypt}(pk, \alpha) \qquad \text{Decrypt}([sk \cdot \beta], ct) = [\alpha \cdot \beta]$

Conclusion

Conclusion

- **Client-server HSS in the CRS Model:** Client-server secure computation with [ideal-world-like efficiency](#).
 - Two rounds
 - Client runtime is function independent
 - Communication cost is function independent
 - Clients, server, and function can be chosen on-the-fly

Conclusion

- **Client-server HSS in the CRS Model:** Client-server secure computation with [ideal-world-like efficiency](#).
 - Two rounds
 - Client runtime is function independent
 - Communication cost is function independent
 - Clients, server, and function can be chosen on-the-fly
- **This work:** Client-server HSS in the [CRS model](#) from [group-based](#) assumptions.

Conclusion

- **Client-server HSS in the CRS Model:** Client-server secure computation with **ideal-world-like efficiency**.
 - Two rounds
 - Client runtime is function independent
 - Communication cost is function independent
 - Clients, server, and function can be chosen on-the-fly
- **This work:** Client-server HSS in the **CRS model** from **group-based** assumptions.
- **Extension:** **Succinct** client-server HSS
 - Clients can have **long input** but **communication to one server** is **sublinear** in the input length.

Conclusion

- **Client-server HSS in the CRS Model:** Client-server secure computation with **ideal-world-like efficiency**.
 - Two rounds
 - Client runtime is function independent
 - Communication cost is function independent
 - Clients, server, and function can be chosen on-the-fly
- **This work:** Client-server HSS in the **CRS model** from **group-based** assumptions.
- **Extension:** **Succinct** client-server HSS
 - Clients can have **long input** but **communication to one server** is **sublinear** in the input length.

Thank You