CWI Centrum Wiskunde & Informatica

Secure Multiparty Computation (MPC)

Serge Fehr

CWI Amsterdam

www.cwi.nl/~fehr

Meeting on Privacy-Enhancing Cryptography

December 8 & 9, 2011

Outline

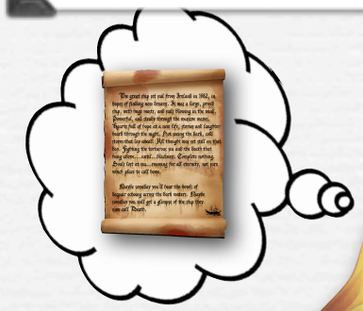
- Intro and problem description
- Possibility result
- High-level idea

Encryption and more

ALICE

BOB

Encryption and more

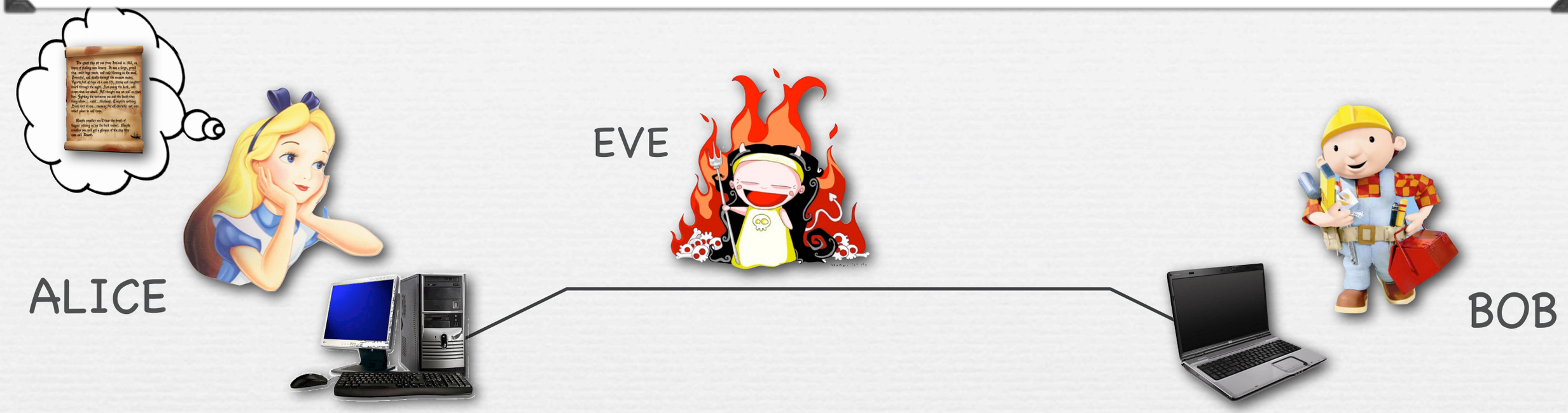


ALICE

EVE

BOB

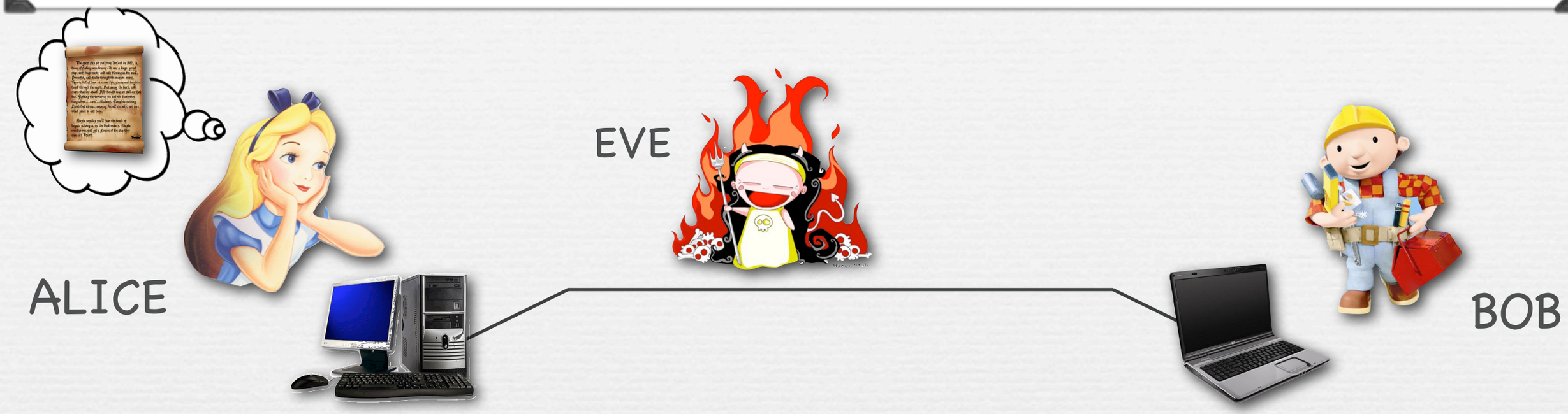
Encryption and more



Eve can:

- eavesdrop the communication
  -> use encryption (symmetric or public-key)

Encryption and more



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 co

- 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

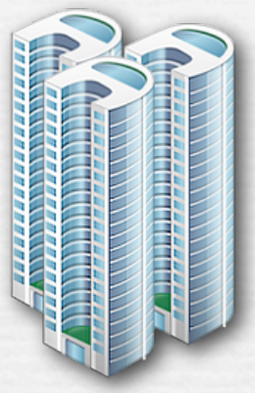
- eav
  ->
- moc
  -> use authentication or digital signatures

But:

The world is not just **black** and 

Examples

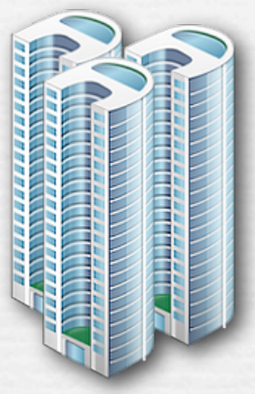
Company A 

 Company B

- A and B want to compare their performance

Examples

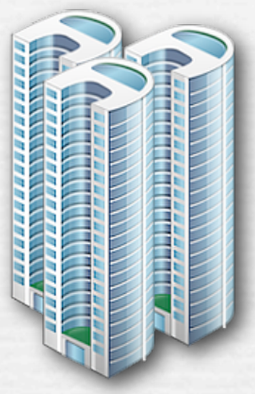
Company A 

 Company B

- A and B want to compare their performance
- Neither is willing to reveal its detailed performance data

Examples

Company A 

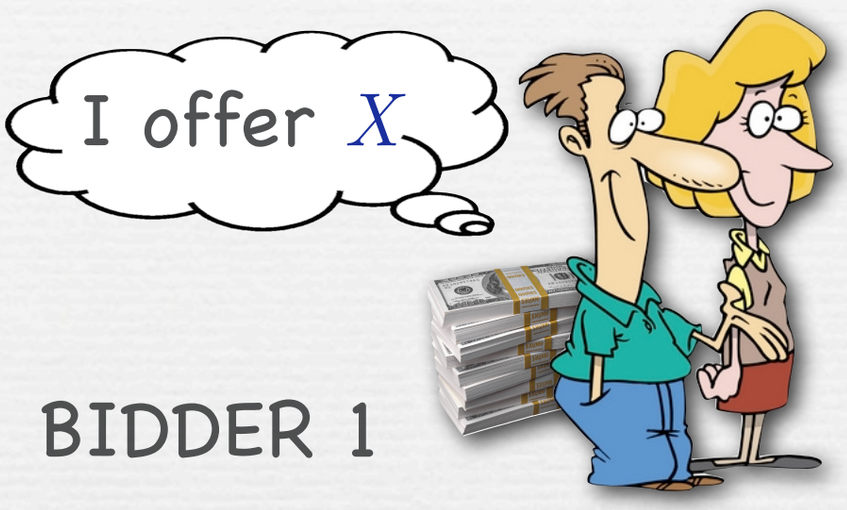
 Company B

- A and B want to compare their performance
- Neither is willing to reveal its detailed performance data

OR

- A and B want to find the overlap in customers
- Neither is willing to reveal its own customer list

Examples



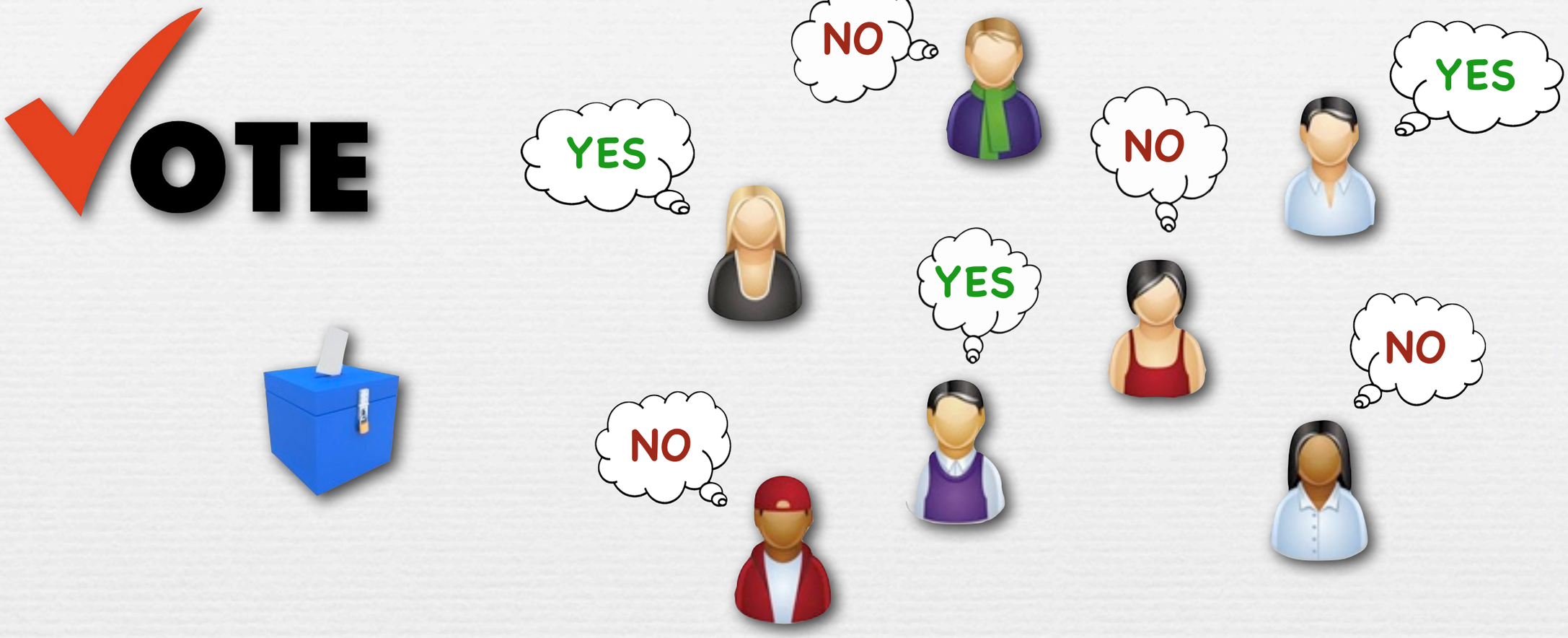
BIDDER 1

BIDDER 2

SELLER

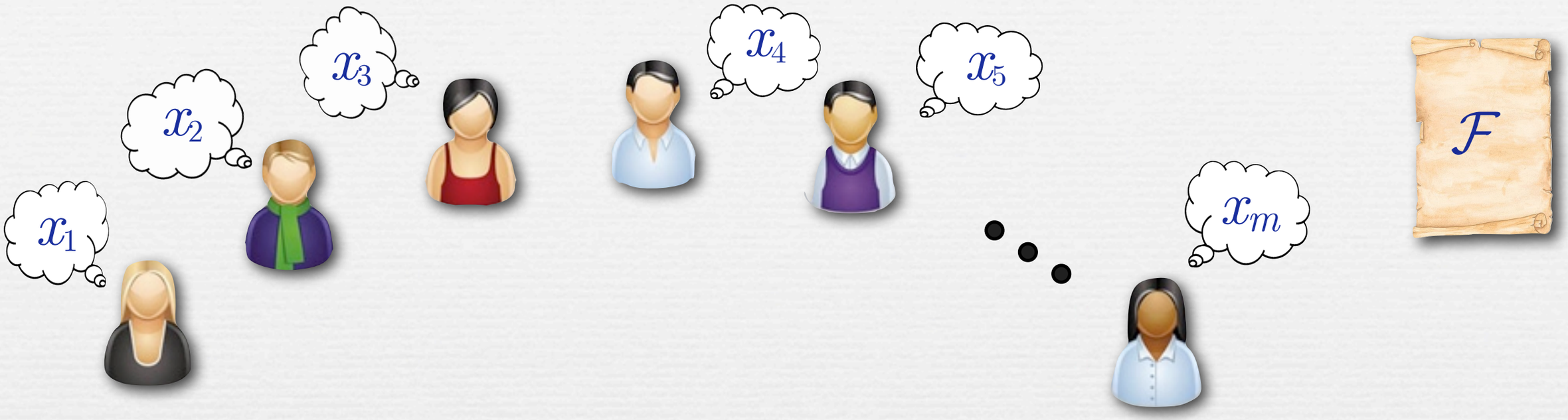
- Want to find out if bids are sufficient and who bids more, and e.g. agree on $\max\{W, \min\{X, Y\}+1\}$ as price.
- No one is willing to reveal his upper/lower bound.

Examples



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

The General Problem



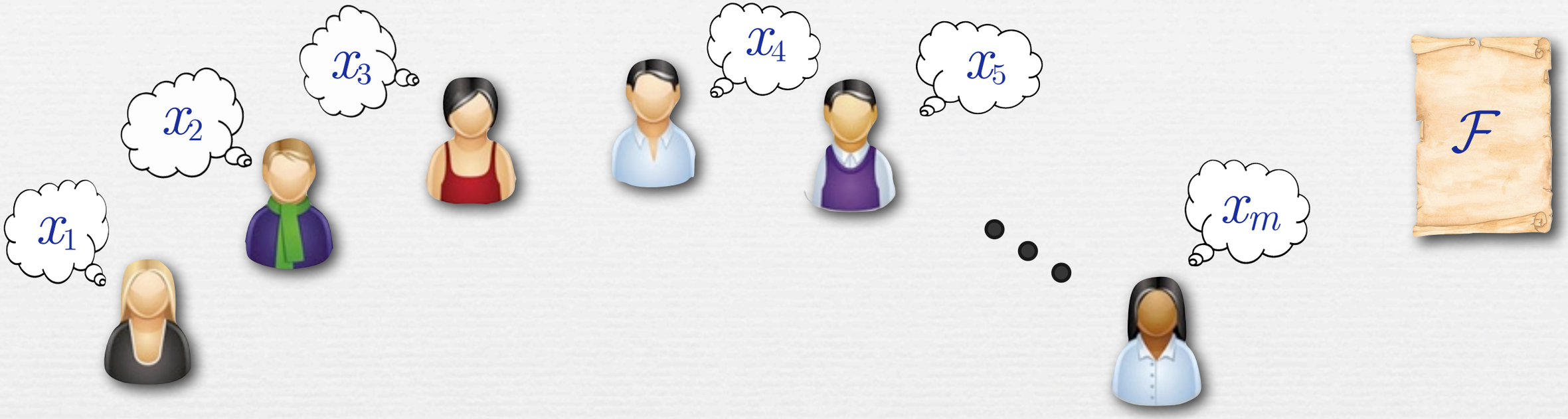
- Every user U_i has a private input x_i .

The General Problem

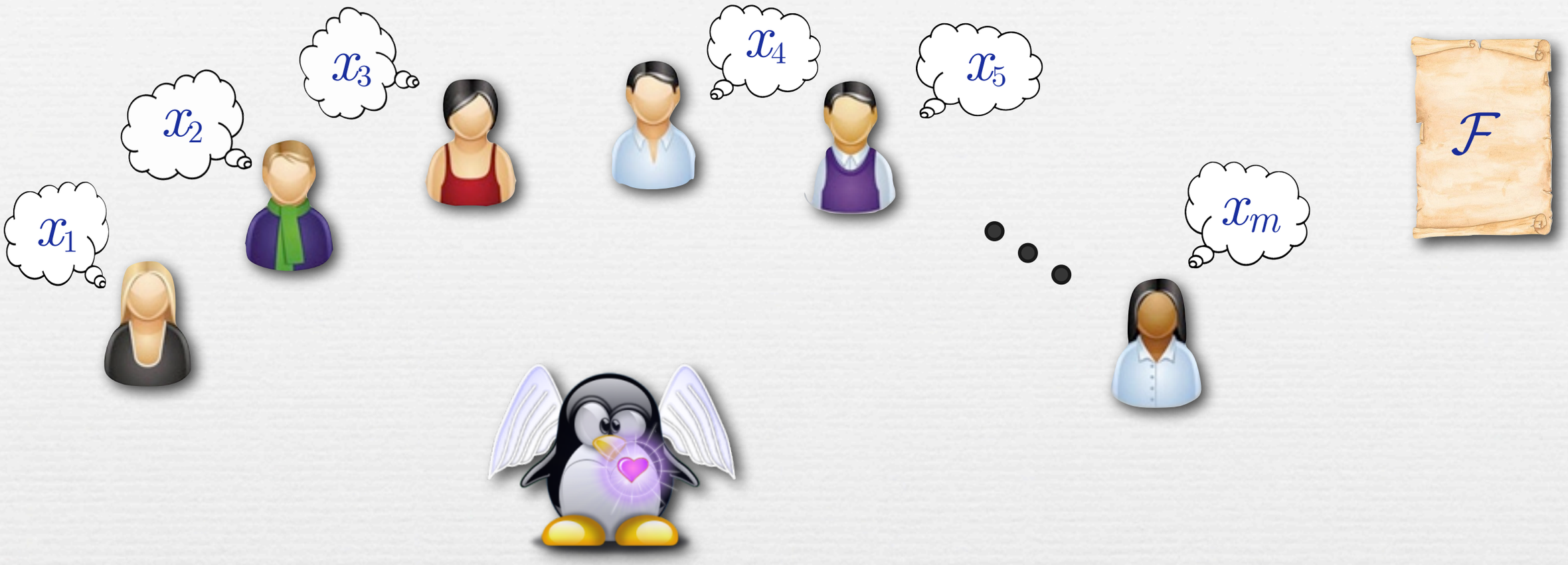
$$\mathcal{F}(x_1, x_2, x_3, \ldots, x_m) = ?$$

- 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.

An Ideal Solution

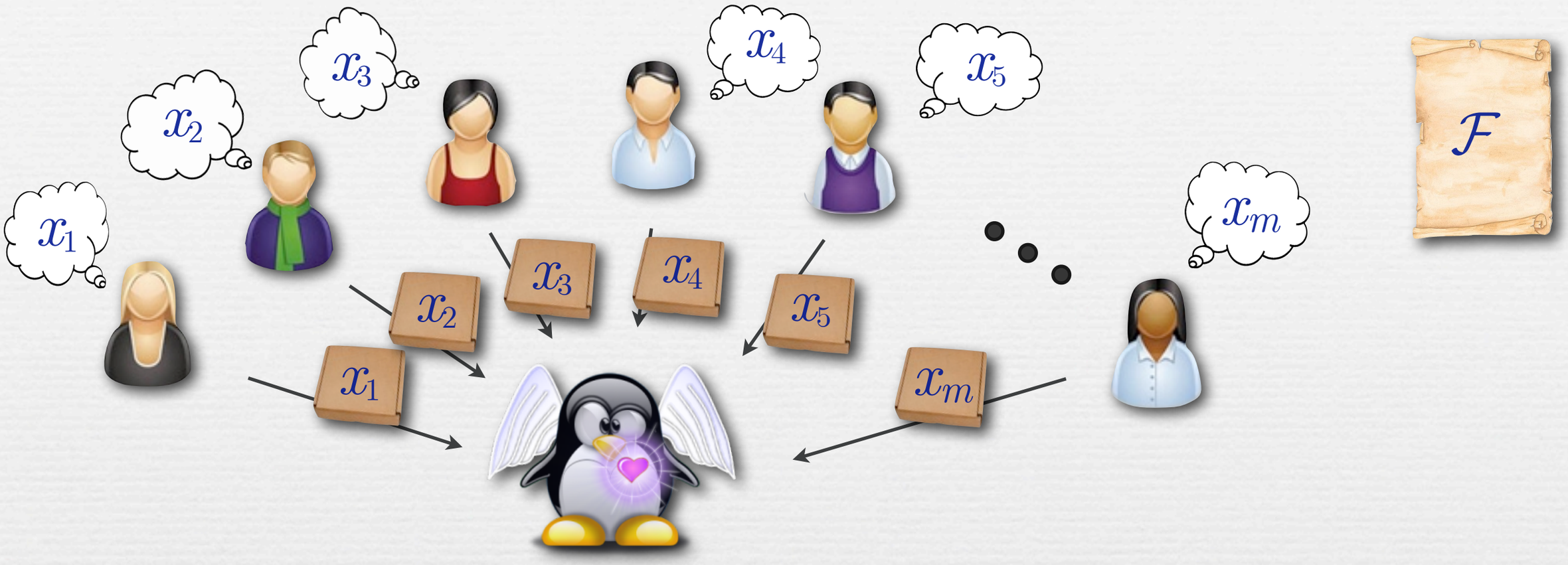


An Ideal Solution



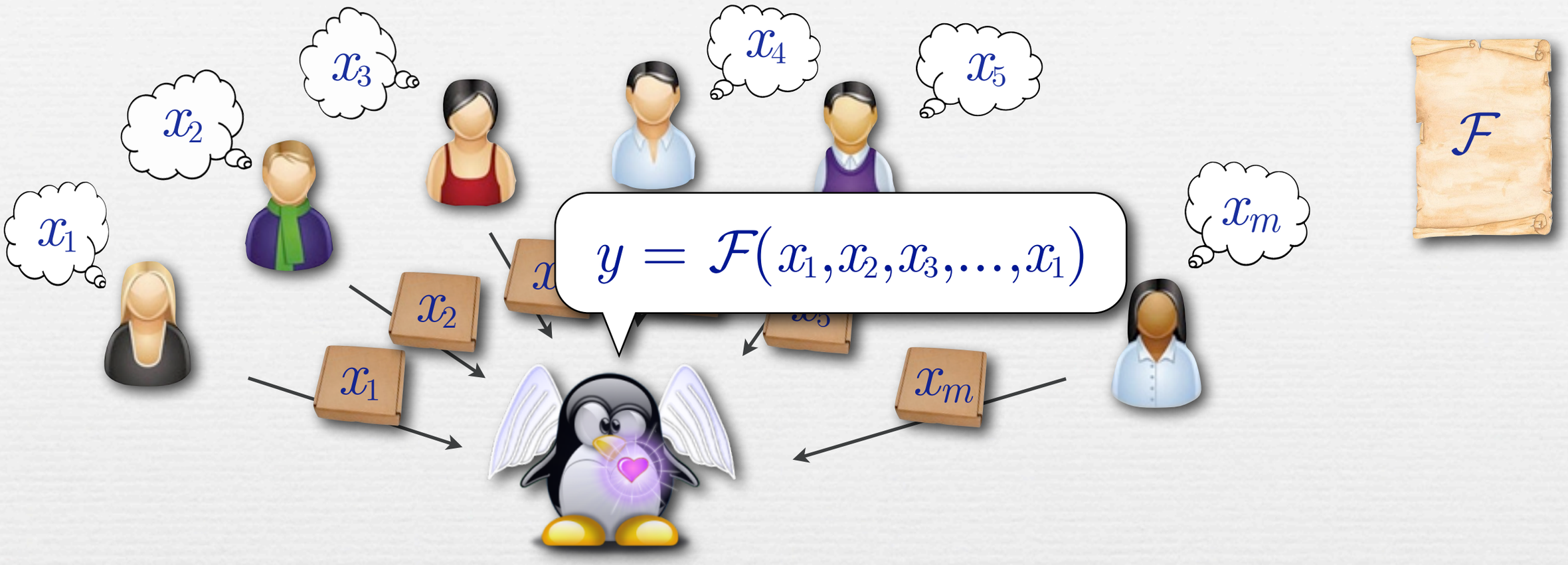
trusted authority TA

An Ideal Solution



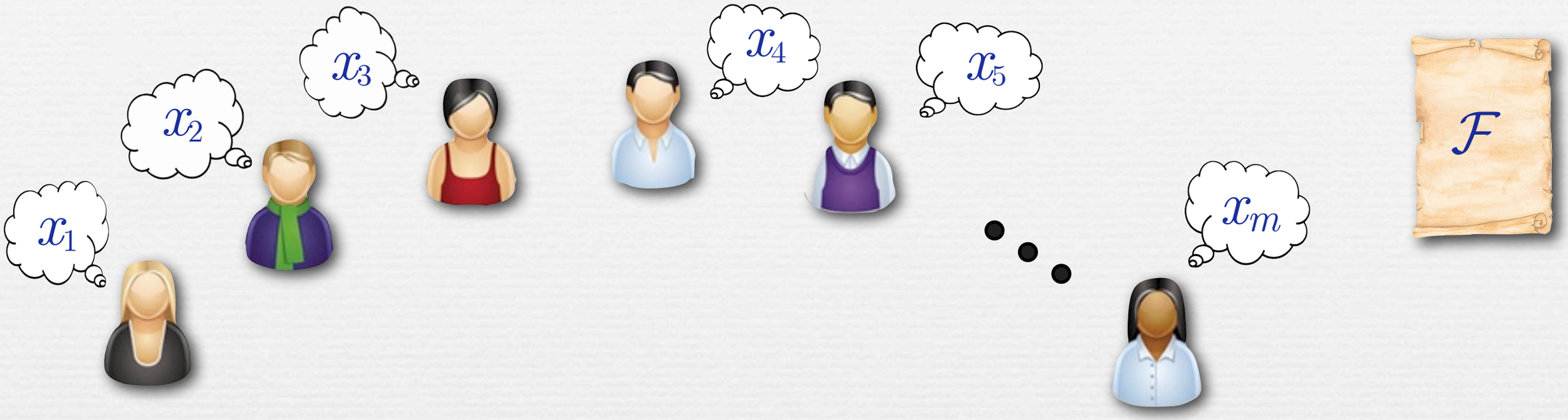
- Every user U_i sends his x_i to trusted authority TA.

An Ideal Solution

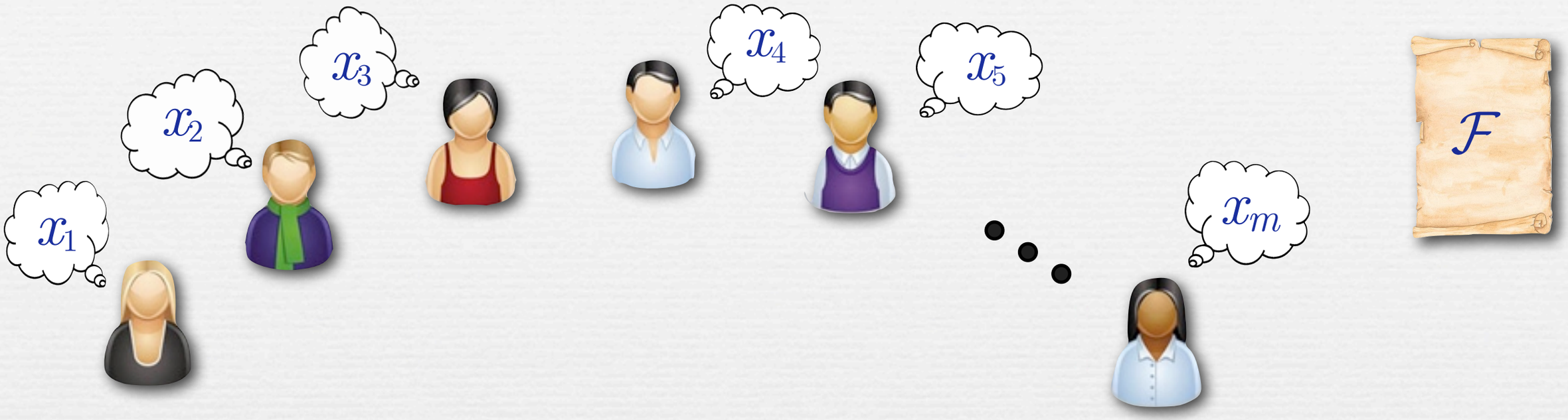


- Every user U_i sends his x_i to trusted authority TA.
- TA computes $y = \mathcal{F}(x_1, x_2, x_3, \ldots, x_1)$, and
- announces y to everyone.

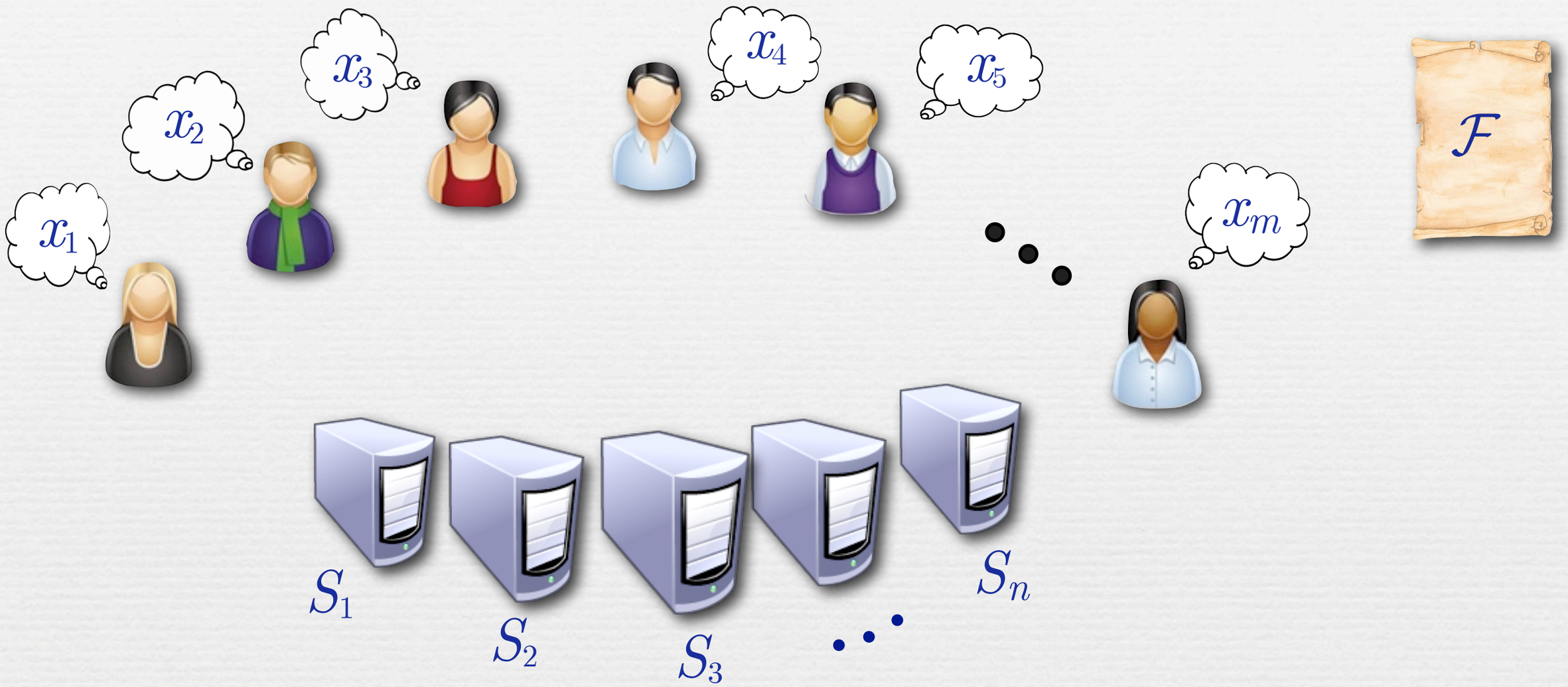
MPC: Removing the Trusted Authority



MPC: Removing the Trusted Authority



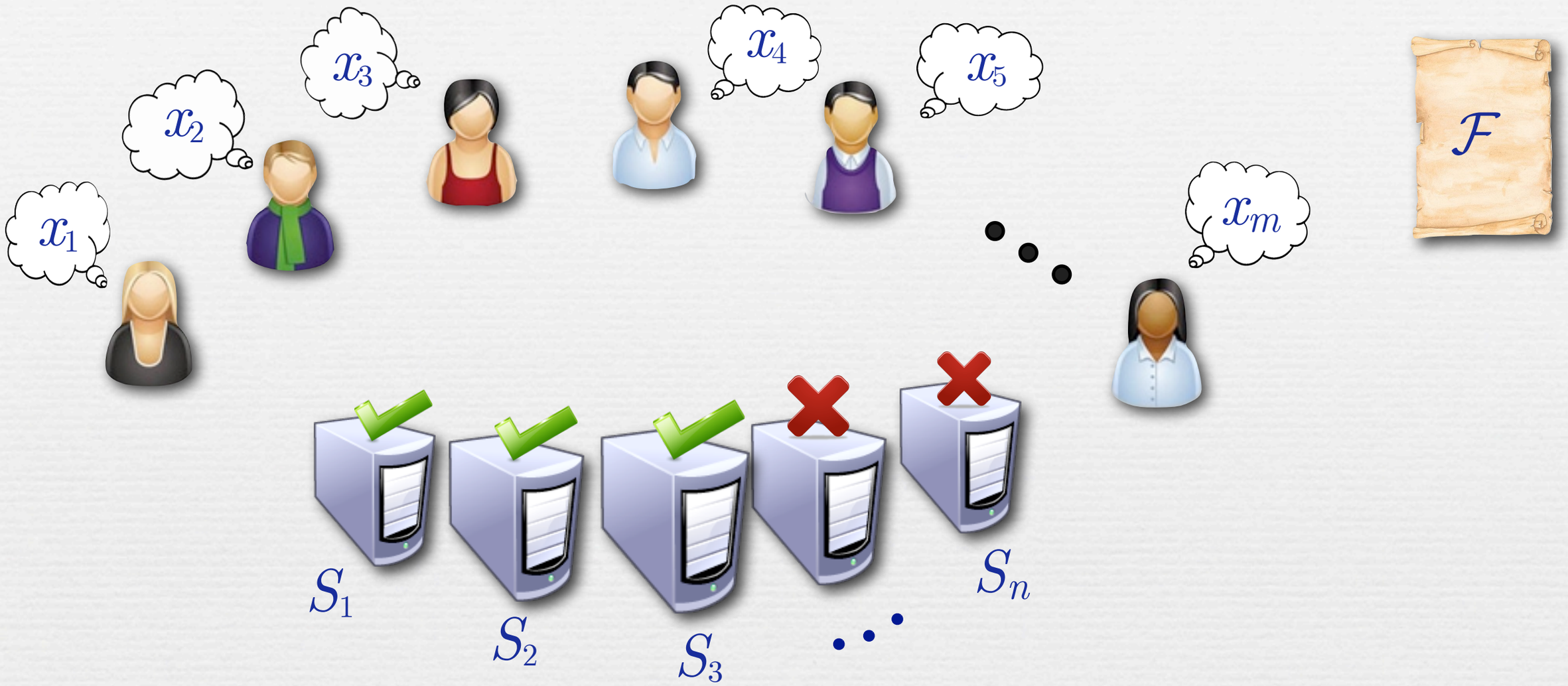
MPC: Removing the Trusted Authority



Idea:

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

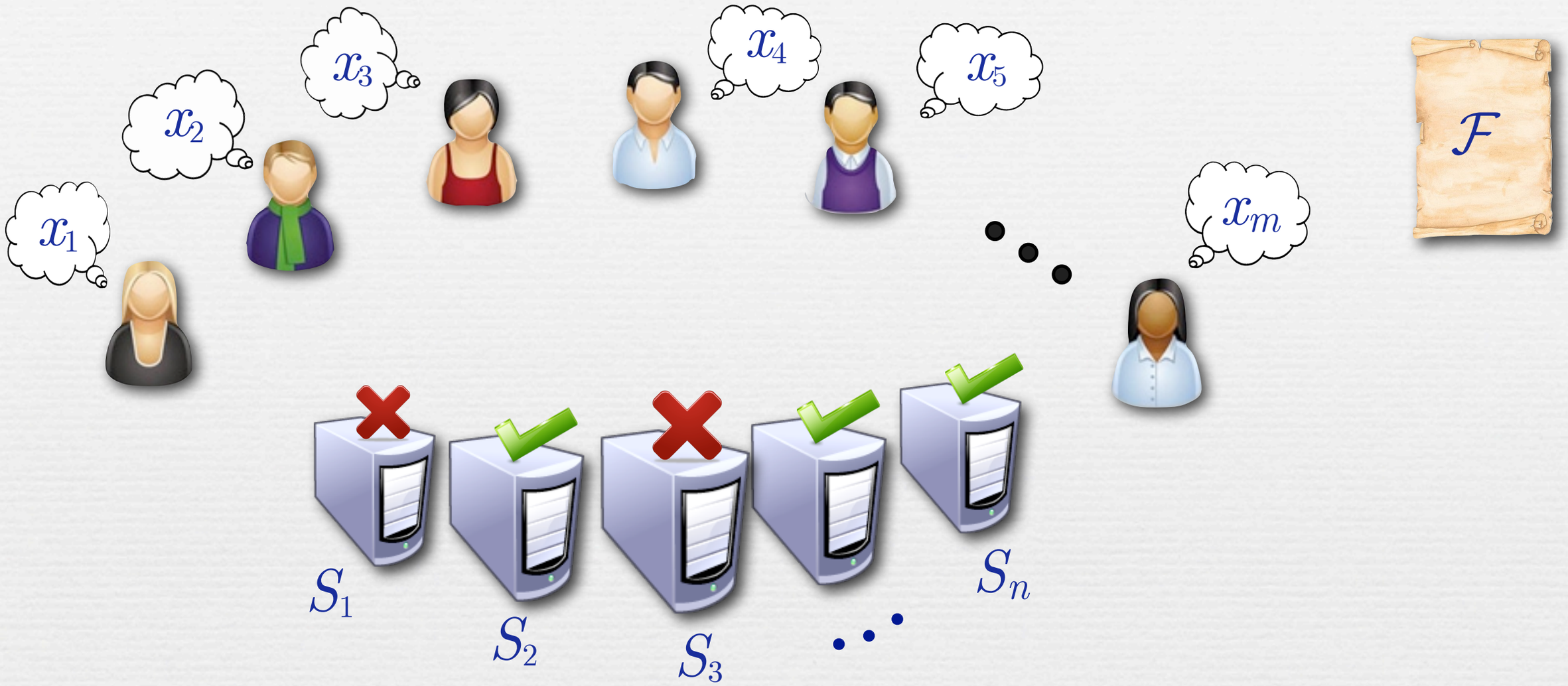
MPC: Removing the Trusted Authority



Idea:

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

MPC: Removing the Trusted Authority



Idea:

- Perform computation by a group of servers.
- 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.

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.

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:

- sufficiently many 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:

- sufficiently many 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:

- sufficiently many servers are honest.

Outline

- Intro and problem description
- Possibility result
- High-level idea

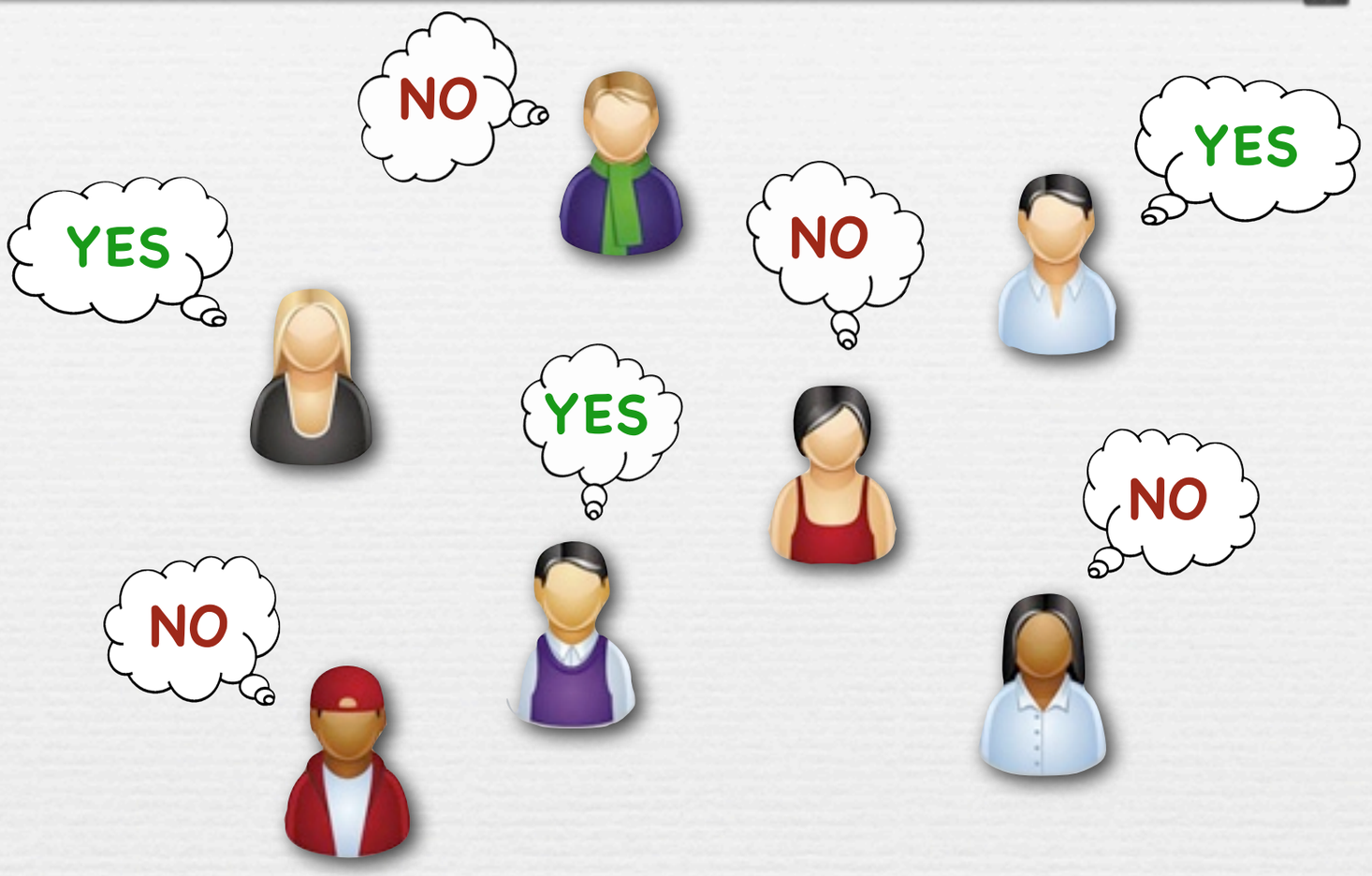
Possibility of MPC

Under reasonable set-up assumptions (e.g. PKI),
general secure MPC is possible if (and only if)
a **majority** of the servers are honest,
i.e., $t < n/2$ of the n servers are malicious.

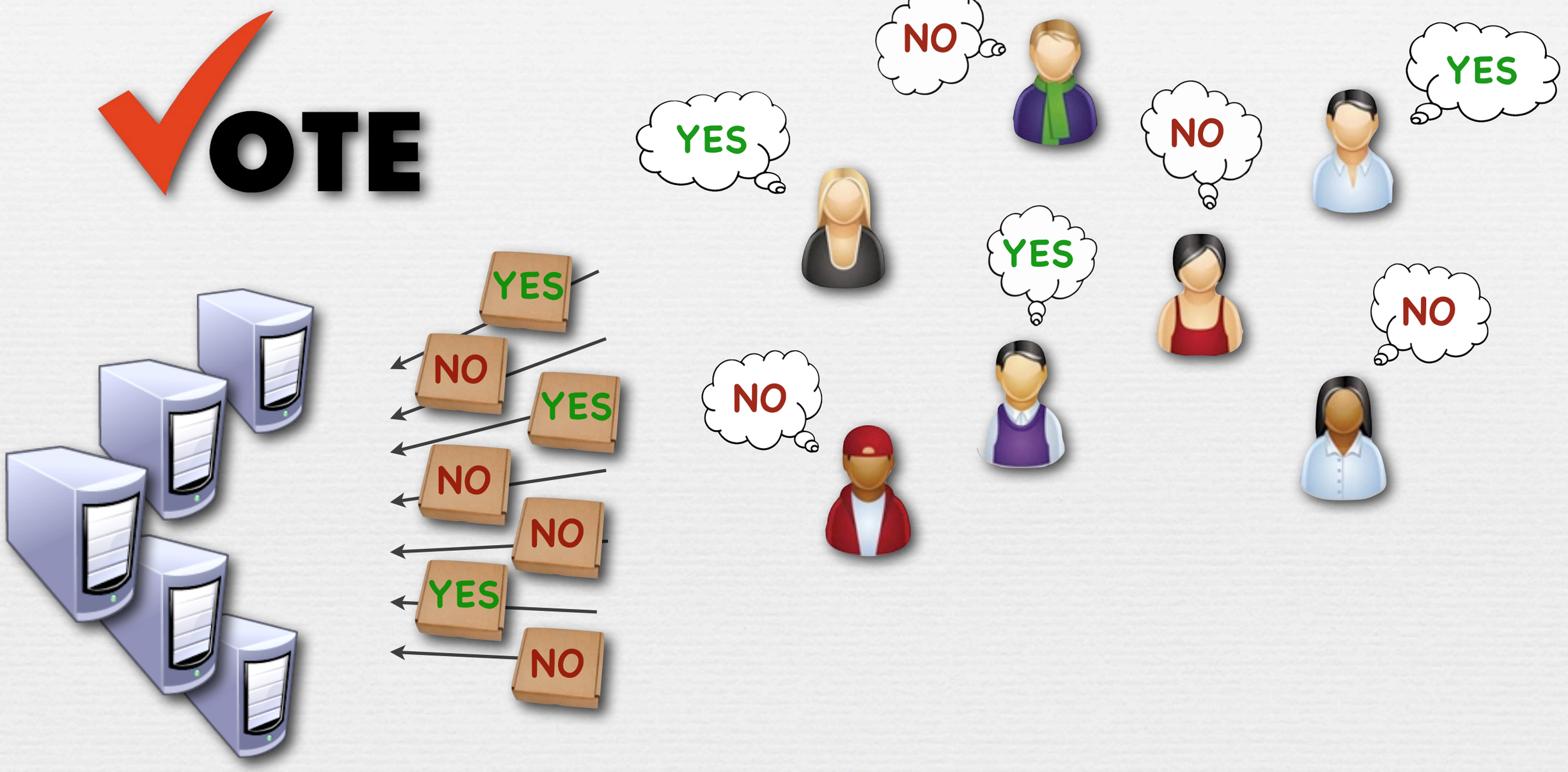
Exist many different variants which differ in:

- flavors of security
- set-up assumptions
- complexity
- # of malicious servers
- communication model
- etc.

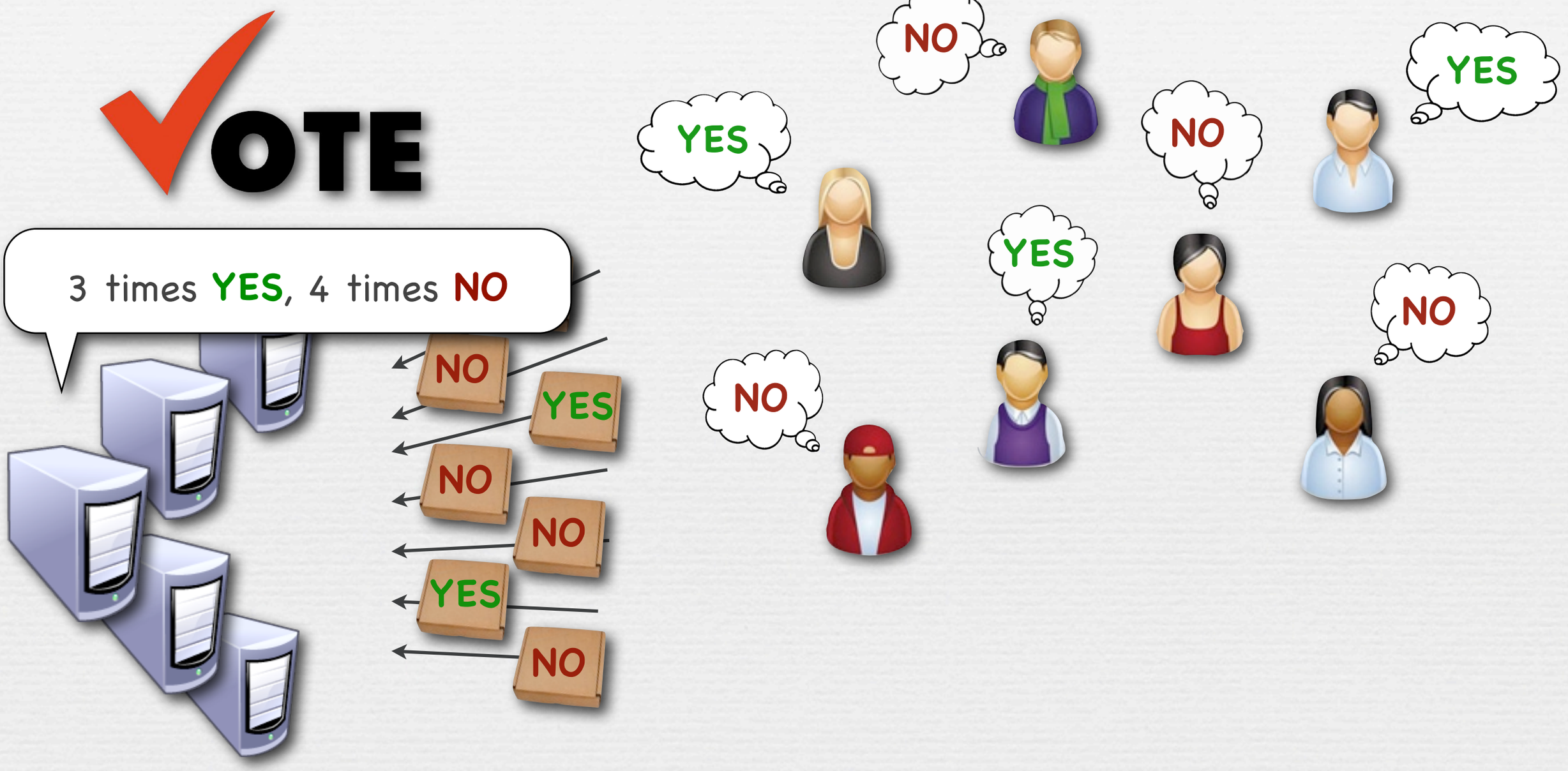
Example



Example



Example



Promise:

- Votes remain private and tally is guaranteed correct
- If a majority of servers is honest.

Outline

- Intro and problem description
- Possibility result
- High-level idea

Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

- Decryption key is ``shared'' among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt

(even if a malicious minority tries to prevent them)

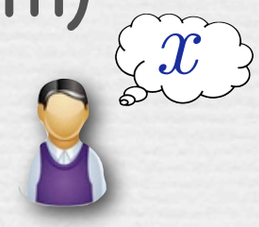
Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

- Decryption key is ``shared'' among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt

(even if a malicious minority tries to prevent them)



Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

- Decryption key is ``shared'' among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt

(even if a malicious minority tries to prevent them)

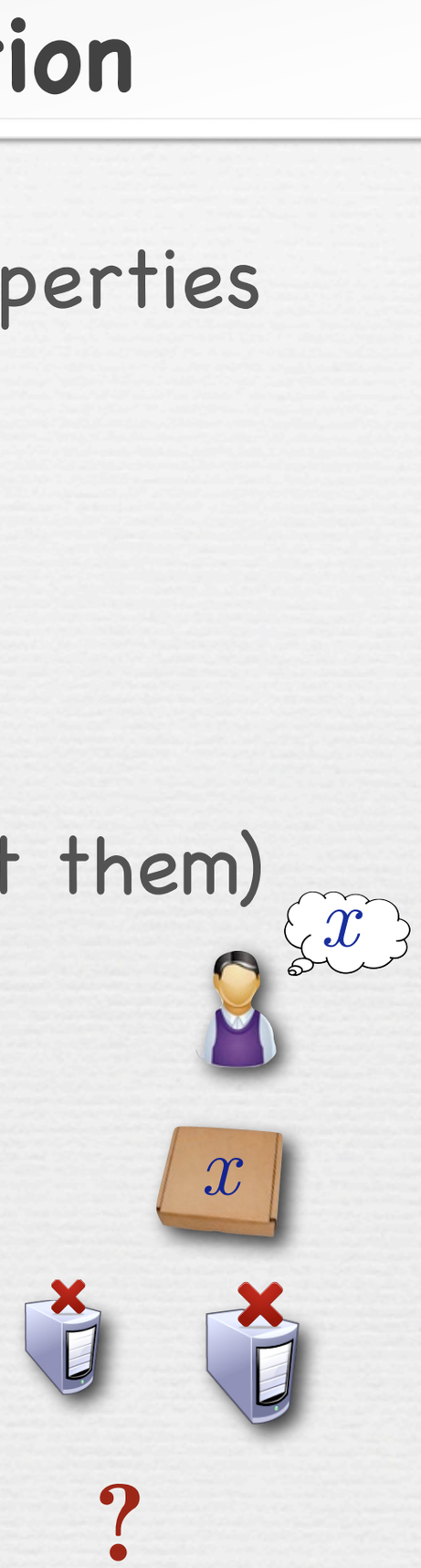
Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

- Decryption key is ``shared'' among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt

(even if a malicious minority tries to prevent them)



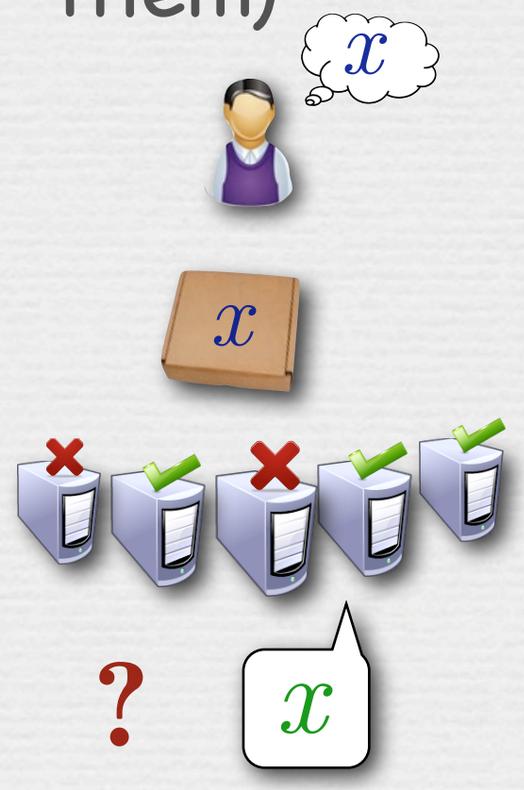
Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

- Decryption key is ``shared'' among servers.
- A malicious minority cannot decrypt
- All servers together can decrypt

(even if a malicious minority tries to prevent them)



Tool: Homomorphic Threshold Encryption

Public-key encryption scheme with special properties

Threshold:

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

Homomorphic:

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

Tool: Homomorphic Threshold Encryption

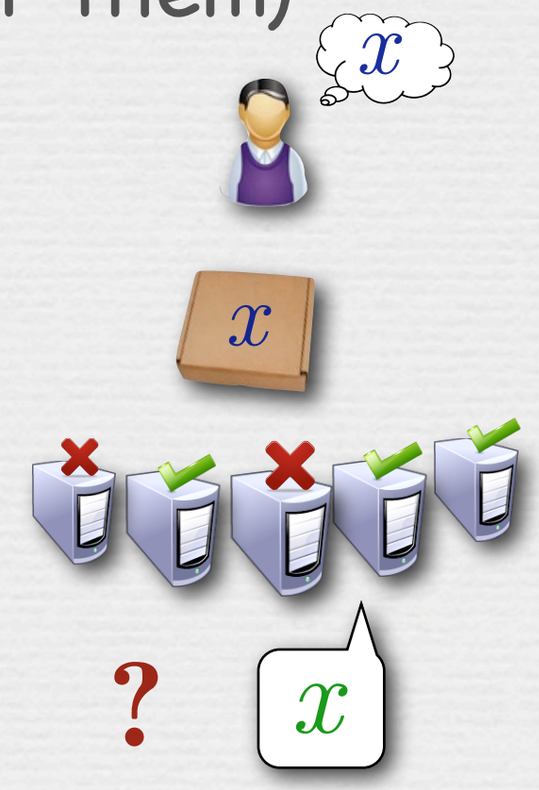
Public-key encryption scheme with special properties

Threshold:

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

Homomorphic:

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



Tool: Homomorphic Threshold Encryption

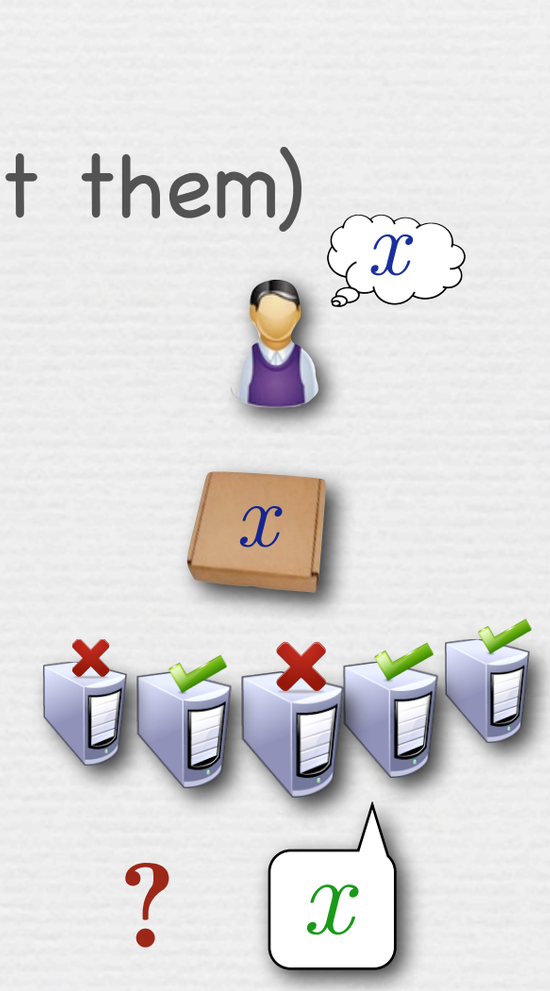
Public-key encryption scheme with special properties

Threshold:

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

Homomorphic:

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

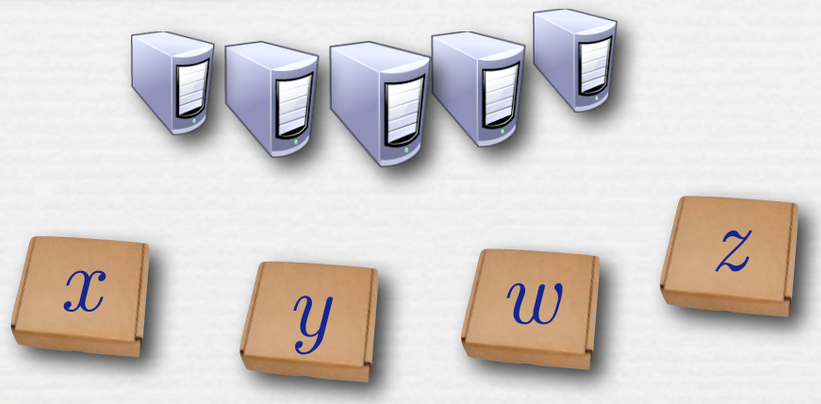


MPC in Action

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

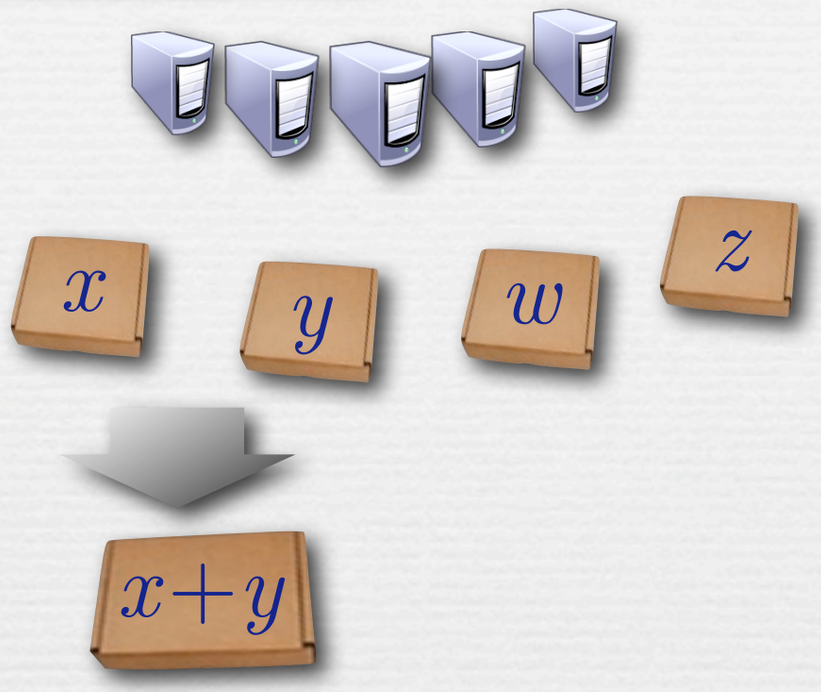
MPC in Action

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



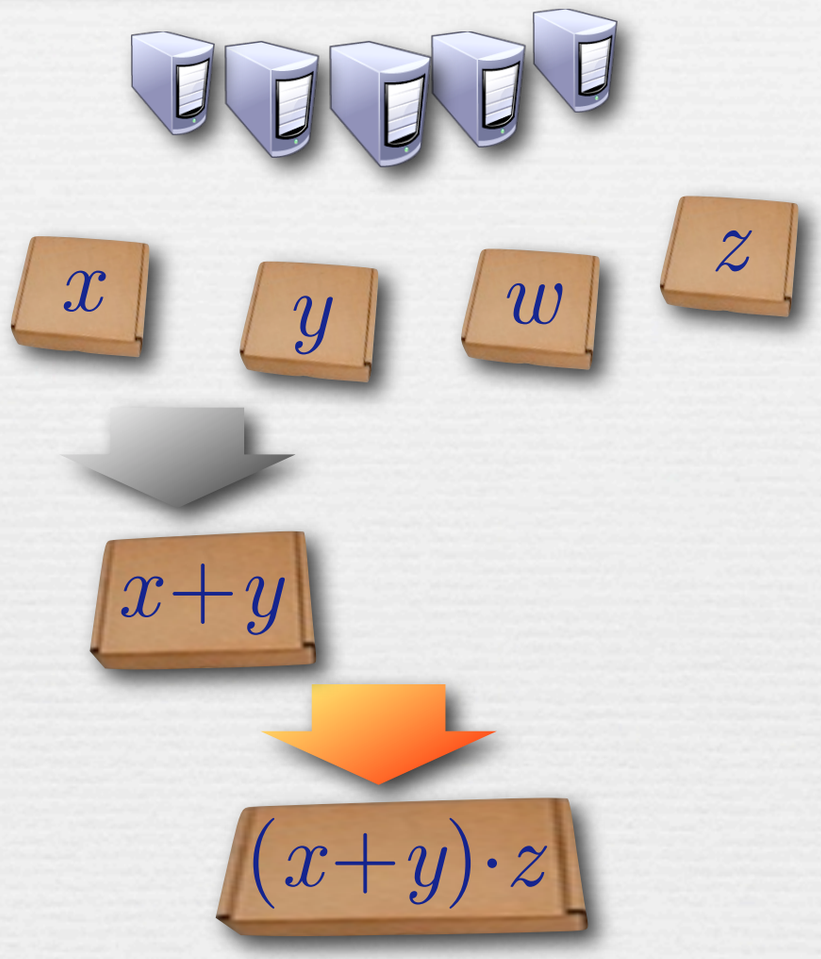
MPC in Action

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



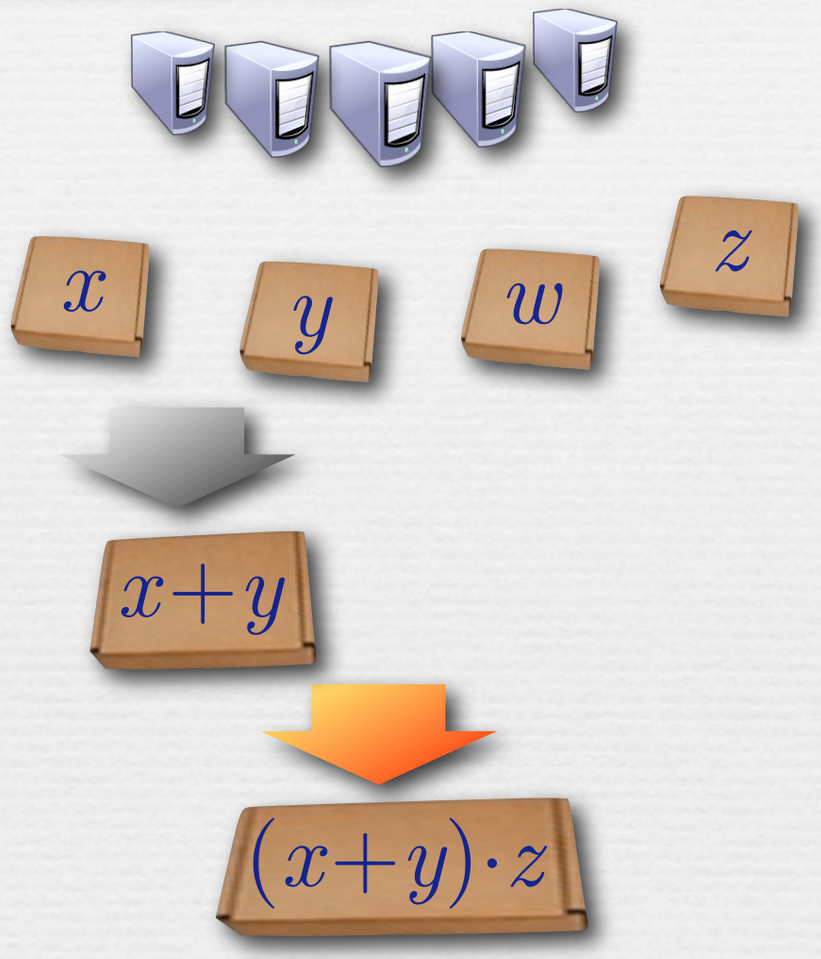
MPC in Action

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



MPC in Action

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

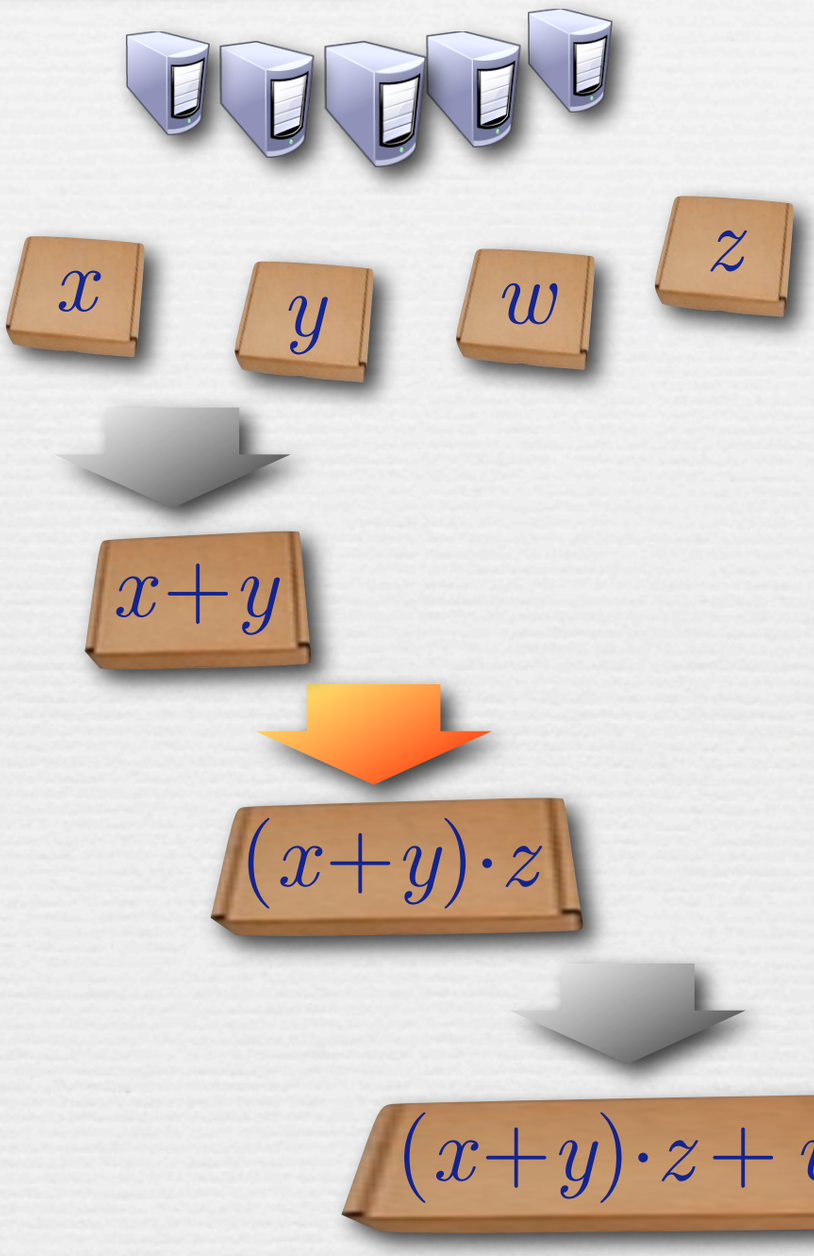


homomorphic property

complex subprotocol, involving
communication among the servers

MPC in Action

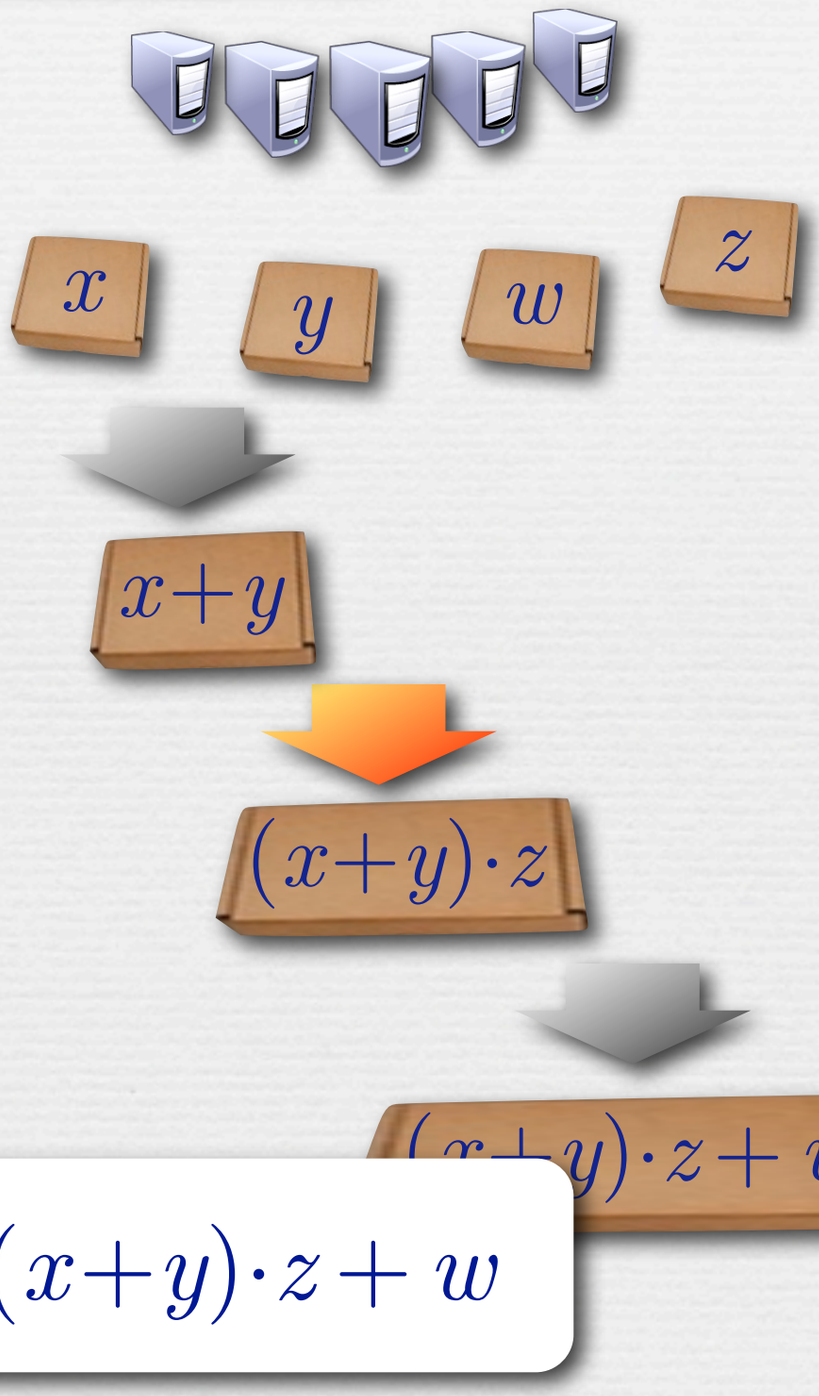
$$\mathcal{F}(x,y,w,z) = (x+y)\cdot z + w$$



complex subprotocol, involving
communication among the servers

MPC in Action

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



Summary

MPC is useful when

- parties have common goal yet conflicting interests
- it is unclear whom we can trust
- there is no fully trusted party available

Summary

MPC is useful when

- parties have common goal yet conflicting interests
- it is unclear whom we can trust
- there is no fully trusted party available

Downside: general solutions are rather inefficient

But: special purpose solutions can be reasonably efficient
(see next talk by Tomas Toft)

Summary

MPC is useful when

- parties have common goal yet conflicting interests
- it is unclear whom we can trust
- there is no fully trusted party available

Downside: general solutions are rather inefficient

But: special purpose solutions can be reasonably efficient
(see next talk by Tomas Toft)

THANK YOU