Centrum Wiskunde & Informatica

CRYPTOGRAPHY

Part II:
Secure Multiparty Computation

Serge Fehr

CWI Amsterdam
www.cwi.nl/~fehr

Encryption and more

ALICE

EVE

BOB

Eve can:

- **eavesdrop** the communication
  -> use **encryption** (symmetric or public-key)
- **modify** (or insert/delete) messages
  -> use **authentication** or **digital signatures**

Encryption and more

ALICE

EVE

BOB

Eve ca

- eav
  ->
- moc
  -> use authentication or digital signatures

Distinguishing features

- clear distinction between good and bad
- know whom to trust
- reveal all-or-nothing

Encryption and more

ALICE

EVE

BOB

Eve c

- eav

  ->

- mod

  -> use authentication or digital signatures

But:

The world is not just **black** and **white**

Secure-Cooperation Problems

Setting

- Two or more parties want to engage into cooperation
- Common interest to perform this cooperation
- Parties do not trust **each other** (fully)

Security Goals

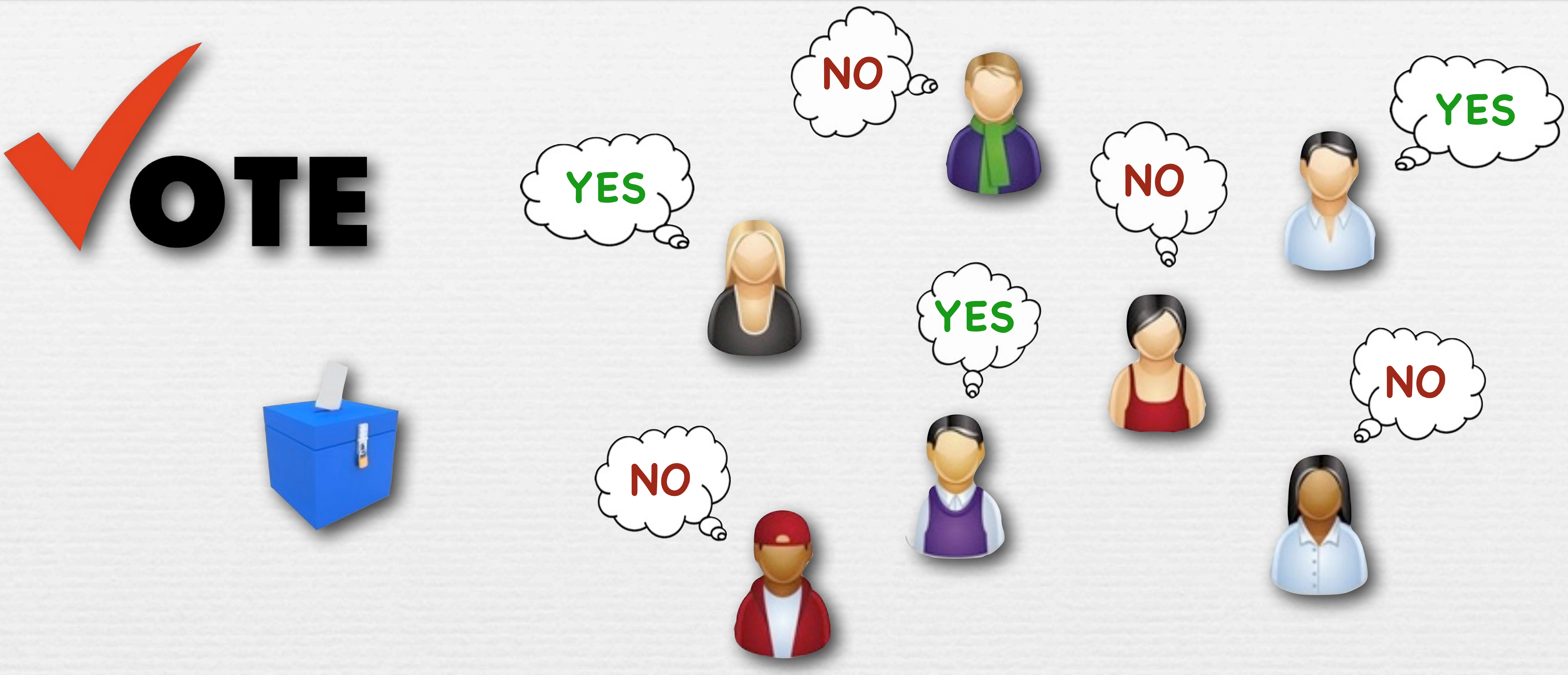
- private data **remains private** - as much as possible
- result of cooperation is **correct**

Examples

"millionaires' problem"

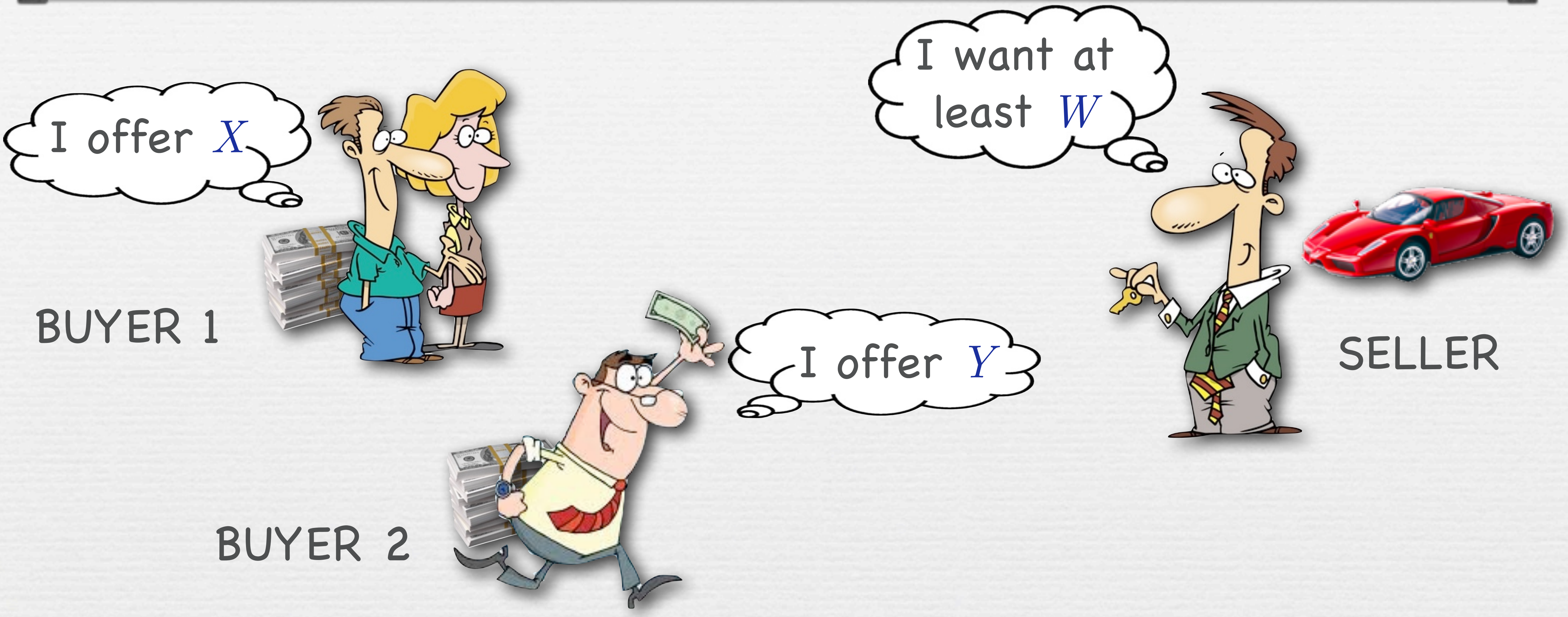
- Two millionaires want to find out who is richer.
- **Neither is willing to reveal** how much he owns.

Examples



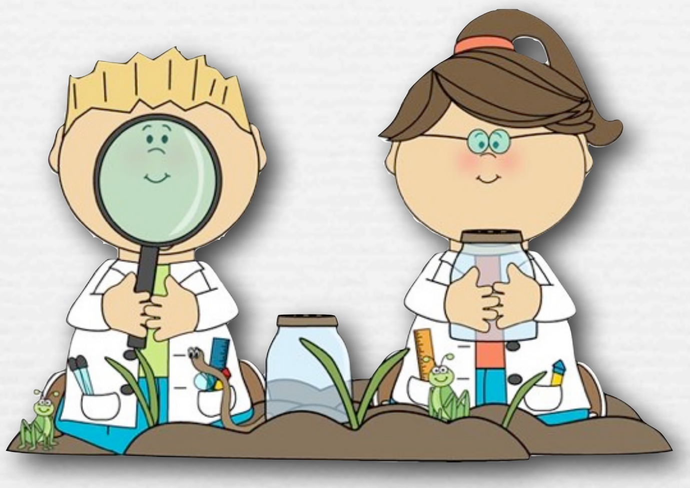
- Voters want to find out outcome of the vote.
- **None is willing to reveal** his individual vote.

Examples



- Want to find out if bids are sufficient and who bids more, and what the winning bid is.
- **No one is willing to reveal** his upper/lower bound.

Examples



Scientists

Hospital

- Scientists want to perform study on patient data
- Hospital **is not allowed to reveal** such sensitive data

Examples

Company A 

 Company B

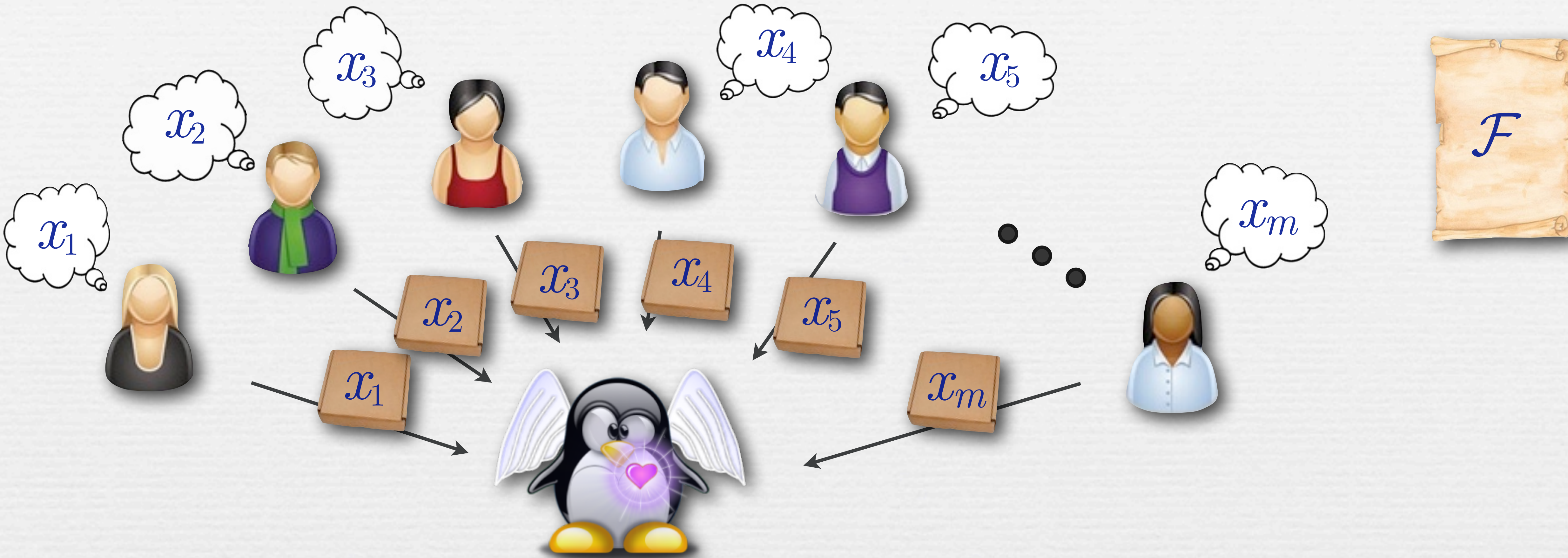
- Want to do joint analysis of individually gathered data (e.g. two competing pharma companies want to pool their clinical data for improved effectiveness study)
- **Neither is willing to reveal** its own data

The General Problem

$$\mathcal{F}(x_1, x_2, x_3, \ldots, x_m) = ?$$

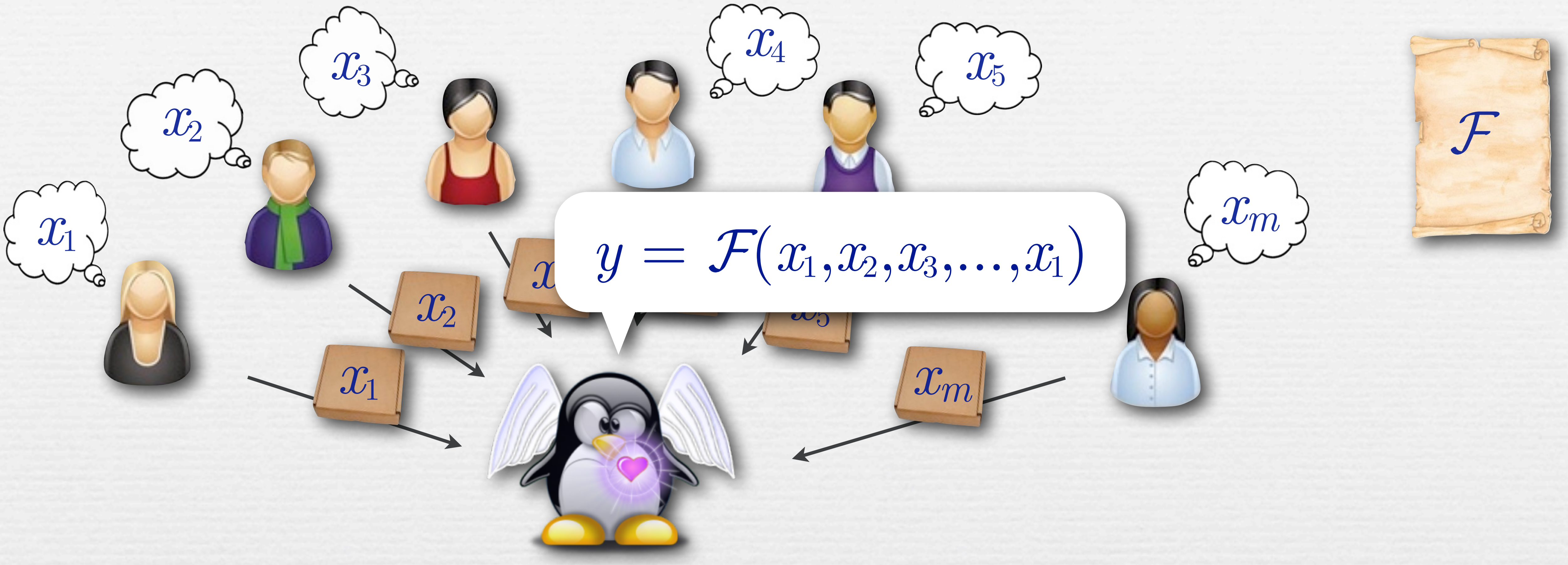
- Every user U_i has a **private input** x_i .
- Users **want to learn** $\mathcal{F}(x_1, x_2, x_3, \ldots, x_1)$.
  (Variation: Different users learn different functions)
- Private inputs should **remain private**.
- Output should be guaranteed to be **correct**.

An Ideal Solution



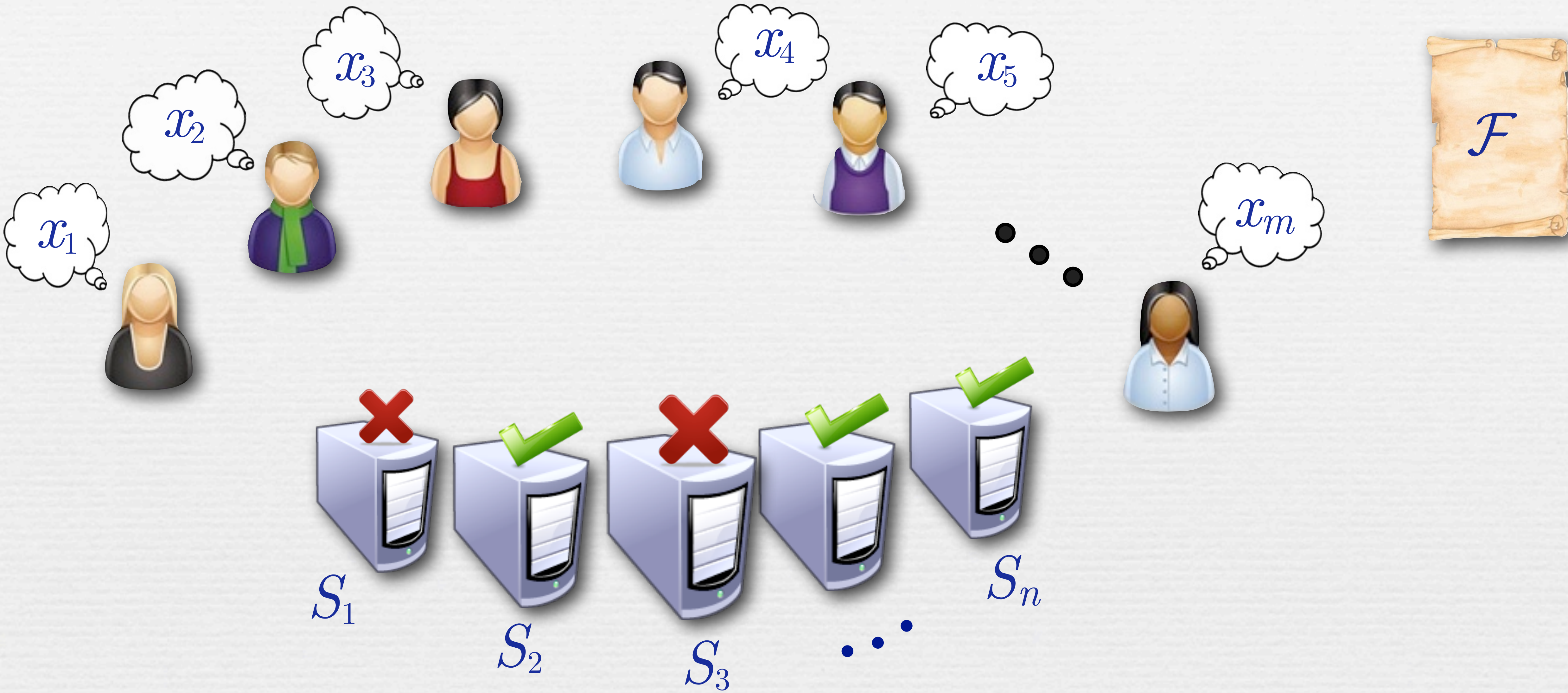
- Every user U_i sends his x_i to trusted authority TA.

An Ideal Solution



- Every user U_i sends his x_i to trusted authority TA.
- TA computes $y = \mathcal{F}(x_1, x_2, x_3, \ldots, x_1)$, and
- announces y to everyone.

MPC: Removing the Trusted Authority



Idea:

- Perform computation by a group of servers (servers could be the users/parties themselves)
- Some of the servers may be malicious.

MPC: Removing the Trusted Authority

Idea:

- Perform computation by a group of servers.
- Some of the servers may be malicious.

Want:

- No single (malicious) server learns any input.
- Malicious servers jointly should not learn any input.
- Also: malicious servers cannot influence outcome y.

Advantages:

- No need to know whom to trust.
- Different users may trust different servers.
- No single point of failure.

Only requirement:

- some servers are honest.

MPC: Removing the Trusted Authority

Idea:

- Perform computation by a group of servers.

> A MPC **emulates** an imaginary **fully trusted** party by means of a **group** of **partly trusted** parties.

- Also: malicious servers cannot influence outcome y.

Advantages:

- No need to know whom to trust.
- Different users may trust different servers.
- No single point of failure

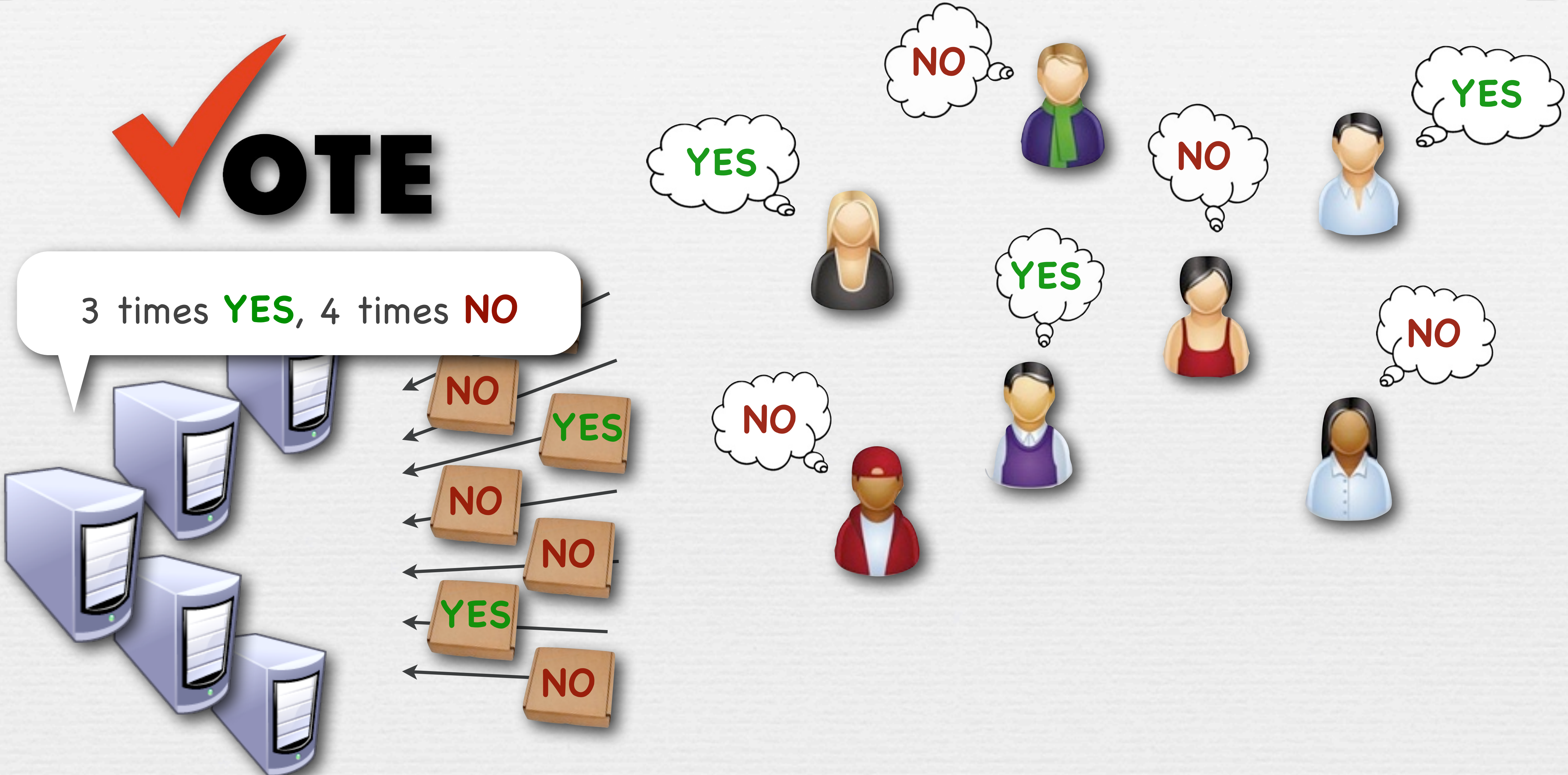
Only requirement:

- some servers are honest.

Theory and Practice of MPC

- Exist many different variants which differ in:
  - notion of security
  - communication model
  - (dis)allowing 'abort'
  - # of malicious servers
  - set-up assumptions
  - etc.
- Strong possibility results
- Theory is very well understood
- Great progress to bring MPC to practice (sugar beet contracts in Denmark traded using MPC)
- Not plug-and-play (yet), solutions are custom made

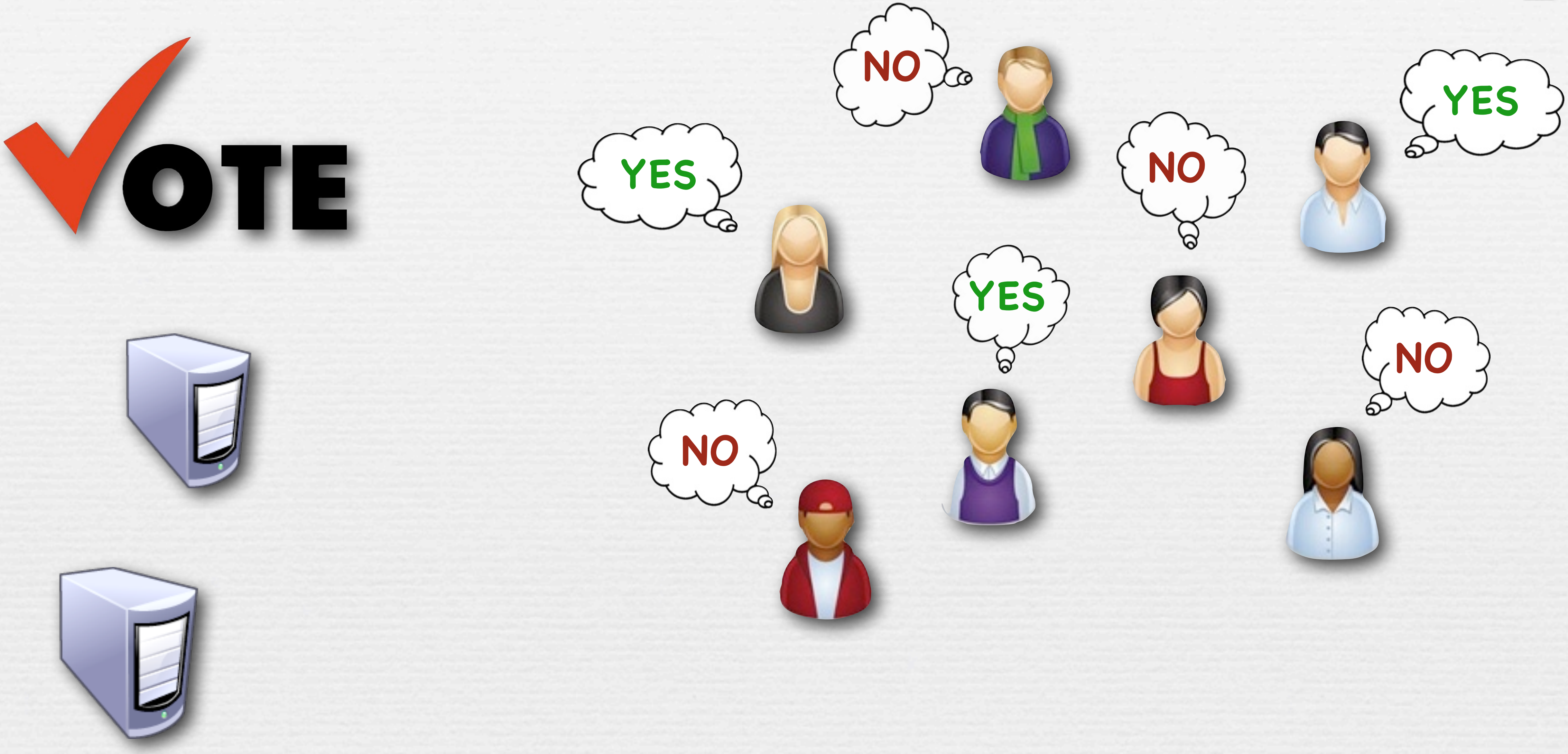
The Functionality of MPC



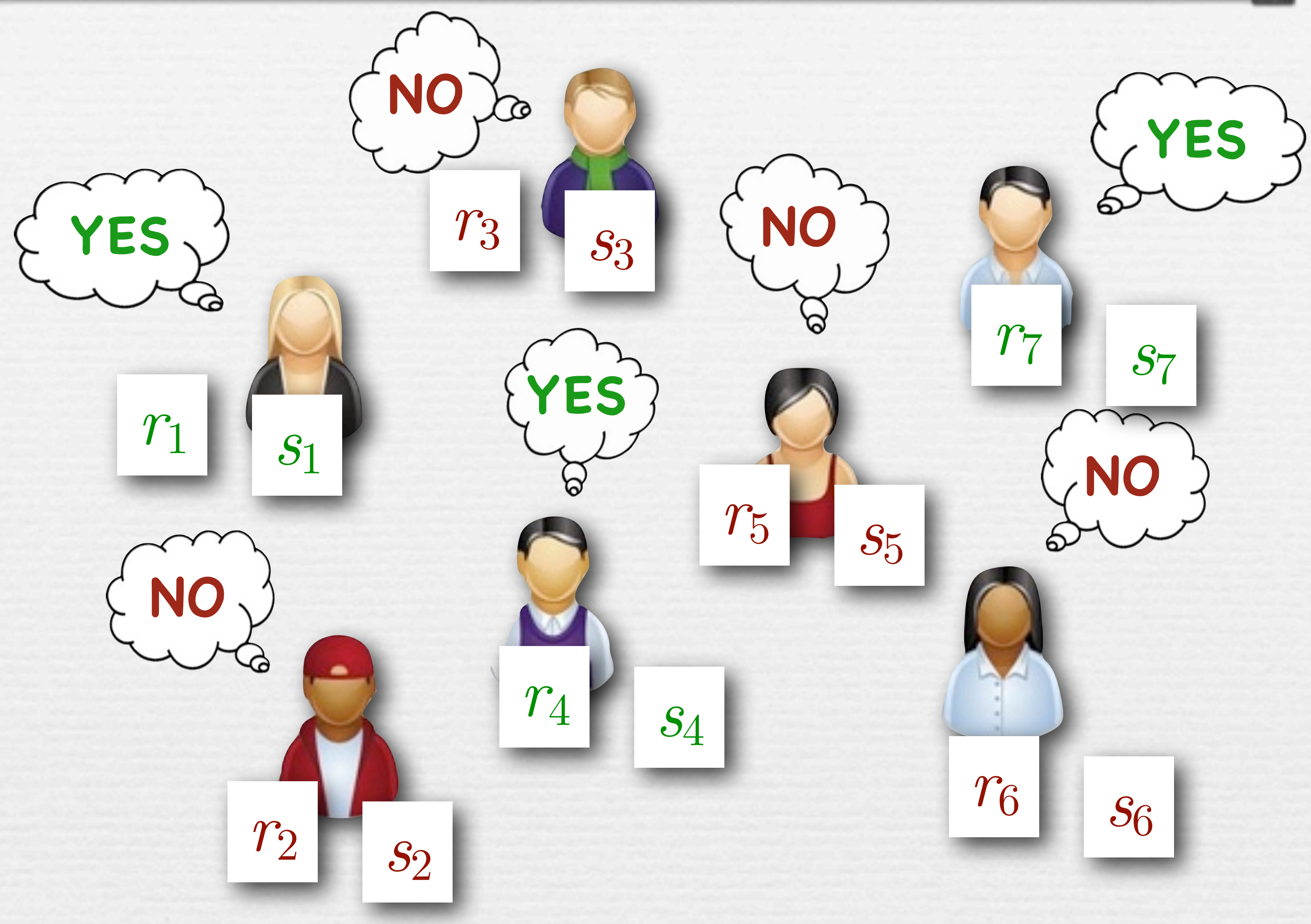
Promise:

- Votes remain private (to coalitions of parties/servers)
- Tally is guaranteed correct

A First Try



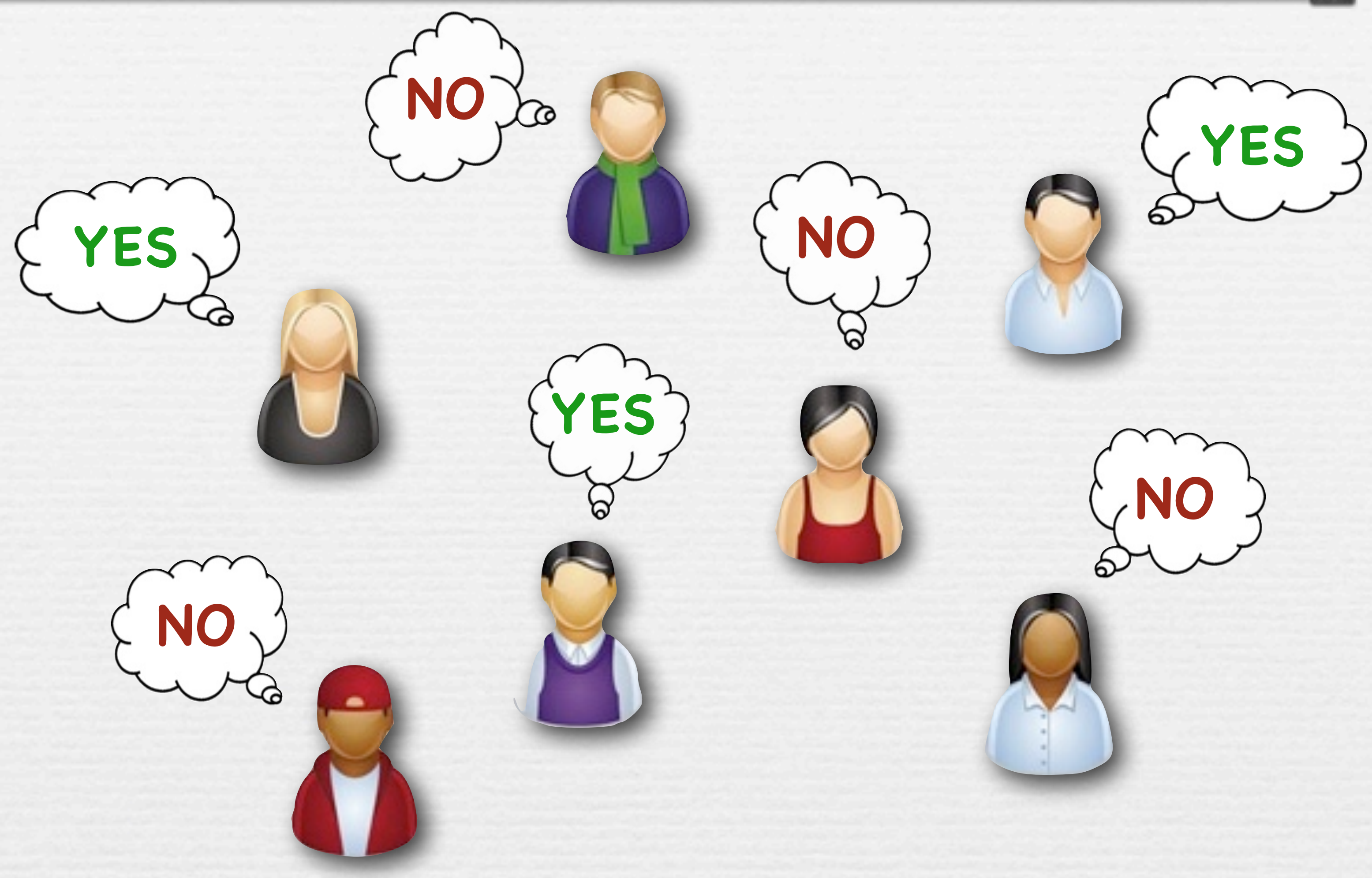
A First Try



Rule:

- if vote = **NO** then $s_i = r_i$
- if vote = **YES** then $s_i = r_i + 1$

A First Try

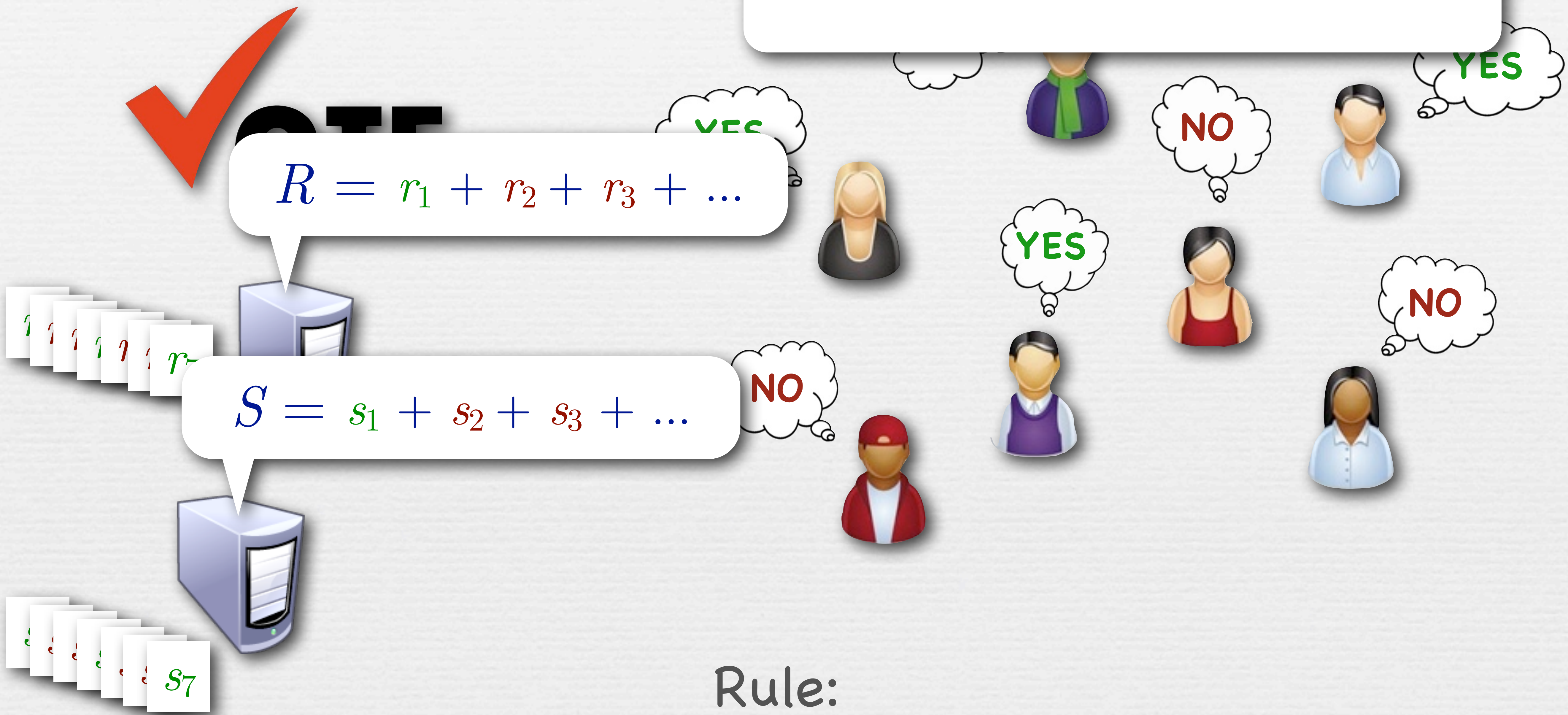


Rule:

- if vote = **NO** then $s_i = r_i$
- if vote = **YES** then $s_i = r_i + 1$

A First Try

Number **YES**-votes = $S-R$

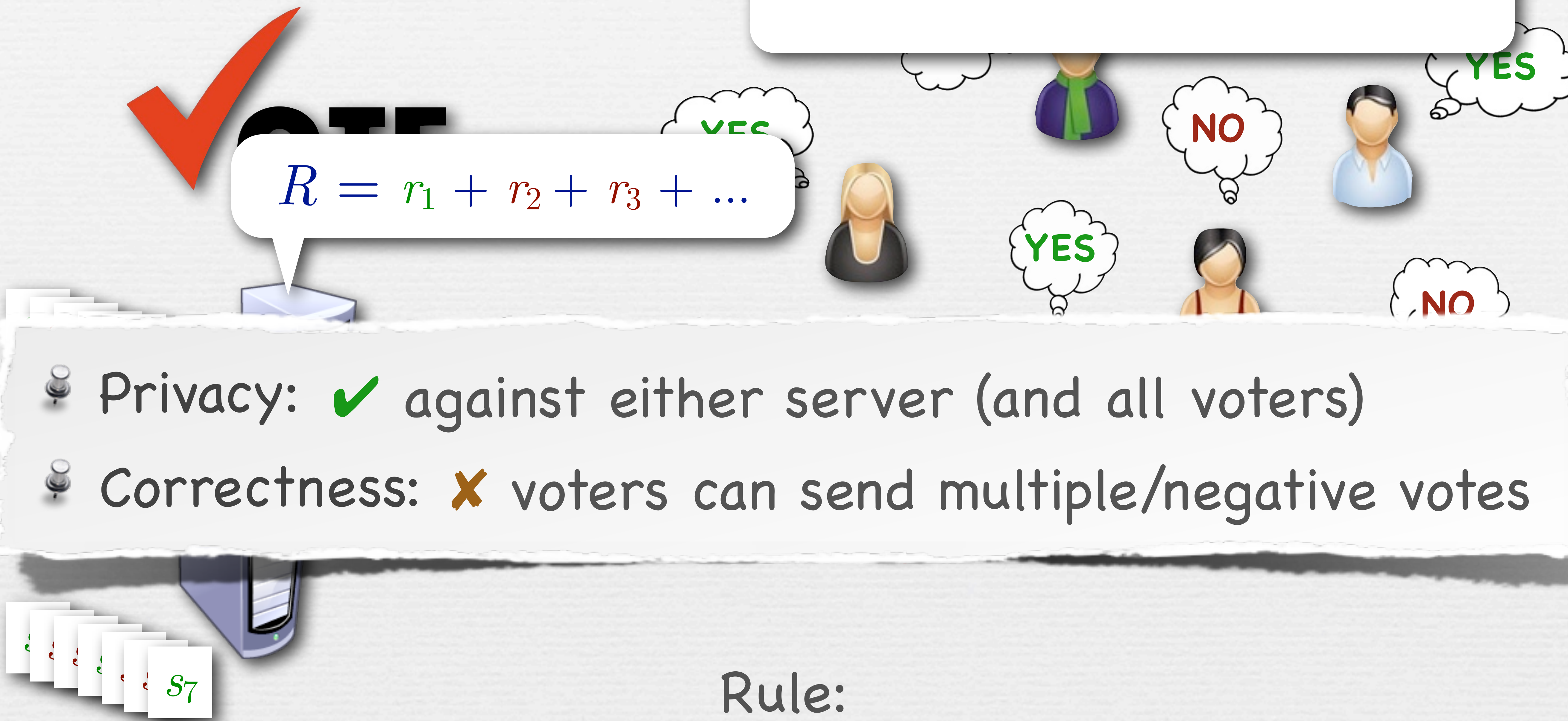


Rule:

- if vote = **NO** then $s_i = r_i$
- if vote = **YES** then $s_i = r_i + 1$

A First Try

Number **YES**-votes = $S - R$



$R = r_1 + r_2 + r_3 + \ldots$

- Privacy: ✔ against either server (and all voters)
- Correctness: ✘ voters can send multiple/negative votes

Rule:

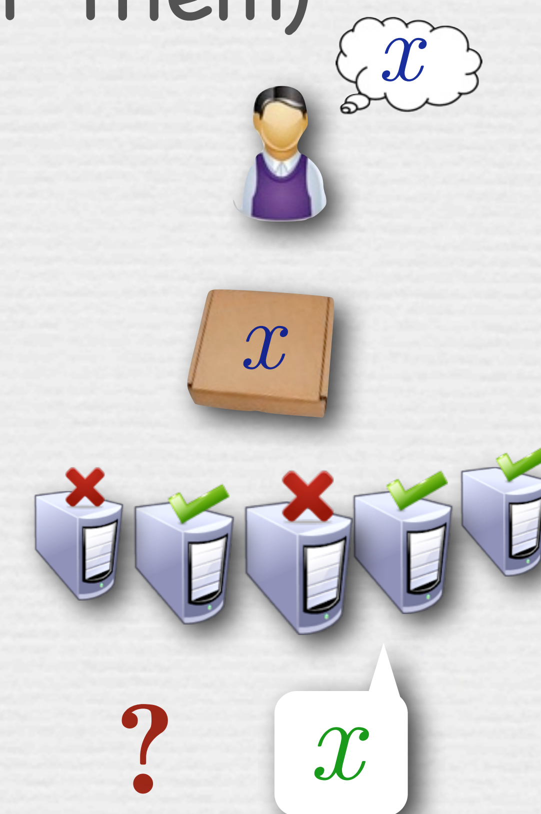
- if vote = **NO** then $s_i = r_i$
- if vote = **YES** then $s_i = r_i + 1$

Tool: Homomorphic Threshold Encryption

"Encryption scheme" with special properties

Threshold:

- Decryption ability is ``shared" among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt
  (even if a malicious minority tries to prevent them)



(Additively) homomorphic:

- When given encryption of x and y
- an encryption of $x+y$ can be computed

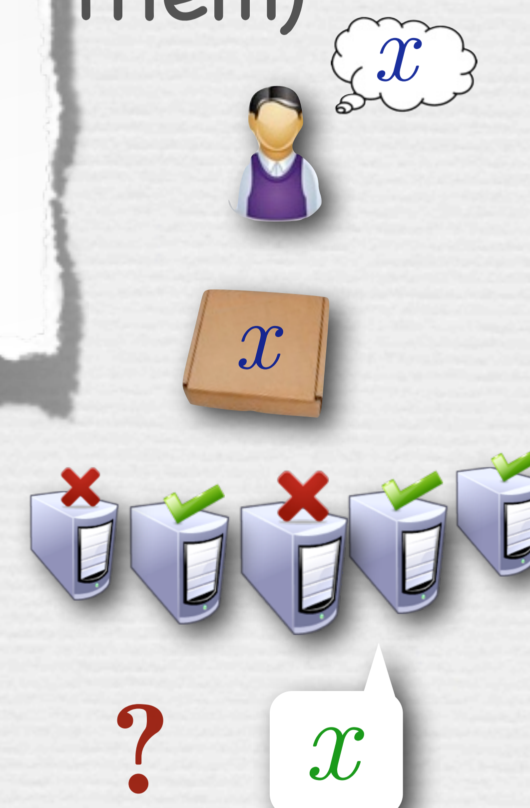
Tool: Homomorphic Threshold Encryption

"Encryption scheme" with special properties

Existence: Based on

- secret sharing
- mathematical structure in certain encryption schemes
- zero-knowledge proofs

- an encryption of $x+y$ can be computed



Tool: Homomorphic Threshold Encryption

"Encryption scheme" with special properties

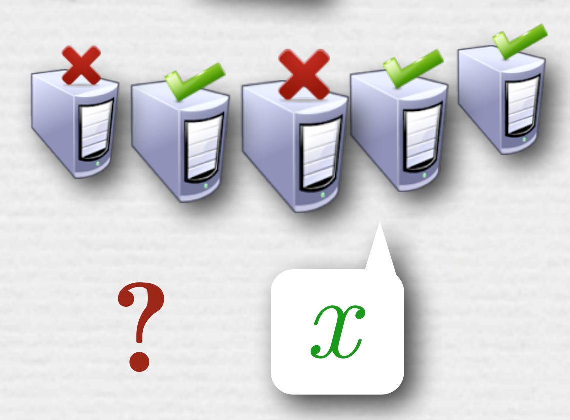
Thr

Hint: RSA is multiplicatively homomorphic

- Recall: $pk = (n, e)$ and $E_{pk}(a) = a^e \pmod{n}$
- Thus:

$$E_{pk}(a) \cdot E_{pk}(b) = a^e \cdot b^e = (a \cdot b)^e = E_{pk}(a \cdot b) \pmod{n}$$

- an encryption of $x+y$ can be computed

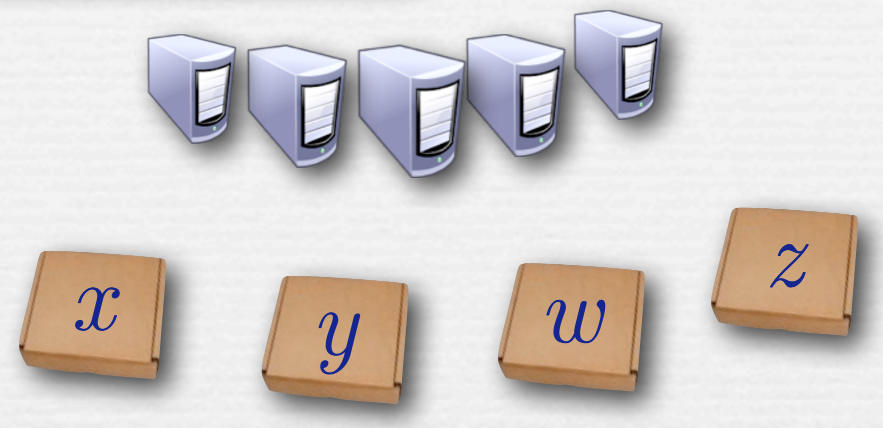


MPC in Action

$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$

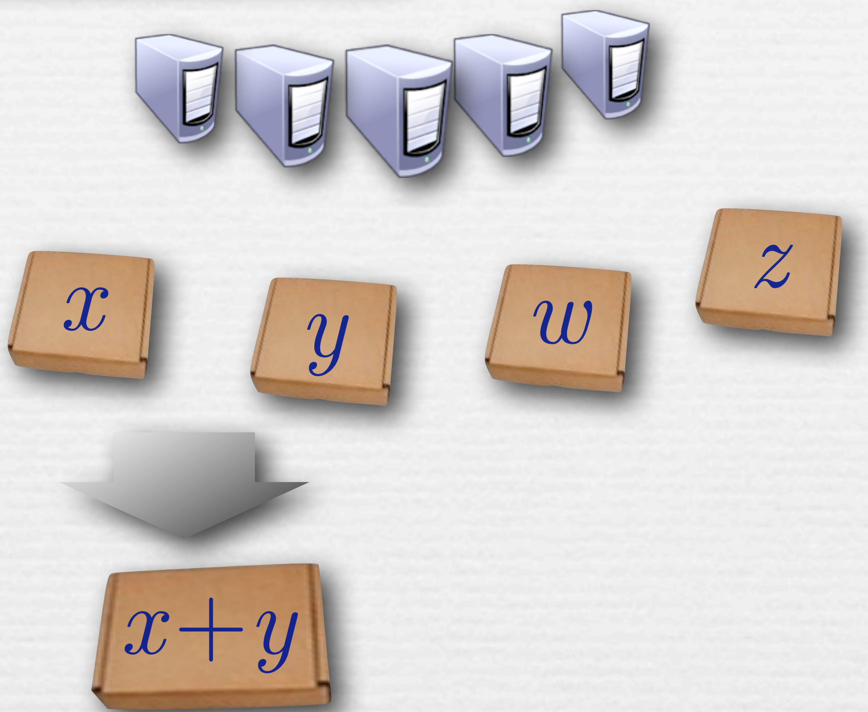
MPC in Action

$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



MPC in Action

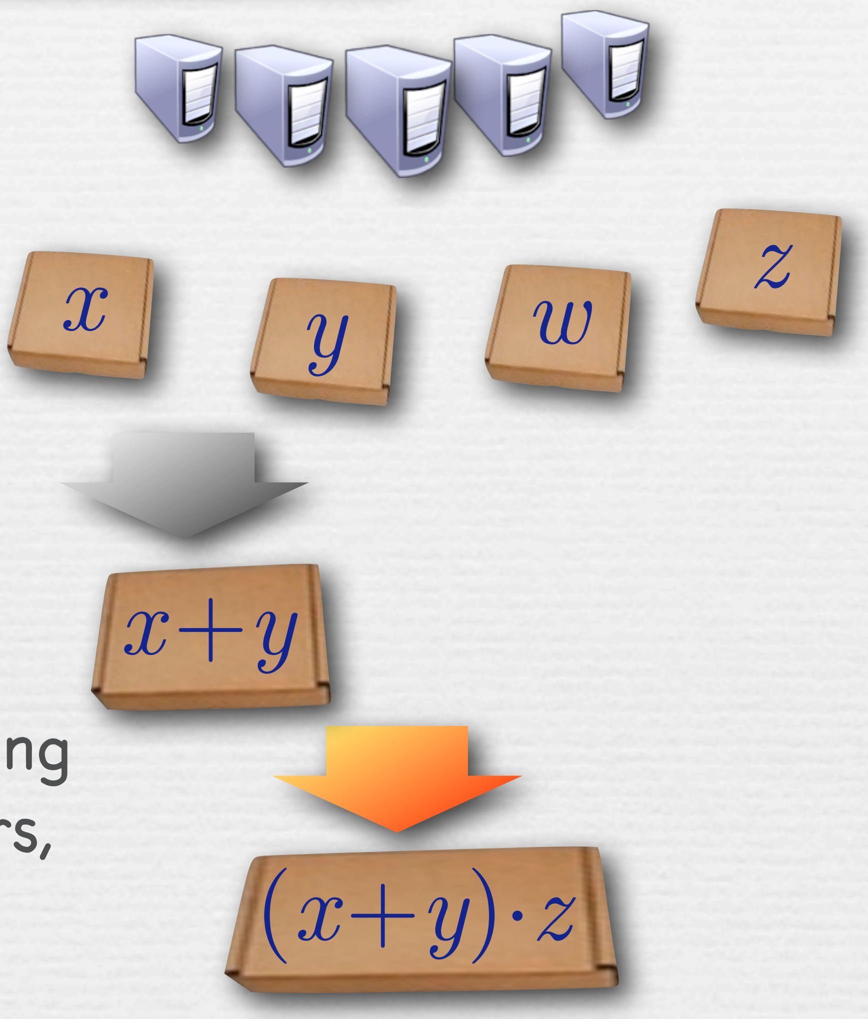
$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



homomorphic property

MPC in Action

$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



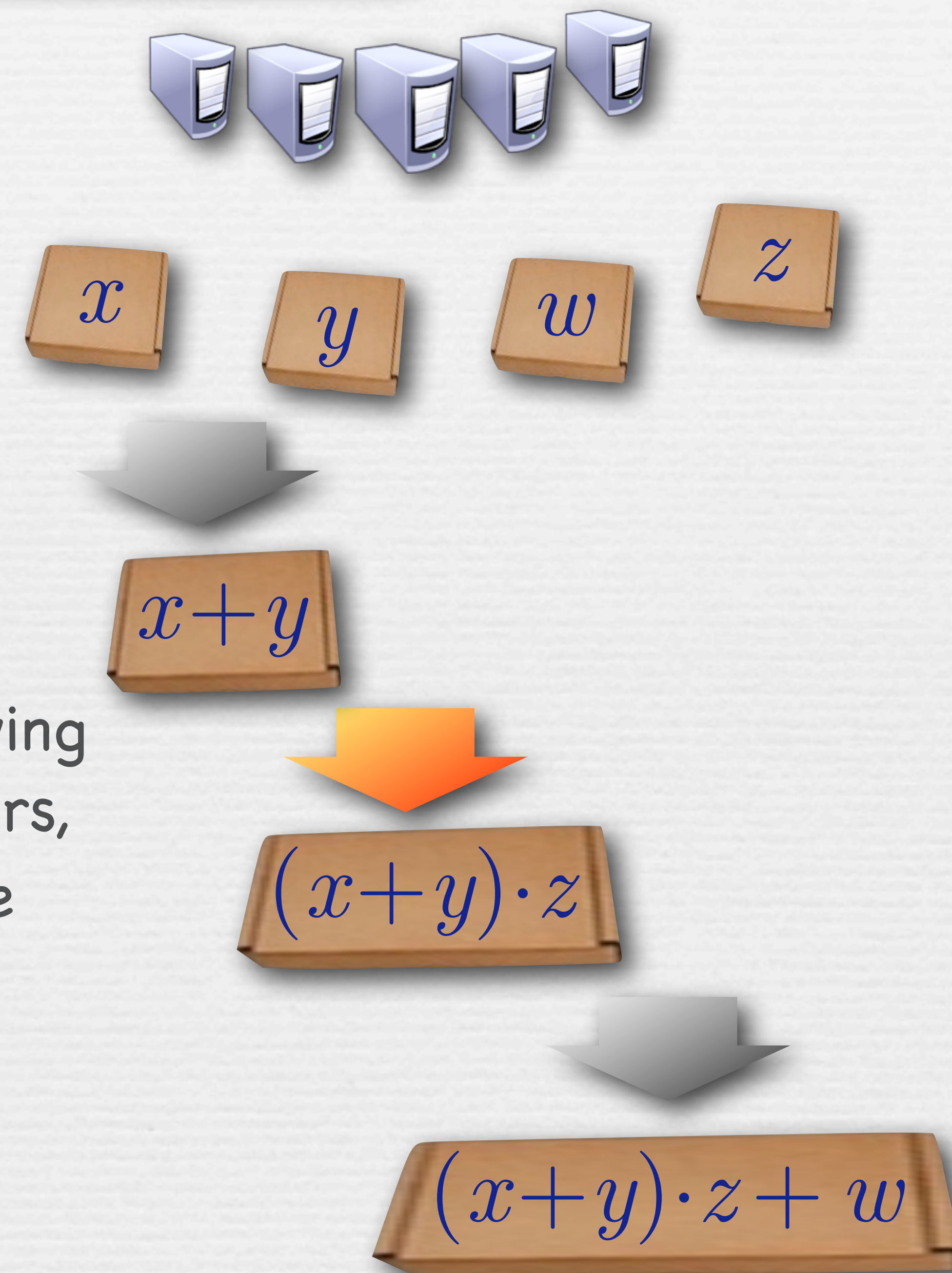
homomorphic property

Based on
- secret sharing
- zero-knowledge proofs
- mathematical structure

using a clever subprotocol, involving communication among the servers, or a fully homomorphic scheme

MPC in Action

$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



homomorphic property

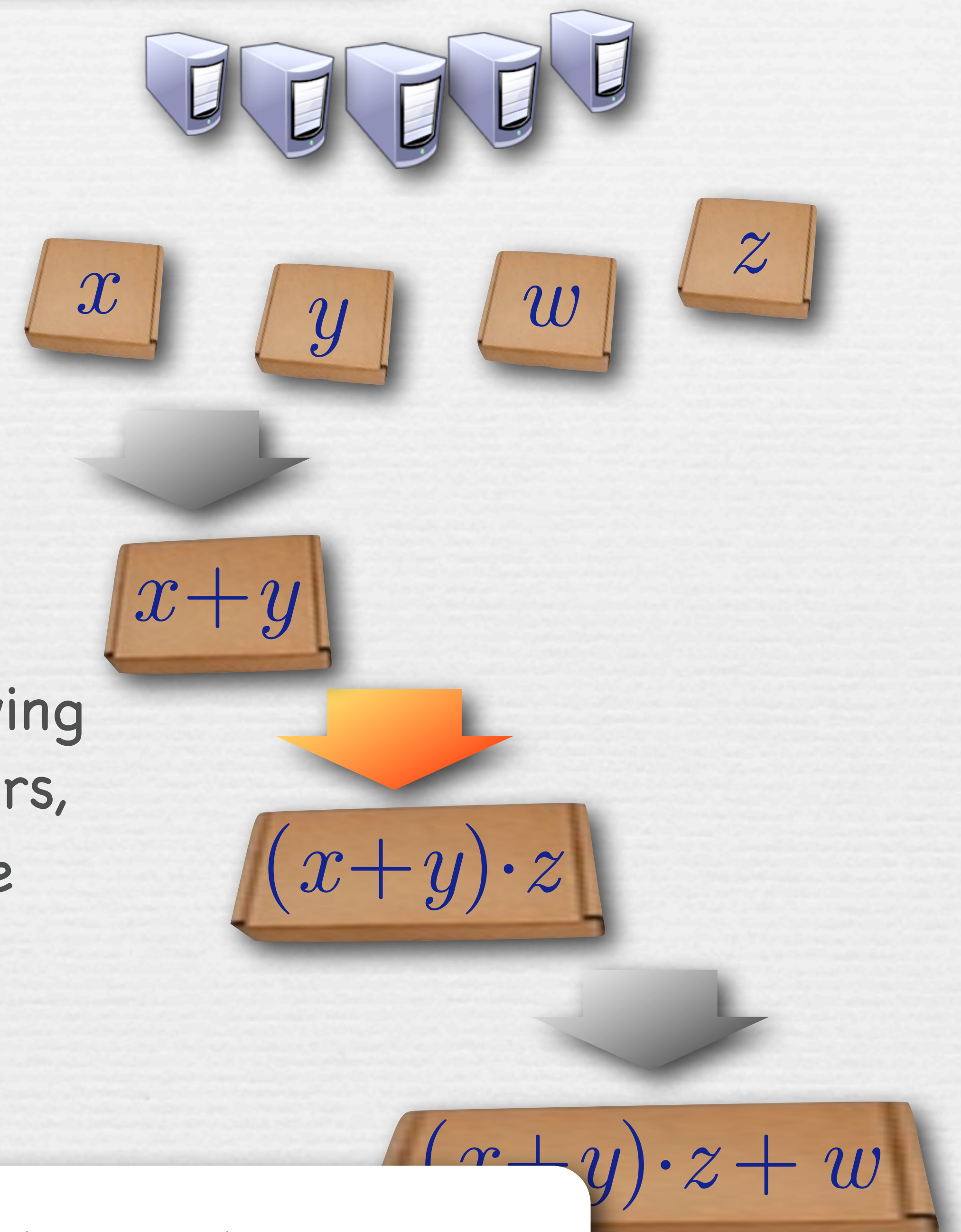
using a clever subprotocol, involving communication among the servers, or a fully homomorphic scheme

Based on
- secret sharing
- zero-knowledge proofs
- mathematical structure

homomorphic property

MPC in Action

$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



homomorphic property

using a clever subprotocol, involving communication among the servers, or a fully homomorphic scheme

homomorphic property

threshold property

$$(x+y)\cdot z + w$$

Based on

- secret sharing
- zero-knowledge proofs
- mathematical structure

Remarks

- Above is general blueprint
- Exist lots of variations and different instantiations
- Exist conceptually different approaches
- Choice will depend on
  - exact setting
  - exact goal(s)
  - exact requirements
  - etc.
- Strong theoretical possibility results
- Transition phase: from theory to practice

Summary

MPC offers a solution whenever

- parties have common goal
- yet conflicting interests

MPC has great potential:

- very general, and thus broadly applicable
- offers strong security guarantees
- replaces **any** (imaginary) trusted party – in principle

Efficiency:

- general MPC used to be considered impractical
- early schemes were very inefficient
- great progress in recent years
- special purpose solutions can be made practical