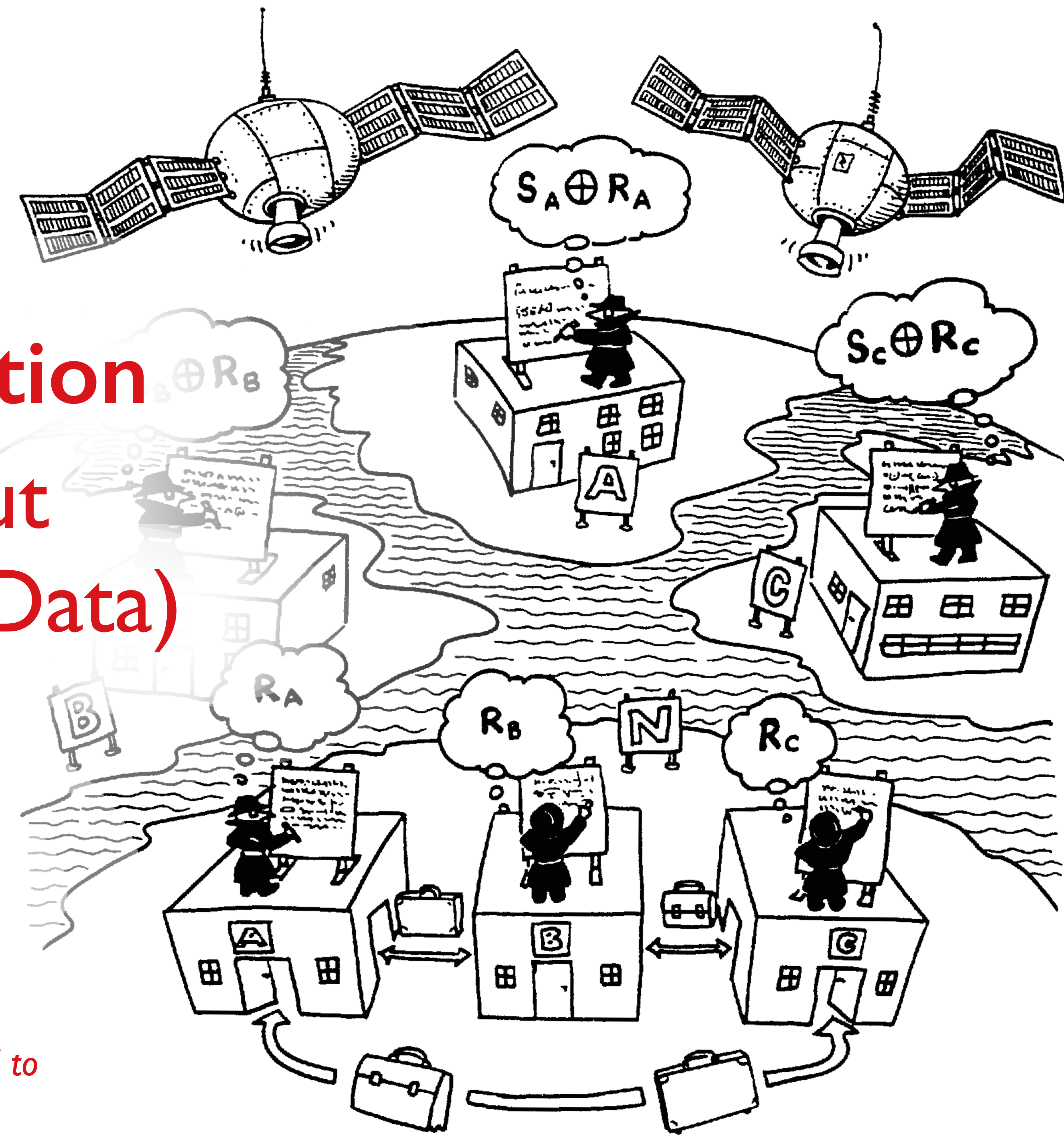
Centrum Wiskunde & Informatica

Universiteit Leiden

CWI Lectures
November 21 & 22, 2019

Multiparty Computation

Collaborate Without Compromise(ing Your Data)



Serge Fehr

Centrum Wiskunde & Informatica (CWI)

Mathematical Institute, Leiden University

On the occasion of the Dijkstra Fellowship being awarded to David Chaum

Centrum Wiskunde & Informatica

Universiteit Leiden

CWI Lectures
November 21 & 22, 2019

Multiparty Computation

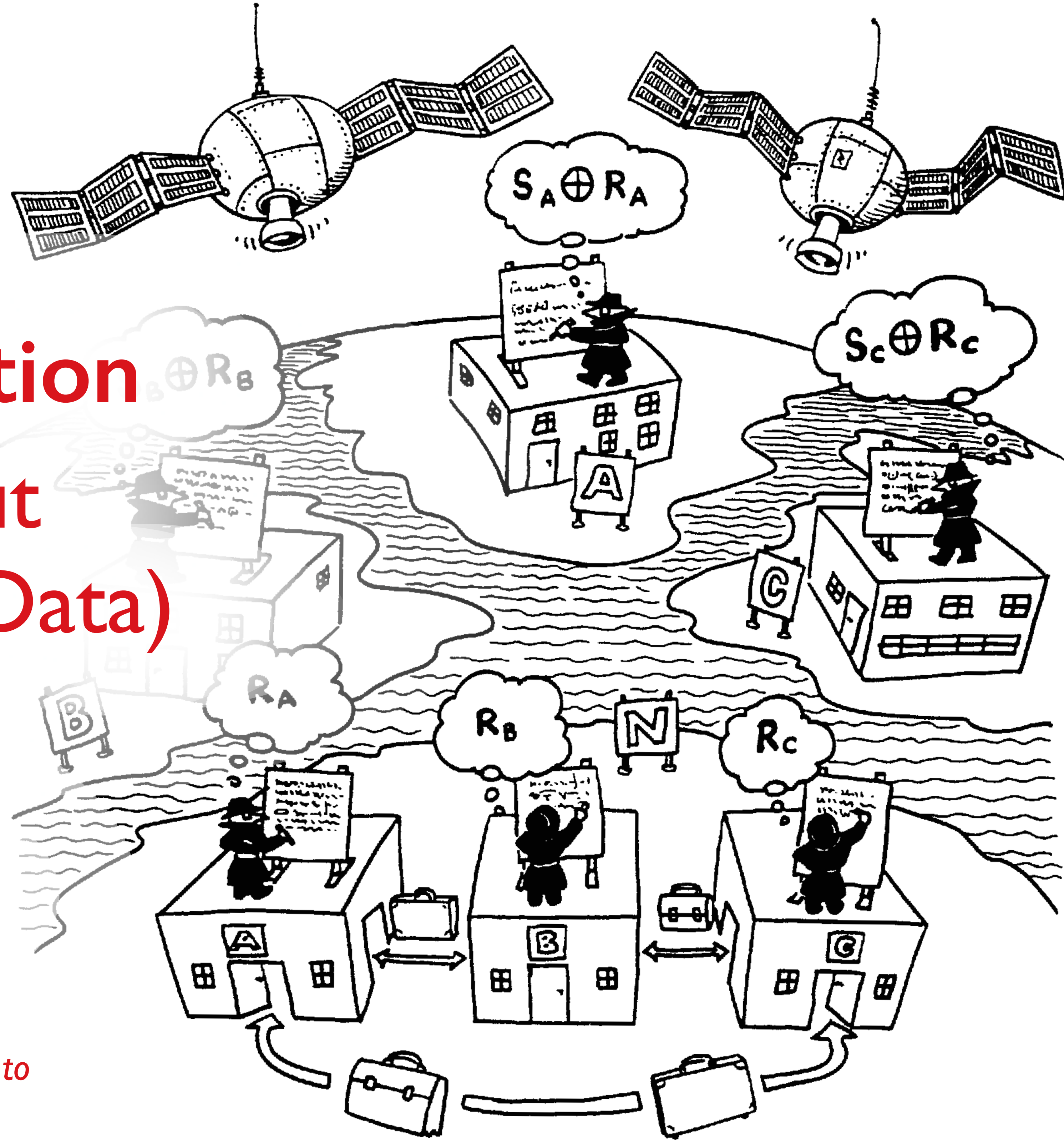
Collaborate Without Compromise(ing Your Data)

Serge Fehr

Centrum Wiskunde & Informatica (CWI)

Mathematical Institute, Leiden University

On the occasion of the Dijkstra Fellowship being awarded to David Chaum



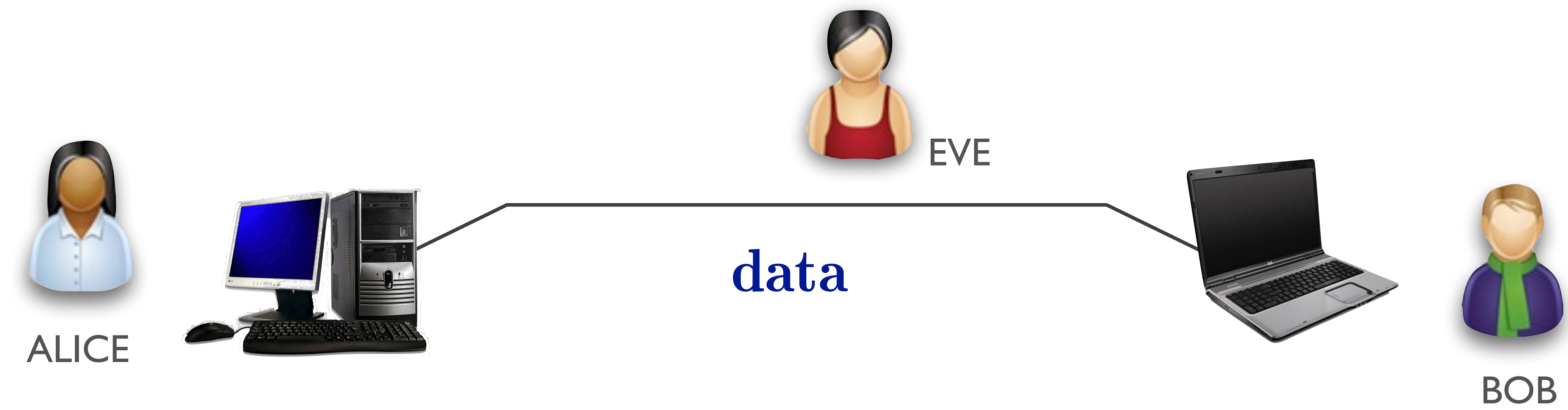
Road Map

- WHAT is multiparty computation?
- HOW does multiparty computation work?
- WHERE can/is multiparty computation be/ used?

Road Map

- WHAT is multiparty computation?
- HOW does multiparty computation work?
- WHERE can/is multiparty computation be/ used?

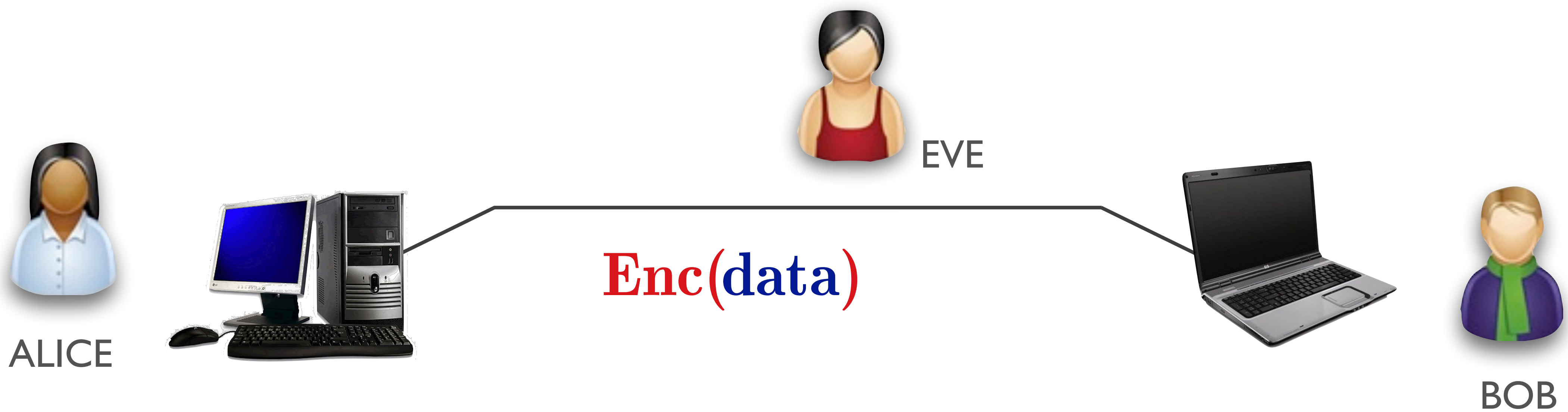
Cryptography



Original goal of cryptography:

Protect data from an eavesdropper/hacker/etc.

Cryptography

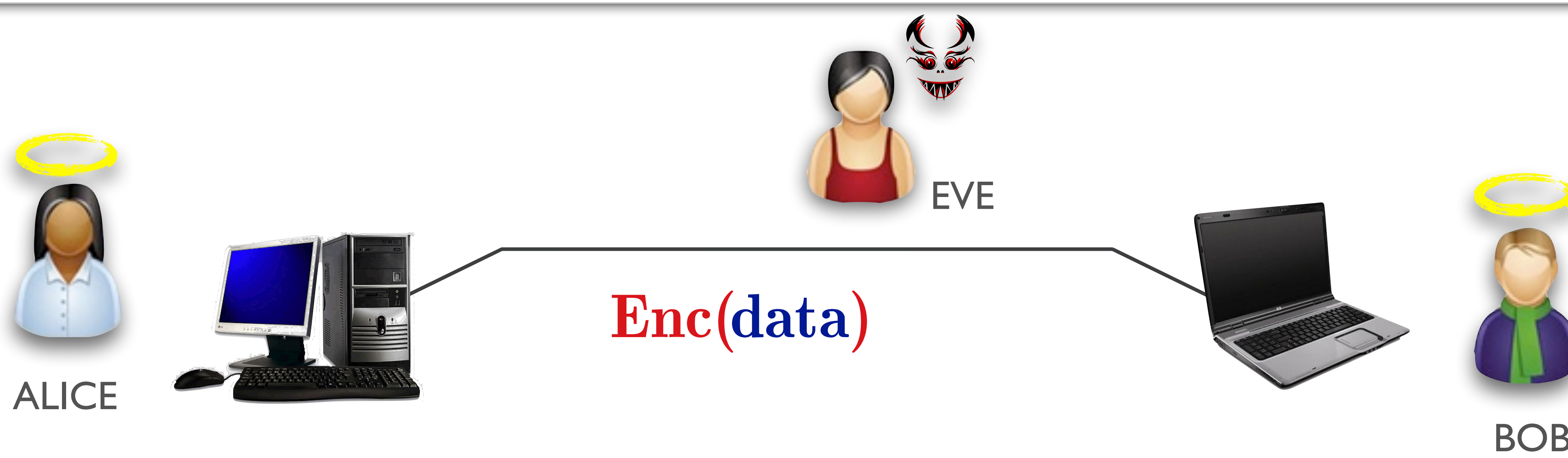


Original goal of cryptography:

Protect data from an eavesdropper/hacker/etc.

Means: Encryption, and authentication/signatures

Cryptography



Original goal of cryptography:

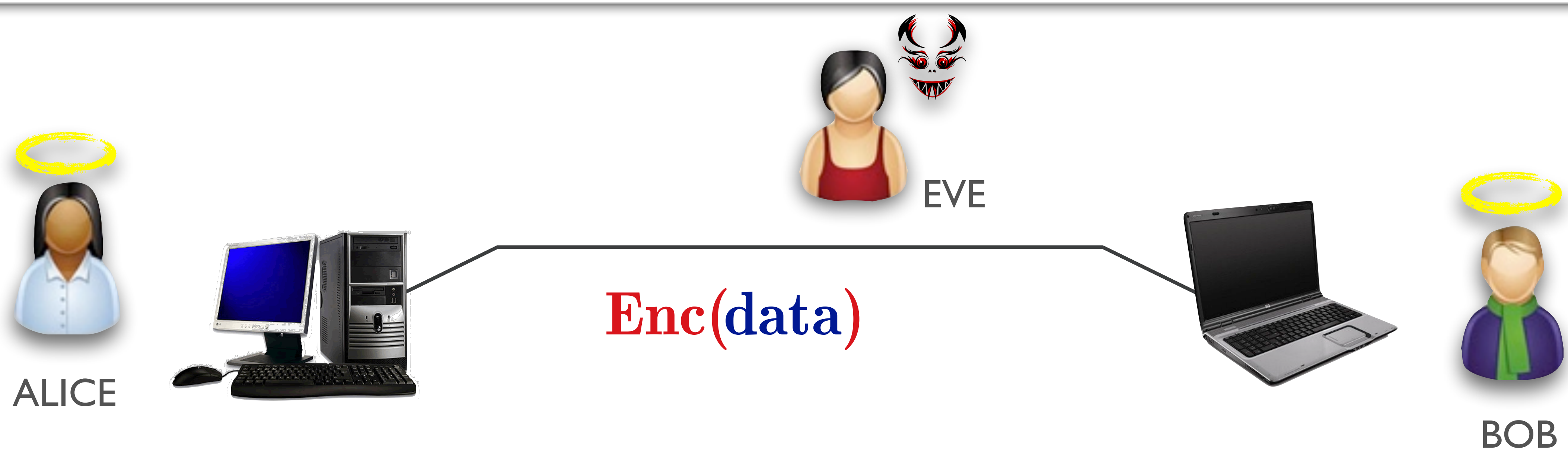
Protect data from an eavesdropper/hacker/etc.

Means: Encryption, and authentication/signatures

Here: Clear distinction between

"good participants" and *"malicious attacker"*

Cryptography



Original goal of cryptography:

Protect data from an eavesdropper/hacker/etc.

Means: Encryption, and authentication/signatures

Here: Clear distinction between

"good participants" and *"malicious attacker"*

Situation may not always be so clear cut…

Cryptography

Sometimes, participants may not trust **each other**:

Cryptography

Sometimes, participants may not trust **each other**:

- from Alice's perspective: Bob may be honest or malicious

Cryptography

Sometimes, participants may not trust **each other**:

- from Alice's perspective: Bob may be honest or malicious
- from Bob's perspective: Alice may be honest or malicious

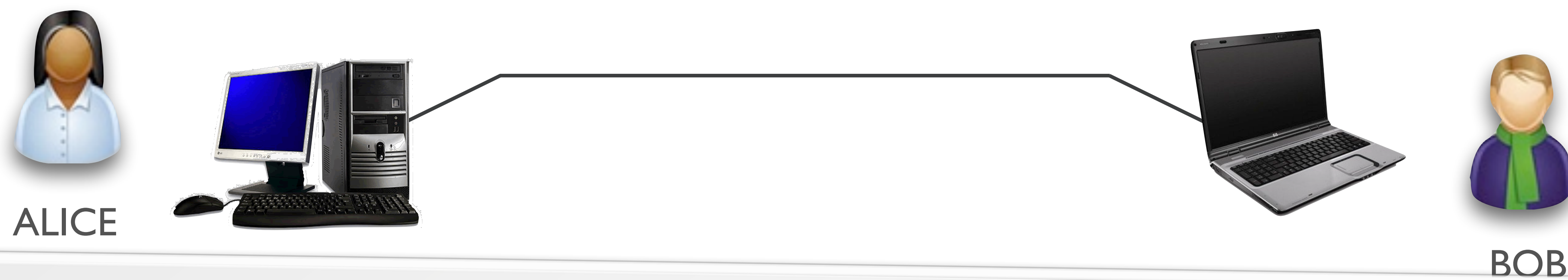
Cryptography

Sometimes, participants may not trust **each other**:

- from Alice's perspective: Bob may be honest or malicious
- from Bob's perspective: Alice may be honest or malicious

Goal: Collaborate **without** the need to trust each other, and so that **nothing gets revealed** beyond what is necessary.

Cryptography



Multiparty Computation (MPC)

An advanced cryptographic concept

- for **protecting individual data** of different parties
- while **using the data** in collaboration with other parties

Goal: Collaborate **without** the need to **trust each other**, and so that **nothing gets revealed** beyond what is necessary.

Example: Yao's "Millionairs' Problem"

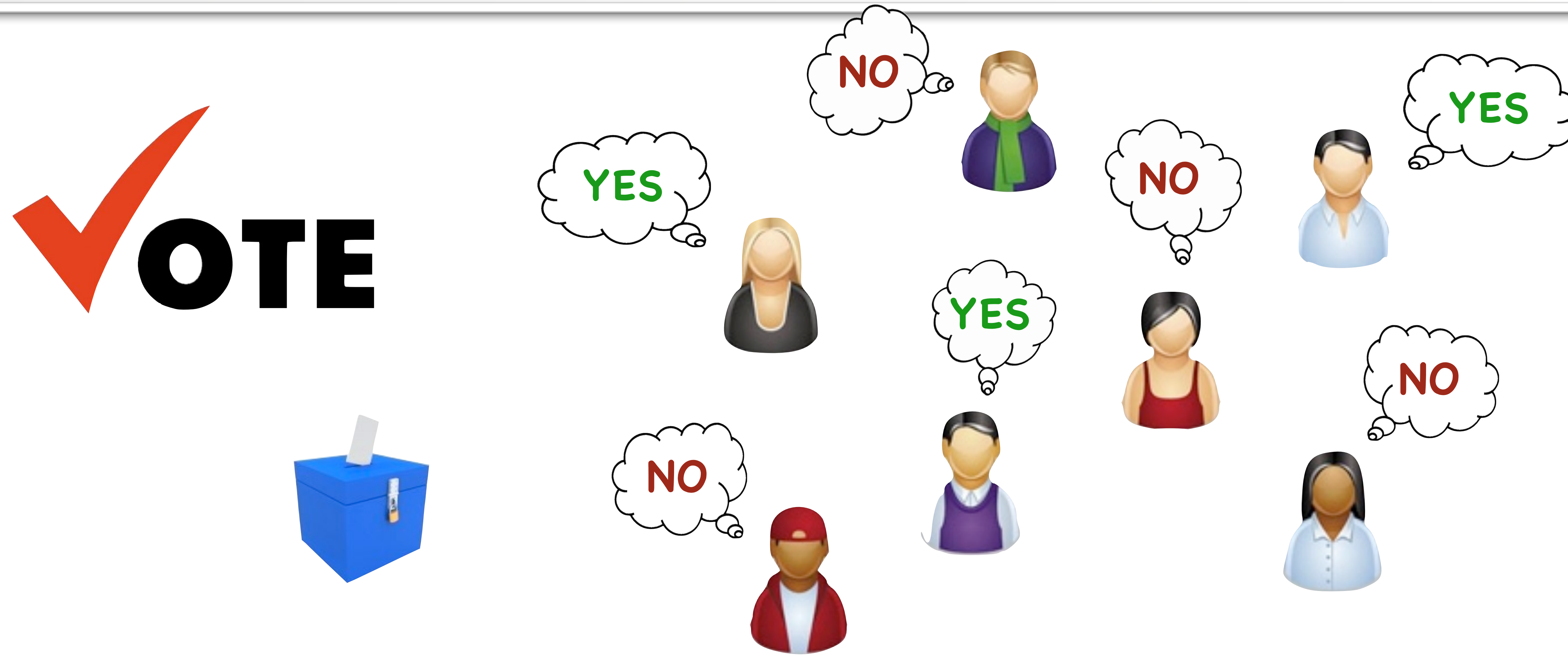
Two millionaires want to find out who is richer,

Example: Yao's "Millionairs' Problem"

Two millionaires want to find out who is richer,
but **without** telling each other how much they own:
both should learn **nothing beyond**

$$y \in \{\text{“Richard is richer”}, \text{“Elon is richer (or equally rich)”}\}$$

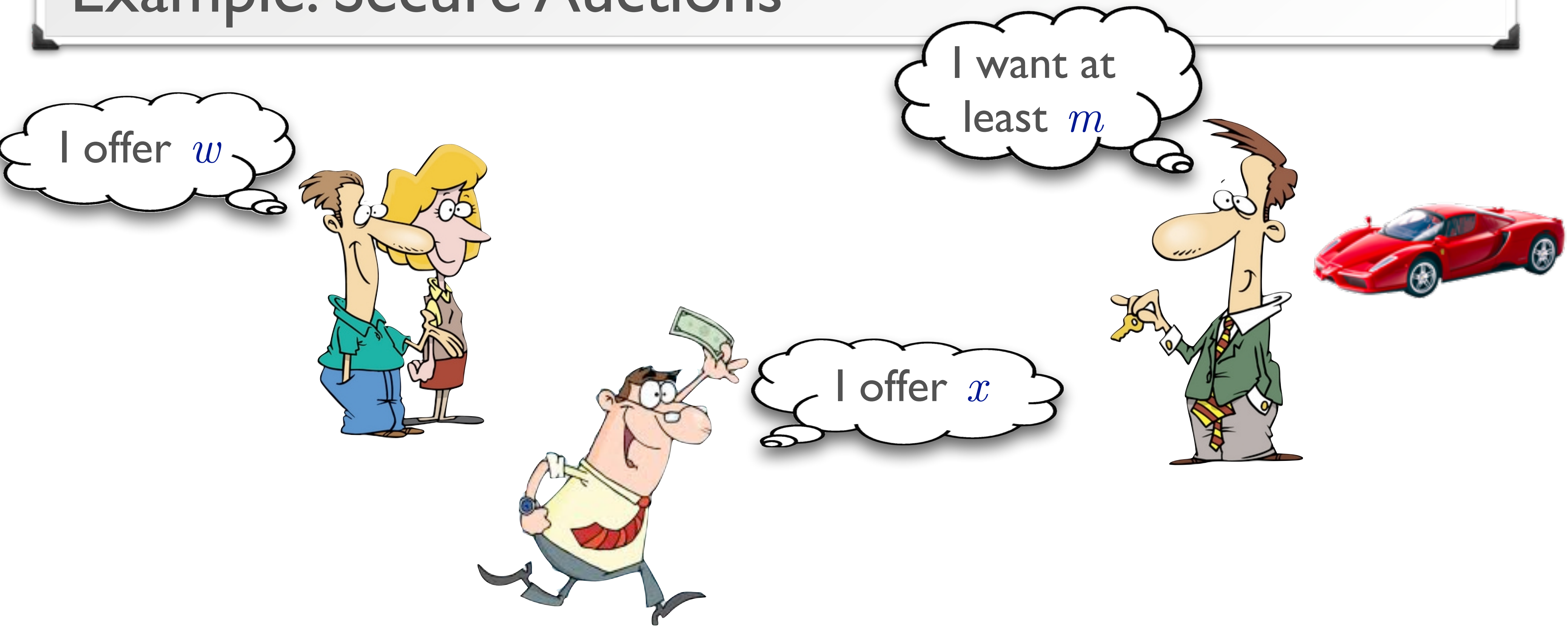
Example: Secure Voting



Find out what the majority wants, i.e., tally the votes,
without revealing individual opinions/votes:
everyone should learn **nothing beyond**, say,

$$y = (\text{sum of YES votes}, \text{sum of NO votes})$$

Example: Secure Auctions



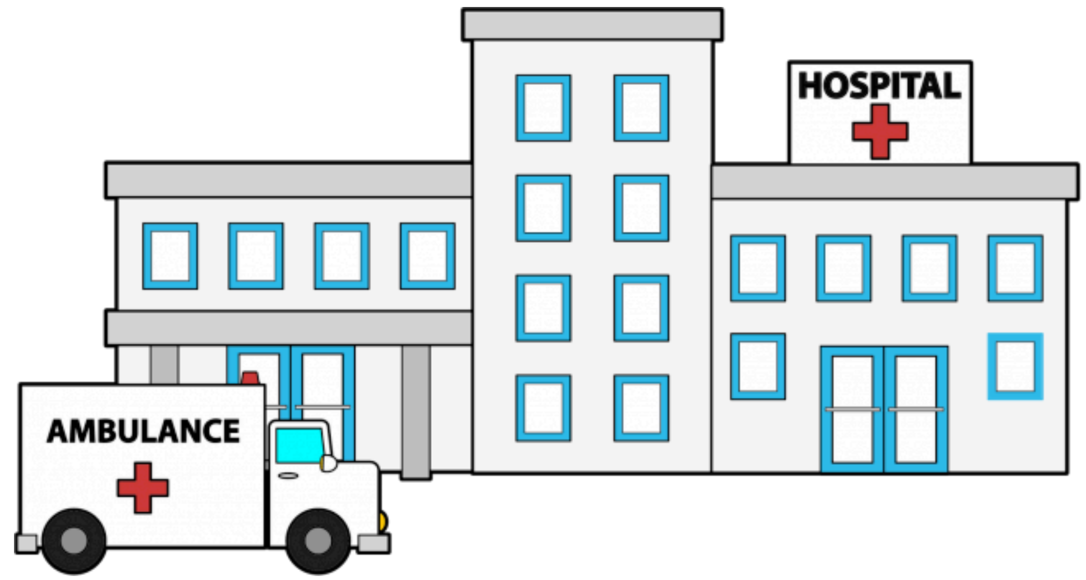
Find the winning bid, while keeping individual bids private. Everyone should learn **nothing beyond**, say,

$$y = \text{"identity of the largest bid} \geq m \text{ if one exists"}$$

i.e., more formally,

$$y = arg\,max\{w,x,m\}.$$

Etc.



Perform a scientific study on patient data,
without the hospital having to reveal such sensitive data.

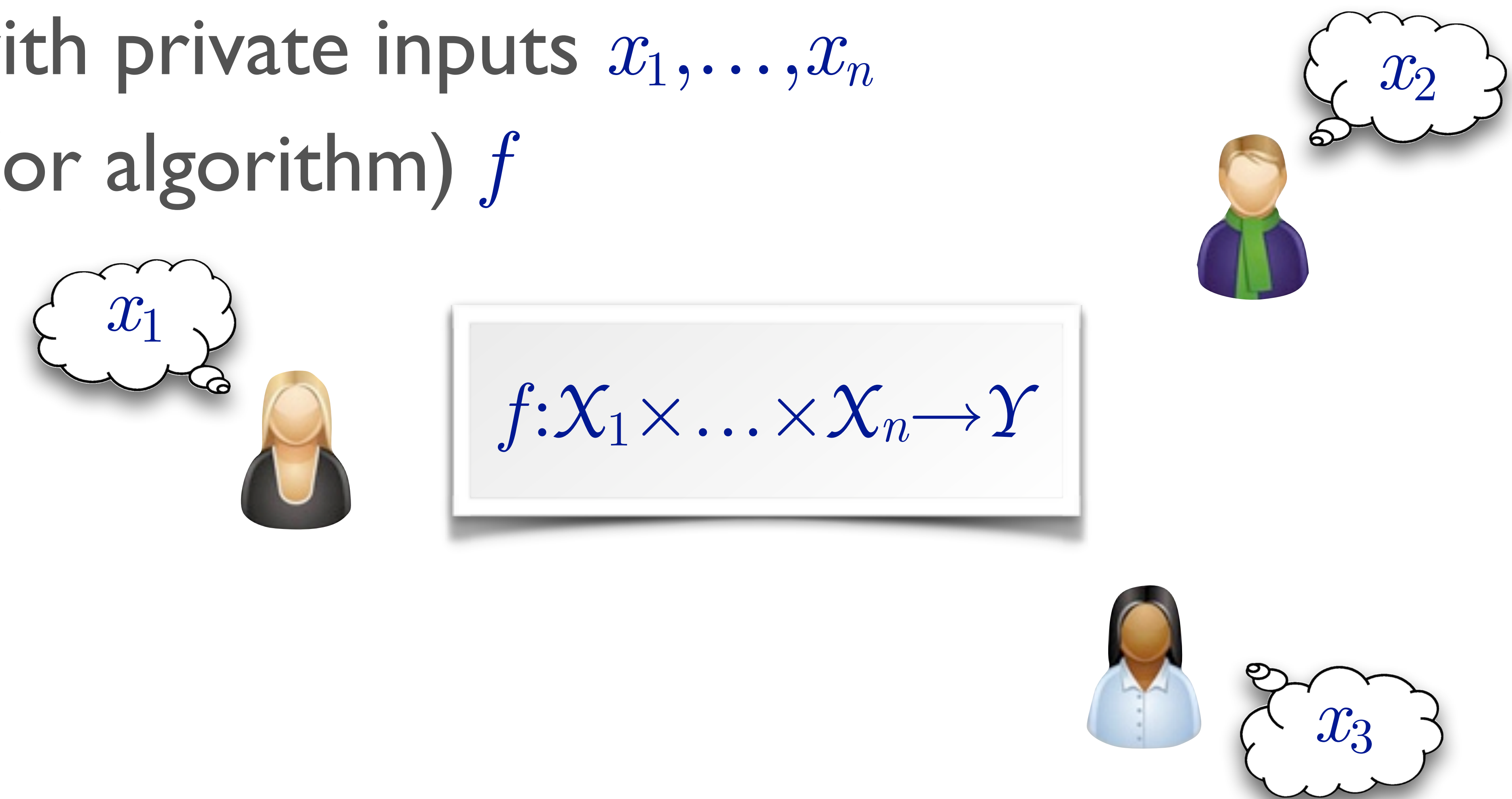
Etc. etc.

Find Facebook friends that are nearby, **without** letting Facebook (or friends not nearby) know where you are.

The General Goal

Given:

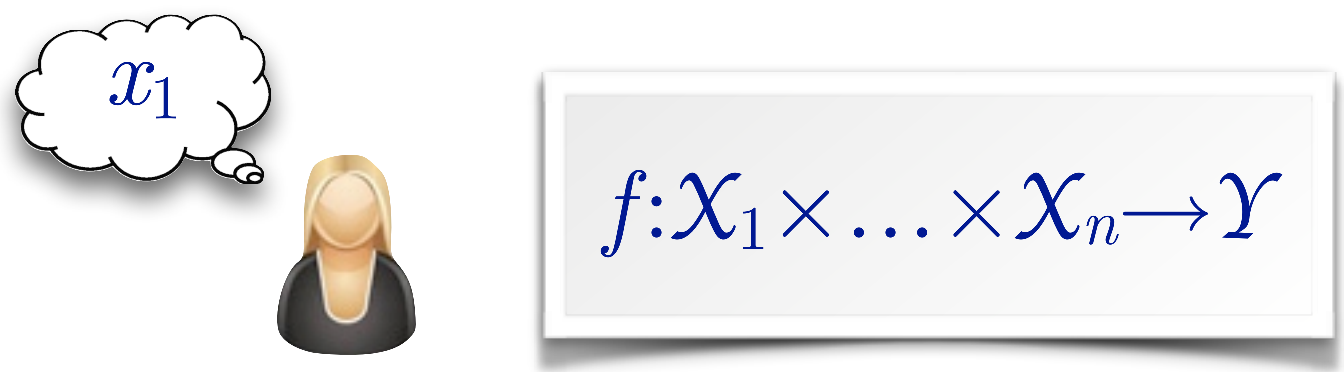
- n parties with private inputs $x_1, \ldots, x_n$
- a function (or algorithm) f

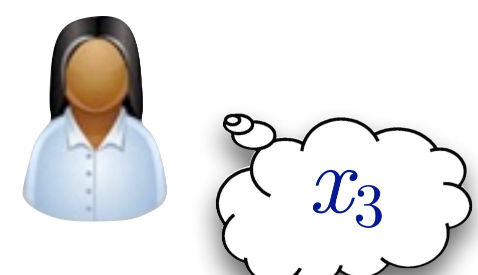


The General Goal

Given:

- n parties with private inputs $x_1,\ldots,x_n$
- a function (or algorithm) f





Want: compute $y=f(x_1,\ldots,x_n)$, so that

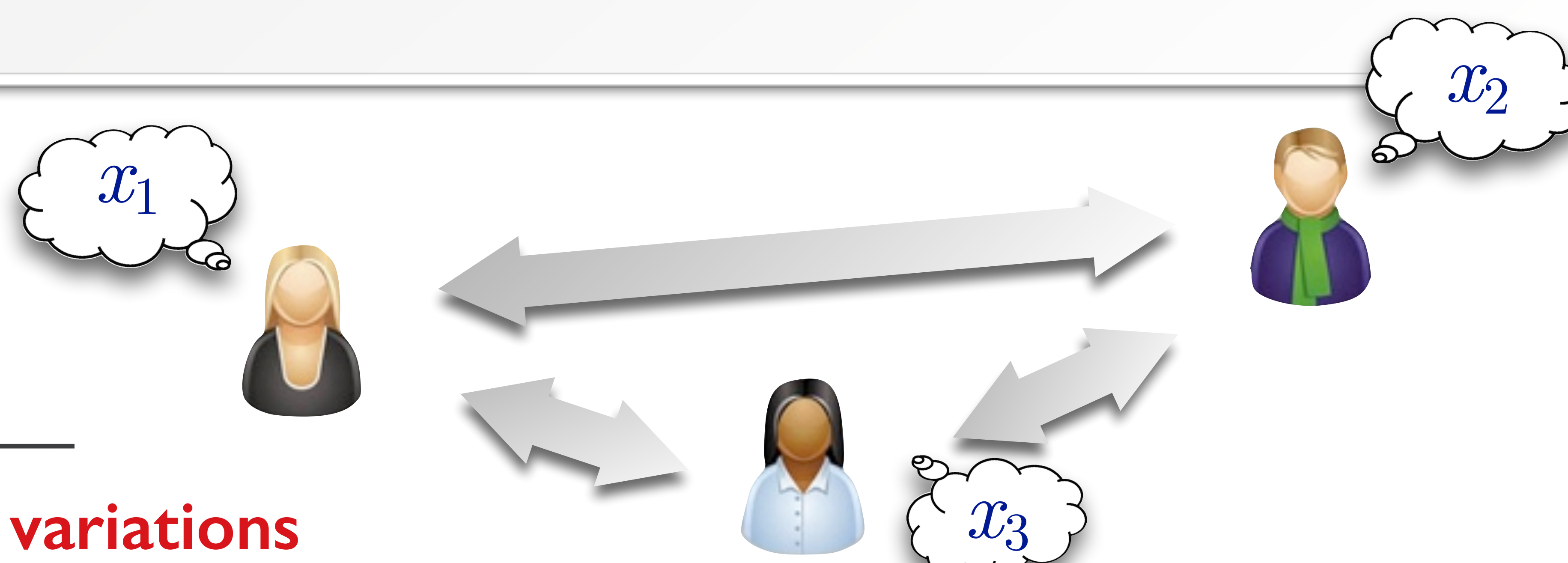
- everyone learns the (correct) result $y=f(x_1,\ldots,x_n)$
- but **nothing more** (in particular, the x_i **remain secret**)

Multiparty Computation (MPC)

Fundamental Theorem of MPC (*)

Originally invented/proven by [Yao 80's, Goldwasser-Micali-Wigderson 87, Chaum-Crépeau-Damgård 88, BenOr-Goldwasser-Wigderson 88]

Any function $f{:}\mathcal{X}_1\times\ldots\times\mathcal{X}_n\rightarrow\mathcal{Y}$ can be jointly computed by means of an **interactive protocol** in a **secure** way, so that:



(*) Comes in **lots of variations**

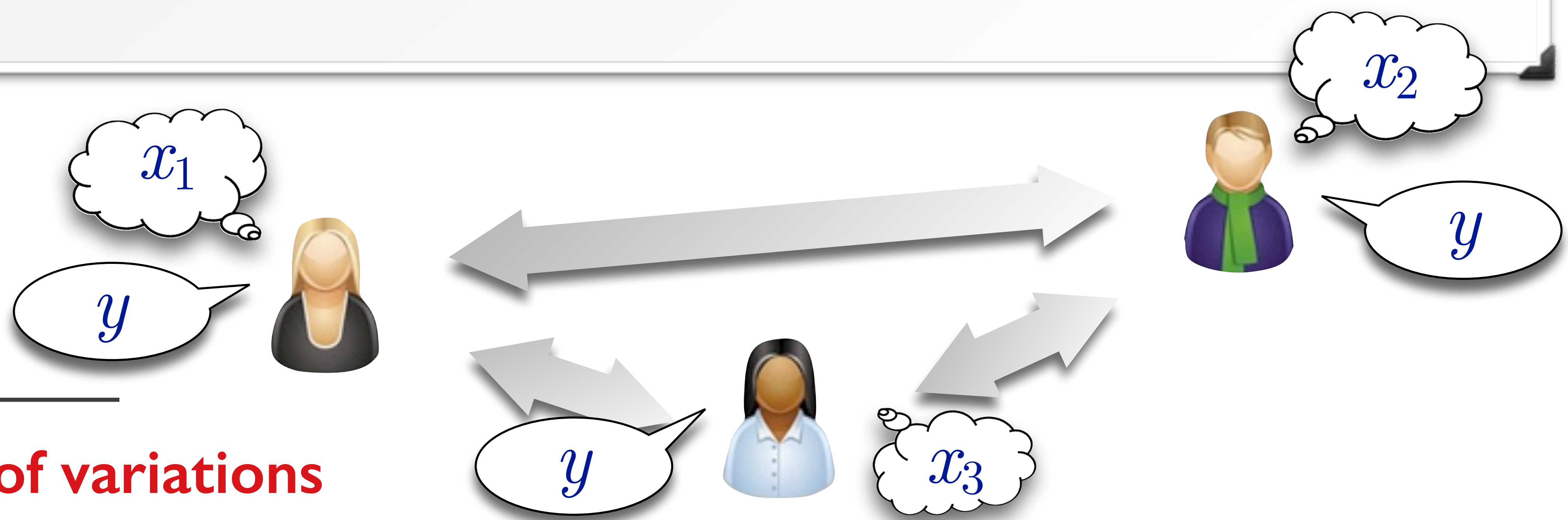
Multiparty Computation (MPC)

Fundamental Theorem of MPC (*)

Originally invented/proven by [Yao 80's, Goldwasser-Micali-Wigderson 87, Chaum-Crépeau-Damgård 88, BenOr-Goldwasser-Wigderson 88]

Any function $f{:}\mathcal{X}_1\times\ldots\times\mathcal{X}_n\rightarrow\mathcal{Y}$ can be jointly computed by means of an interactive protocol in a **secure** way, so that:

- everyone learns the **correct result** $y=f(x_1,\ldots,x_n)$,
- yet **nothing more** than than,



(*) Comes in **lots of variations**

Multiparty Computation (MPC)

Fundamental Theorem of MPC (*)

Originally invented/proven by [Yao 80's, Goldwasser-Micali-Wigderson 87, Chaum-Crépeau-Damgård 88, BenOr-Goldwasser-Wigderson 88]

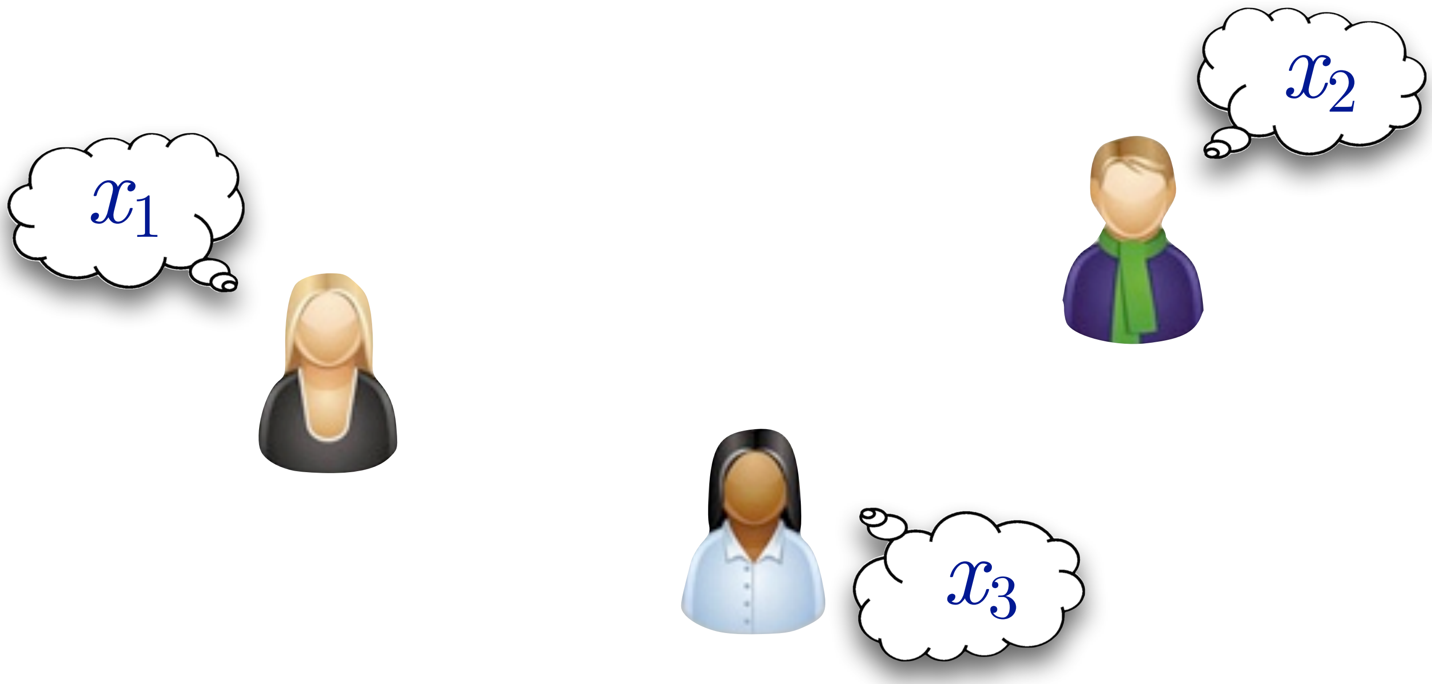
Any function $f{:}\mathcal{X}_1\times\ldots\times\mathcal{X}_n\rightarrow\mathcal{Y}$ can be jointly computed by means of an **interactive protocol** in a **secure** way, so that:

- everyone learns the **correct result** $y = f(x_1,\ldots,x_n)$,
- yet **nothing more** than than,
- even if some of the parties are **dishonest**.

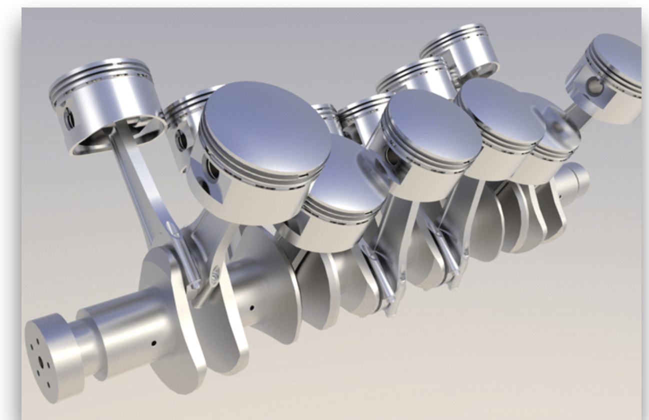
(*) Comes in **lots of variations**

Multiparty Computation (MPC) - In Clip Arts

Fundamental Theorem of MPC (*)

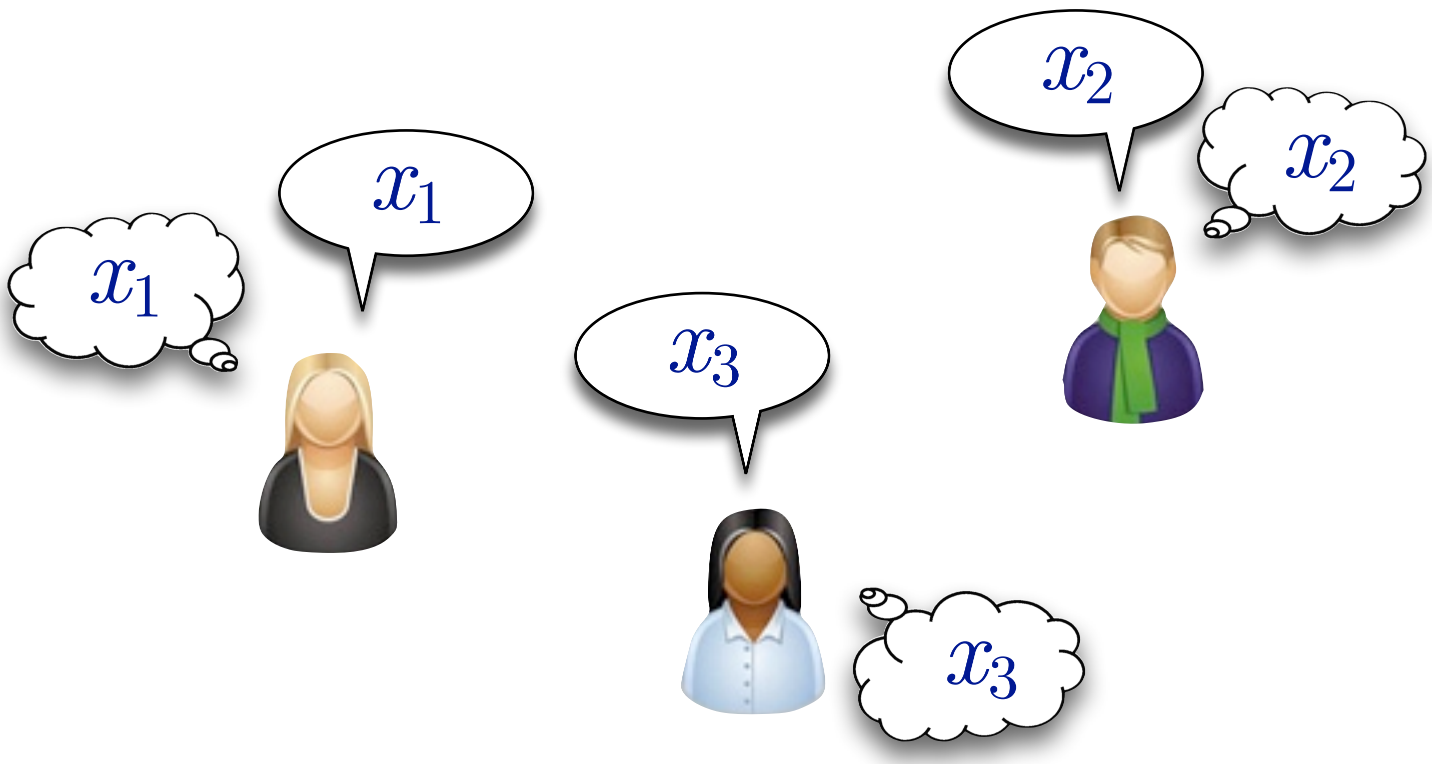


$\forall$ algorithm

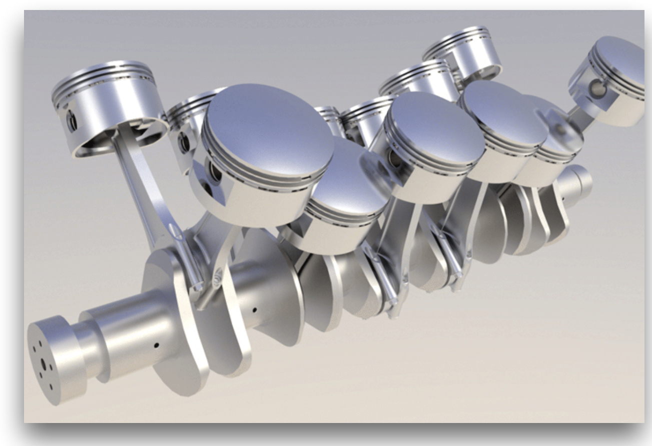


Multiparty Computation (MPC) - In Clip Arts

Fundamental Theorem of MPC (*)

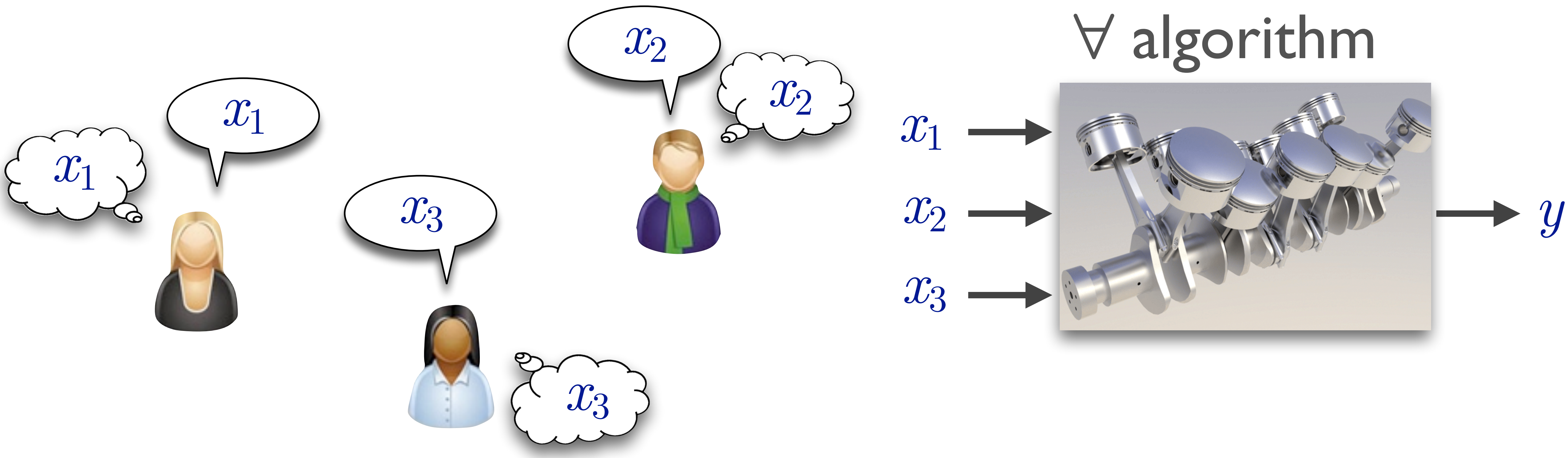


$\forall$ algorithm



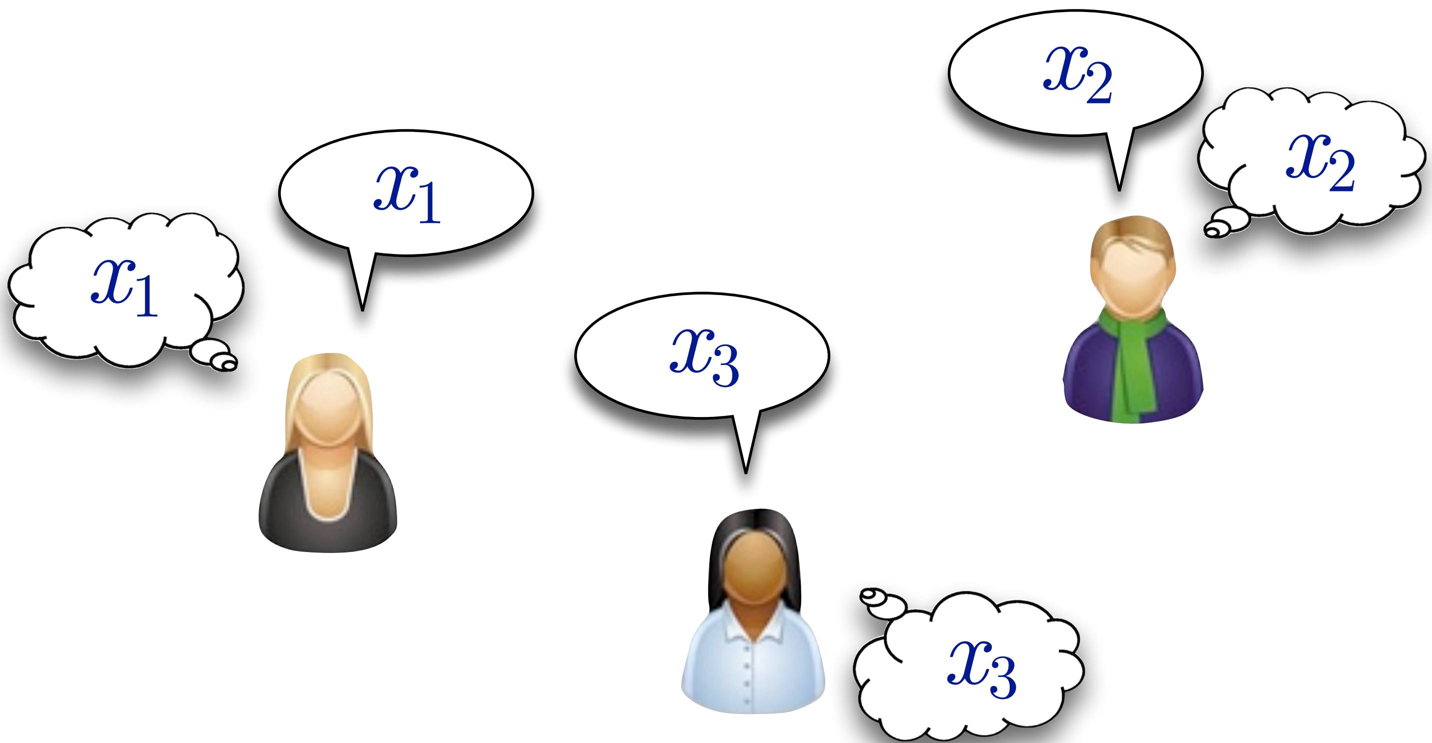
Multiparty Computation (MPC) - In Clip Arts

Fundamental Theorem of MPC (*)

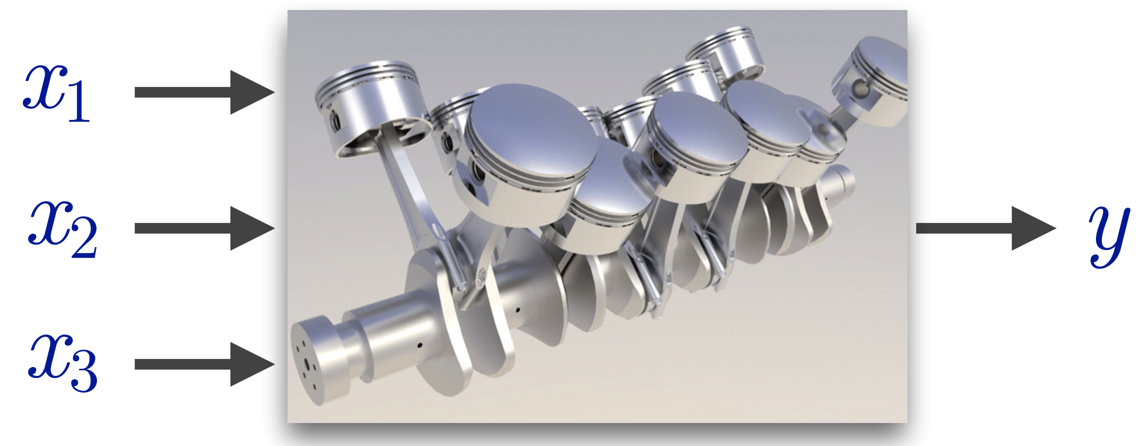


Multiparty Computation (MPC) - In Clip Arts

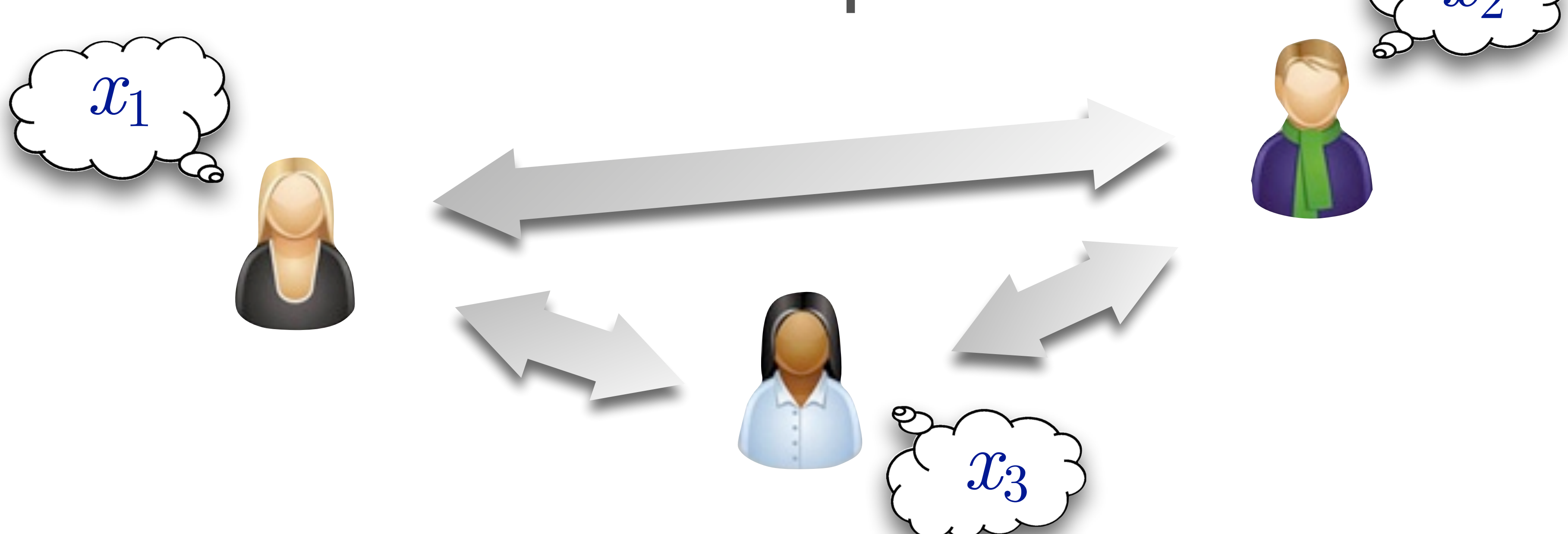
Fundamental Theorem of MPC (*)



$\forall$ algorithm

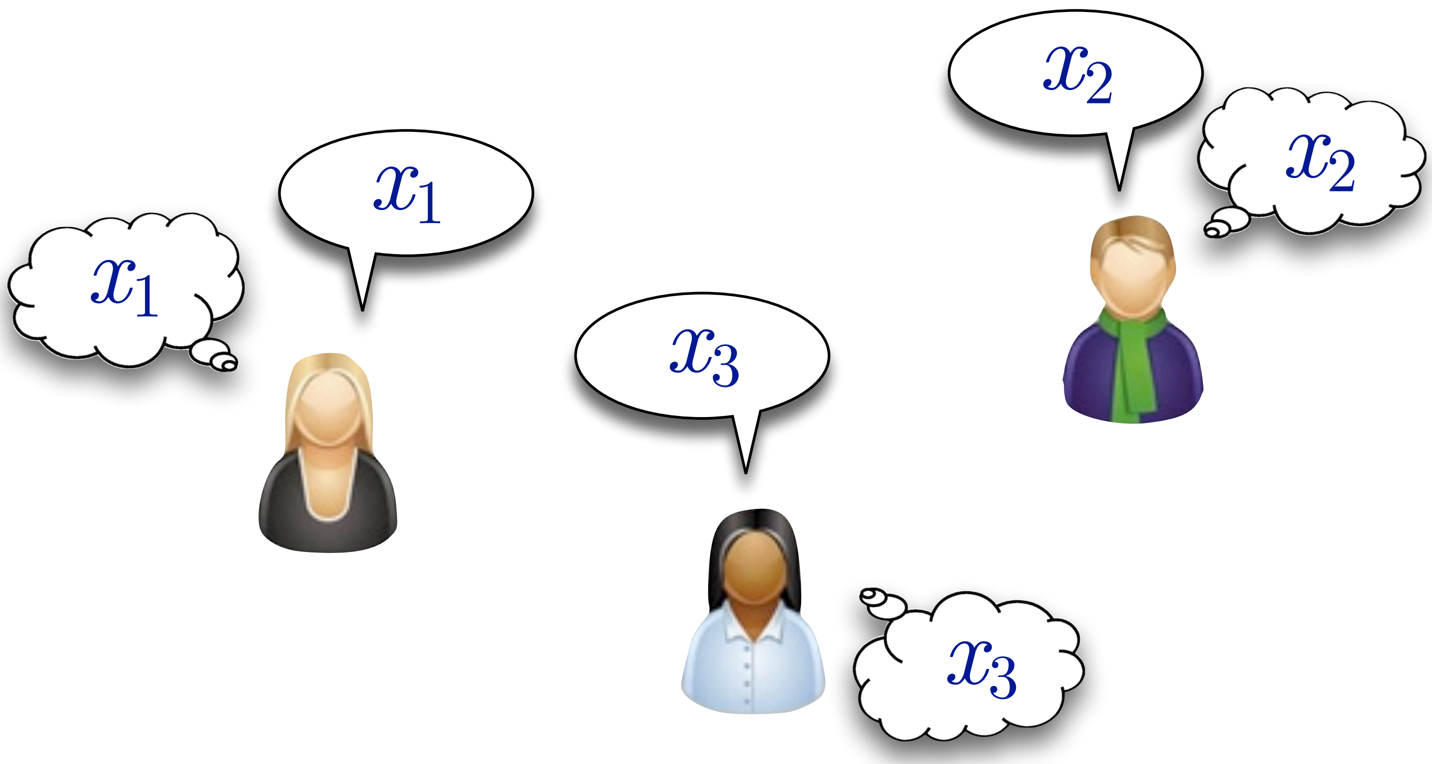


$\exists$ interactive protocol

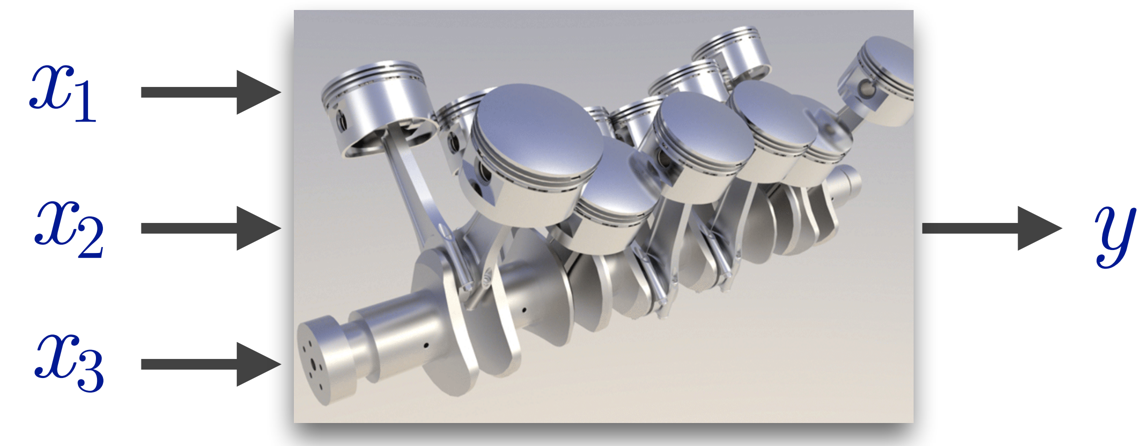


Multiparty Computation (MPC) - In Clip Arts

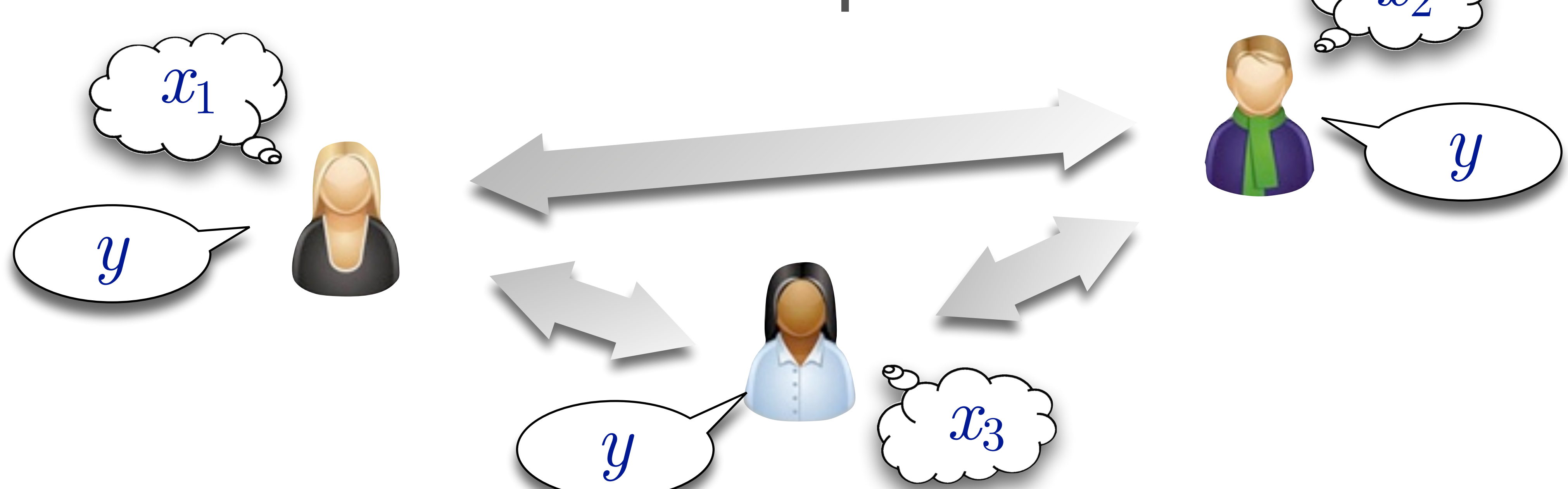
Fundamental Theorem of MPC (*)



$\forall$ algorithm

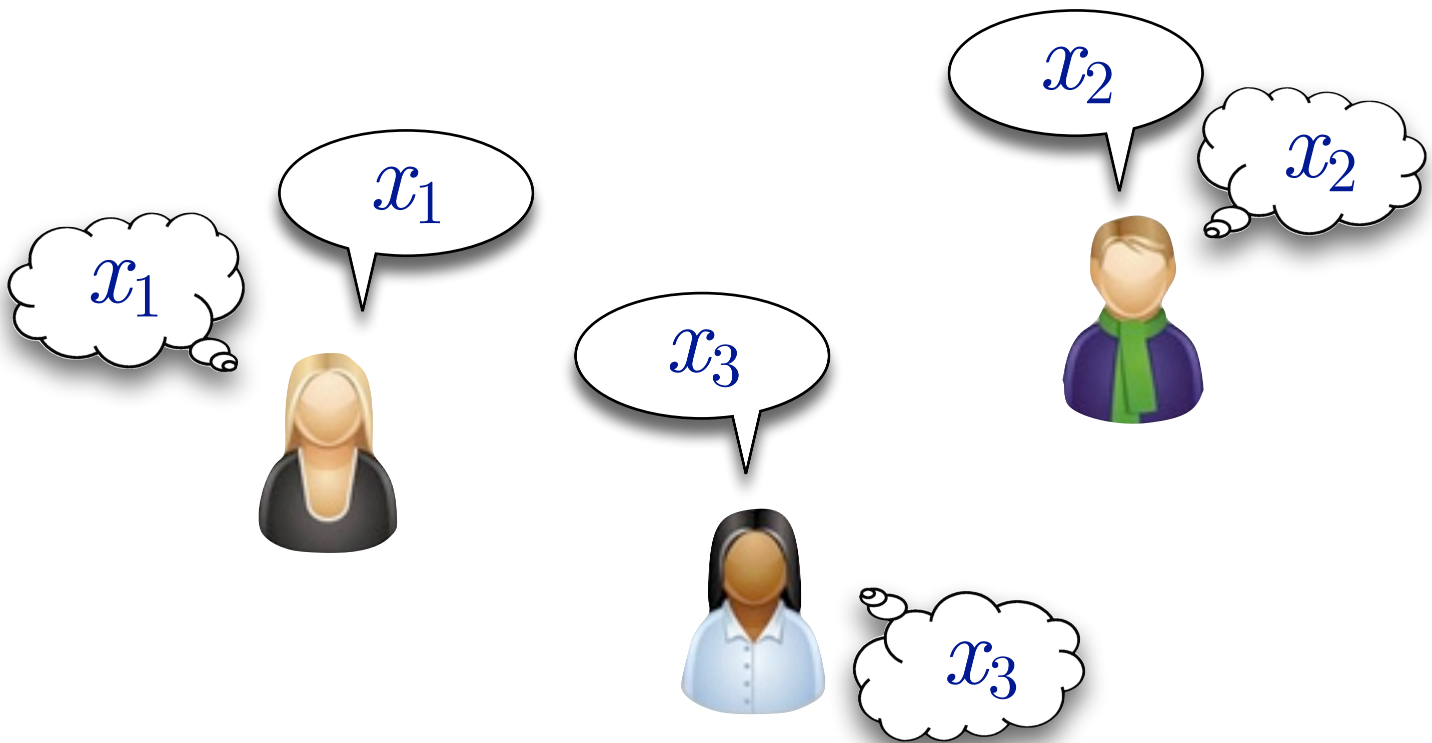


$\exists$ interactive protocol

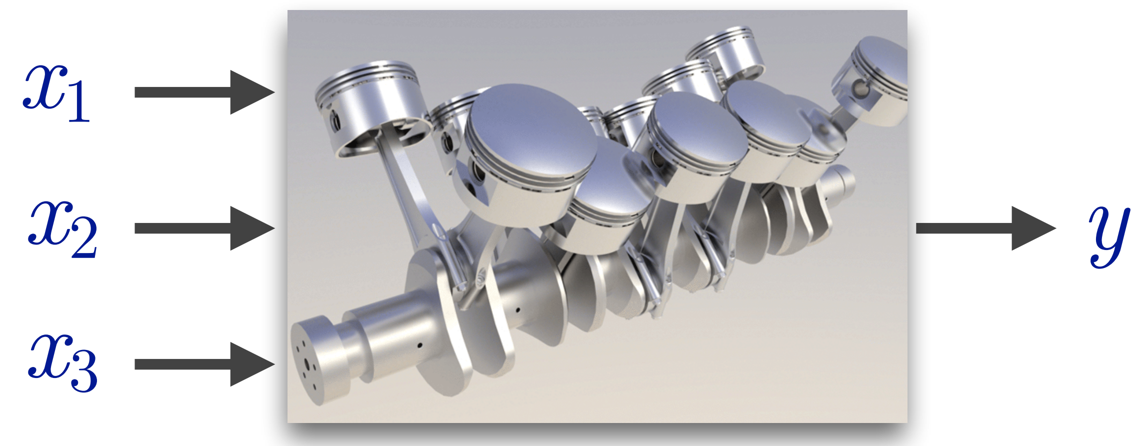


Multiparty Computation (MPC) - In Clip Arts

Fundamental Theorem of MPC (*)

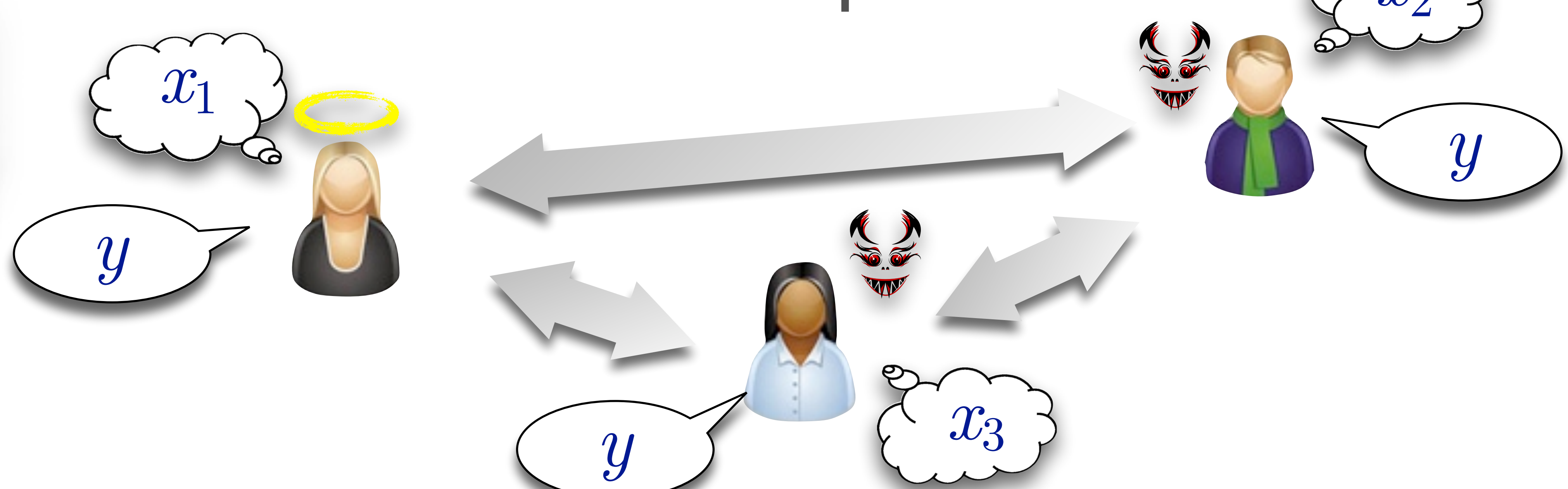


$\forall$ algorithm



$\exists$ interactive protocol

x_i's remain secret



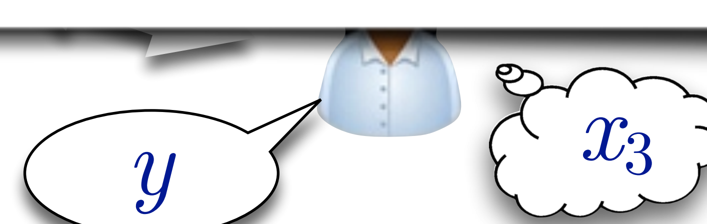
Multiparty Computation (MPC) - In Clip Arts

Fundamental Theorem of MPC (*)

(*) Comes in lots of **different variations**, in terms of:

- number of conspiring dishonest parties it tolerates
- assumed capabilities of dishonest parties
- considered communication infrastructure
- (dis)allowing the protocol to abort
- (not) requiring fairness and/or cheater detection
- etc.

Also, comes with a (significant) **overhead** in computation and communication.

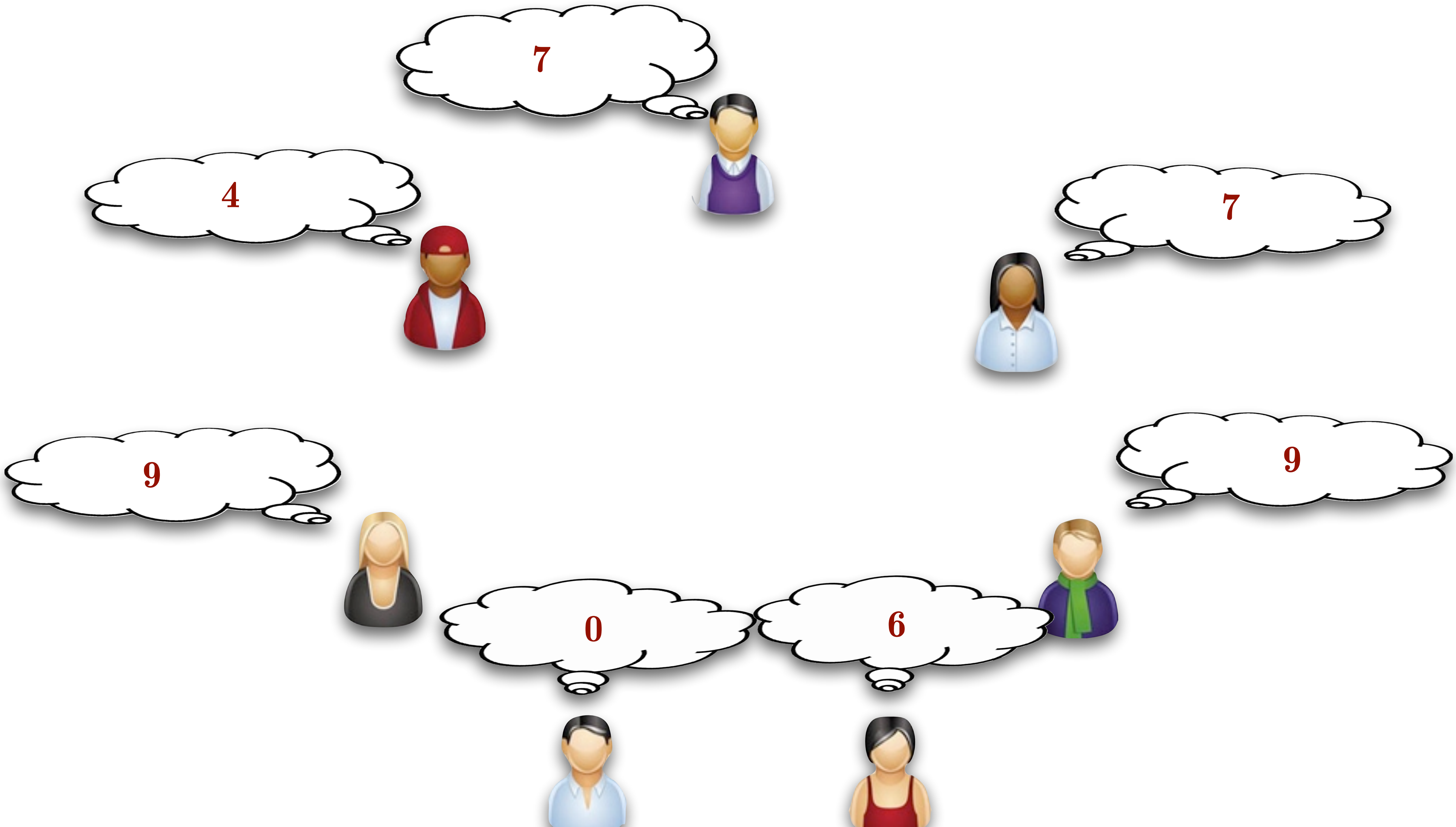


Road Map

- WHAT is multiparty computation?
- HOW does multiparty computation work?
- WHERE can/is multiparty computation be/ used?

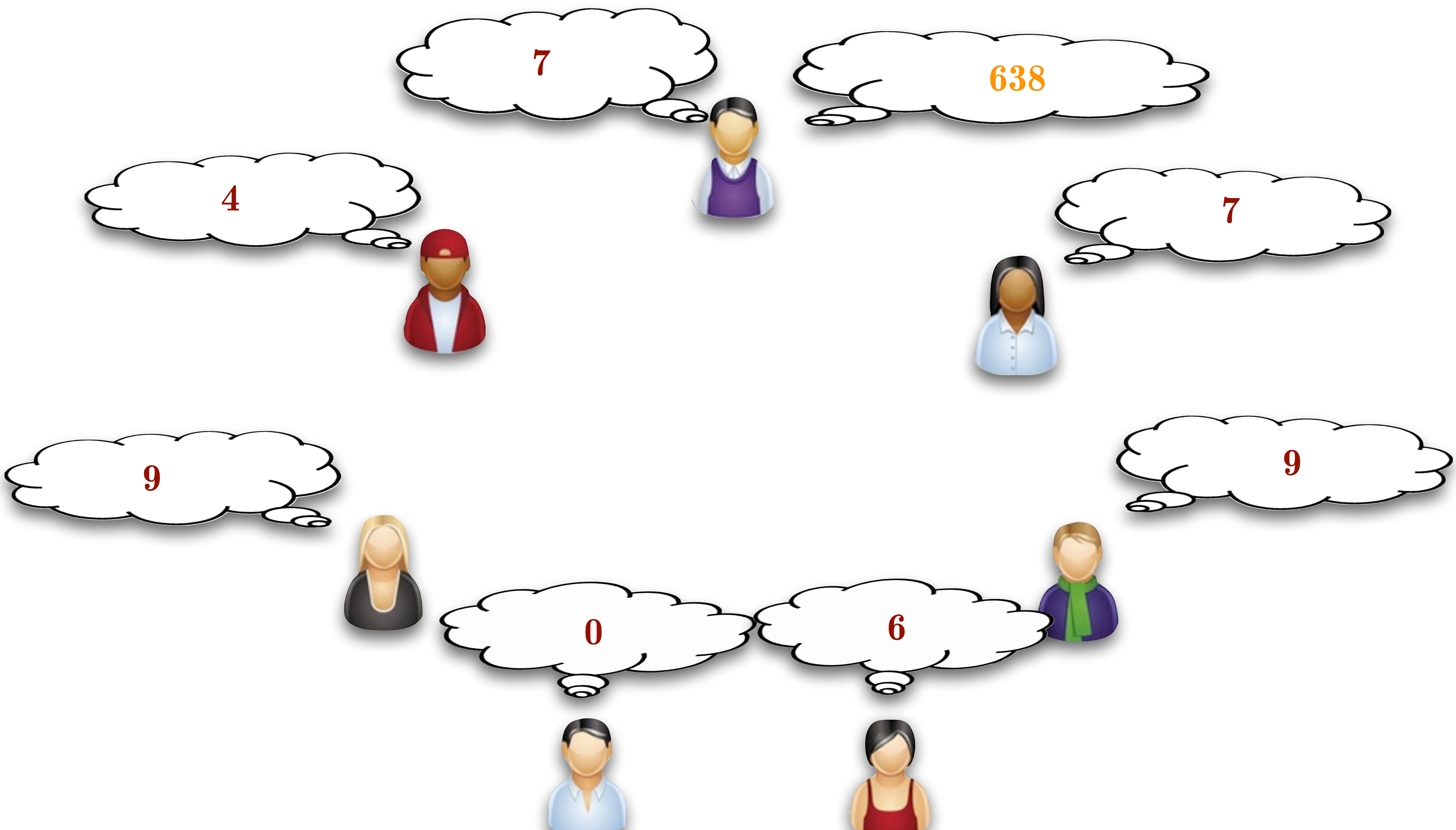
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



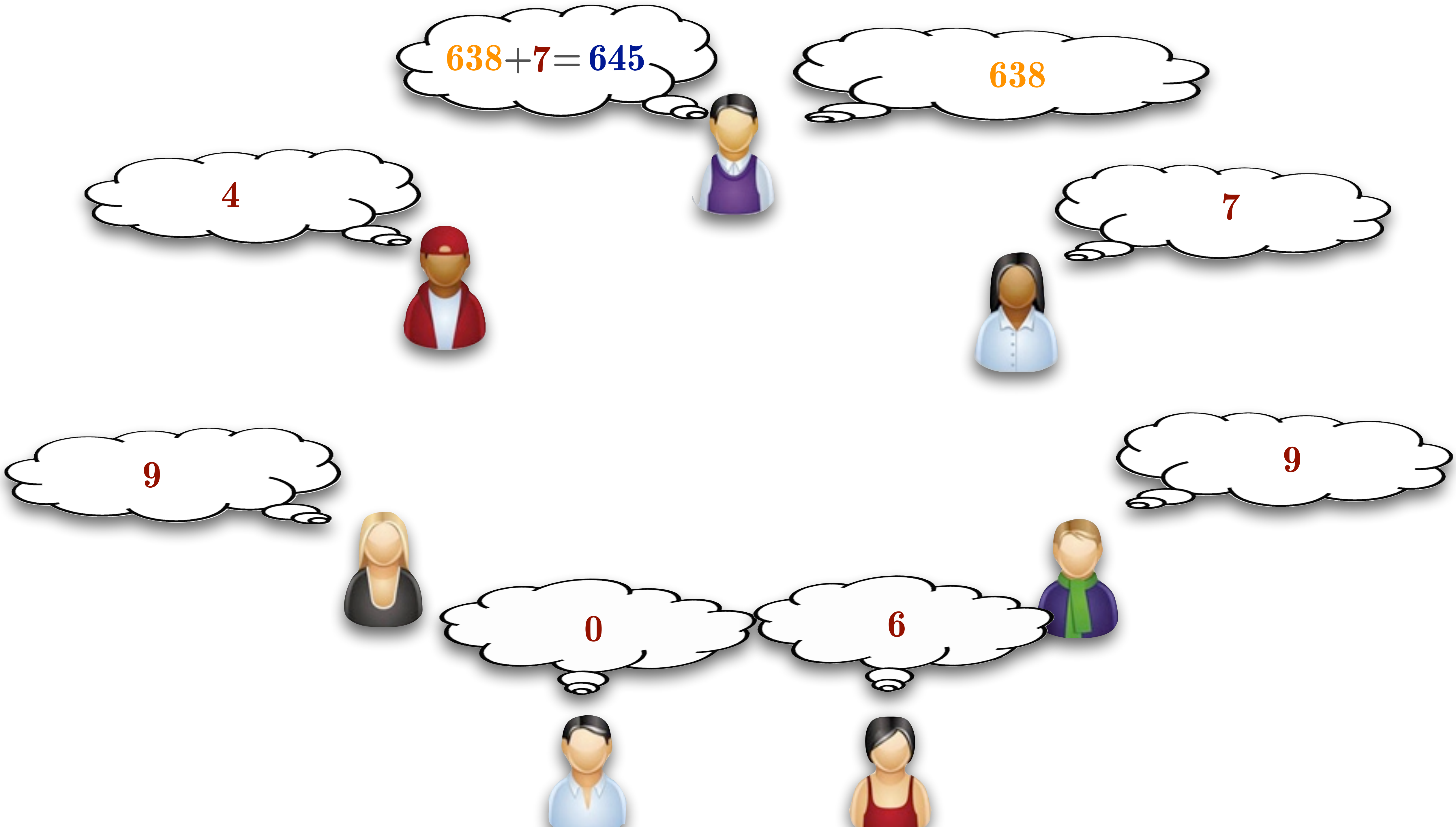
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



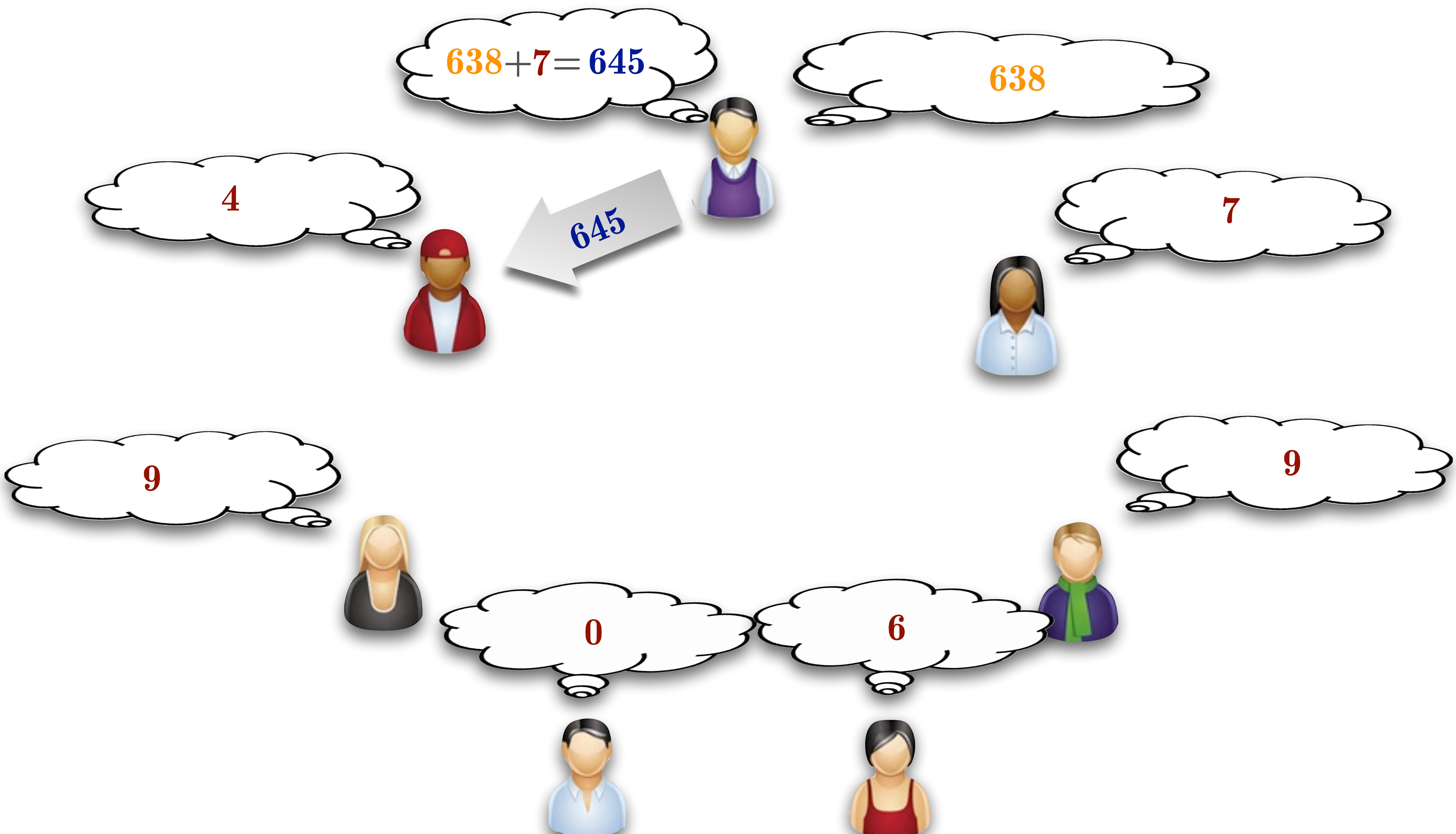
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



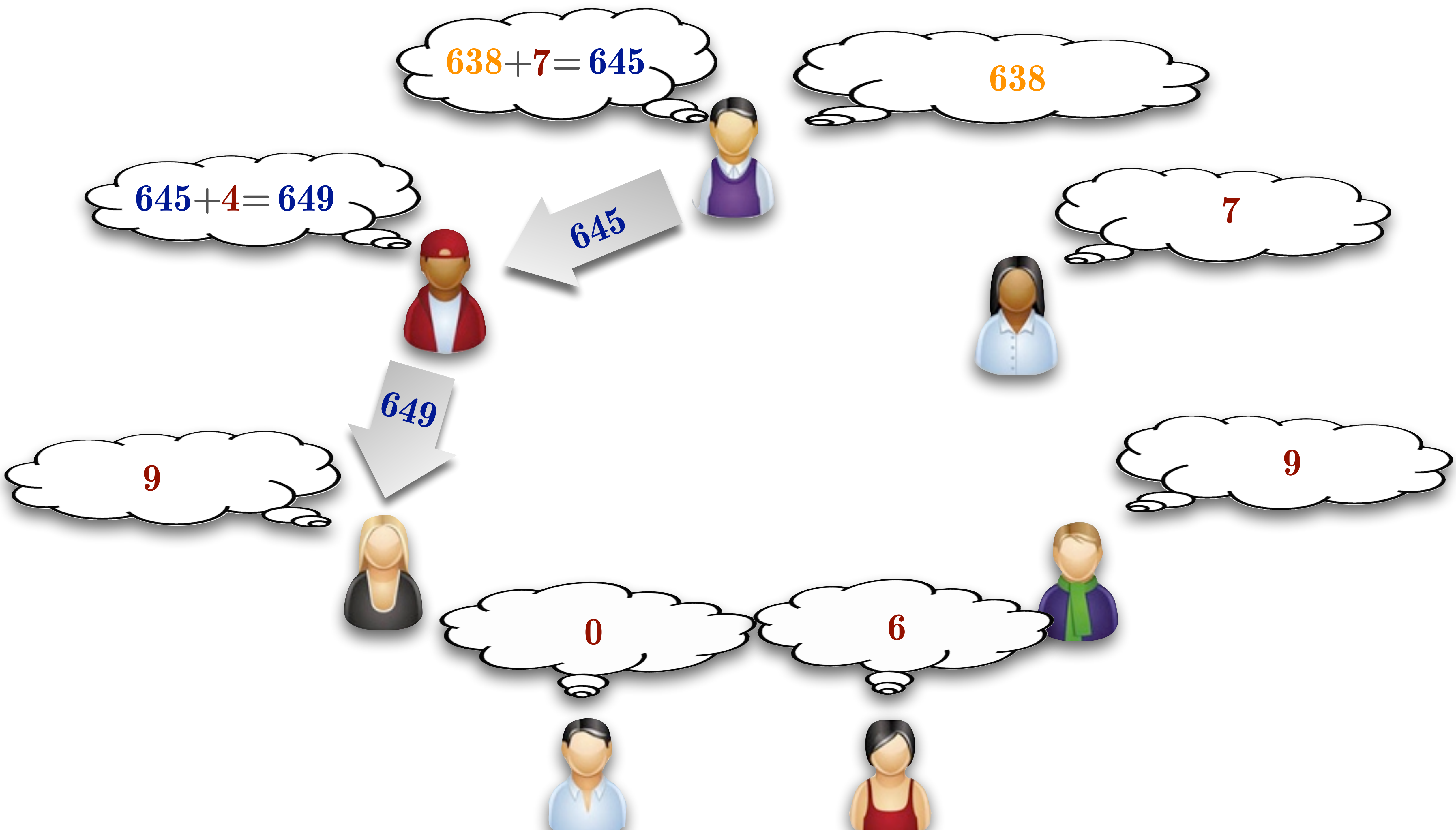
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



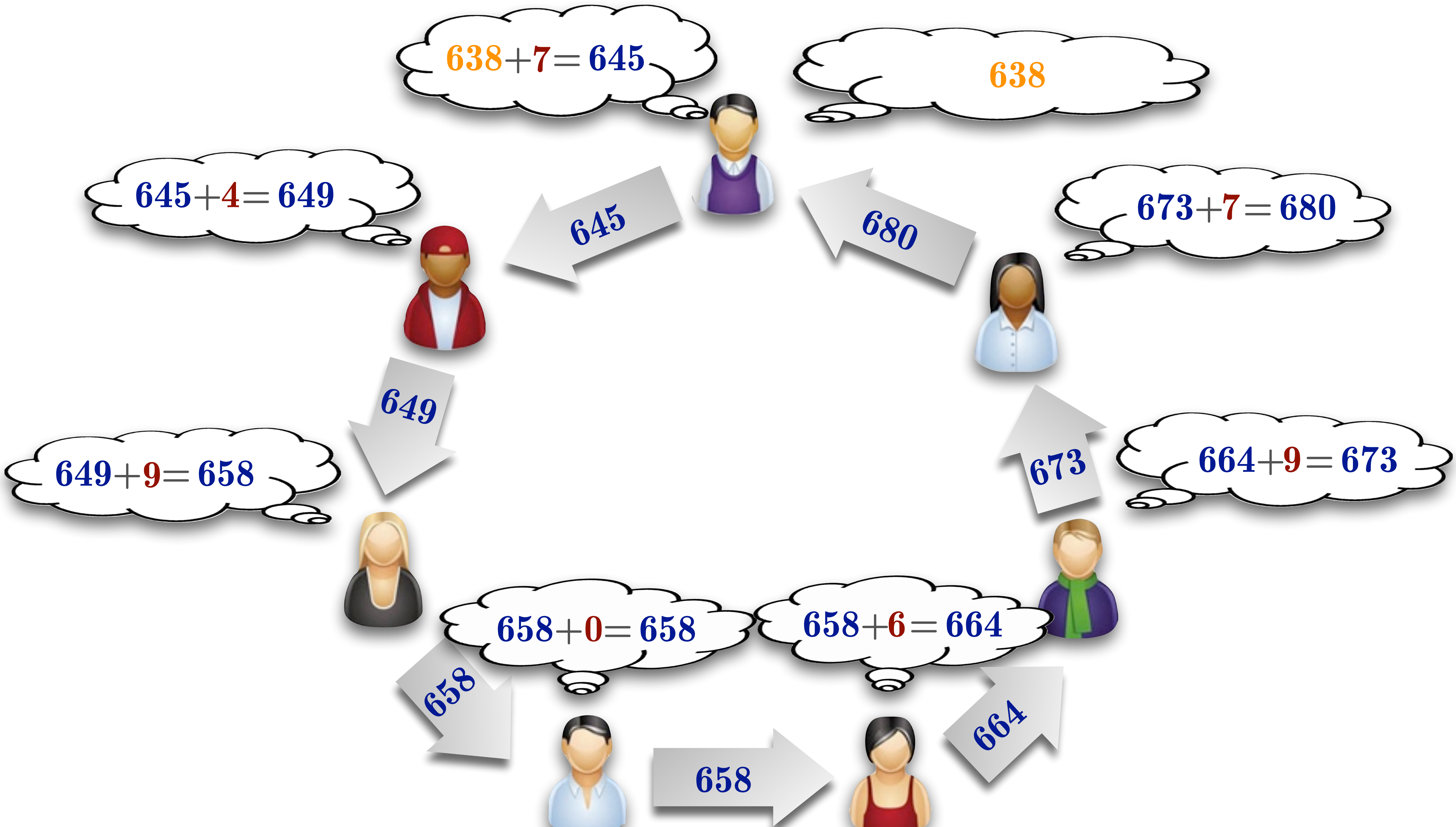
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1, \dots, x_n) = \sum x_i$



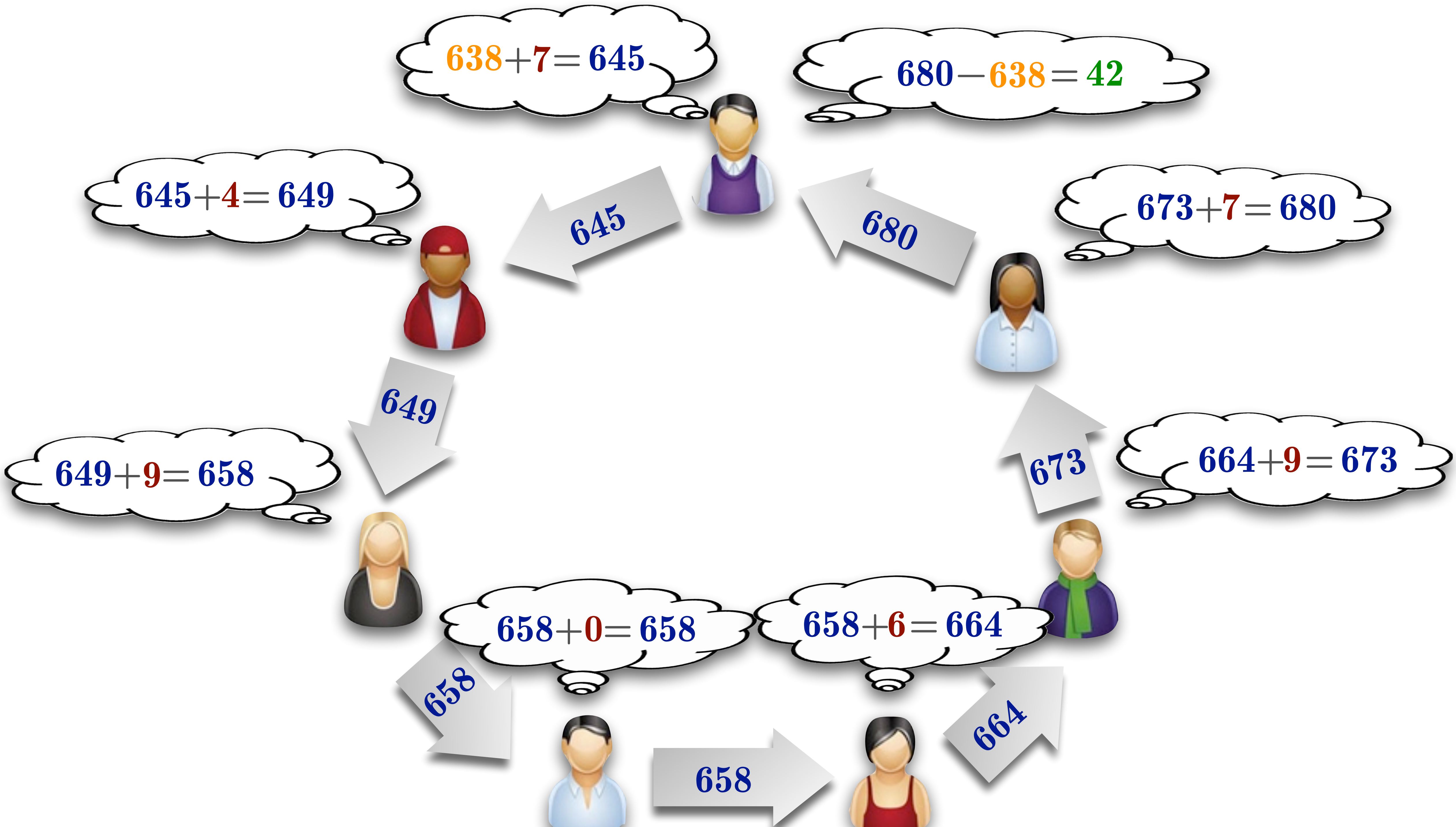
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



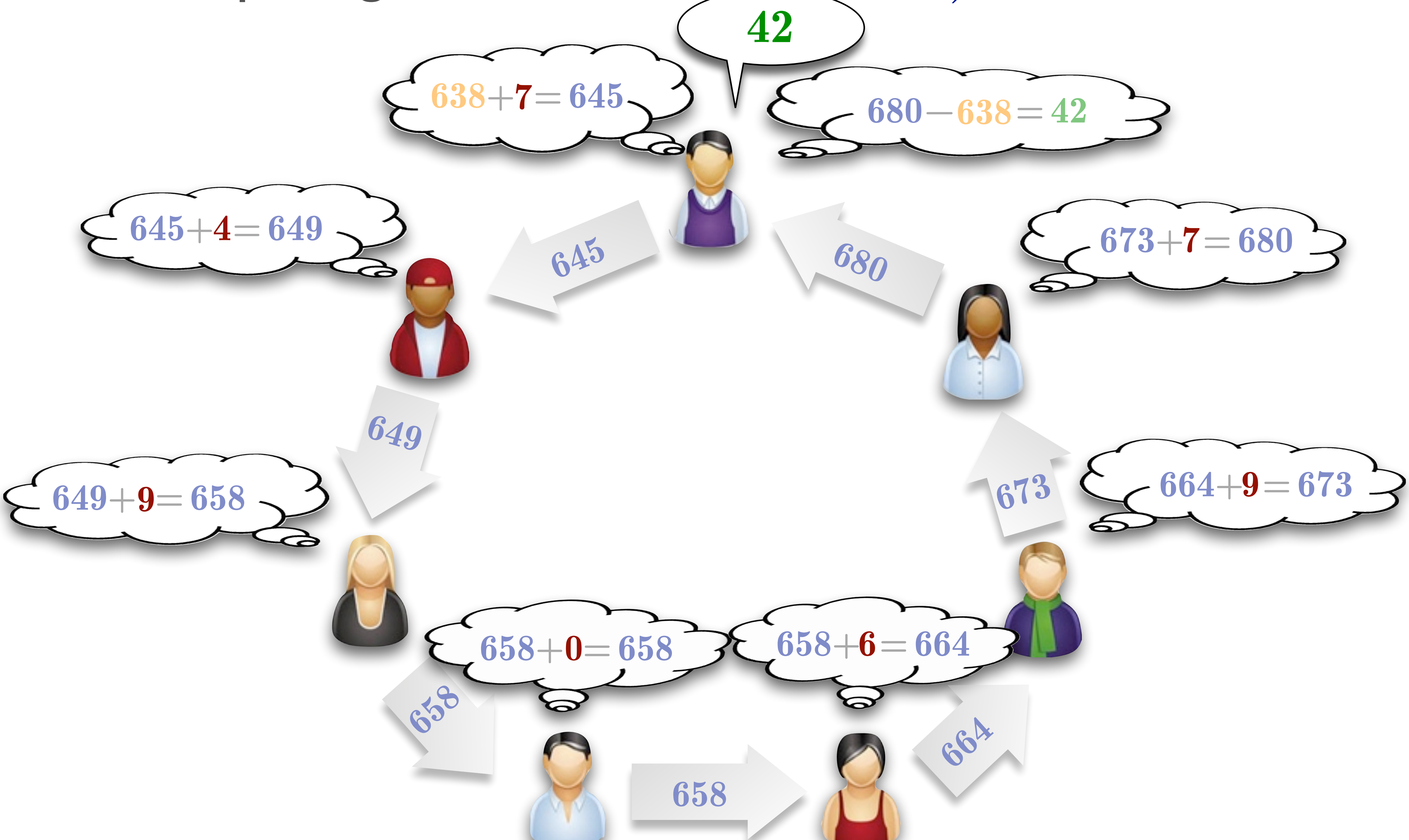
MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$

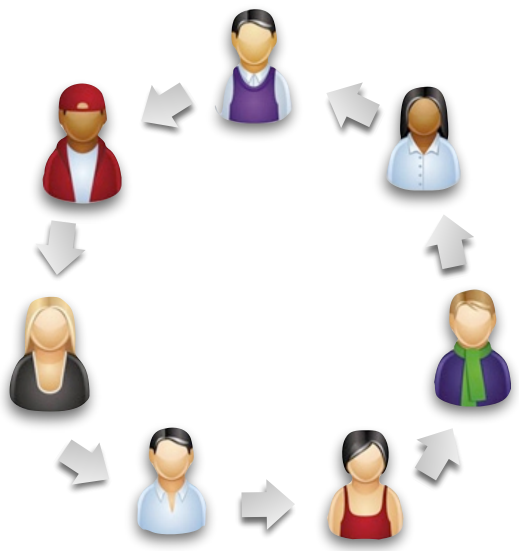


MPC: A very first try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$

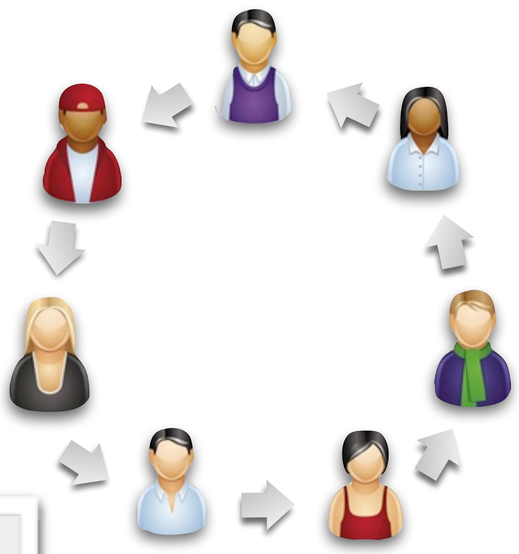


Analysis



Works sort-of, but:

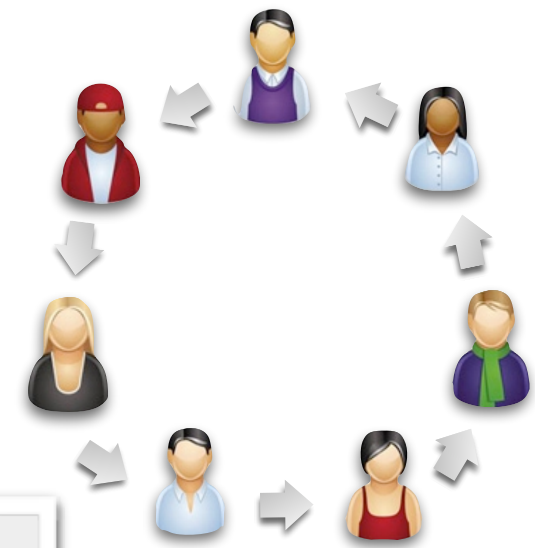
Analysis



Works sort-of, but:

"Leader" can lie about the result:
→ **no *correctness* or *fairness* guaranteed**

Analysis

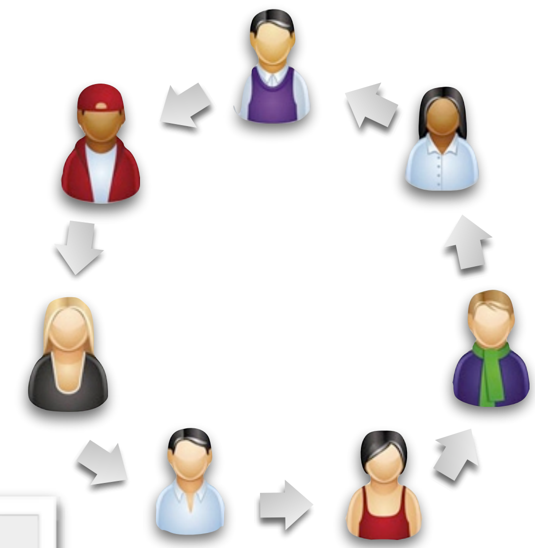


Works sort-of, but:

"Leader" can lie about the result:
→ **no *correctness*** or ***fairness*** guaranteed

Two parties together learn input of party in-between
→ **no *privacy*** of inputs **against certain coalitions**

Analysis



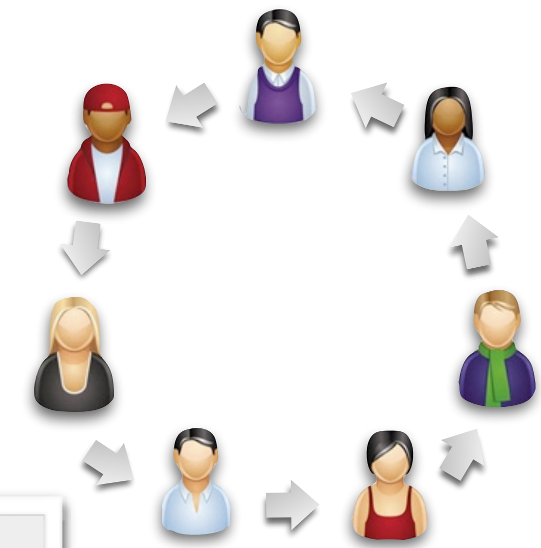
Works sort-of, but:

"Leader" can lie about the result:
→ **no** ***correctness*** or ***fairness*** guaranteed

Two parties together learn input of party in-between
→ **no** ***privacy*** of inputs **against certain coalitions**

Any party can stall the procedure → **not** ***abort-free***

Analysis



Works sort-of, but:

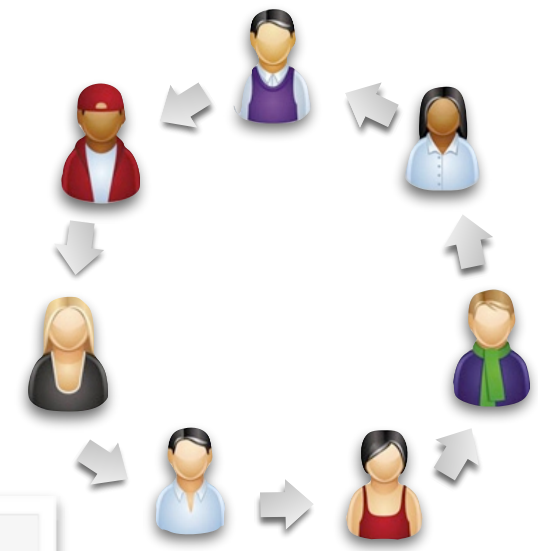
"Leader" can lie about the result:
→ **no** ***correctness*** or ***fairness*** guaranteed

Two parties together learn input of party in-between
→ **no** ***privacy*** of inputs **against certain coalitions**

Any party can stall the procedure → **not** ***abort-free***

Approach/solution **limited** to linear functions

Analysis



Works sort-of, but:

"Leader" can lie about the result:
→ **no** *correctness* or *fairness* guaranteed

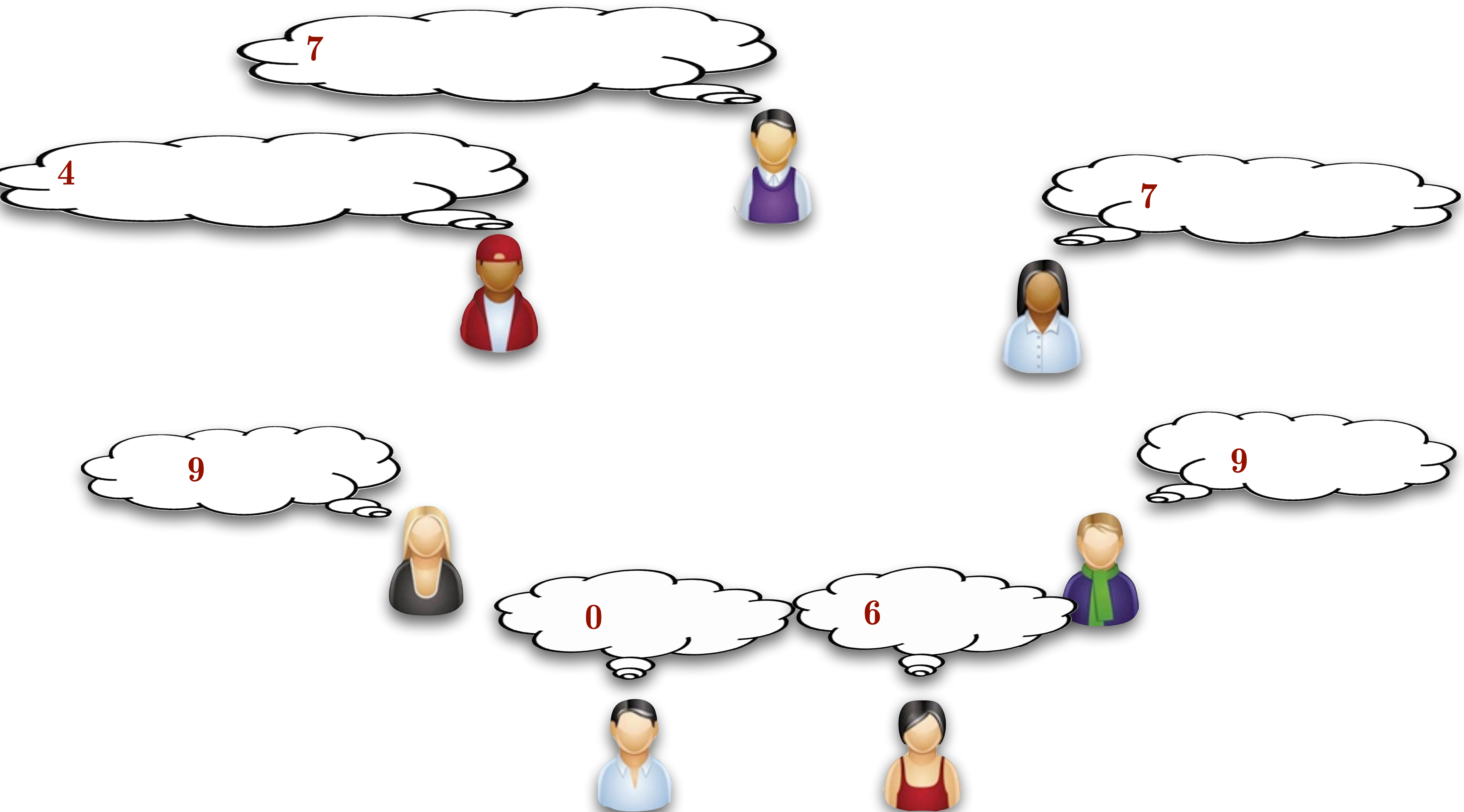
Two parties together learn input of party in-between
→ **no** *privacy* of inputs **against certain coalitions**

Any party can stall the procedure → **not** *abort-free*

Approach/solution **limited** to linear functions

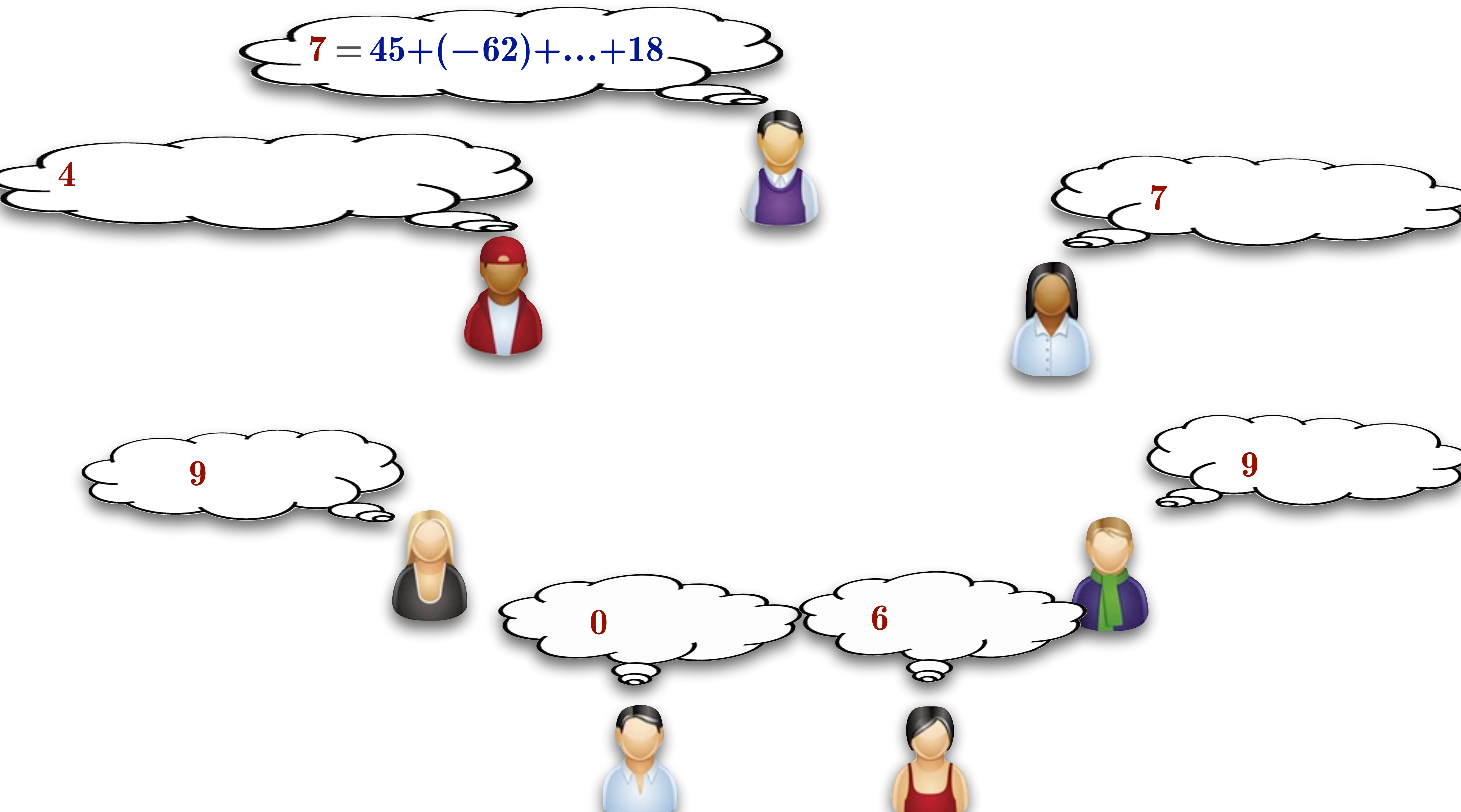
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



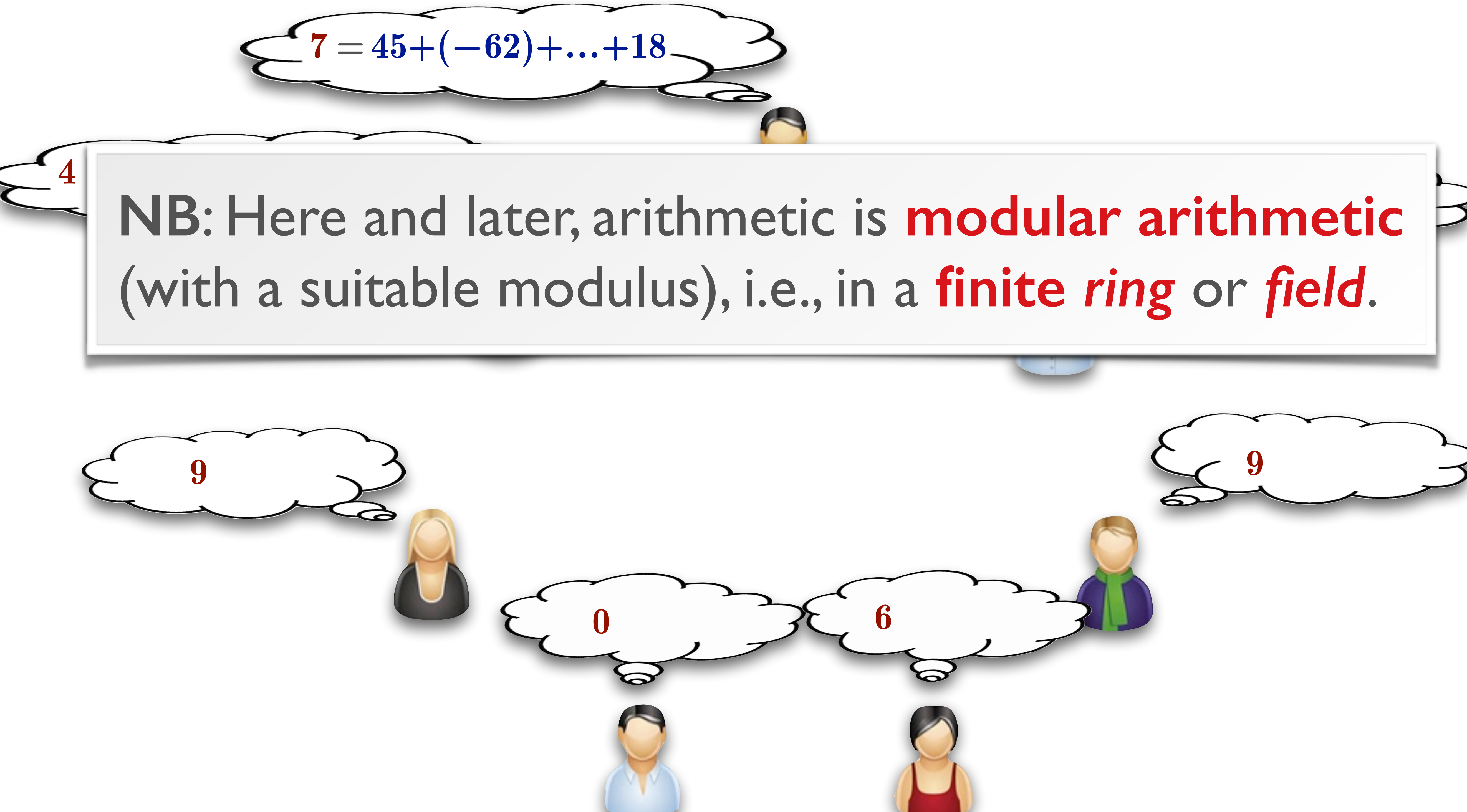
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



MPC: A second try

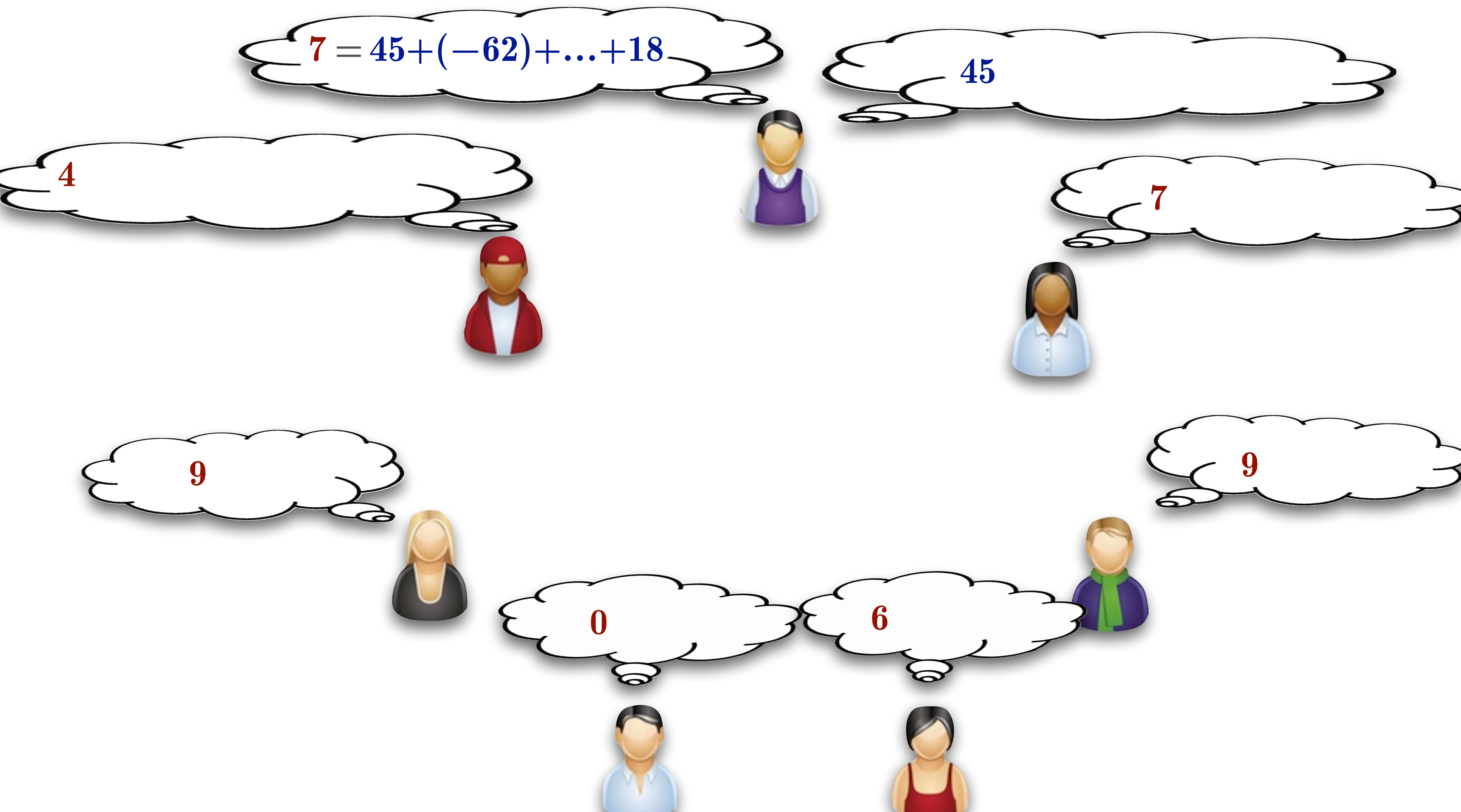
Goal: Computing the **sum**, i.e., $f(x_1,\ldots,x_n)=\sum x_i$



NB: Here and later, arithmetic is **modular arithmetic** (with a suitable modulus), i.e., in a **finite *ring*** or ***field***.

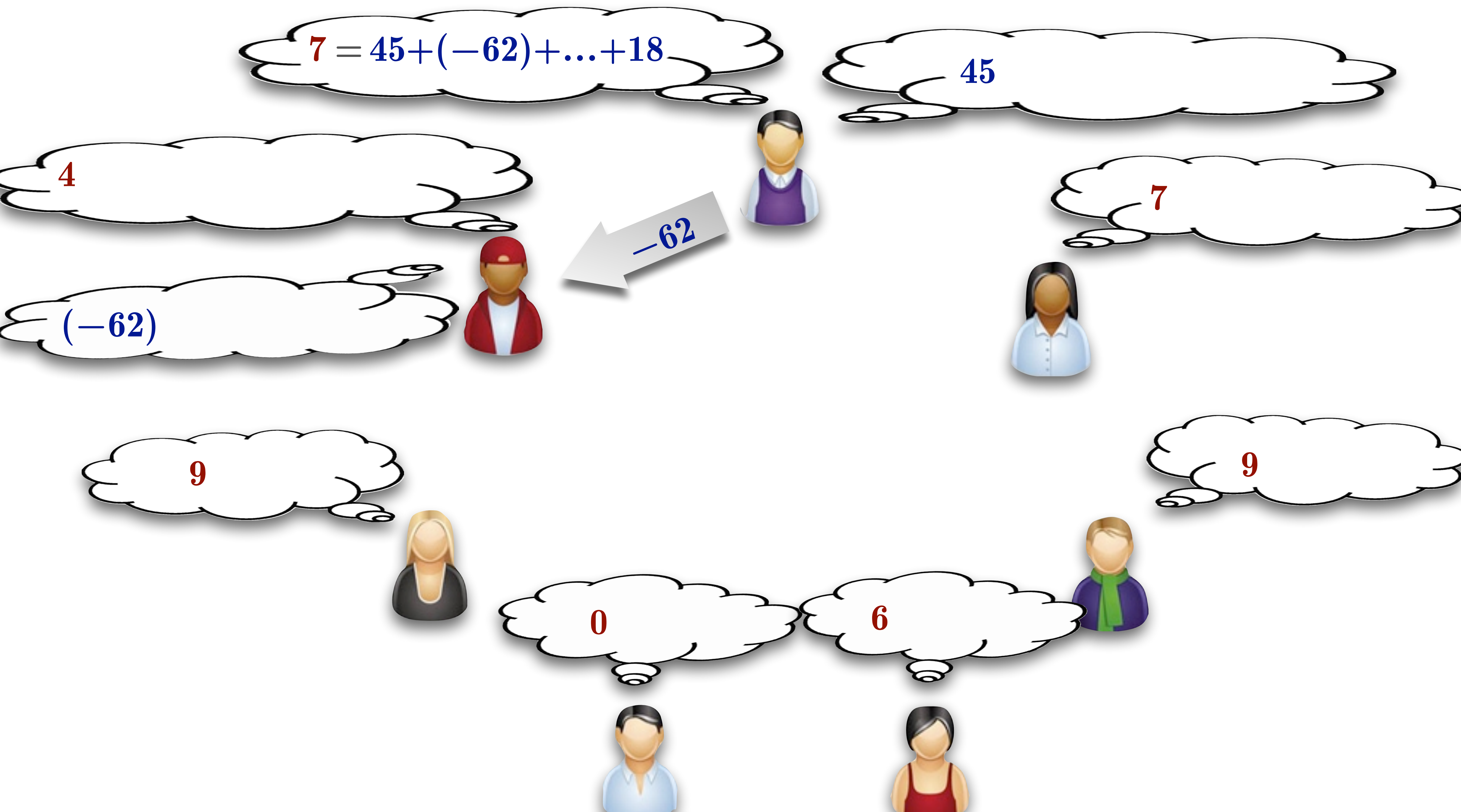
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



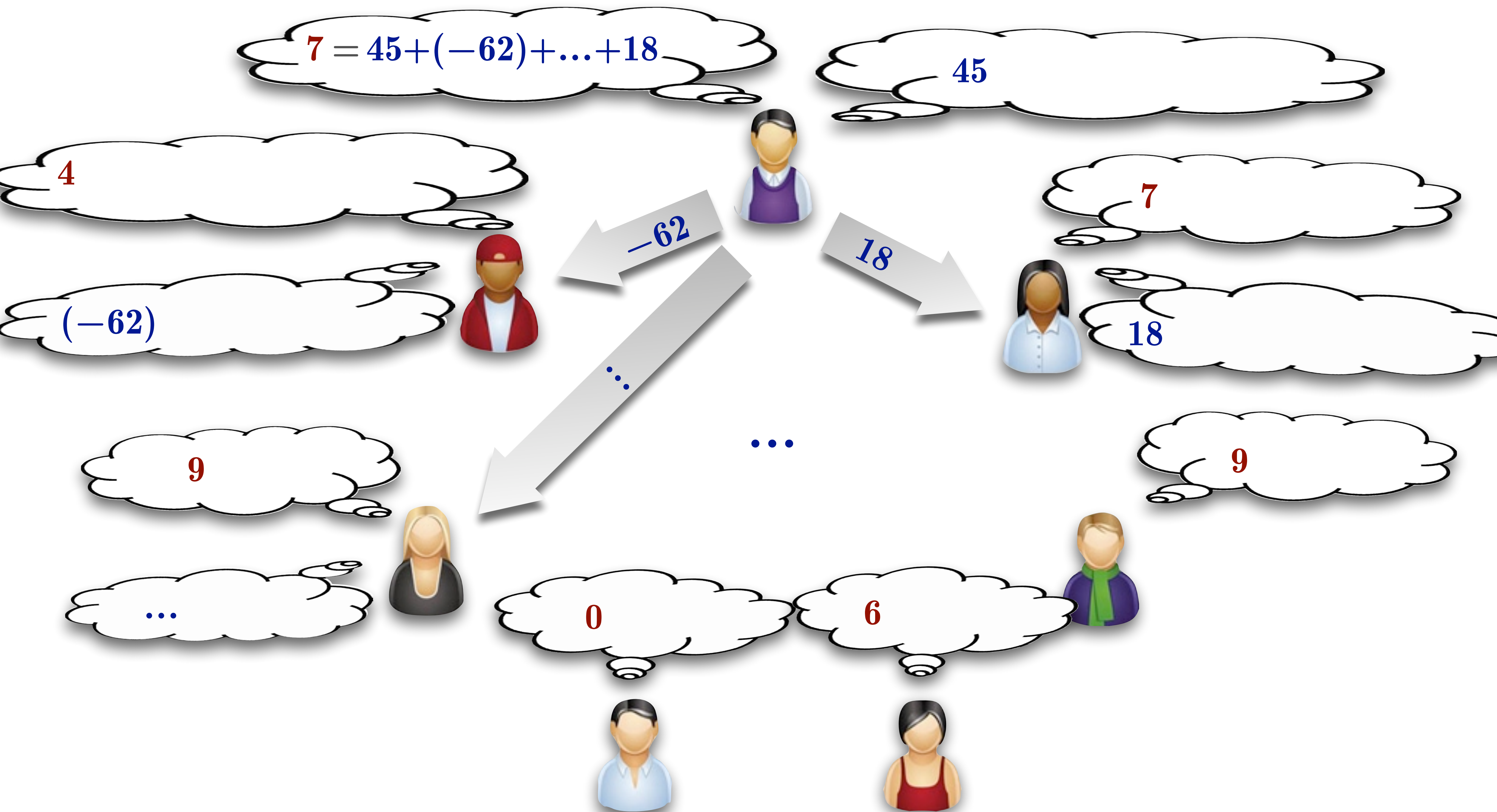
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



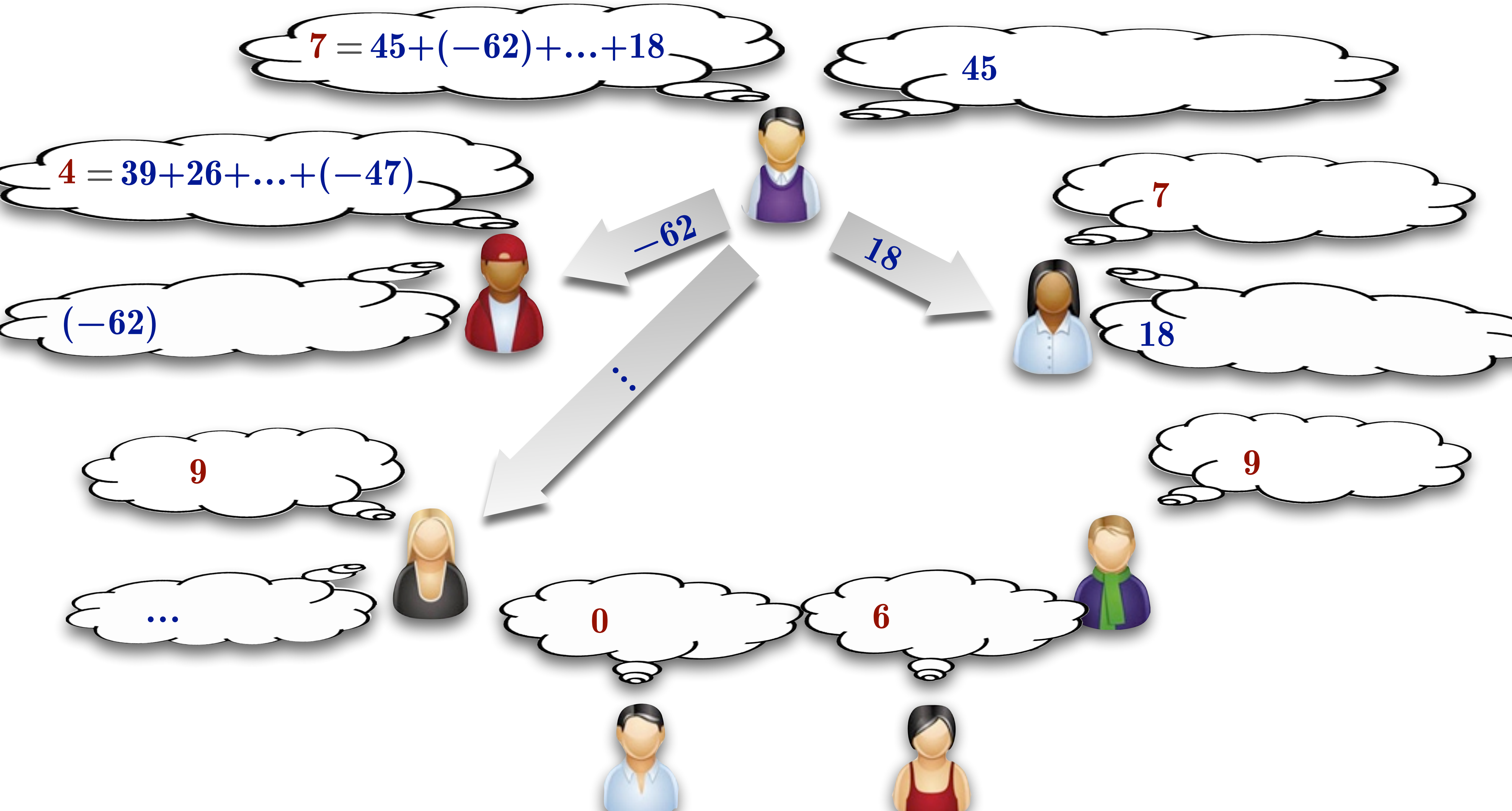
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



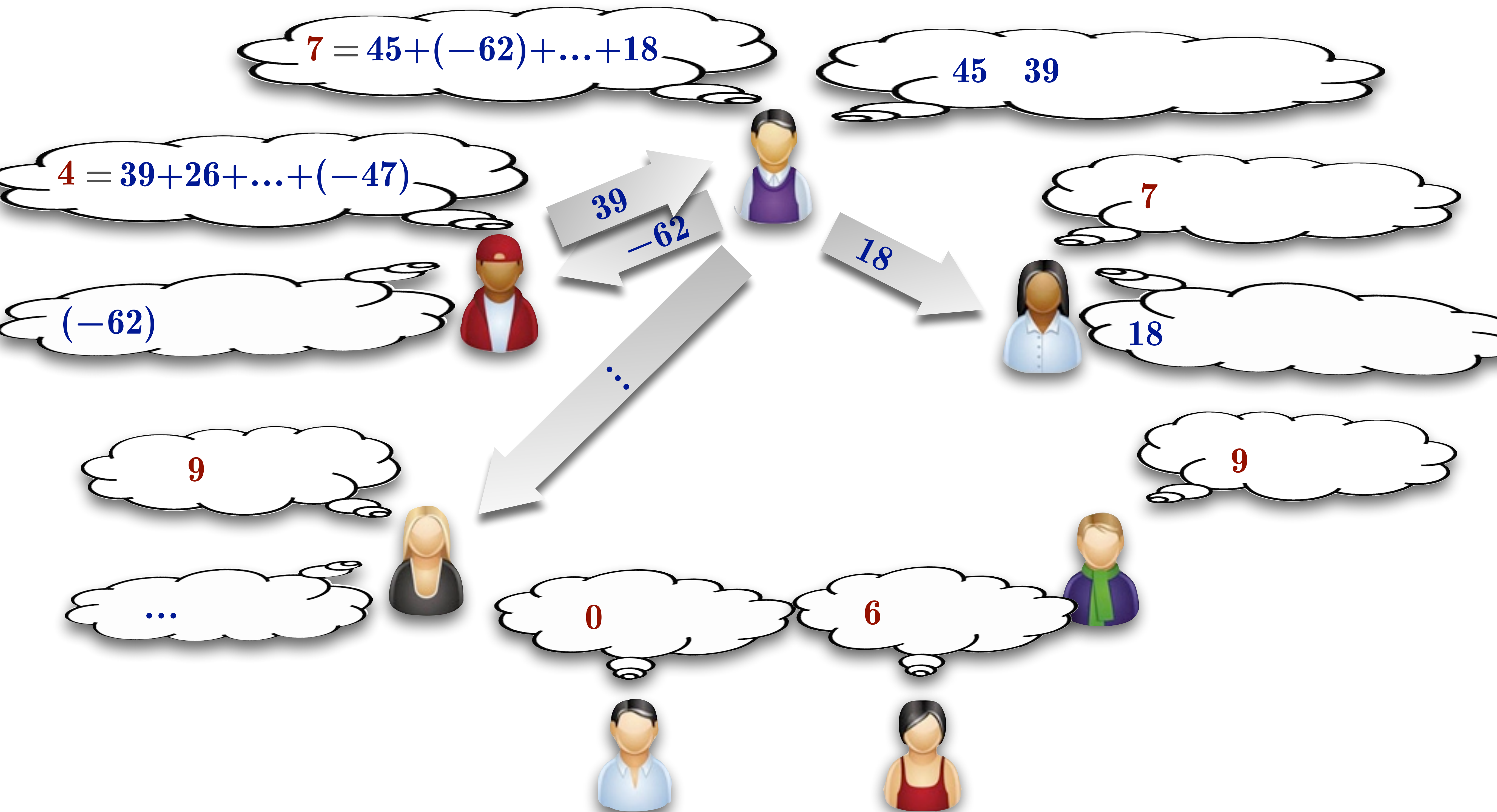
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



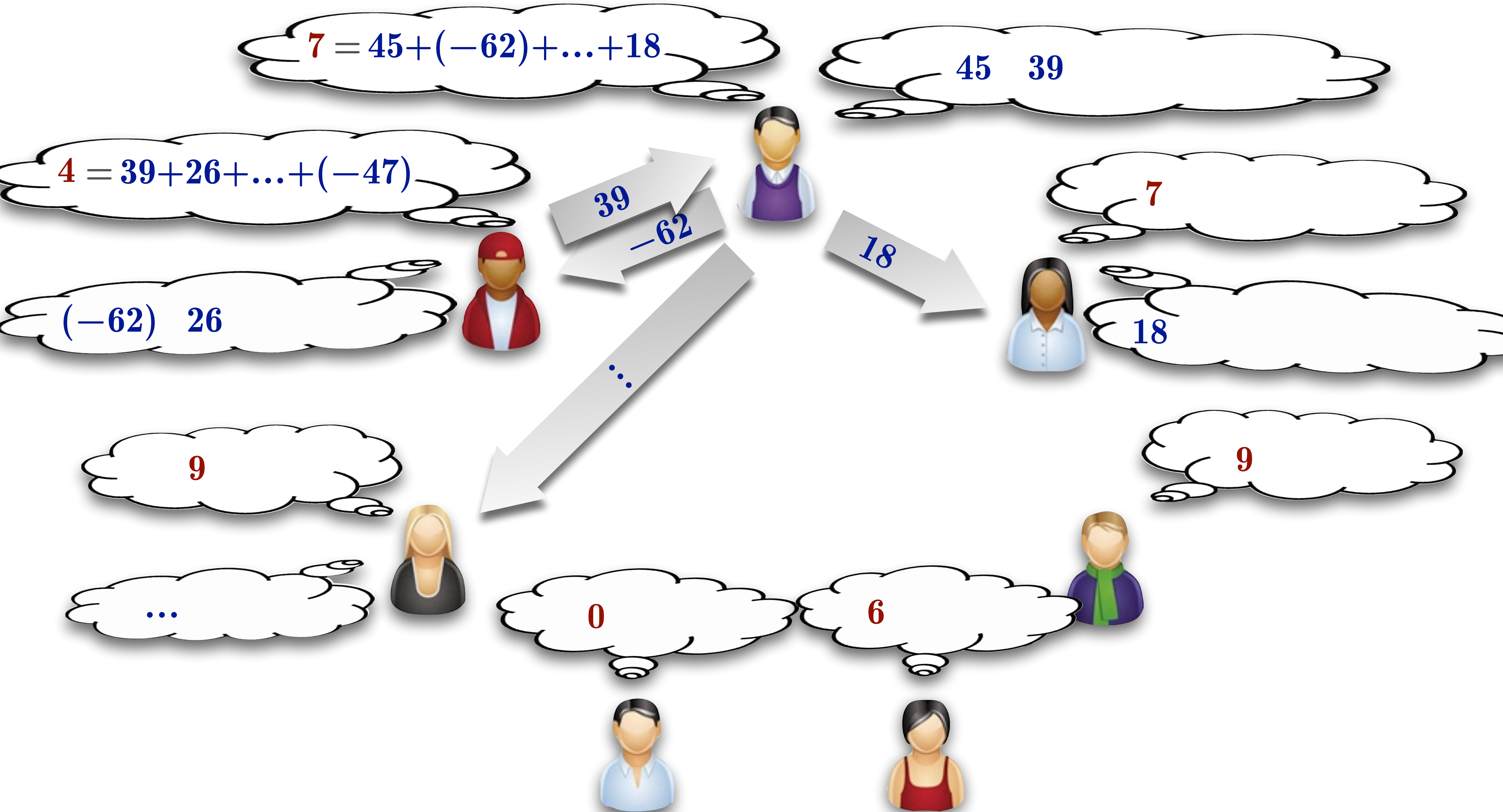
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



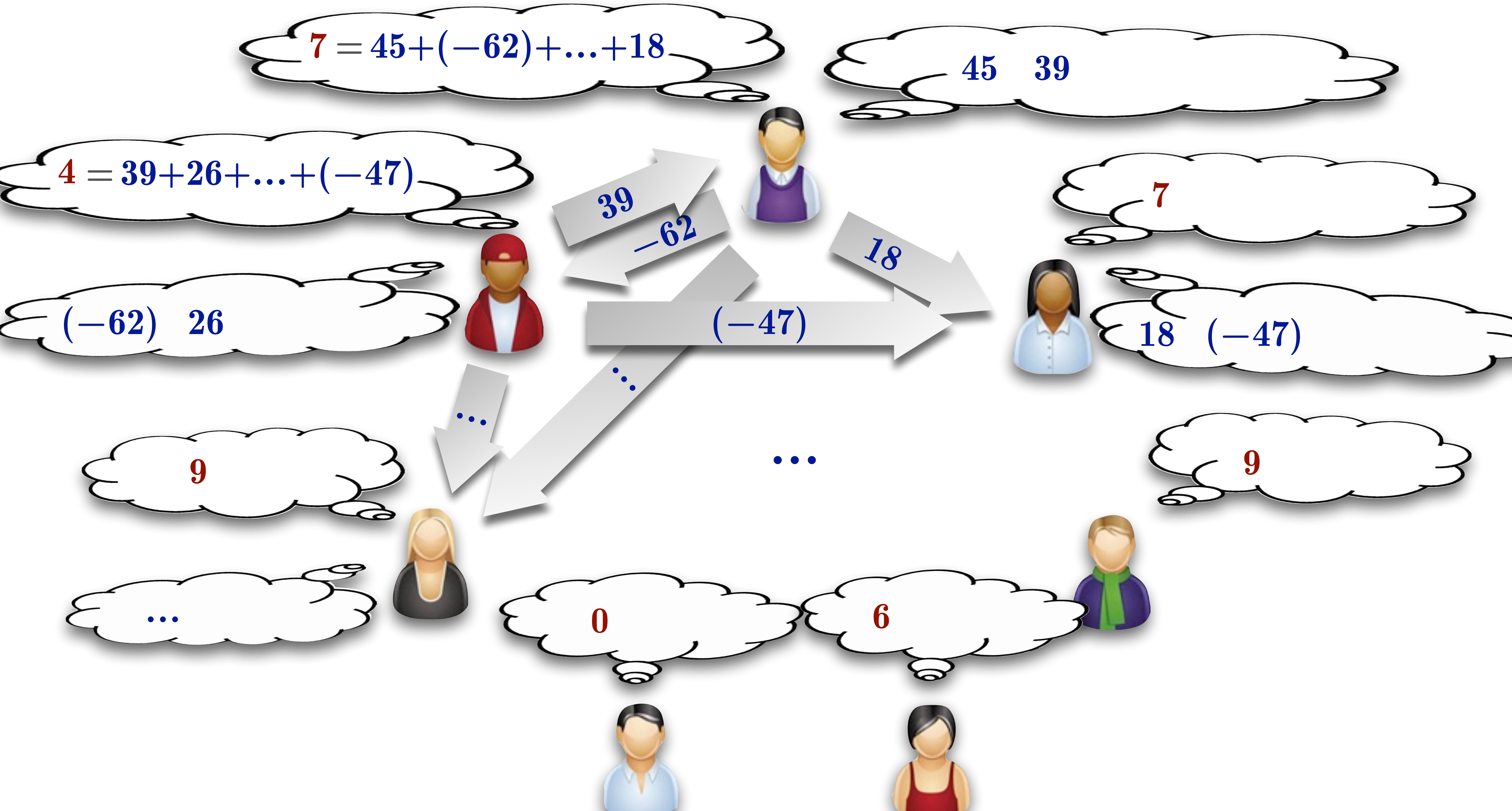
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



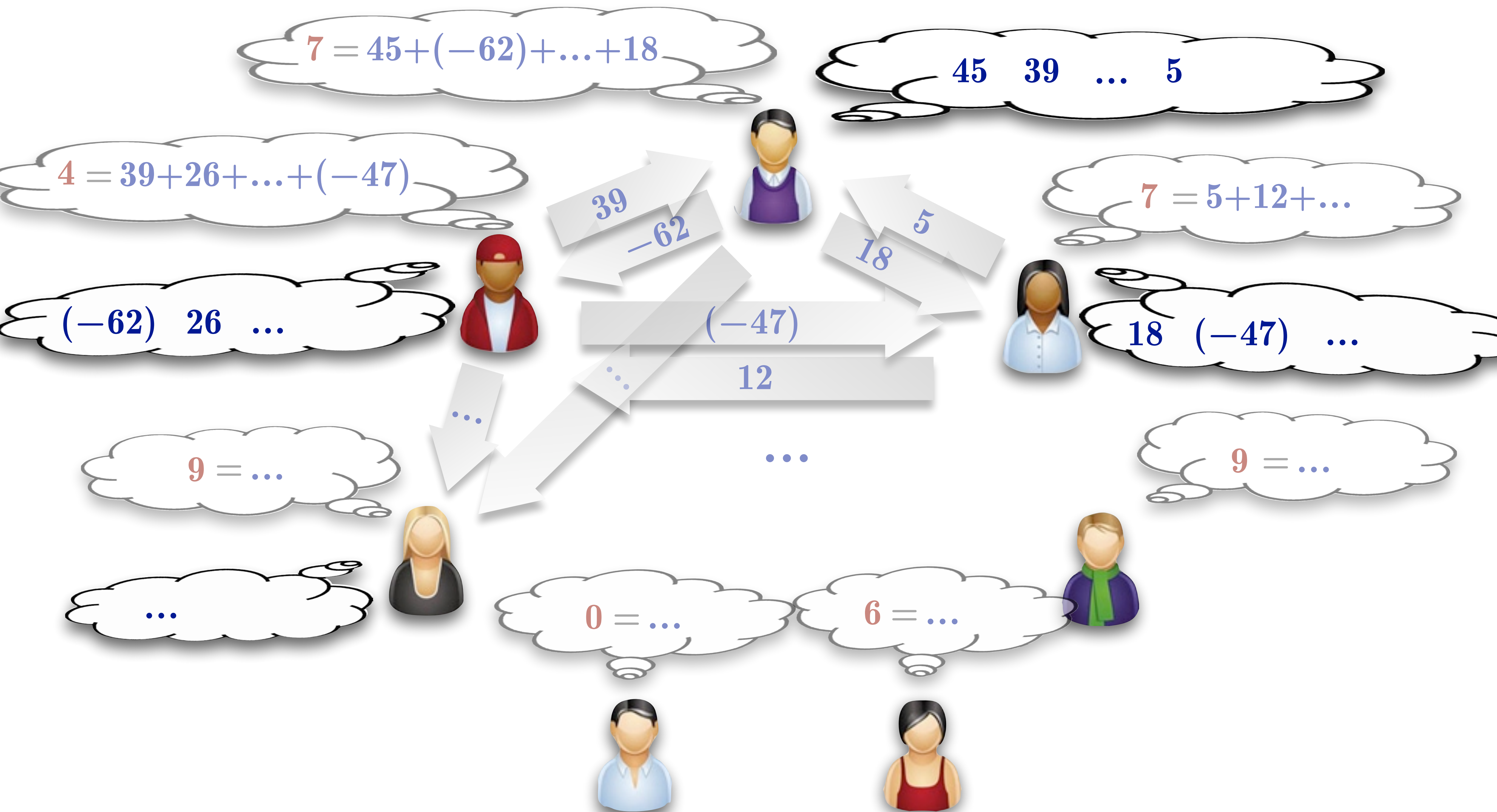
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



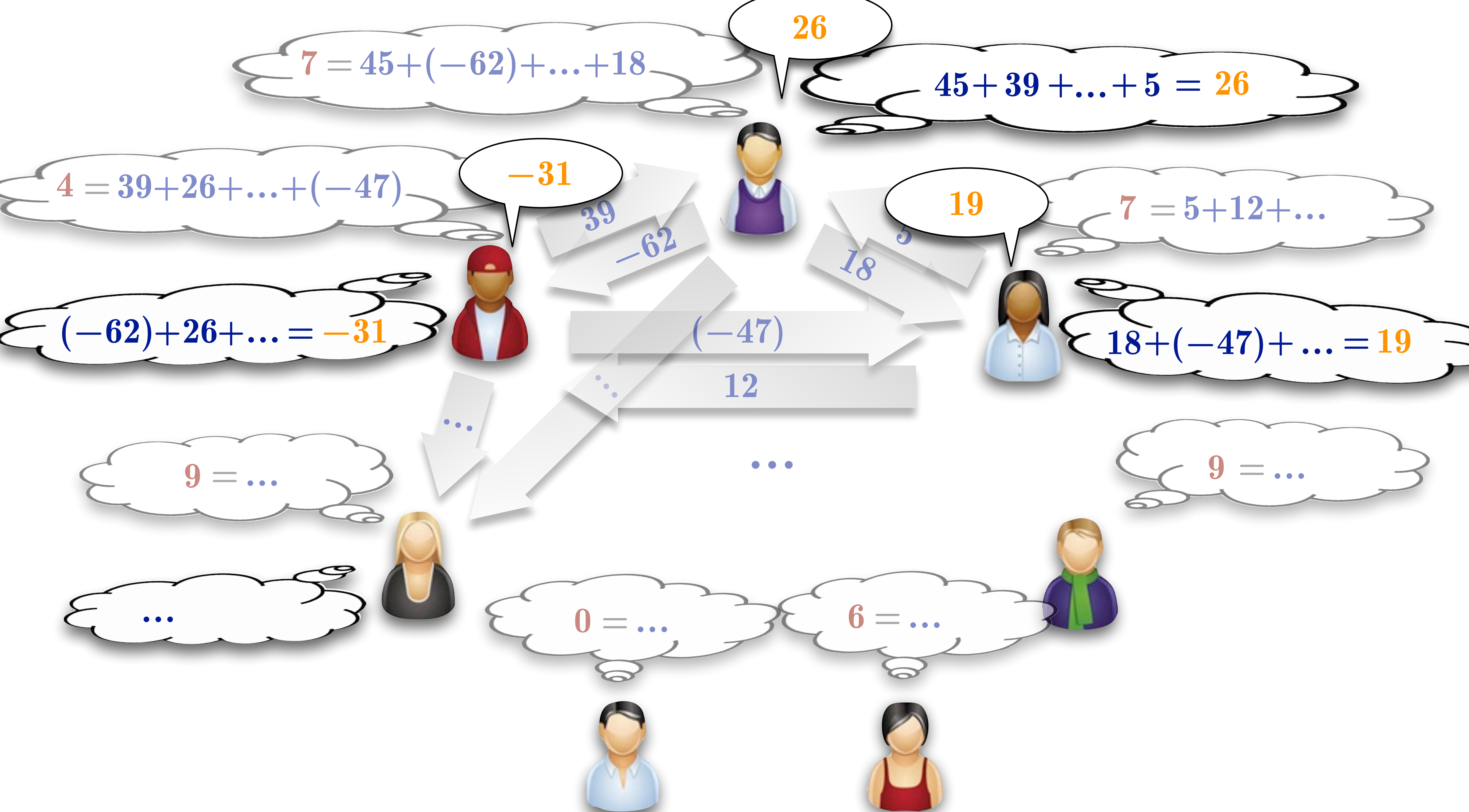
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1, \ldots, x_n) = \sum x_i$



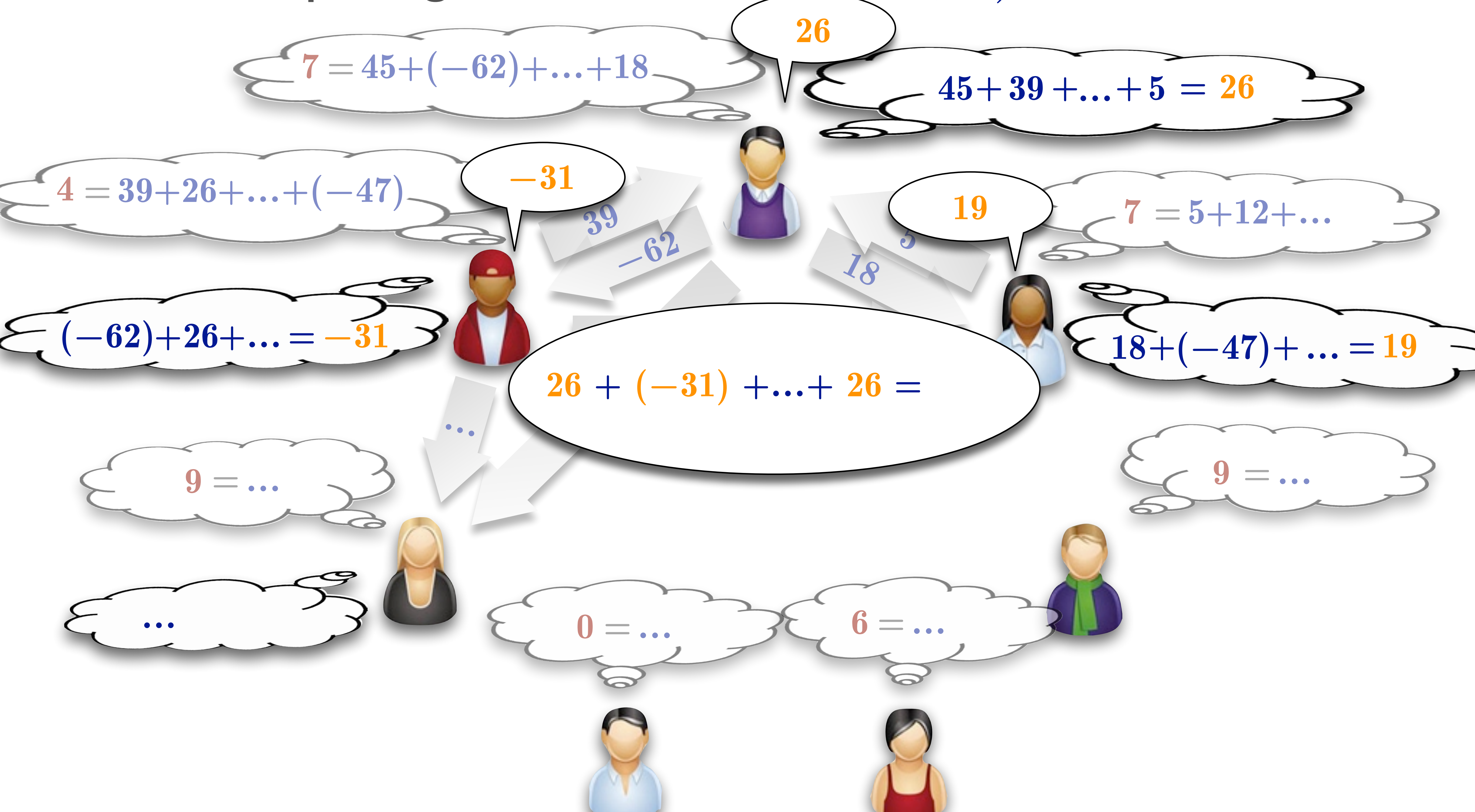
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



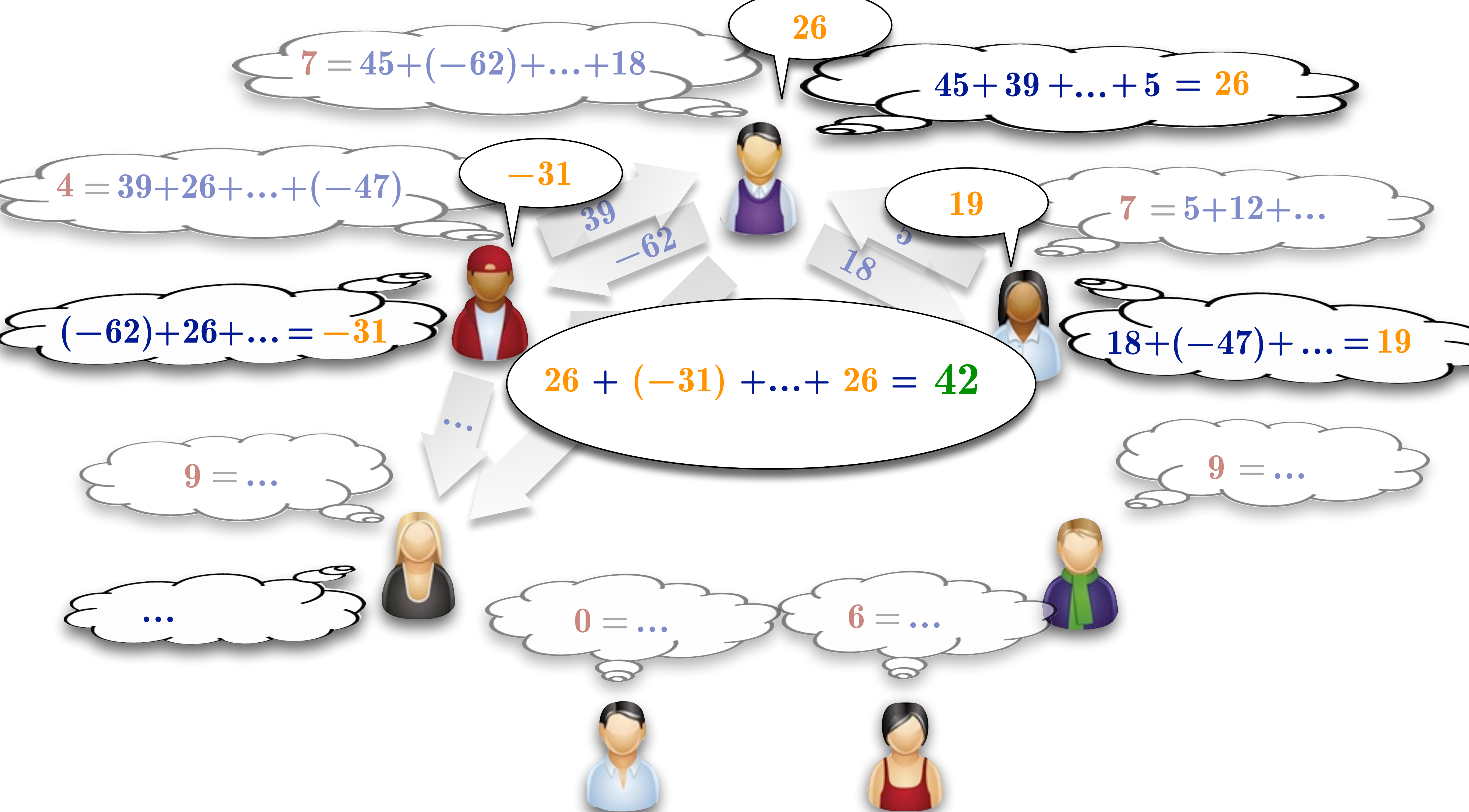
MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$

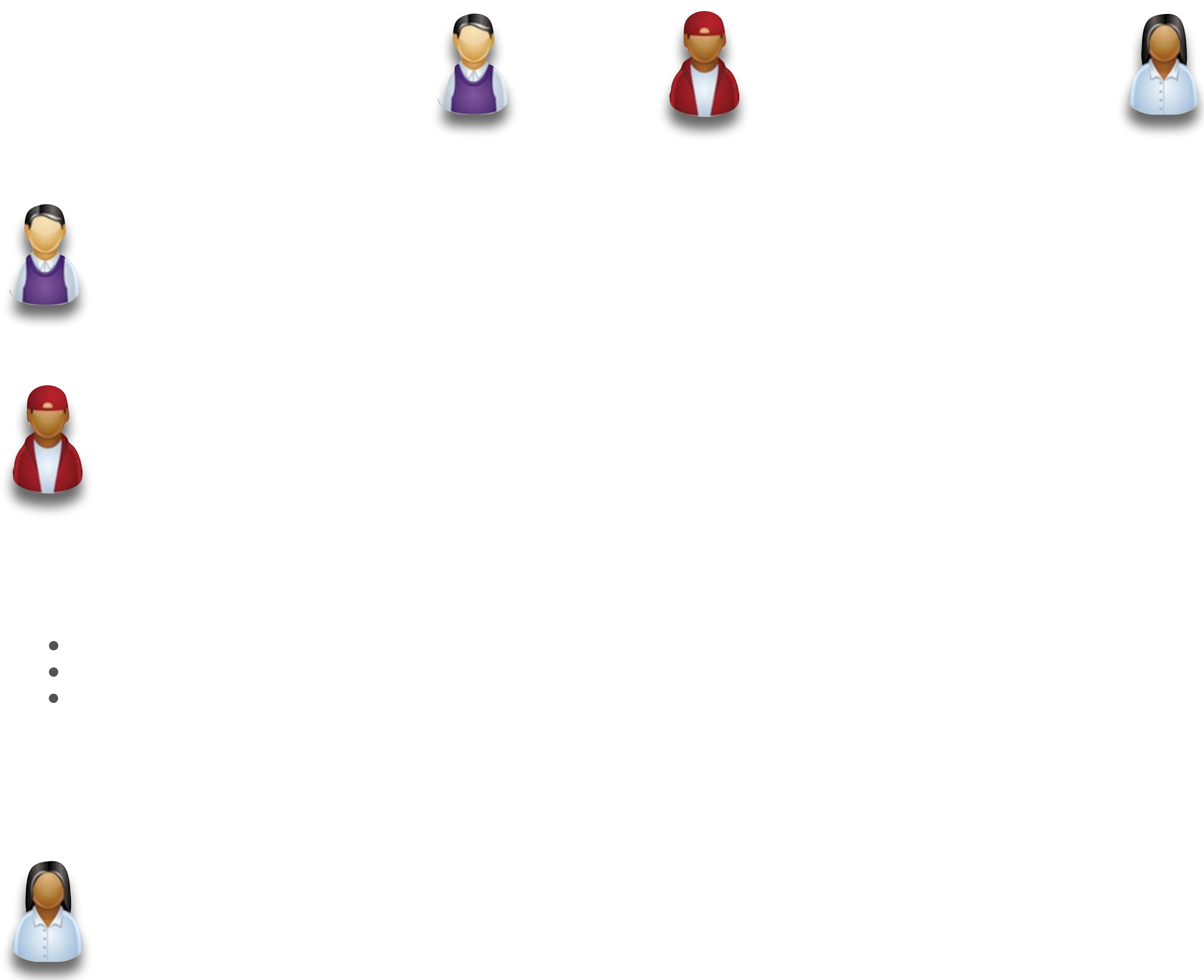


MPC: A second try

Goal: Computing the sum, i.e., $f(x_1,\ldots,x_n) = \sum x_i$



A More Abstract Description



A More Abstract Description

 $$x_1 = x_{11} + x_{12} + \ldots + x_{1n}$$

 $$x_2 = x_{21} + x_{22} + \ldots + x_{2n}$$

$\vdots$ $\qquad$ $\vdots$

 $$x_n = x_{n1} + x_{n2} + \ldots + x_{nn}$$

A More Abstract Description



$$
\begin{array}{lcccccccc}
x_1 & = & x_{11} & + & x_{12} & + & \ldots & + & x_{1n} \\
x_2 & = & x_{21} & + & x_{22} & + & \ldots & + & x_{2n} \\
\vdots & & & & & & & & \\
x_n & = & x_{n1} & + & x_{n2} & + & \ldots & + & x_{nn}
\end{array}
$$

A More Abstract Description

$$
\begin{array}{ccccccccc}
 & & & & & & & & \\
x_1 & = & x_{11} & + & x_{12} & + & \ldots & + & x_{1n} \\
 & & + & & + & & & & + \\
x_2 & = & x_{21} & + & x_{22} & + & \ldots & + & x_{2n} \\
 & & + & & + & & & & + \\
\vdots & & & & & & & & \\
 & & + & & + & & & & + \\
x_n & = & x_{n1} & + & x_{n2} & + & \ldots & + & x_{nn} \\
 & & = & & = & & & & = \\
 & & y_1 & & y_2 & & & & y_n
\end{array}
$$

A More Abstract Description

$$\begin{array}{ccccccccc}
x_1 & = & x_{11} & + & x_{12} & + & \dots & + & x_{1n} \\
+ & & + & & + & & & & + \\
x_2 & = & x_{21} & + & x_{22} & + & \dots & + & x_{2n} \\
+ & & + & & + & & & & + \\
 & \vdots & & & & & & & \\
+ & & + & & + & & & & + \\
x_n & = & x_{n1} & + & x_{n2} & + & \dots & + & x_{nn} \\
= & & = & & = & & & & = \\
y & = & y_1 & + & y_2 & + & \dots & + & y_n
\end{array}$$

A More Abstract Description

$$
\begin{array}{ccccccccc}
x_1 & = & \boxed{x_{11}} & + & \boxed{x_{12}} & + & \ldots & + & \boxed{x_{1n}} \\
+ & & + & & + & & & & + \\
x_2 & = & x_{21} & + & x_{22} & + & & + & x_{2n} \\
\vdots & & \vdots & & & & & & \\
+ & & + & & + & & & & + \\
x_n & = & x_{n1} & + & x_{n2} & + & \ldots & + & x_{nn} \\
= & & = & & = & & & & = \\
y & = & y_1 & + & y_2 & + & \ldots & + & y_n
\end{array}
$$

Offers ***privacy*** of inputs against **arbitrary** coalitions ✔

A More Abstract Description

Parties can lie about their partial result:
→ no ***correctness*** or ***fairness*** guaranteed

$$x_2 = x_{21} + x_{22} + \dots + x_{2n}$$

Offers ***privacy*** of inputs against **arbitrary** coalitions

$$\begin{array}{ccccccccc}
\vdots & & \vdots & & & & & & \\
+ & & + & & + & & & & + \\
x_n & = & x_{n1} & + & x_{n2} & + & \dots & + & x_{nn} \\
= & & = & & = & & & & = \\
y & = & y_1 & + & y_2 & + & \dots & + & y_n
\end{array}$$

A More Abstract Description

Parties can lie about their partial result:
→ **no *correctness* or *fairness* guaranteed**

$$x_2 = x_{21} + x_{22} + \quad + x_{2n}$$

Offers ***privacy*** **of inputs against arbitrary coalitions** ✔

$$\vdots \qquad \vdots$$

Any party can **stall** the procedure → **not *abort-free***

$$\begin{array}{ccccccccc} x_n & = & x_{n1} & + & x_{n2} & + & \dots & + & x_{nn} \\ \| & & \| & & \| & & & & \| \\ y & = & y_1 & + & y_2 & + & \dots & + & y_n \end{array}$$

A More Abstract Description

Parties can lie about their partial result:
→ **no *correctness* or *fairness* guaranteed**

$x_2 = x_{21} + x_{22} + \ldots + x_{2n}$

Offers ***privacy*** of inputs against **arbitrary coalitions**

Any party can **stall** the procedure → **not *abort-free***

$x_n = x_{n1} + x_{n2} + \ldots + x_{nn}$

Approach/solution **limited** to linear functions

$y = y_1 + y_2 + \ldots + y_n$

A Useful Tool: (Linear) Secret Sharing

At the core is a cryptographic primitive for

distributing ("***sharing***") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that:

A Useful Tool: (Linear) Secret Sharing

At the core is a cryptographic primitive for

distributing ("*sharing*") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that:

reconstructability

- from **all** n shares $s_1, \ldots, s_n$, the secret s can be recovered

A Useful Tool: (Linear) Secret Sharing

At the core is a cryptographic primitive for

distributing ("***sharing***") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i ,

so that:

reconstructability

- from **all** n shares $s_1, \ldots, s_n$, the secret s can be recovered
- given **less** than n shares, no info on s is revealed

privacy

A Useful Tool: (Linear) Secret Sharing

At the core is a cryptographic primitive for

distributing (“***sharing***”) a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that:

reconstructability

- from **all** n shares $s_1, \ldots, s_n$, the secret s can be recovered
- given **less** than n shares, no info on s is revealed *privacy*
- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

  $s_1 + s'_1, \ldots, s_n + s'_n$ is a sharing of $s + s'$. *linearity*

A Useful Tool: (Linear) Secret Sharing

At the core is a cryptographic primitive for

distributing ("*sharing*") a secret input s

Prime example:

$s_1, \ldots, s_n$ random subject to $s = s_1 + \ldots + s_n \pmod{p}$

reconstructability

- from **all** n shares $s_1, \ldots, s_n$, the secret s can be recovered
- given **less** than n shares, no info on s is revealed

privacy

- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

  $s_1 + s'_1, \ldots, s_n + s'_n$ is a sharing of $s + s'$.

linearity

A Paradigm for Doing MPC

Sharing phase:

Computation phase:

Reconstruction phase:

A Paradigm for Doing MPC

Sharing phase:

Every party P_i **shares** his input x_i .

Computation phase:

Reconstruction phase:

A Paradigm for Doing MPC

Sharing phase:

Every party P_i **shares** his input x_i .

Computation phase:

The function f is computed on the **shared** inputs, resulting in a sharing of $y = f(x_1, \ldots, x_n)$.

Reconstruction phase:

A Paradigm for Doing MPC

Sharing phase:

Every party P_i **shares** his input x_i .

Computation phase:

The function f is computed **on the shared inputs**, resulting in a **sharing** of $y = f(x_1, \ldots, x_n)$.

So far: only know how to do for linear f.

Reconstruction phase:

A Paradigm for Doing MPC

Sharing phase:
Every party P_i **shares** his input x_i .

Computation phase:

The function f is computed on the **shared** inputs, resulting in a sharing of $y = f(x_1, \ldots, x_n)$.

So far: only know how to do for linear f.

Reconstruction phase:

The share result $y = f(x_1, \ldots, x_n)$ is reconstructed.

A Paradigm for Doing MPC

Sharing phase:

Every party P_i **shares** his input x_i .

Computation phase:

The function f is computed **on the shared inputs**, resulting in a **sharing** of $y = f(x_1, \ldots, x_n)$.

So far: only know how to do for linear f.

Reconstruction phase:

The share result $y = f(x_1, \ldots, x_n)$ is reconstructed.

Still some issues about dishonest parties lying.

Threshold Secret Sharing

At the core is a cryptographic primitive for

distributing ("***sharing***") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that

reconstructability

- from **all** n shares, the secret s can be recovered
- given **less** than n shares, no info on s is revealed

privacy

- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

$s_1 + s'_1, \ldots, s_n + s'_n$ is a sharing of $s + s'$.

linearity

Threshold Secret Sharing

At the core is a cryptographic primitive for

distributing ("***sharing***") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that for some t

reconstructability

- from **all** n shares, the secret s can be recovered
- given **less** than n shares, no info on s is revealed

privacy

- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

$s_1 + s'_1, \ldots, s_n + s'_n$ is a sharing of $s + s'$.

linearity

Threshold Secret Sharing

At the core is a cryptographic primitive for

distributing ("*sharing*") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that for some t

- from ~~**all** n~~ **any** $t+1$ shares, the secret s can be recovered

reconstructuctability

- given **less** than n shares, no info on s is revealed

privacy

- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

$s_1+s'_1, \ldots, s_n+s'_n$ is a sharing of $s+s'$.

linearity

Threshold Secret Sharing

At the core is a cryptographic primitive for

distributing ("***sharing***") a secret input s

by means of

preparing ***shares*** $s_1, s_2, \ldots, s_n$ and giving s_i to party P_i,

so that for some t

reconstructability

- from ~~**all** n~~ **any** $t+1$ shares, the secret s can be recovered

- given ~~**less** than n~~ **at most** t shares, no info on s is revealed *privacy*

- if $s_1, \ldots, s_n$ is a sharing of s, and $s'_1, \ldots, s'_n$ of s' then

  $s_1+s'_1, \ldots, s_n+s'_n$ is a sharing of $s+s'$. *linearity*

Example: Shamir Secret Sharing

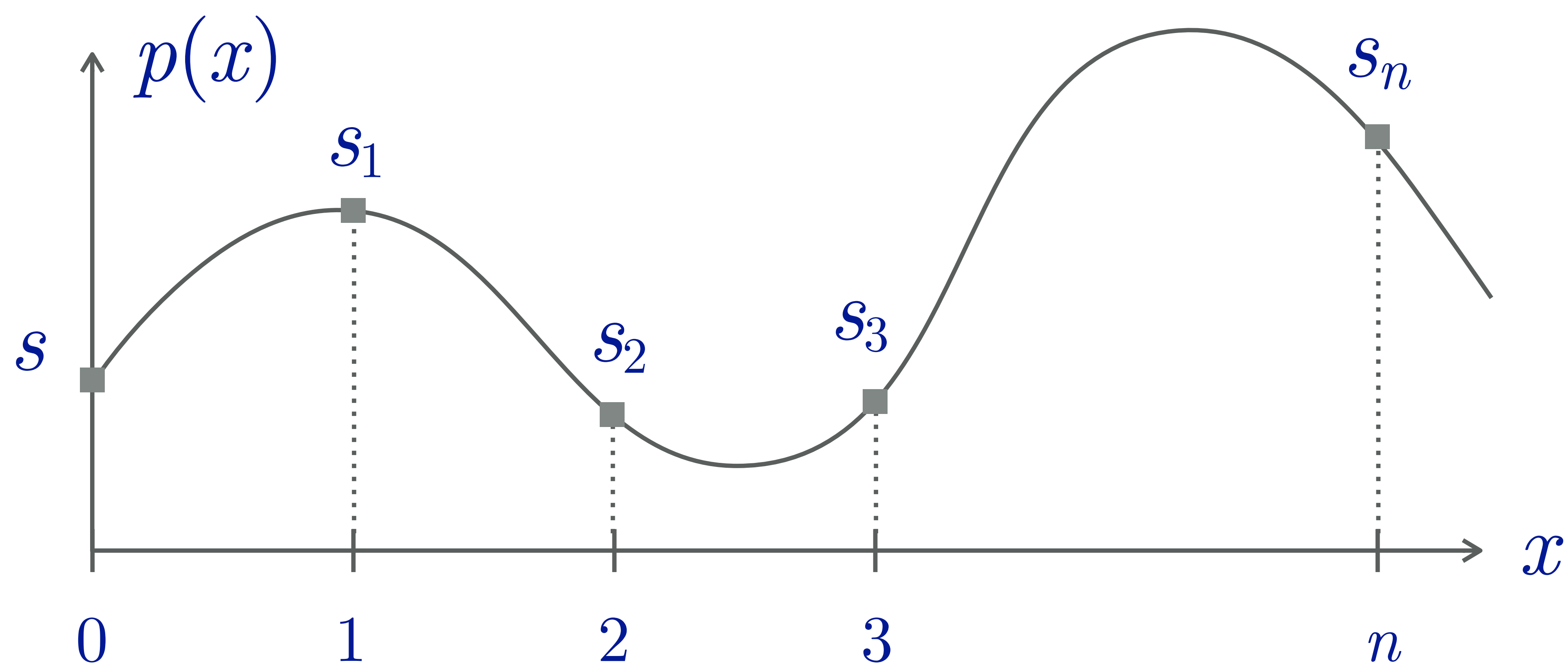
To share s: choose a polynomial

$$p(x) = s + a_1 x + \ldots + a_t x^t$$

with ***random*** $a_1, \ldots, a_t$ and constant coefficient s, and set

$$s_i = p(i)$$

for $i = 1, \ldots, n$.



Example: Shamir Secret Sharing

To share s: choose a polynomial

$$p(x) = s + a_1 x + \ldots + a_t x^t$$

with ***random*** $a_1, \ldots, a_t$ and constant coefficient s, and set

$$s_i = p(i)$$

for $i = 1, \ldots, n$.

Reconstructability & privacy hold by ***Lagrange interpolation***

Example: Shamir Secret Sharing

To share s: choose a polynomial

$$p(x) = s + a_1 x + \ldots + a_t x^t$$

with ***random*** $a_1, \ldots, a_t$ and constant coefficient s, and set

$$s_i = p(i)$$

for $i = 1, \ldots, n$.

Reconstructability & privacy hold by ***Lagrange interpolation***

As for linearity: if

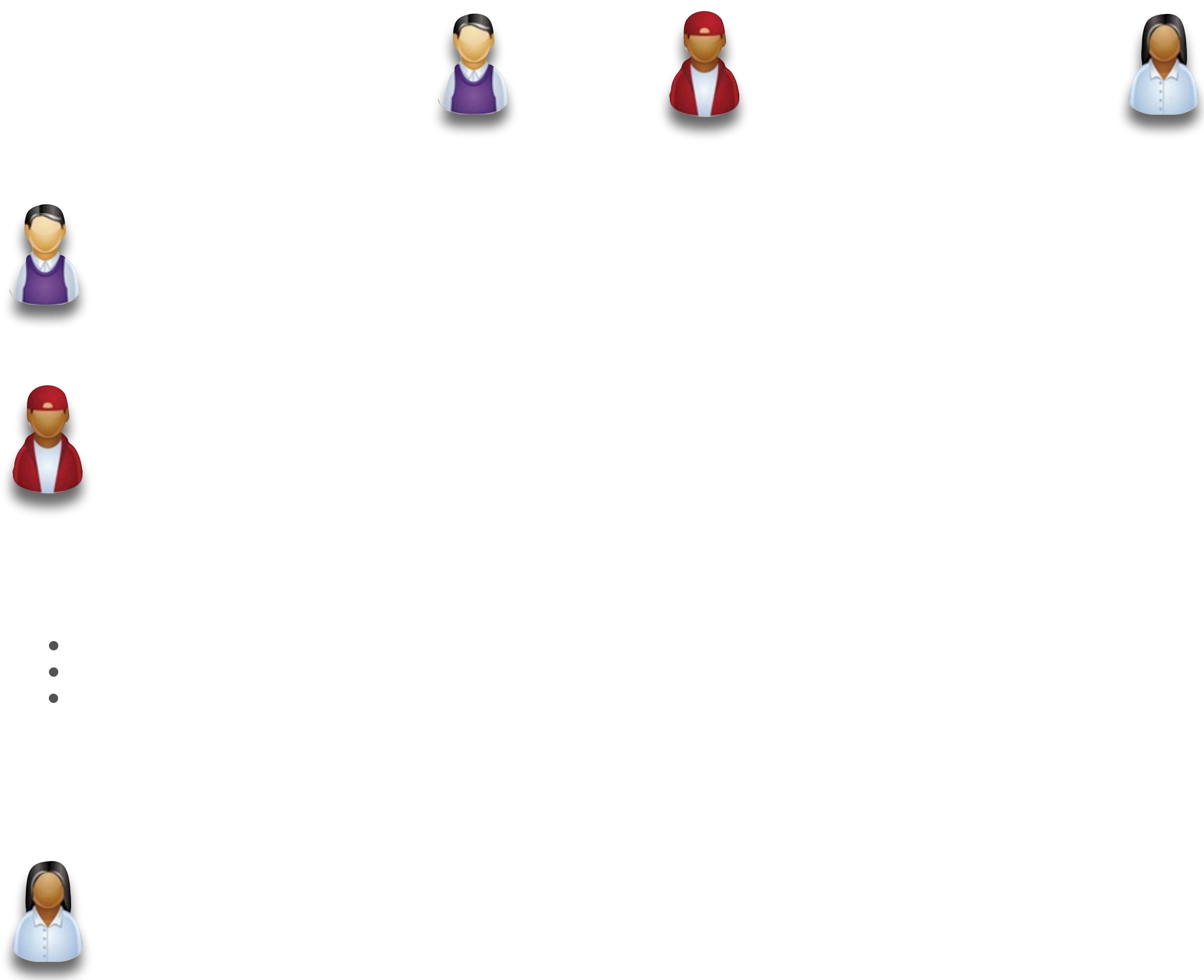
$$s_i = p(i) \text{ for } p(x) = s + a_1 x + \ldots + a_t x^t$$

$$s'_i = p'(i) \text{ for } p'(x) = s' + a'_1 x + \ldots + a'_t x^t$$

then

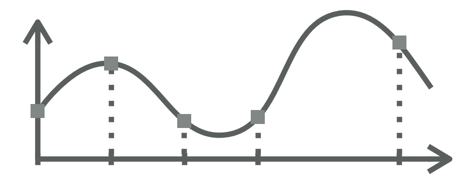
$$s_i + s'_i = p''(i) \text{ for } p''(x) = p(x) + p'(x) = (s + s') + \ldots .$$

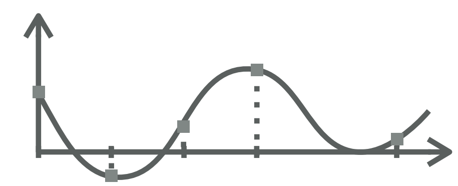
Using Shamir's Secret Sharing Scheme



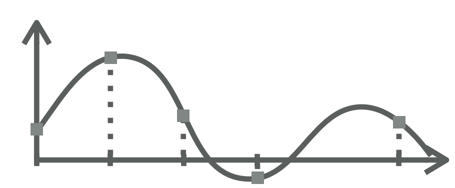
Using Shamir's Secret Sharing Scheme

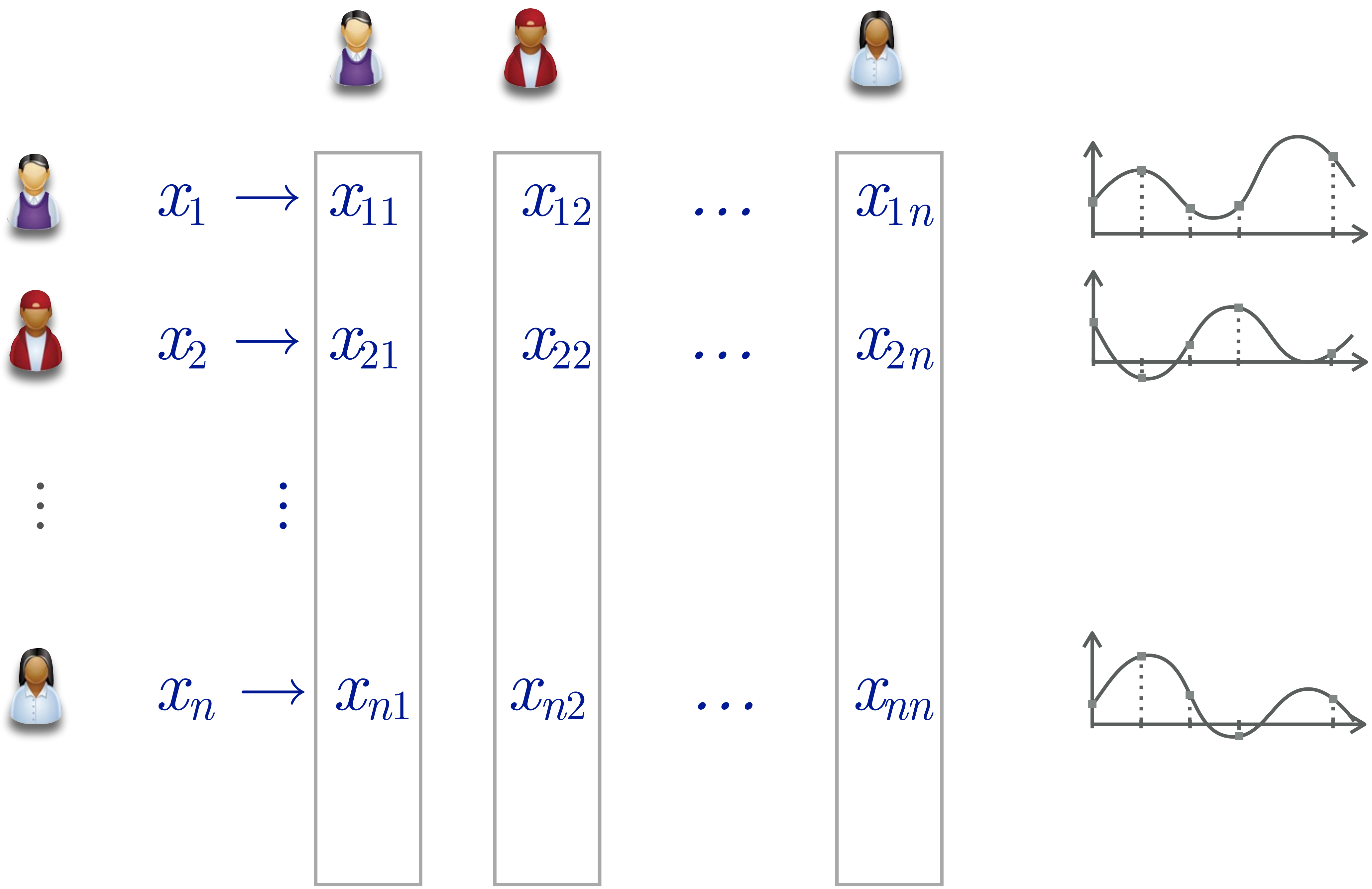
 $x_1 \longrightarrow x_{11} \quad x_{12} \quad \ldots \quad x_{1n}$ 

 $x_2 \longrightarrow x_{21} \quad x_{22} \quad \ldots \quad x_{2n}$ 

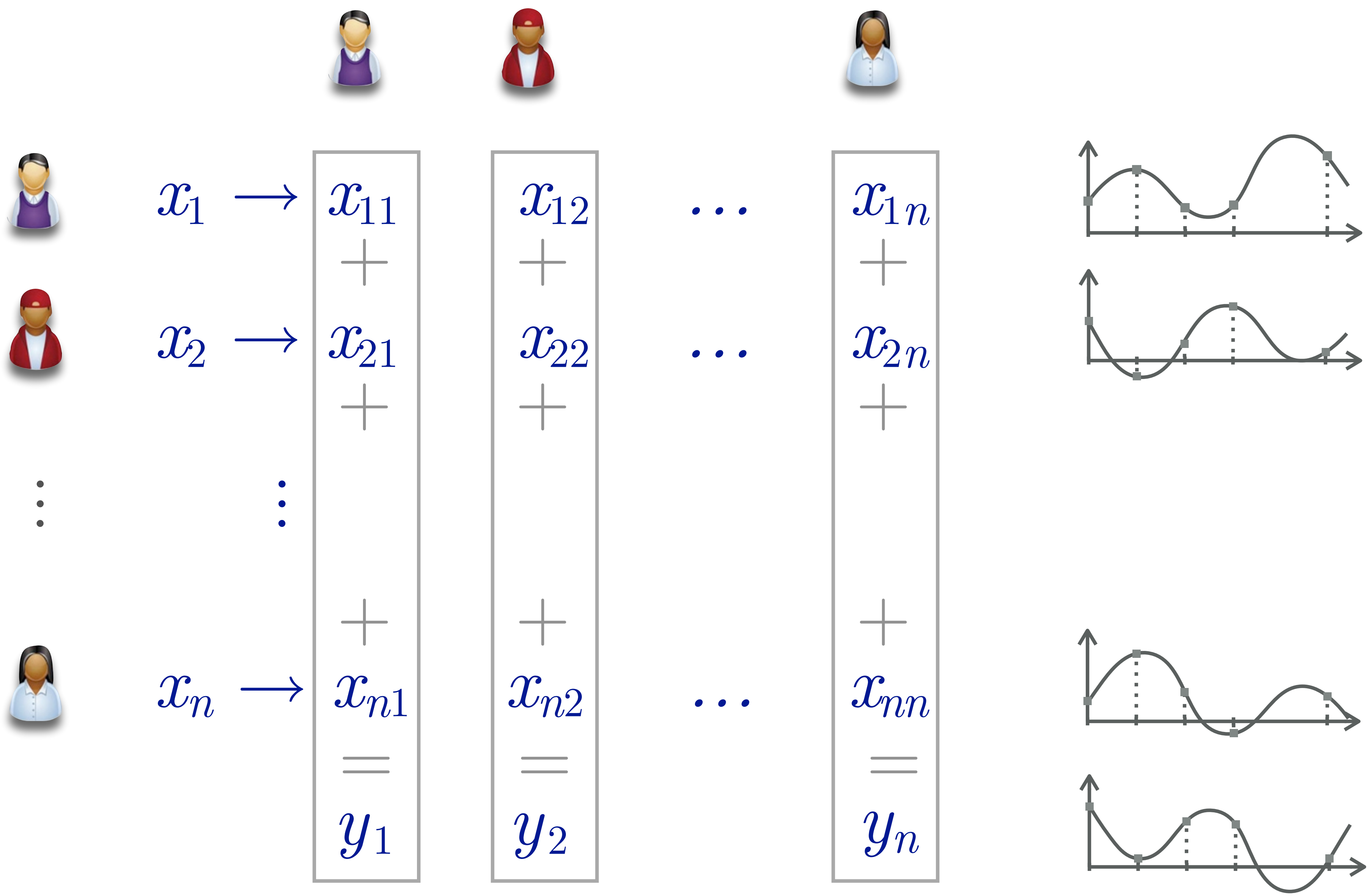
$\vdots \qquad \vdots$

 $x_n \longrightarrow x_{n1} \quad x_{n2} \quad \ldots \quad x_{nn}$ 

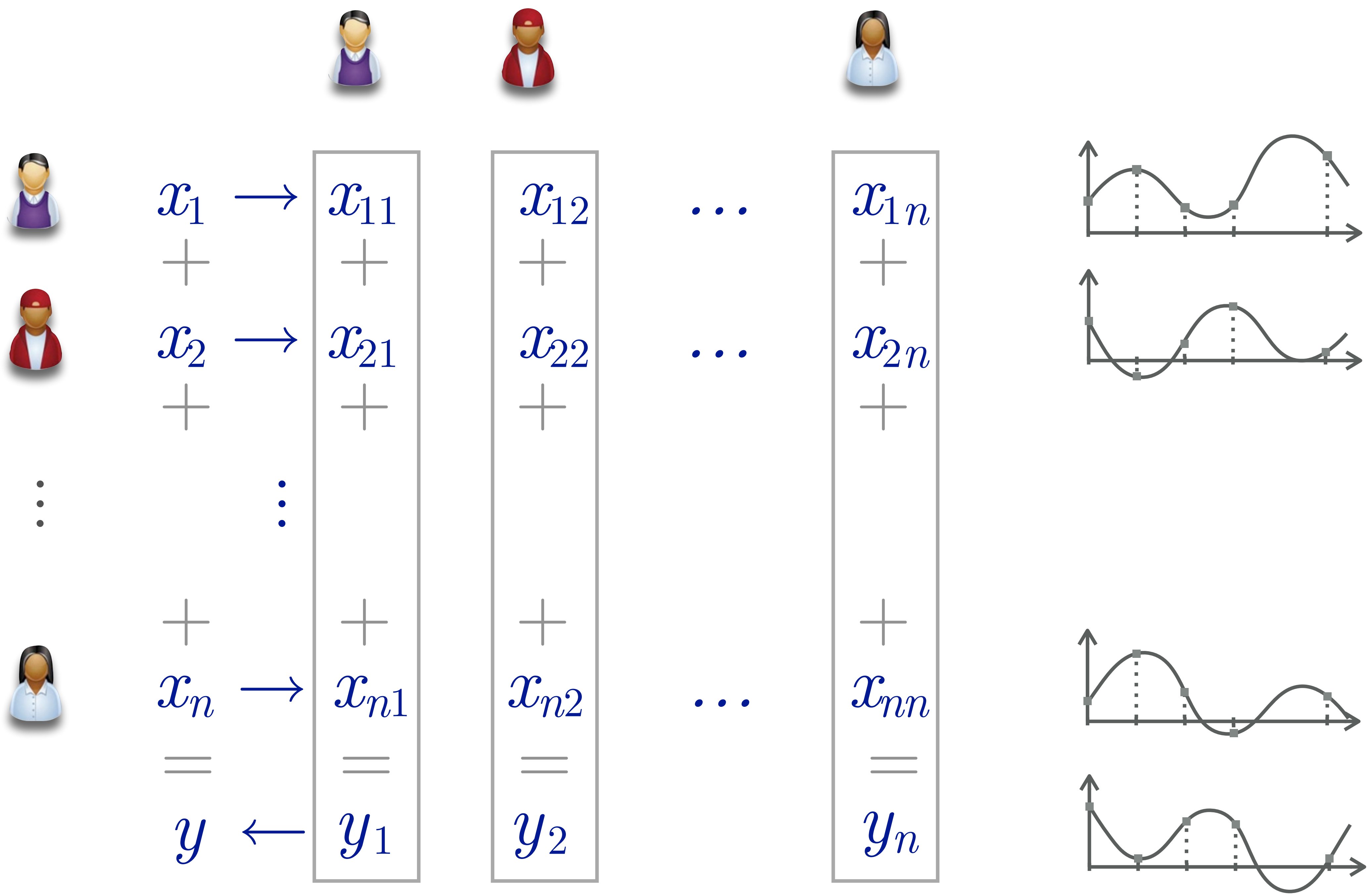
Using Shamir's Secret Sharing Scheme



Using Shamir's Secret Sharing Scheme



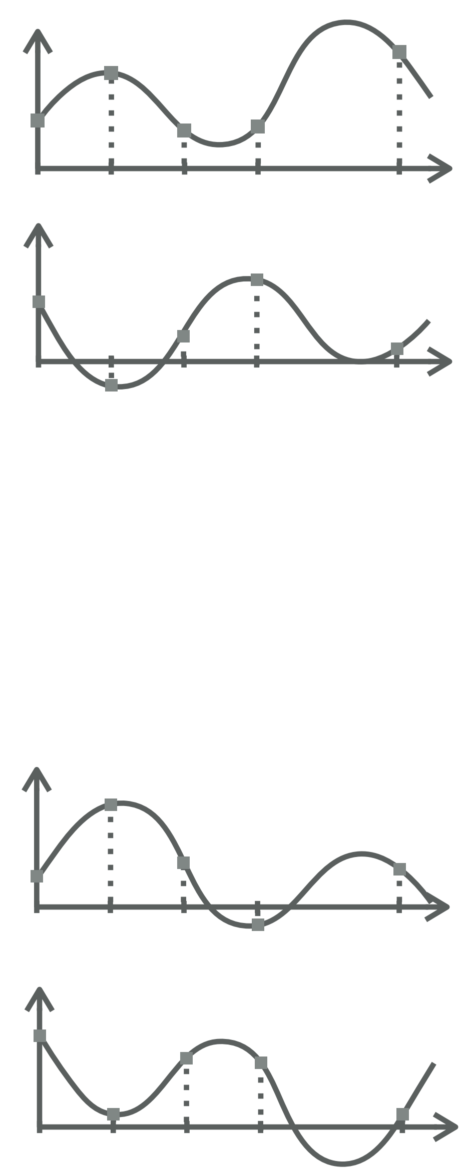
Using Shamir's Secret Sharing Scheme



Using Shamir's Secret Sharing Scheme

Offers *privacy* of inputs against t dishonest parties ✔

$$
\begin{array}{ccccccccc}
x_1 & \longrightarrow & x_{11} & & x_{12} & & \cdots & & x_{1n} \\
+ & & + & & + & & & & + \\
x_2 & \longrightarrow & x_{21} & & x_{22} & & \cdots & & x_{2n} \\
+ & & + & & + & & & & + \\
\vdots & & \vdots & & & & & & \\
+ & & + & & + & & & & + \\
x_n & \longrightarrow & x_{n1} & & x_{n2} & & \cdots & & x_{nn} \\
= & & = & & = & & & & = \\
y & \longleftarrow & y_1 & & y_2 & & & & y_n
\end{array}
$$



Using Shamir's Secret Sharing Scheme

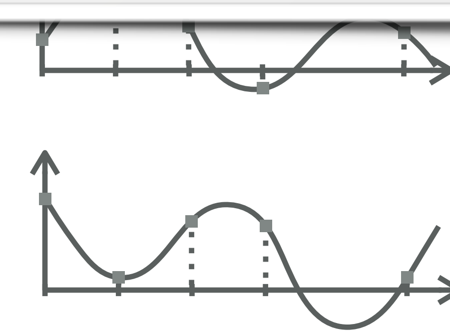
Offers ***privacy*** of inputs **against** t **dishonest parties**

Redundancy in shares ($y_1, \ldots, y_n$ must lie on deg-t poly):

→ cheating will be detected

→ *correctness* (but not *abort-free* nor *fair*)

$$
\begin{array}{ccccccc}
x_n & \longrightarrow & x_{n1} & x_{n2} & \cdots & x_{nn} \\
= & & = & = & & = \\
y & \longleftarrow & y_1 & y_2 & & y_n
\end{array}
$$



Using Shamir's Secret Sharing Scheme

Offers ***privacy*** of inputs **against** t **dishonest parties**

Redundancy in shares ($y_1, \ldots, y_n$ must lie on deg-t poly):

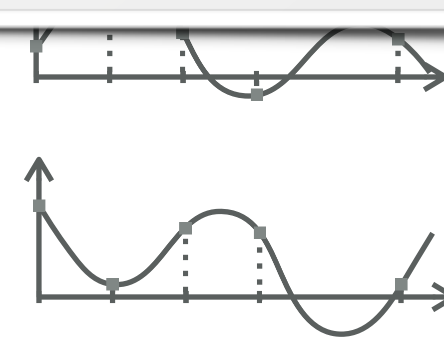
→ cheating will be detected

→ ***correctness*** (but not *abort-free* nor *fair*)

If we can enforce **consistent sharings** (**we can!**) of x_i's, set $t < n/3$, and use Reed-Solomon error correction:

→ ***correctness*** (with guaranteed output delivery)

$$\begin{array}{ccccccc} x_n & \to & x_{n1} & x_{n2} & \cdots & x_{nn} \\ = & & = & = & & = \\ y & \leftarrow & y_1 & y_2 & & y_n \end{array}$$



Using Shamir's Secret Sharing Scheme

Offers ***privacy*** of inputs **against** t **dishonest parties**

Redundancy in shares ($y_1, \ldots, y_n$ must lie on deg-t poly):

→ cheating will be detected

→ ***correctness*** (but not *abort-free* nor *fair*)

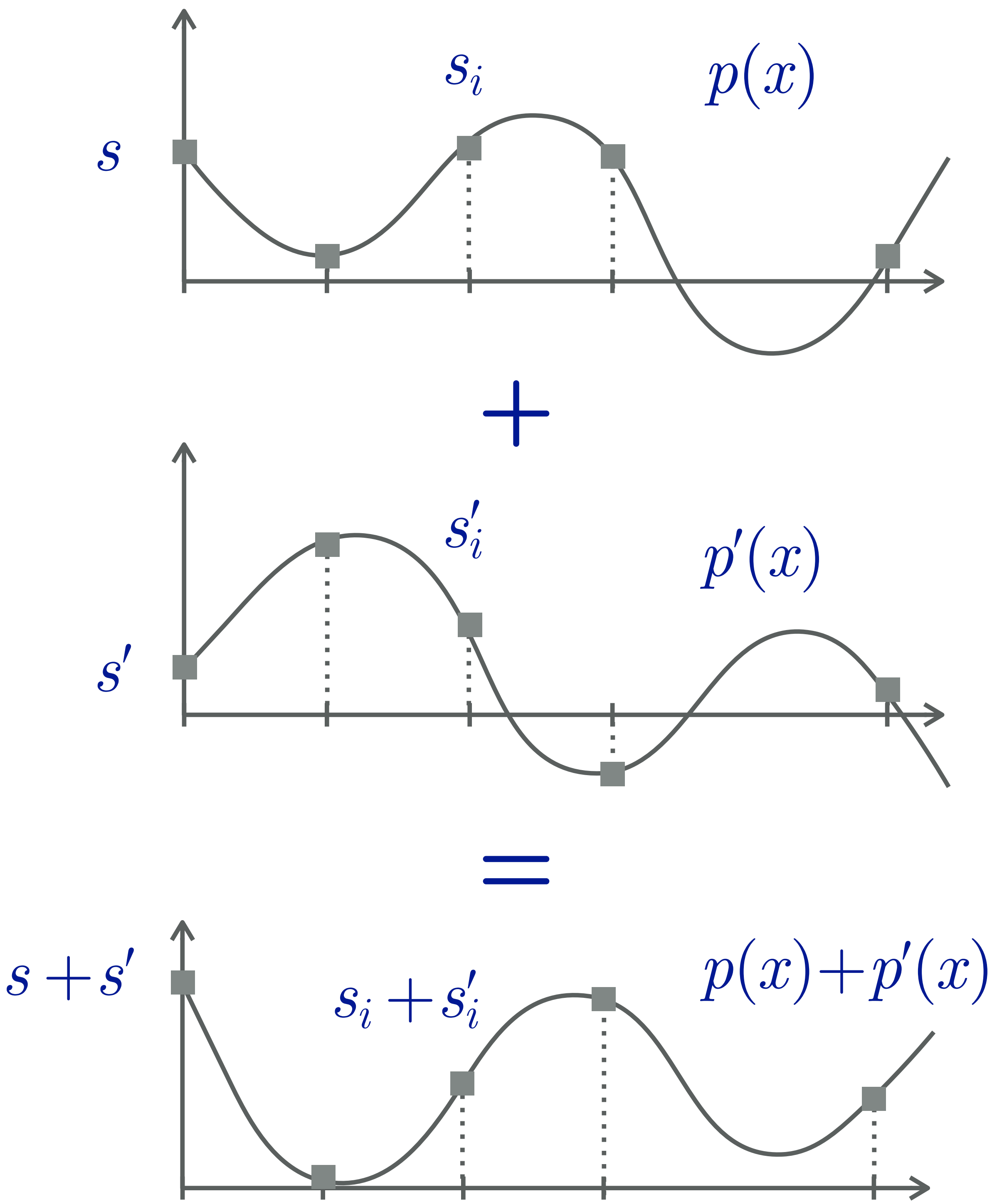
If we can enforce **consistent sharings** (**we can!**) of x_i's, set $t < n/3$, and use Reed-Solomon error correction:

→ ***correctness*** (with guaranteed output delivery)

⇒ Works for ***addition*** / ***linear function evaluation*** only

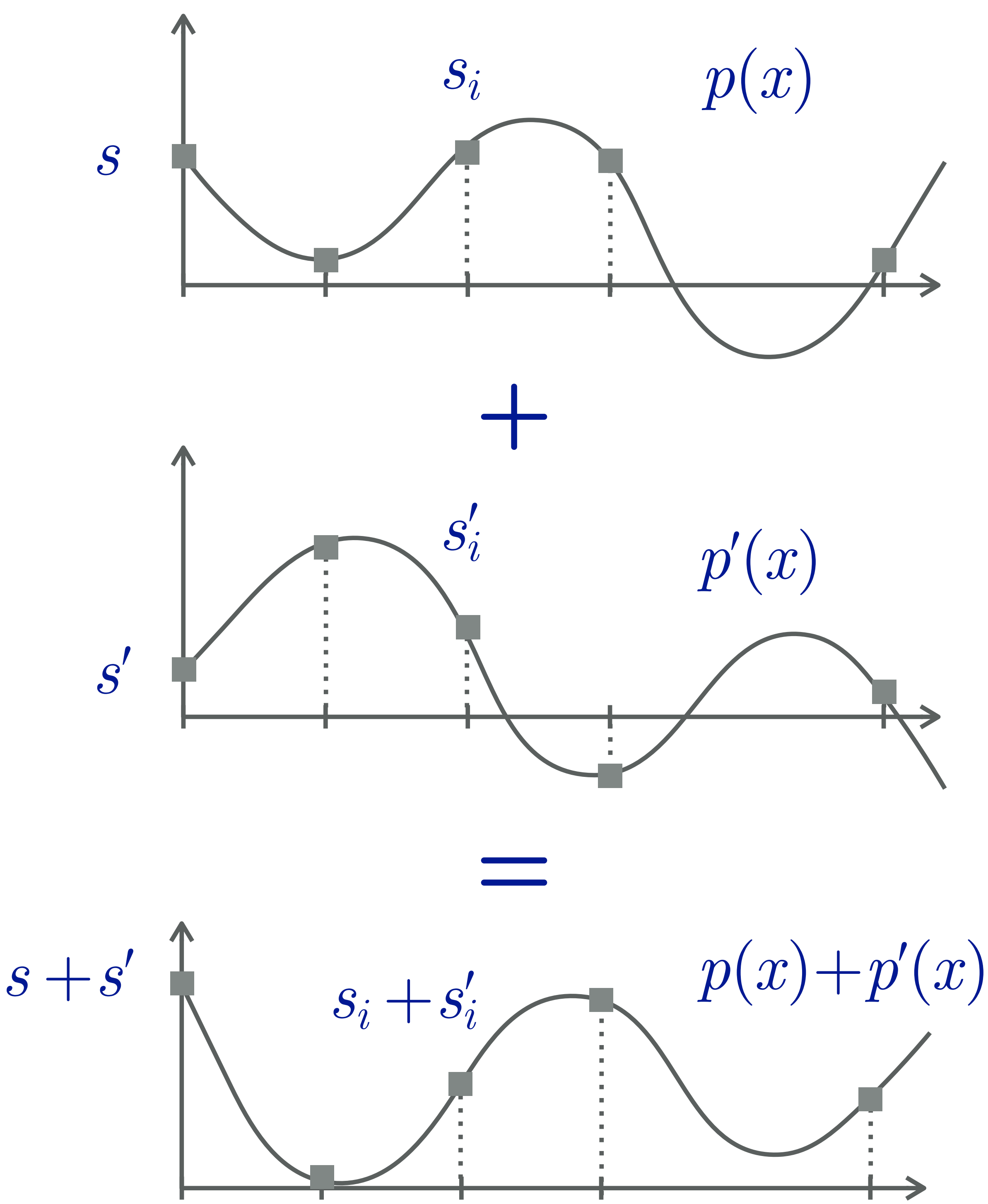
Towards Secure Multiplications

For addition, exploited:

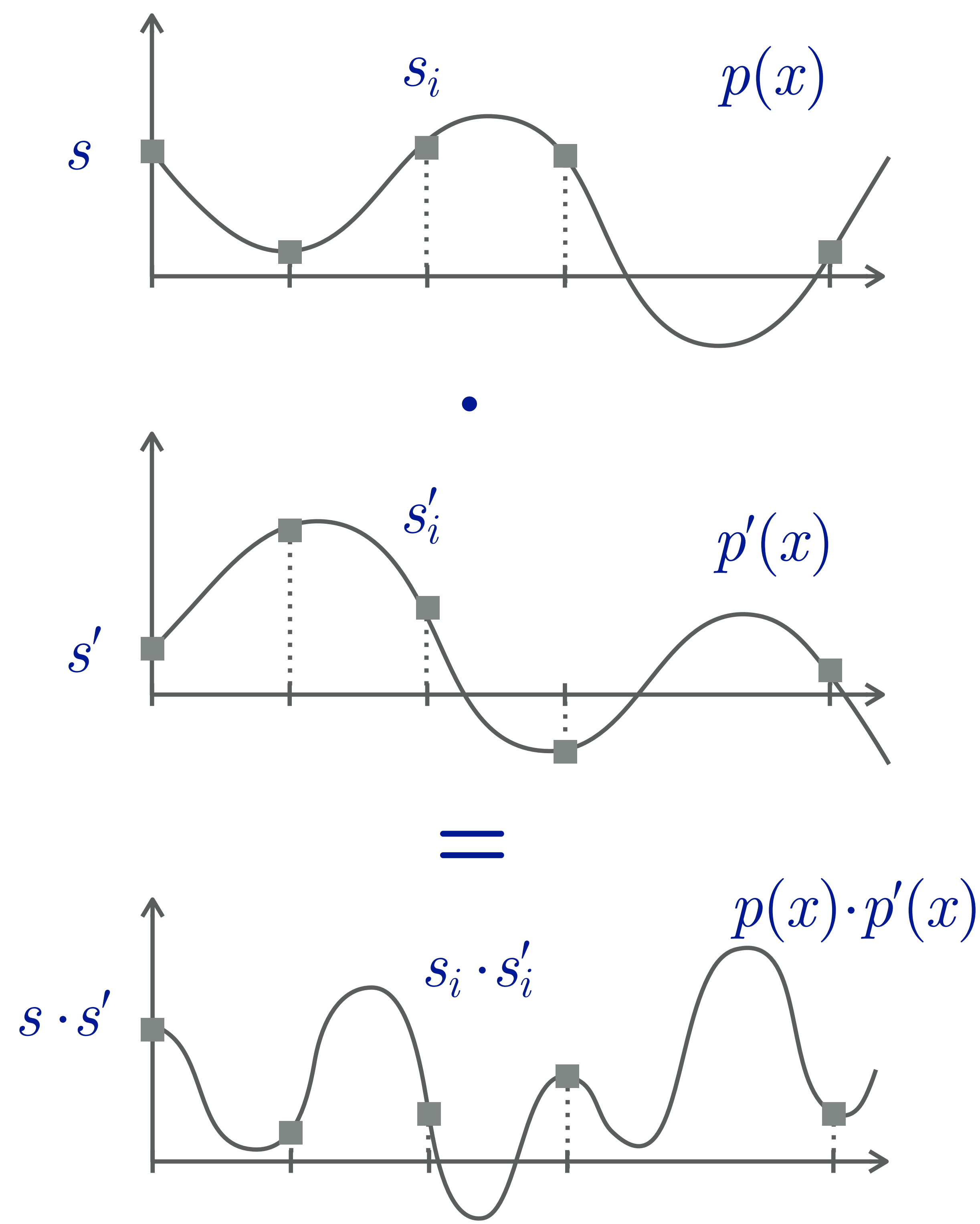


Towards Secure Multiplications

For addition, exploited:

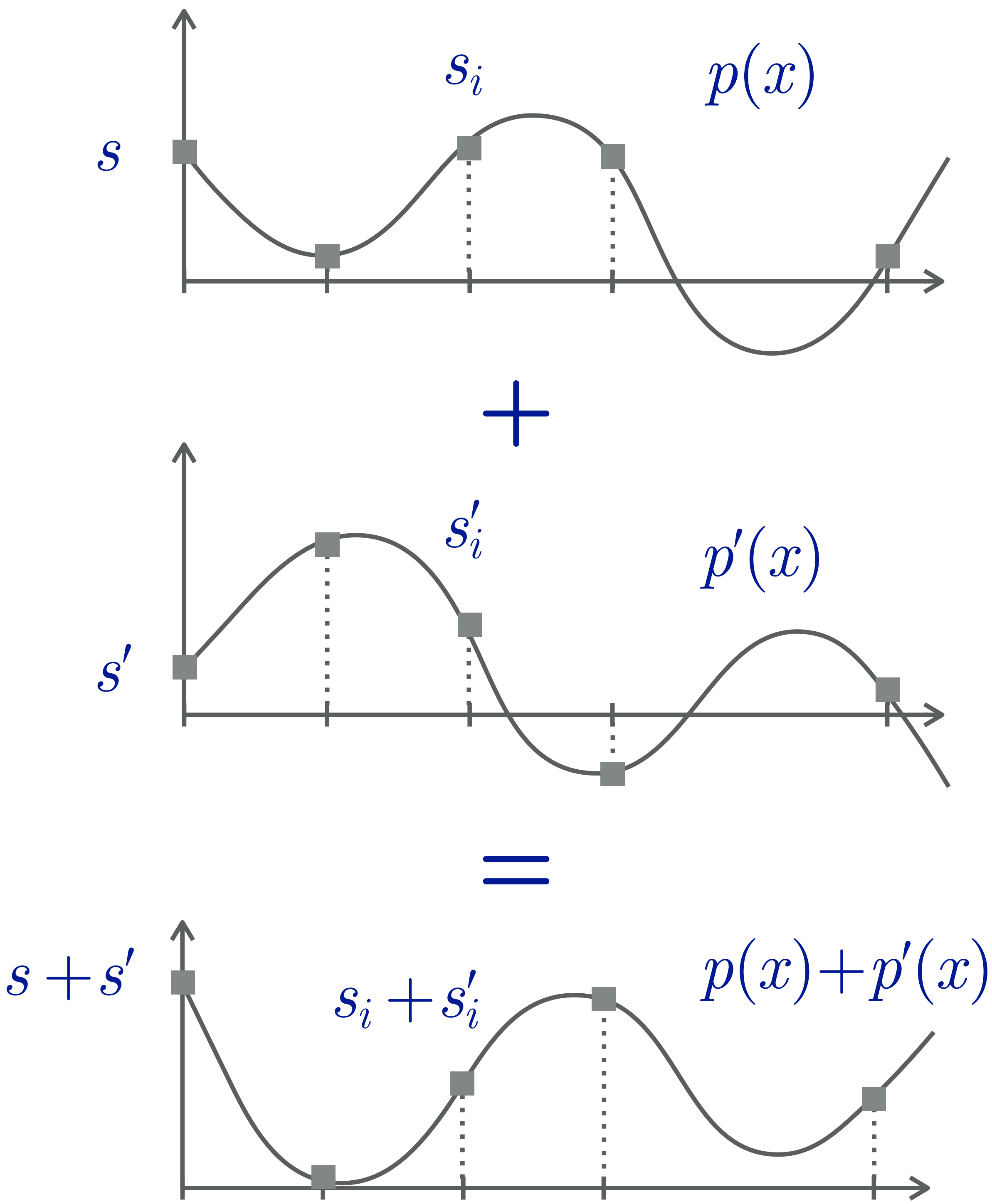


Similarly, for multiplication:

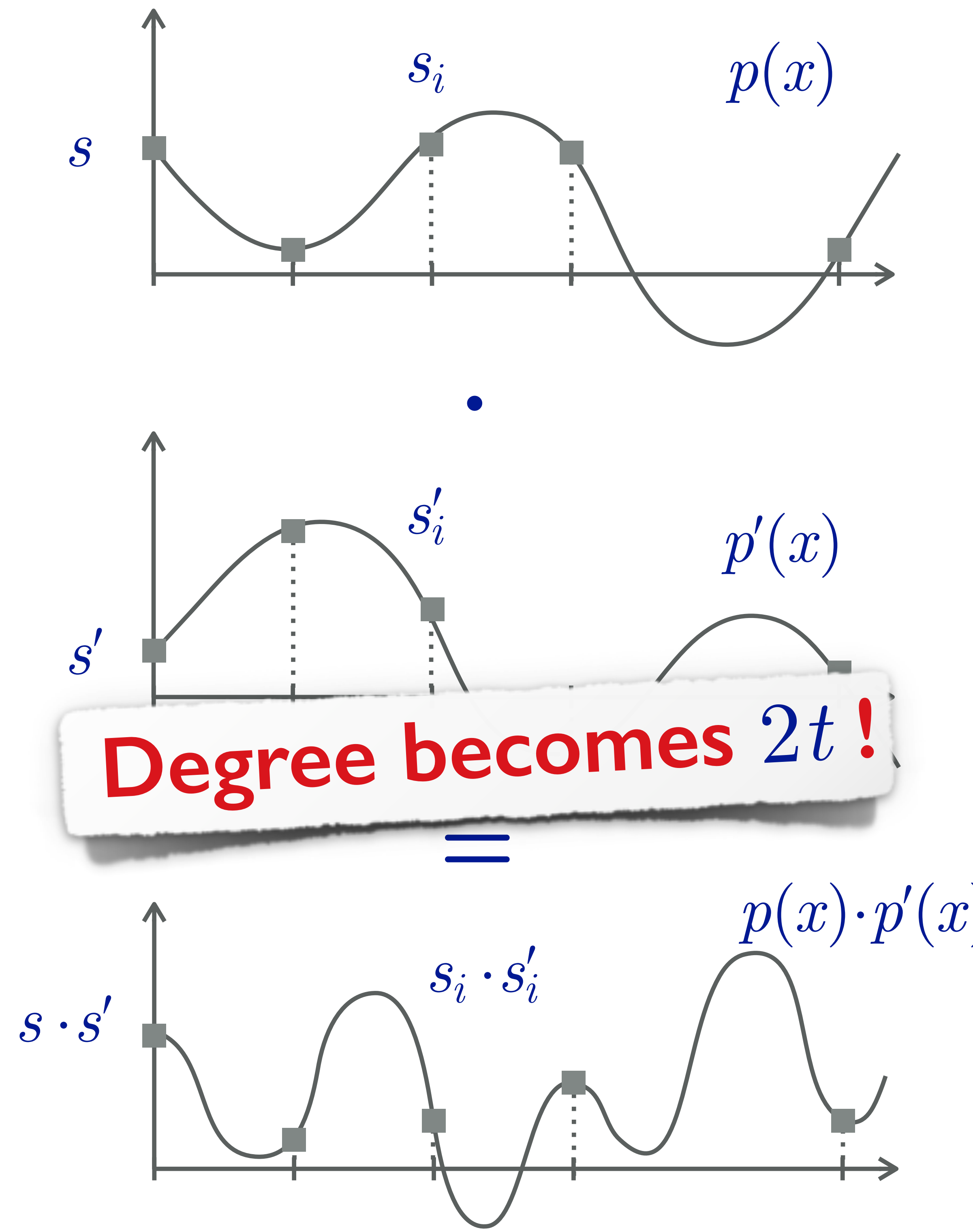


Towards Secure Multiplications

For addition, exploited:

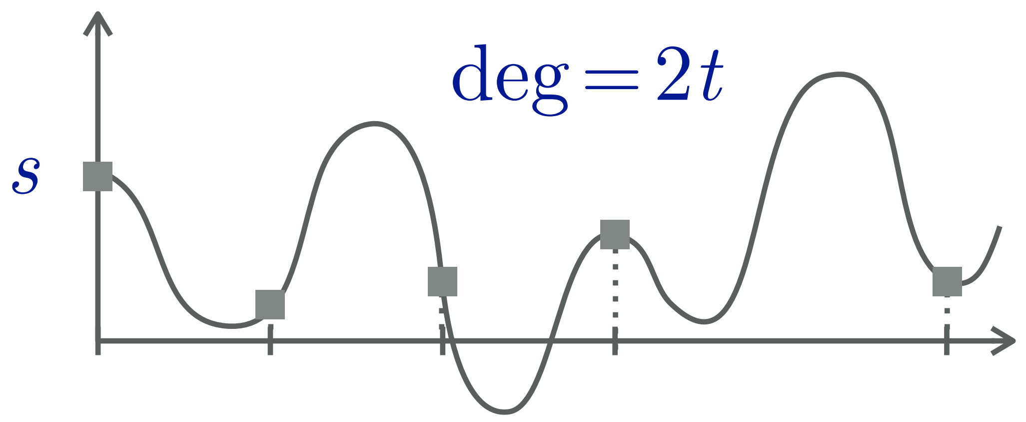


Similarly, for multiplication:



Degree Reduction

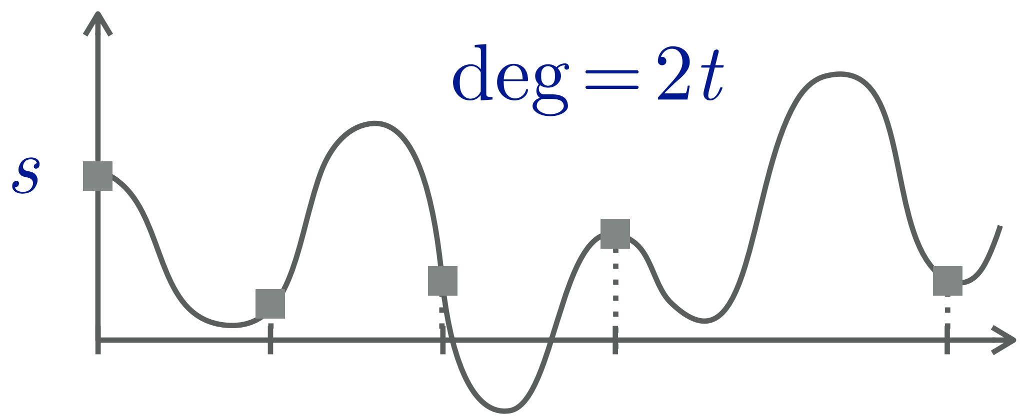
Due to [Chaum et al. 88], reinvented again in 2007.

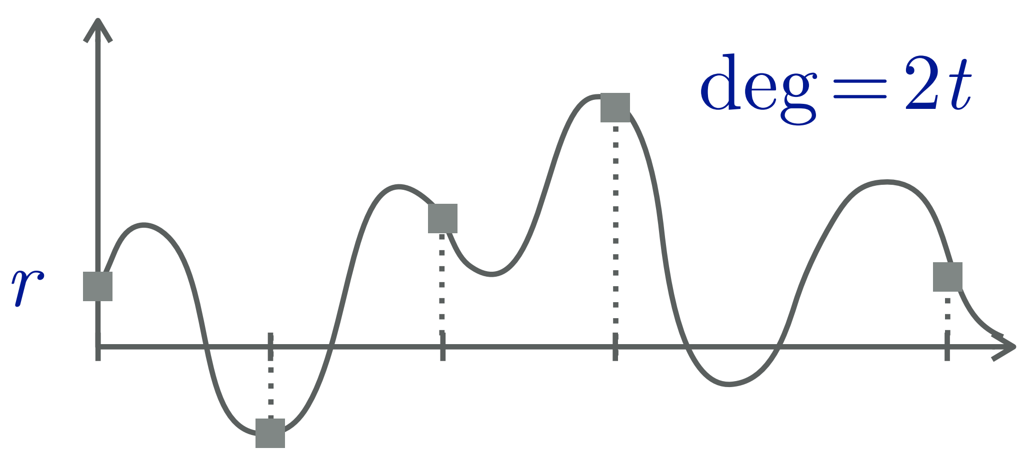


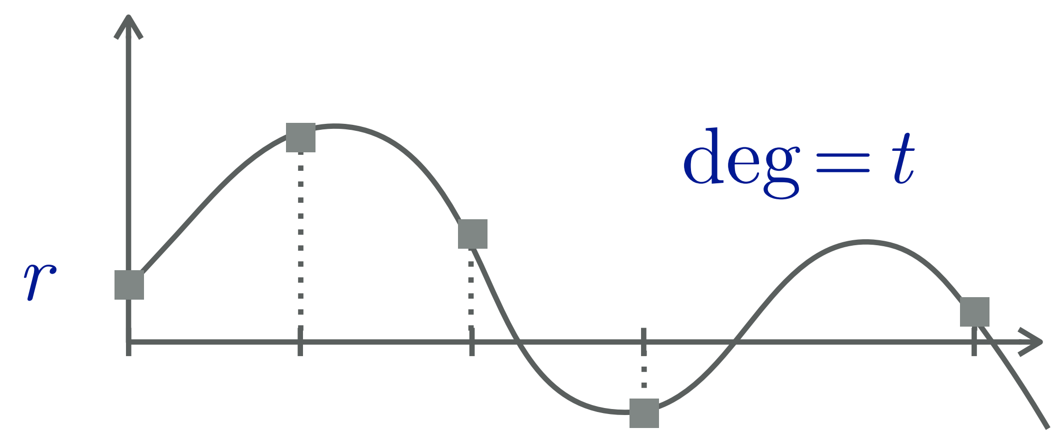
Degree Reduction

Due to [Chaum et al. 88], reinvented again in 2007.

Produce a deg-$2t$ and a deg-t sharing of random **unknown** r.

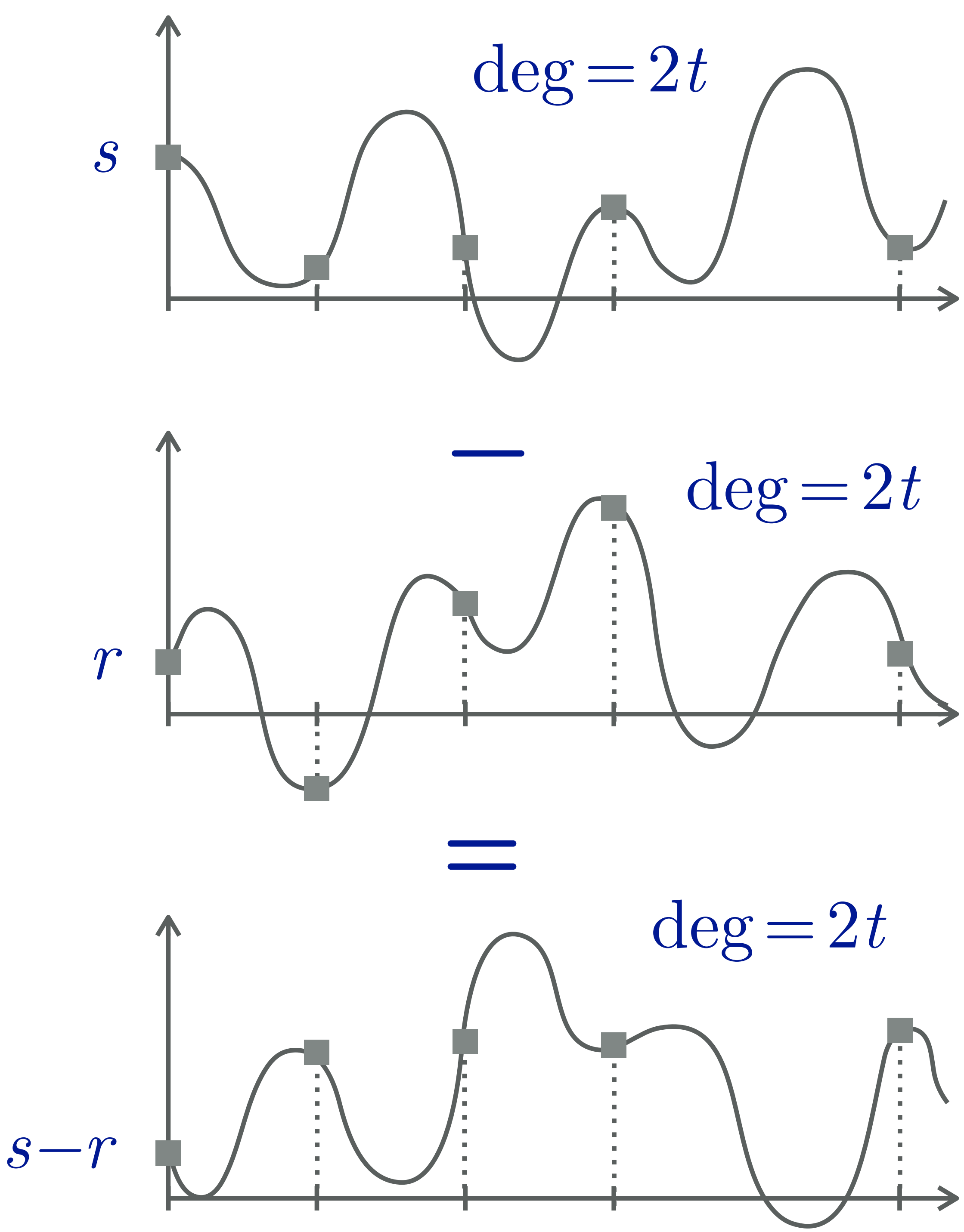




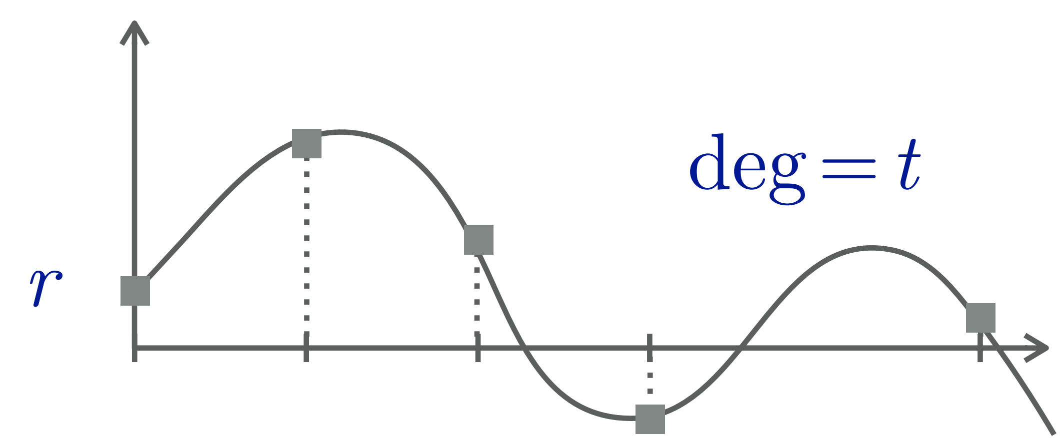


Degree Reduction

Due to [Chaum et al. 88], reinvented again in 2007.

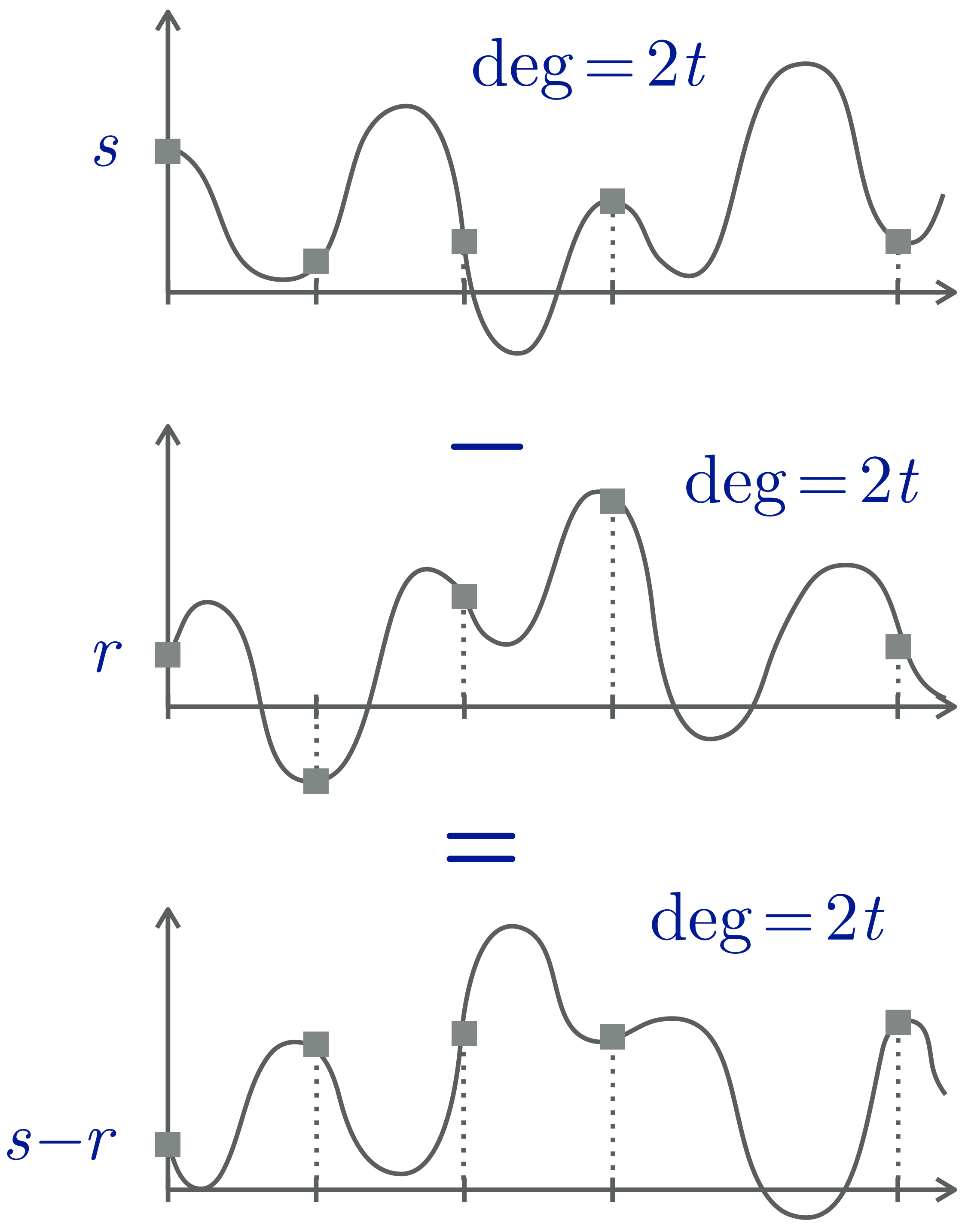


Locally compute the deg-$2t$ sharing of $\delta = s - r$.

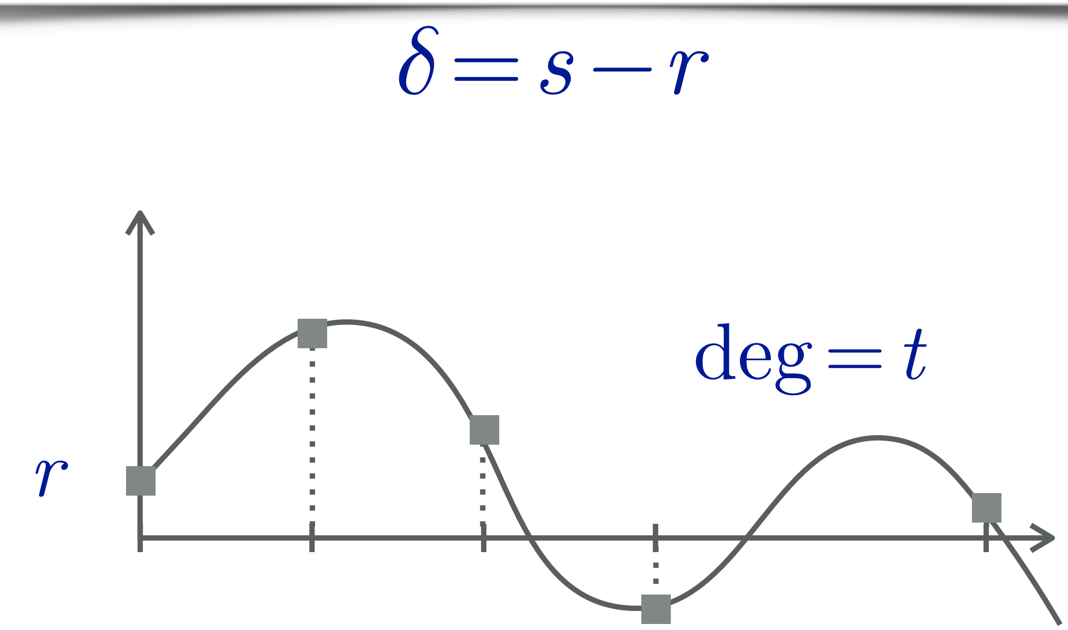


Degree Reduction

Due to [Chaum et al. 88], reinvented again in 2007.



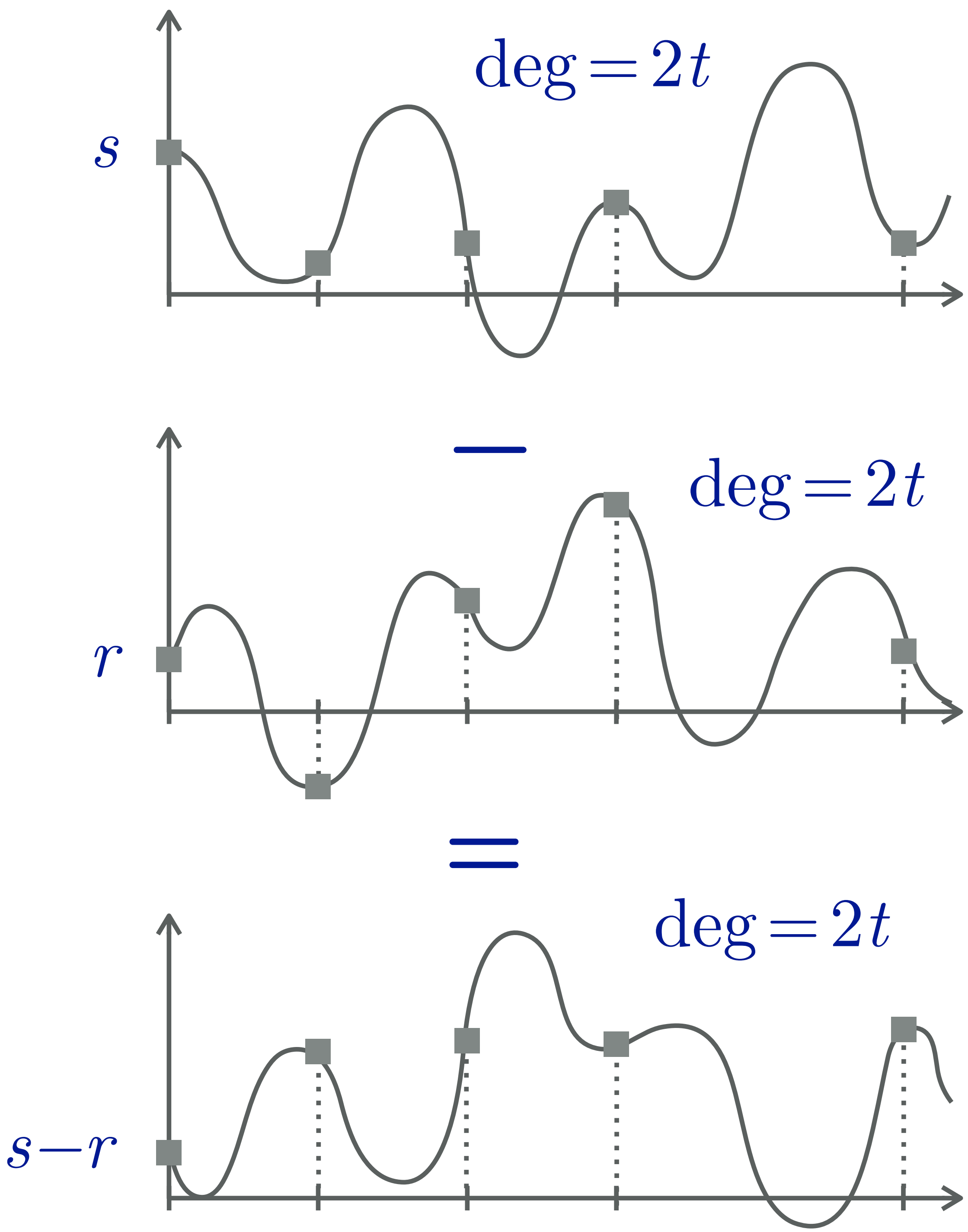
Reconstruct $\delta = s - r$, and



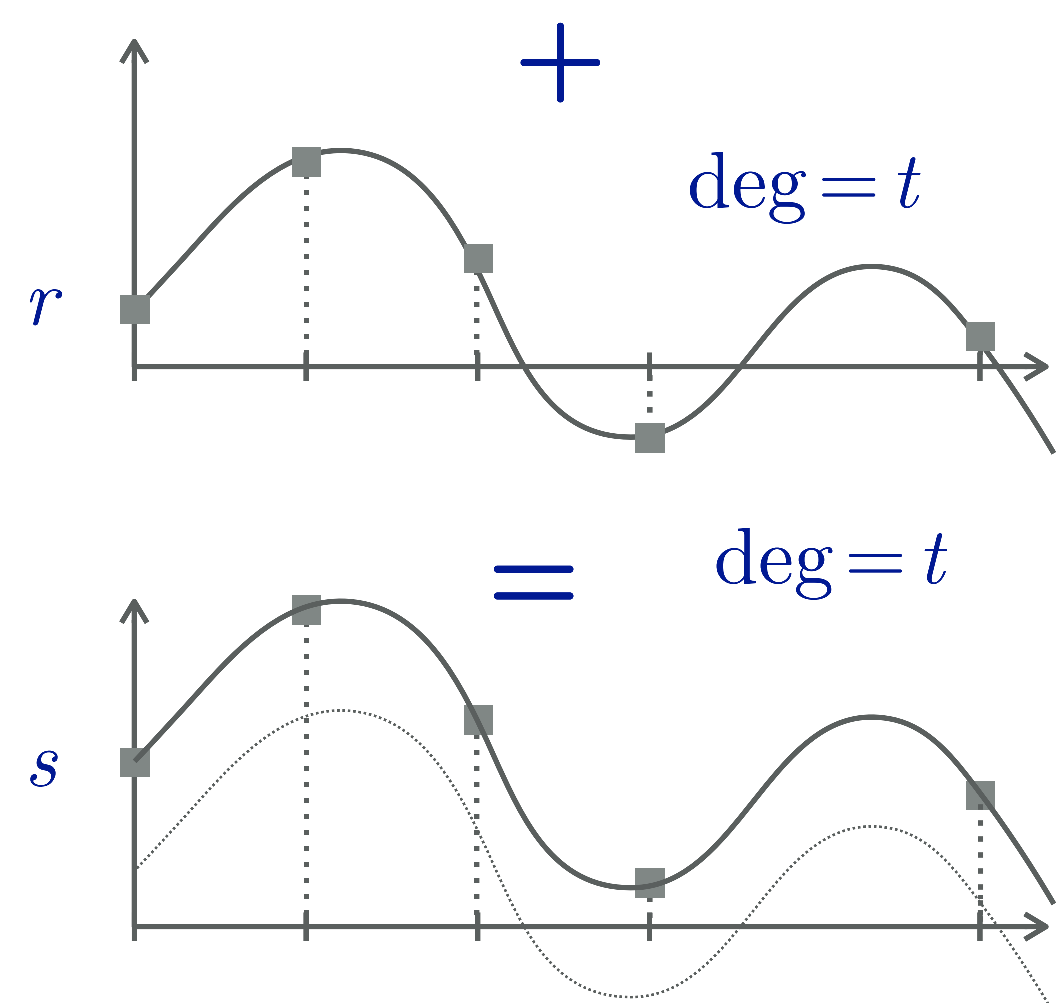
Degree Reduction

Due to [Chaum et al. 88], reinvented again in 2007.

Reconstruct $\delta = s - r$, and
add δ to the deg-t sharing of r.



$\delta = s - r$



Putting Above (And More) Things Together

Techniques for secure ***addition*** & secure ***multiplication***

$\Rightarrow$ secure ***arithmetic***

Putting Above (And More) Things Together

Techniques for secure *addition* & secure *multiplication*

$\Rightarrow$ secure *arithmetic*

&

Together with basic result from theory of computation:

*"**any** computation can be put as an arithmetic computation"*

Putting Above (And More) Things Together

Techniques for secure *addition* & secure *multiplication*

$\Rightarrow$ secure *arithmetic*

&

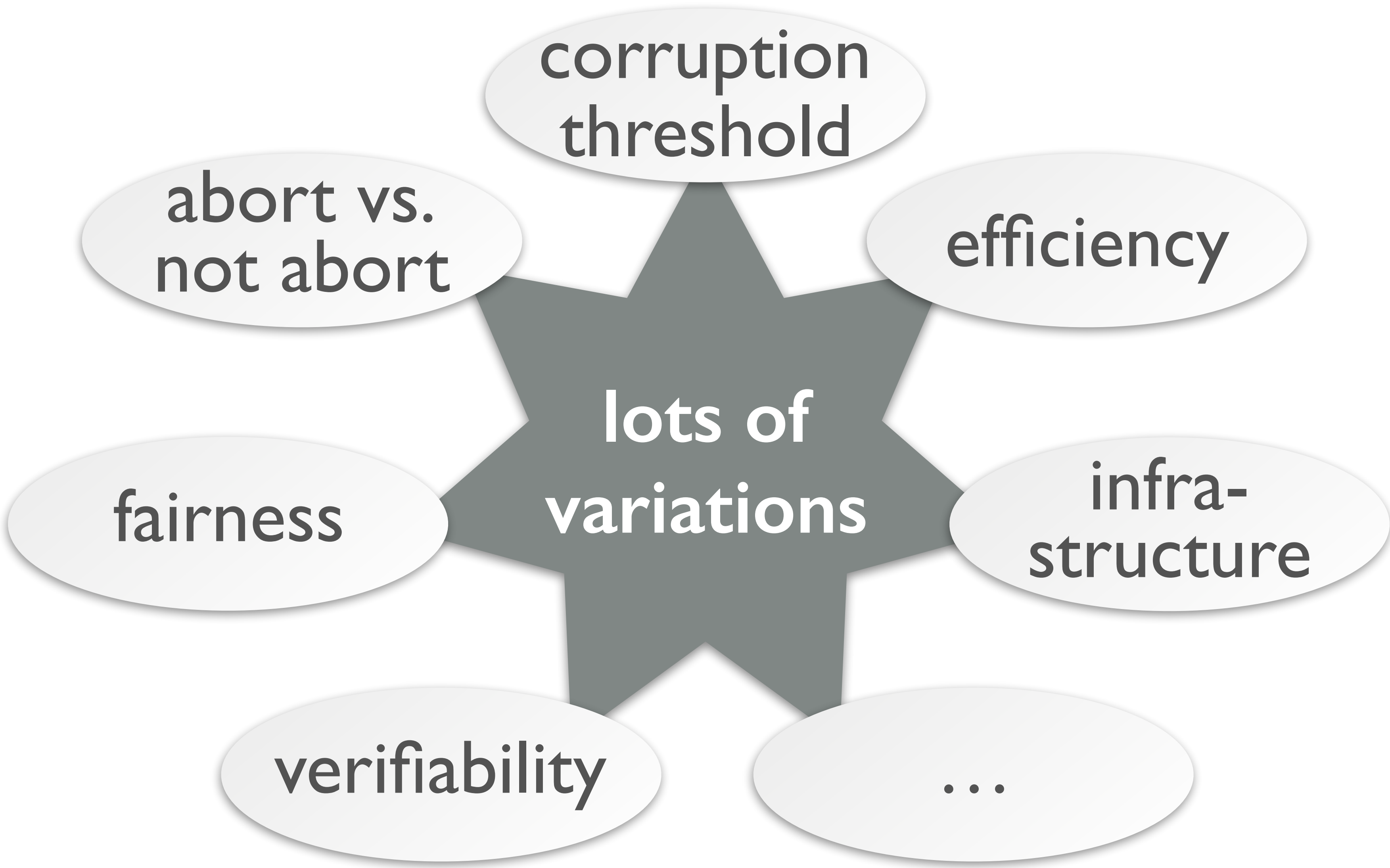
Together with basic result from theory of computation:
"*any computation can be put as an arithmetic computation*"

$\Downarrow$

Every computation can be done **securely**, i.e., so that

- everyone learns the correct result,
- yet nothing more than than,
- even if some of the parties are dishonest.

Various Relations & Dependencies

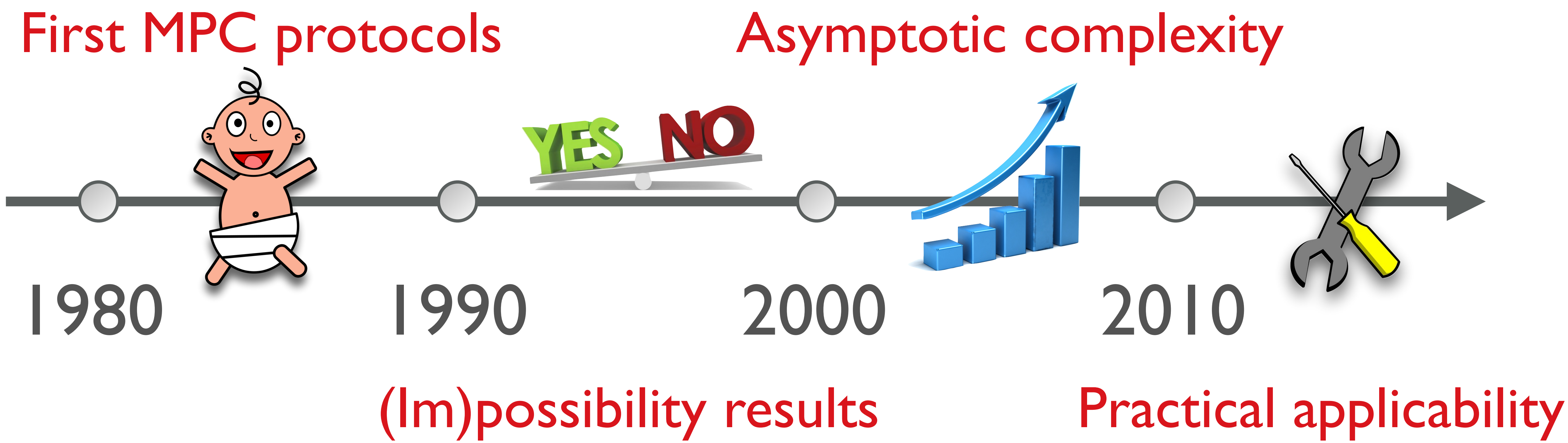


Optimal solution being very much **application dependent**.

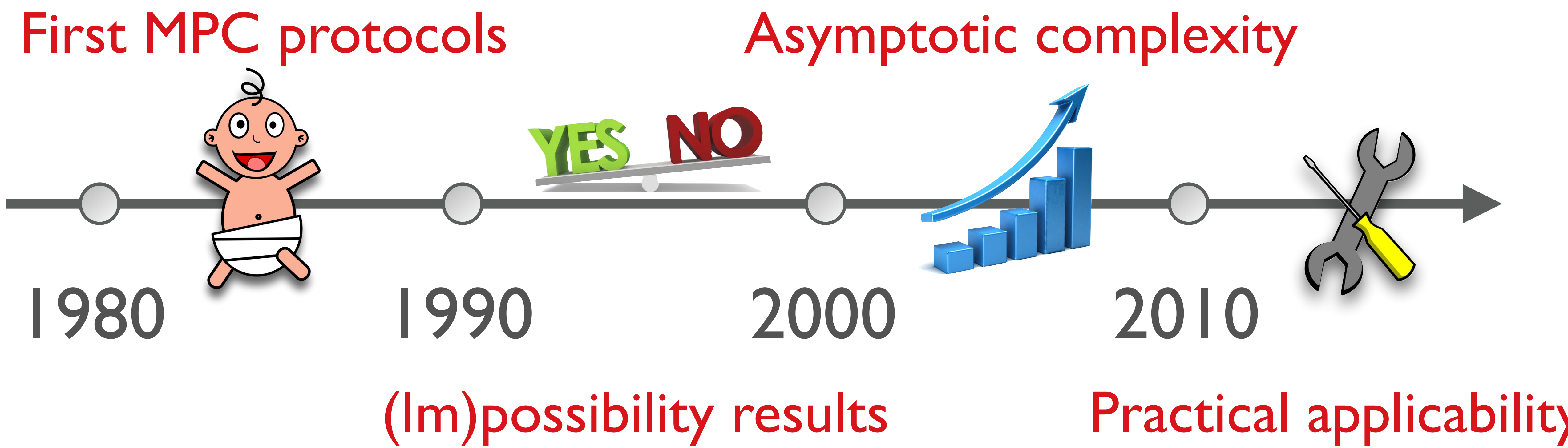
Road Map

- WHAT is multiparty computation?
- HOW does multiparty computation work?
- WHERE can/is multiparty computation be/ used?

Timeline from Theory to Practice



Timeline from Theory to Practice

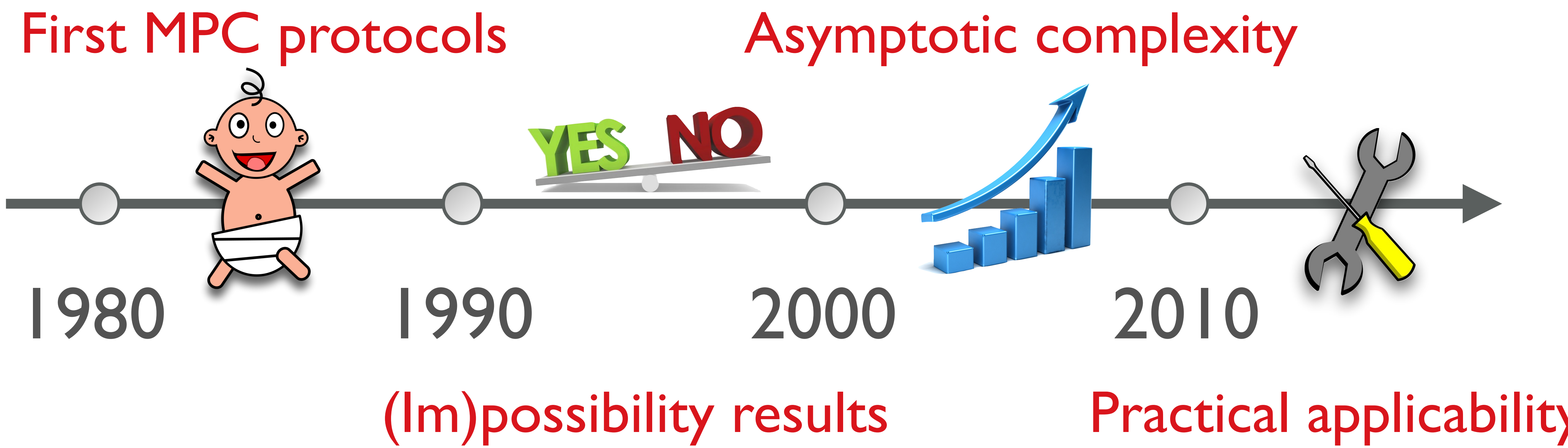


Current state of "practical MPC":

- ∃ companies that offer MPC solutions
- ∃ software libraries that facilitate "writing MPC code"
- ∃ isolated cases of real-life MPC deployment

But: **no** plug'n'play solution (seems to be inherent)

Timeline from Theory to Practice



Current state of "practical MPC":

- ∃ companies that offer MPC solutions
- ∃ software libraries that facilitate "writing MPC code"
- ∃ isolated cases of real-life MPC deployment

But: no plug'n'play solution (seems to be inherent)

Real-life MPC Example 1: **Trading Contracts**

Application scenario:

- Farmers in Demark wish to trade sugar beet contracts, giving them rights to produce/sell to a certain price.
- Danisco (buying the beets) needs to be involved as well.

Reference: Secure Multiparty Computation Goes Live (eprint.iacr.org/2008/068)

Real-life MPC Example 1: **Trading Contracts**

Application scenario:

- Farmers in Demark wish to trade sugar beet contracts, giving them rights to produce/sell to a certain price.
- Danisco (buying the beets) needs to be involved as well.

Problem: Farmers **do not want to reveal** their bids (as they leak info on economic position and productivity).

Reference: Secure Multiparty Computation Goes Live (eprint.iacr.org/2008/068)

Real-life MPC Example 1: **Trading Contracts**

Application scenario:

- Farmers in Demark wish to trade sugar beet contracts, giving them rights to produce/sell to a certain price.
- Danisco (buying the beets) needs to be involved as well.

Problem: Farmers **do not want to reveal** their bids (as they leak info on economic position and productivity).

Solution: **Use MPC**

- Since 2008, auction runs as a 3-party computation.
- Market clearing price computed in a secure way, i.e., **without** revealing individual bids.

Reference: Secure Multiparty Computation Goes Live (eprint.iacr.org/2008/068)

Real-life MPC Example 2: **Data Mining**

Application scenario:

- Researchers in Estonia wanted to study the correlation between *working during university* and *failing to graduate*.
- Required: linking databases from *Estonian Tax & Customs Board* and from *Ministry of Education & Research*.

Reference: Students and Taxes: A Privacy-Preserving Study Using Secure Computation (PET 2016)

Real-life MPC Example 2: **Data Mining**

Application scenario:

- Researchers in Estonia wanted to **study the correlation** between *working during university* and *failing to graduate*.
- Required: linking databases from *Estonian Tax & Customs Board* and from *Ministry of Education & Research*.

***Problem:* By law**, these databases **may not to revealed** (*Estonian Personal Data Protection Act* and *Taxation Act*).

Reference: Students and Taxes: A Privacy-Preserving Study Using Secure Computation (PET 2016)

Real-life MPC Example 2: **Data Mining**

Application scenario:

- Researchers in Estonia wanted to study the correlation between *working during university* and *failing to graduate*.
- Required: linking databases from *Estonian Tax & Customs Board* and from *Ministry of Education & Research*.

Problem: **By law**, these databases **may not to revealed** (*Estonian Personal Data Protection Act* and *Taxation Act*).

Solution: **Use MPC**

- Statistical analysis was done by a 3-party computation, **without** revealing the data bases.

Reference: Students and Taxes: A Privacy-Preserving Study Using Secure Computation (PET 2016)

Real-life MPC Example 3: **Password Checkup**

Application scenario:

- Have every user name & password you enter on a site checked against credentials that are known to be unsafe.

Reference: Helping Organisations Do More Without Collecting Data (Google Security Blog)

Real-life MPC Example 3: **Password Checkup**

Application scenario:

- Have every user name & password you enter on a site checked against credentials that are known to be unsafe.

Problem: You **do not want to reveal** your password.

Reference: Helping Organisations Do More Without Collecting Data (Google Security Blog)

Real-life MPC Example 3: **Password Checkup**

Application scenario:

- Have every user name & password you enter on a site checked against credentials that are known to be unsafe.

Problem: You **do not want to reveal** your password.

Solution: **Use MPC**

- Google offers a *Password Checkup* extension for Chrome, which uses a 2-party computation to check your credentials, **without** Google learning your credentials.

Reference: Helping Organisations Do More Without Collecting Data (Google Security Blog)

Potential Future Real-life MPC Example

[Joint work with CWI Crypto, TNO, UvA - **Demonstrator only**]

Reference: A New Approach to Privacy-Preserving Clinical Decision Support Systems for HIV Treatment (arXiv:1810.01.01107)

Potential Future Real-life MPC Example

[Joint work with CWI Crypto, TNO, UvA - **Demonstrator only**]

Application scenario:

- Effective HIV treatment is a very complicated matter.
- Effectiveness of a drug is related to genotype of HIV virus.
- Not well understood: $\exists > 10^{1250}$ possible HIV virus strains!
- Having an "experience database" would be very valuable.

Reference: A New Approach to Privacy-Preserving Clinical Decision Support Systems for HIV Treatment (arXiv:1810.01.01107)

Potential Future Real-life MPC Example

[Joint work with CWI Crypto, TNO, UvA - **Demonstrator only**]

Application scenario:

- Effective HIV treatment is a very complicated matter.
- Effectiveness of a drug is related to genotype of HIV virus.
- Not well understood: $\exists > 10^{1250}$ possible HIV virus strains!
- Having an "experience database" would be very valuable.

Problem:

- Genotype of HIV virus is very sensitive data.
- Doctors are not willing to share treatment (liability).

Reference: A New Approach to Privacy-Preserving Clinical Decision Support Systems for HIV Treatment (arXiv:1810.01.01107)

Potential Future Real-life MPC Example

[Joint work with CWI Crypto, TNO, UvA - **Demonstrator only**]

Application scenario:

- Effective HIV treatment is a very complicated matter.
- Effectiveness of a drug is related to genotype of HIV virus.
- Not well understood: $\exists > 10^{1250}$ possible HIV virus strains!
- Having an "experience database" would be very valuable.

Problem:
- Genotype of HIV virus is very sensitive data.
- Doctors are not willing to share treatment (liability).

Solution: **Use MPC**

- We built a MPC prototype for a "experience database" with support for *time-to-treatment-failure* queries.

Reference: A New Approach to Privacy-Preserving Clinical Decision Support Systems for HIV Treatment (arXiv:1810.01.01107)

Recap

Recap

- MPC has its roots in seminal work from the 80’s.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.
- Now at the verge of being practically relevant.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.
- Now at the verge of being practically relevant.
- Several (though still isolated) real-life deployments.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.
- Now at the verge of being practically relevant.
- Several (though still isolated) real-life deployments.
- In principle: Applicable in lots and lots of scenarios.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.
- Now at the verge of being practically relevant.
- Several (though still isolated) real-life deployments.
- In principle: Applicable in lots and lots of scenarios.
- Comes with a "price tag": considerable loss in efficiency.

Recap

- MPC has its roots in seminal work from the 80's.
- Originally considered to be of theoretical interest only.
- Stayed a "hot topic" within the research community.
- Now at the verge of being practically relevant.
- Several (though still isolated) real-life deployments.
- In principle: Applicable in lots and lots of scenarios.
- Comes with a "price tag": considerable loss in efficiency.
- No plug'n'play: need for tailor-made solution is inherent

CWI Centrum Wiskunde & Informatica

Universiteit Leiden

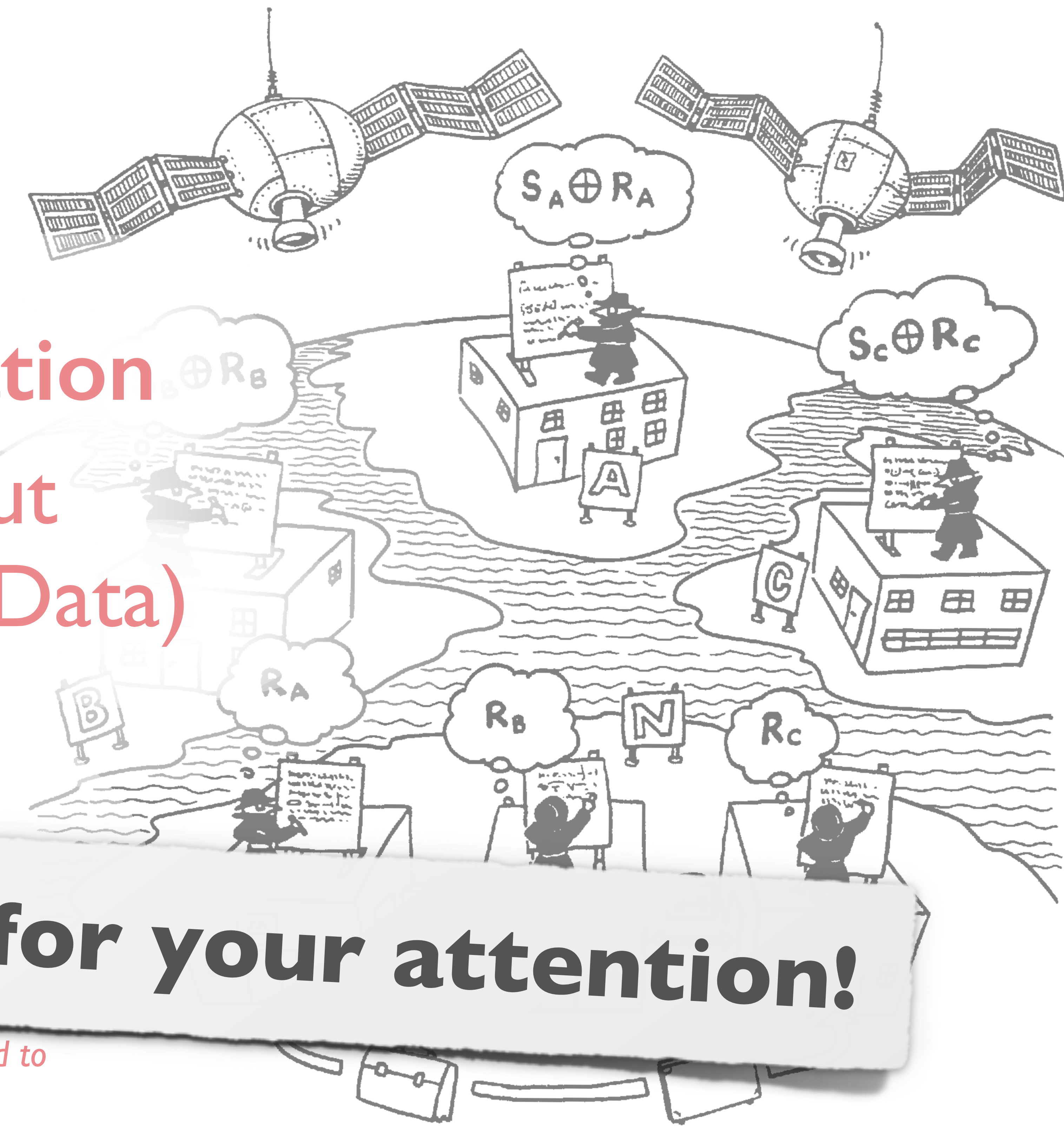
CWI Lectures
November 21 & 22, 2019

Multiparty Computation
Collaborate Without Compromise(ing Your Data)

Serge Fehr

Centrum Wiskunde & Informatica (CWI)

Mathem



Thank you for your attention!

On the occasion of the Dijkstra Fellowship being awarded to David Chaum