The Rush Dilemma: Attacking and Repairing Smart Contracts on Forking Blockchains

Vincenzo Botta
DIEM
University of Salerno

Daniele Friolo
Sapienza
University of Rome

Daniele Venturi
Sapienza
University of Rome

Ivan Visconti
DIEM
University of Salerno

Abstract use of a Blockchain

- There is a decentralized list of blocks that can be extended

Abstract use of a Blockchain

- There is a decentralized list of blocks that can be extended
- The past is immutable

Abstract use of a Blockchain

- There is a decentralized list of blocks that can be extended
- The past is immutable
- Eventually, all transactions are added to a block

Many scientific results follow this abstraction and many implementations assume that the specific Blockchain used behaves like that

Abstract use of a Blockchain

- There is a decentralized list of blocks that can be extended
- The past is immutable
- Eventually, all transactions are added to a block

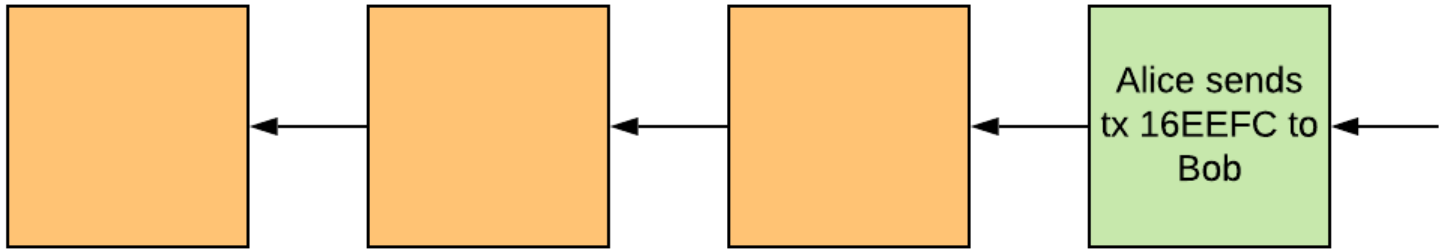
Many scientific results follow this abstraction and many implementations assume that the specific Blockchain used behaves like that

However, reality can be different…

Double Spending Attack

Double spending attack:

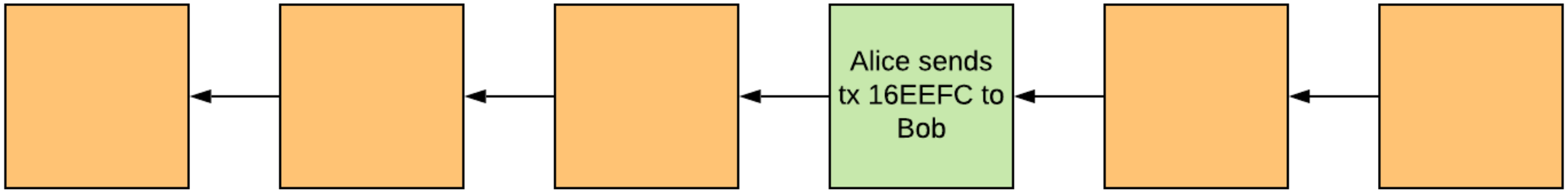
same money used in different transactions



Double Spending Attack

Double spending attack:

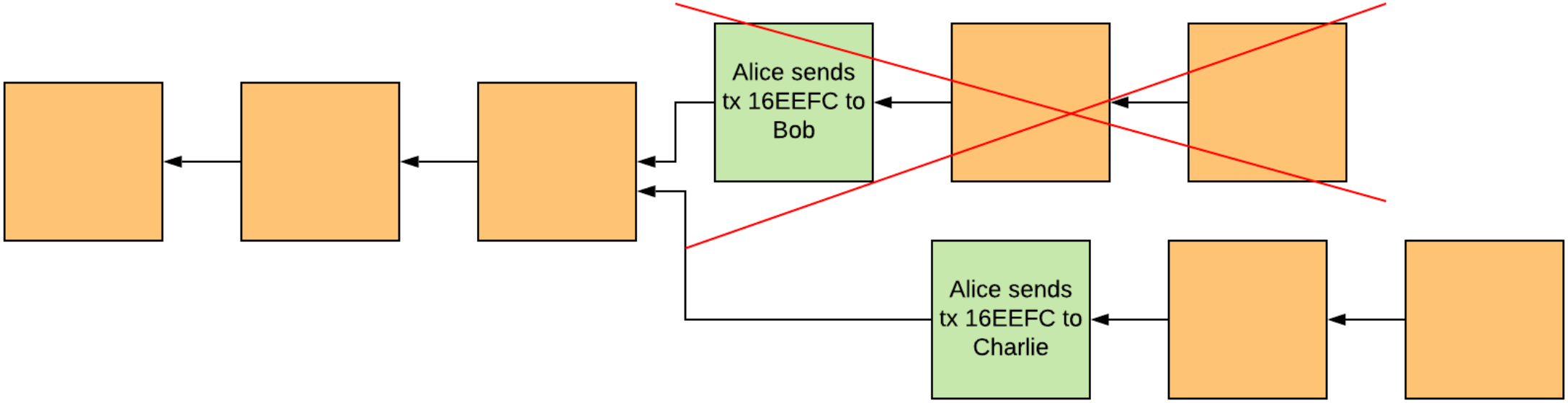
same money used in different transactions



Double Spending Attack

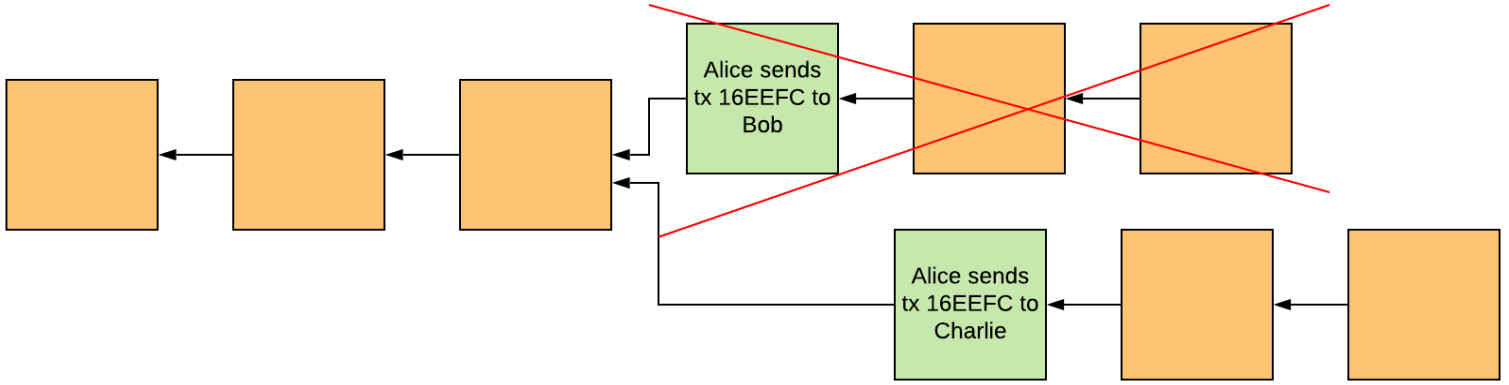
Double spending attack:

same money used in different transactions



Double Spending Attack

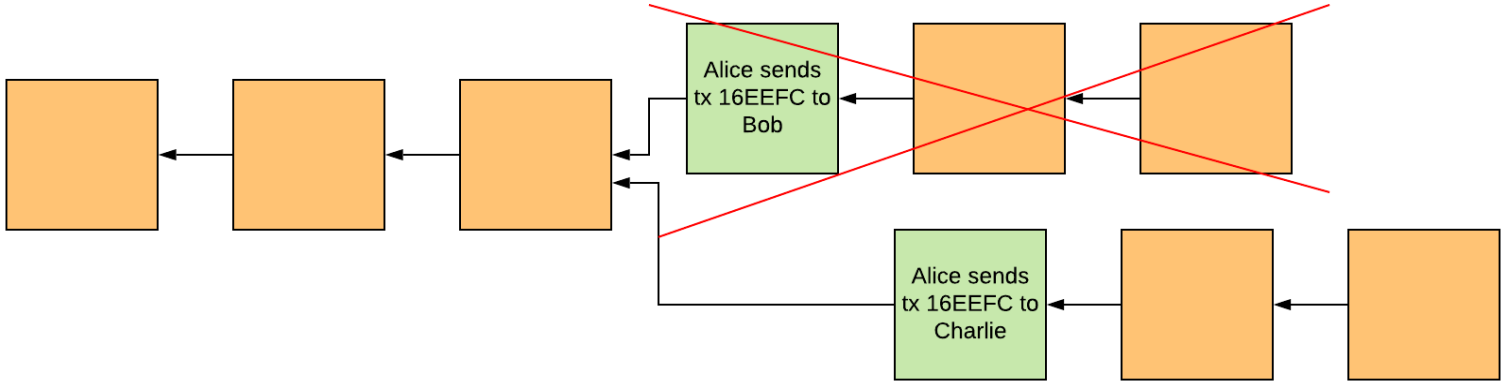
Double spending attack:



Issue in case of off-chain services

Double Spending Attack

Double spending attack:



Issue in case of off-chain services

What happens if there is no off-chain impact?

Forks and Double Spending Attack

Classic solution:

Forks and Double Spending Attack

Classic solution:

wait until the associated transaction is confirmed on the blockchain

Forks and Double Spending Attack

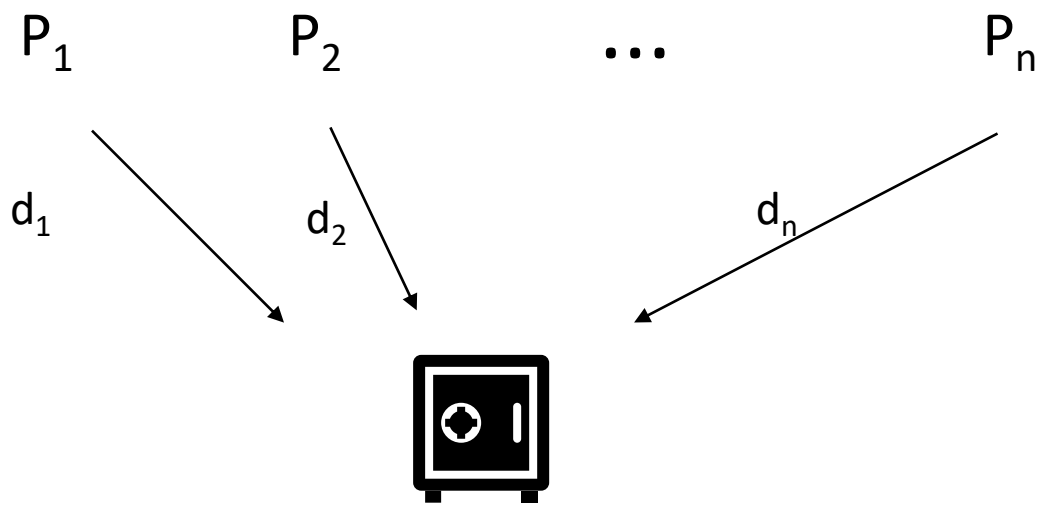
Classic solution:

wait until the associated transaction is confirmed on the blockchain

huge delays

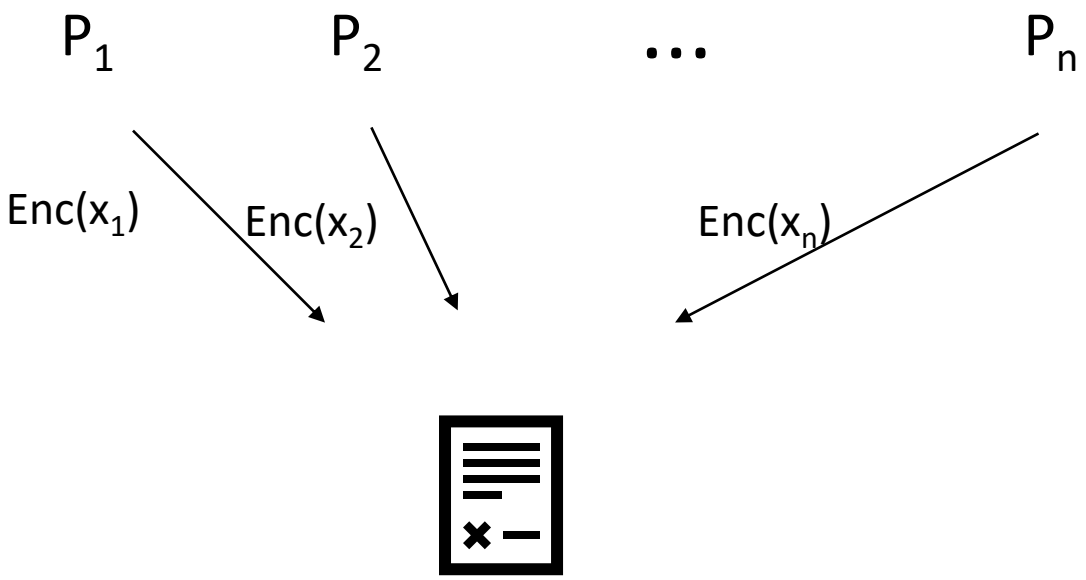
Lottery Protocol(IEEE S&P 2014)

Step 1:



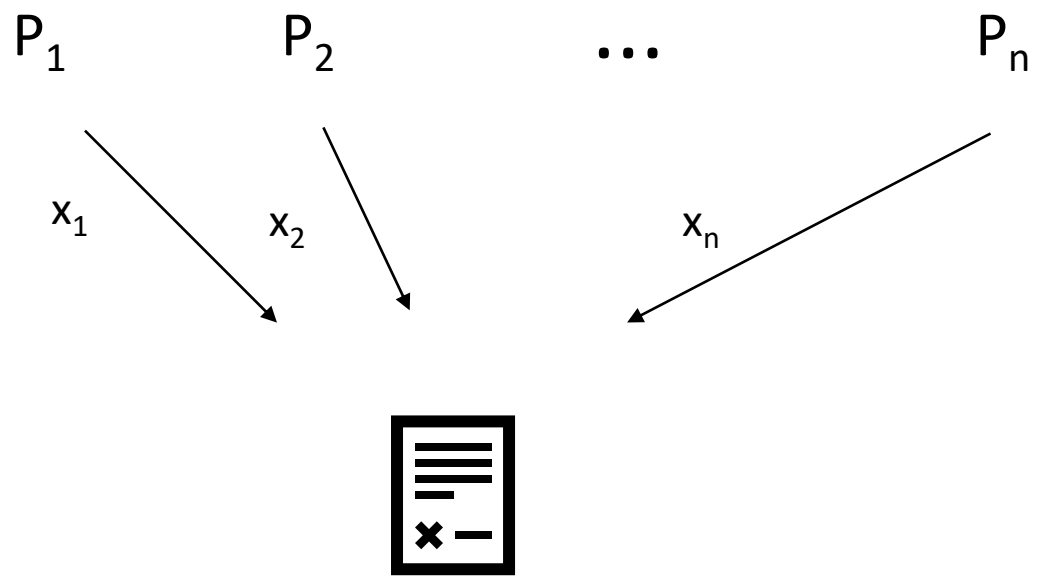
Lottery Protocol (IEEE S&P 2014)

Step 2:



Lottery Protocol (IEEE S&P 2014)

Step 3:



Lottery Protocol (IEEE S&P 2014)

Step 4:

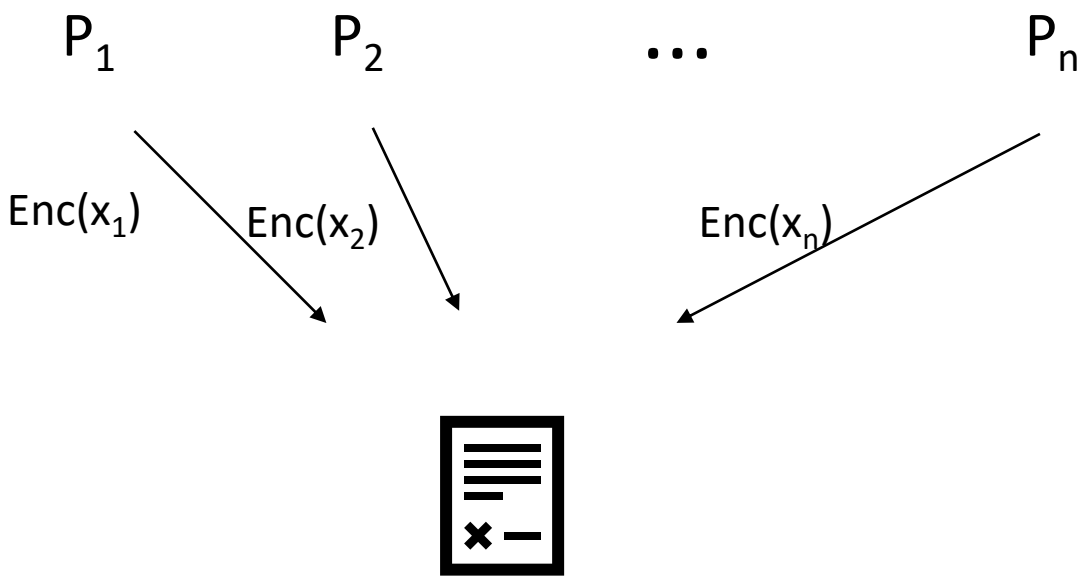
Computes the winner of the lottery as

$w = x_1 + \ldots + x_n \bmod n$

and sends the moneys to P_w

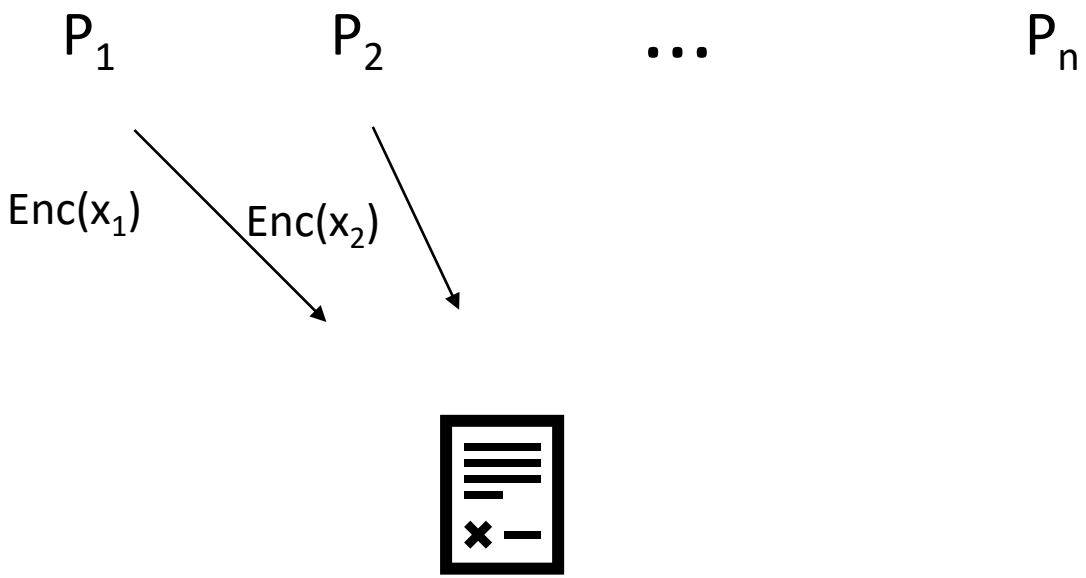
Lottery Protocol with Rushing Players

Step 2:



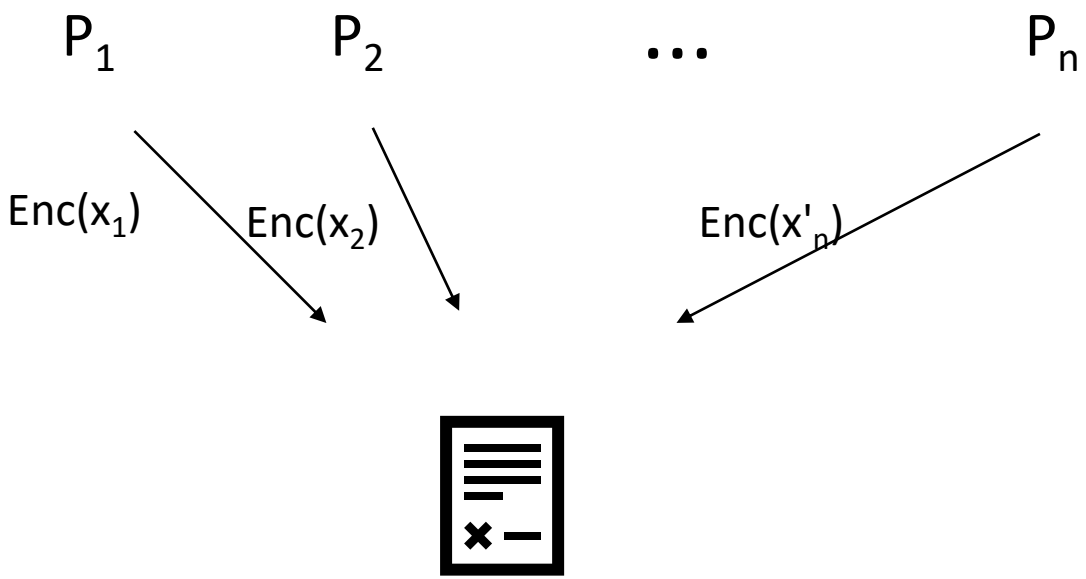
Lottery Protocol with Rushing Players

Step 2':



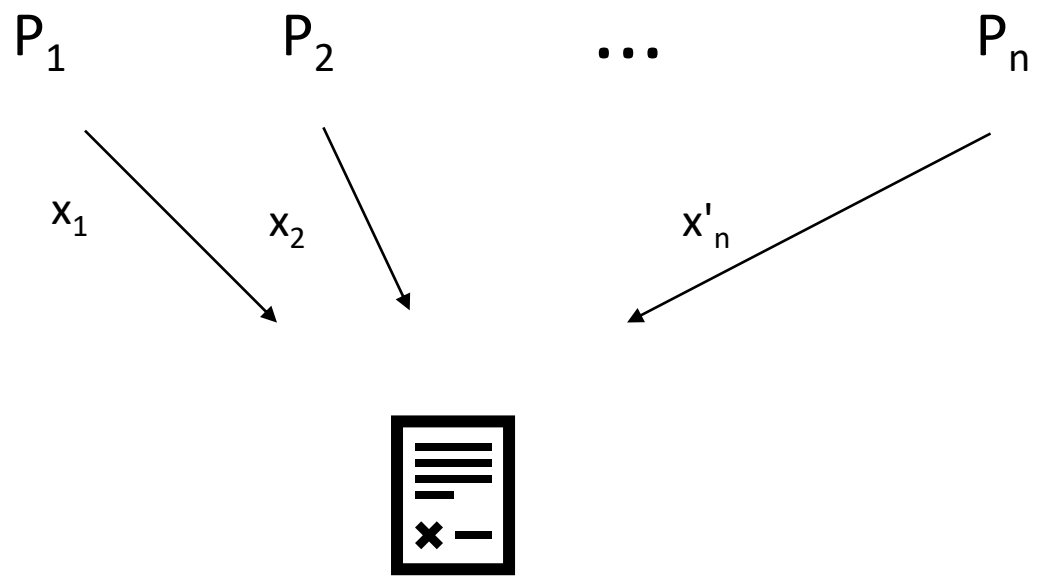
Lottery Protocol with Rushing Players

Step 2':



Lottery Protocol

Step 3’:



Lottery Protocol

Step 4’:

Computes the winner of the lottery as

$w' = x_1 + \ldots + x'_n \bmod n$

and sends the moneys to $P_{w'}$

Our solution

The previous scenarion is insecure in presence of forks and rushing players

But we have **good news**!

Our solution

The previous scenarion is insecure in presence of forks and rushing players

But we have **good news**!

Using crittographic tools we can produce an efficient protocol that evaluates a random string in presence of forks and rushing players!

Our solution

The contribution to the coin-tossing protocol of each player P_i depends on the identifiers of other players

Our solution

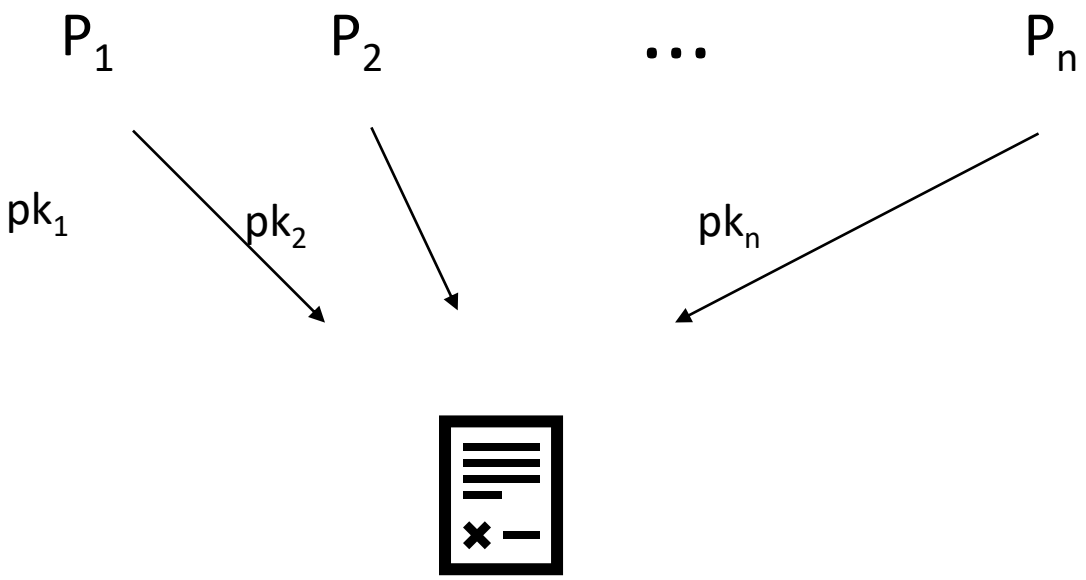
The contribution to the coin-tossing protocol of each player P_i depends on the identifiers of other players

If an adversary changes its input all honest parties will automatically change their inputs too

We achieved this result using a unique signature scheme

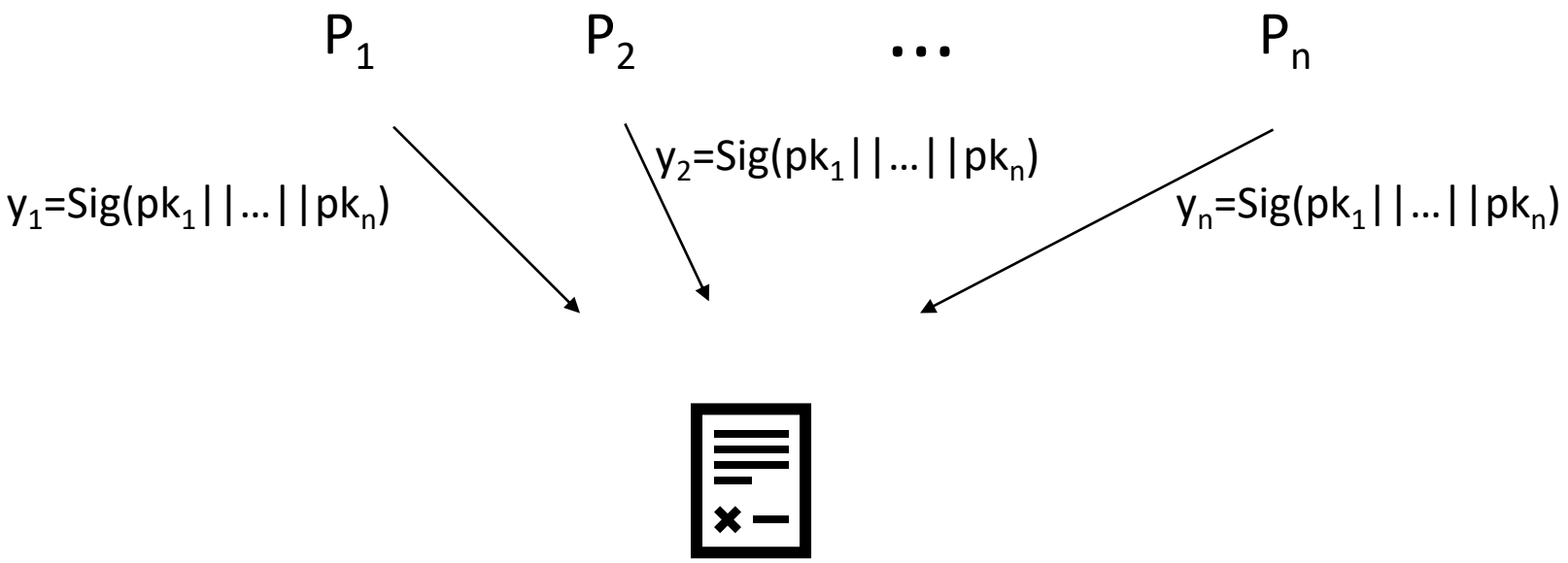
Our Solution

Step 1:



Our Solution

Step 2:



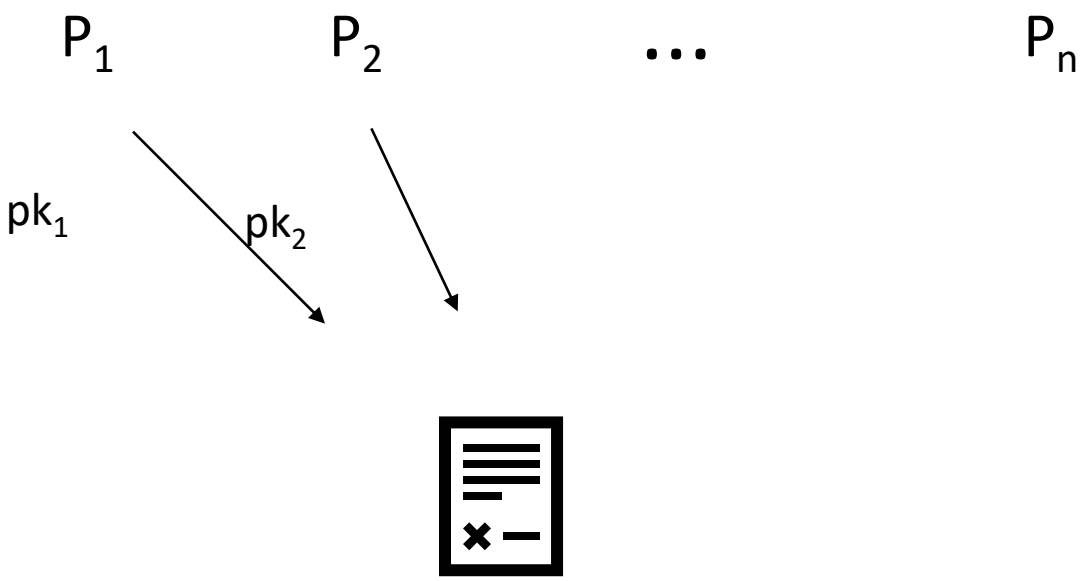
Our Solution

Step 3:

Computes Hash($y_1||\ldots||y_n$)

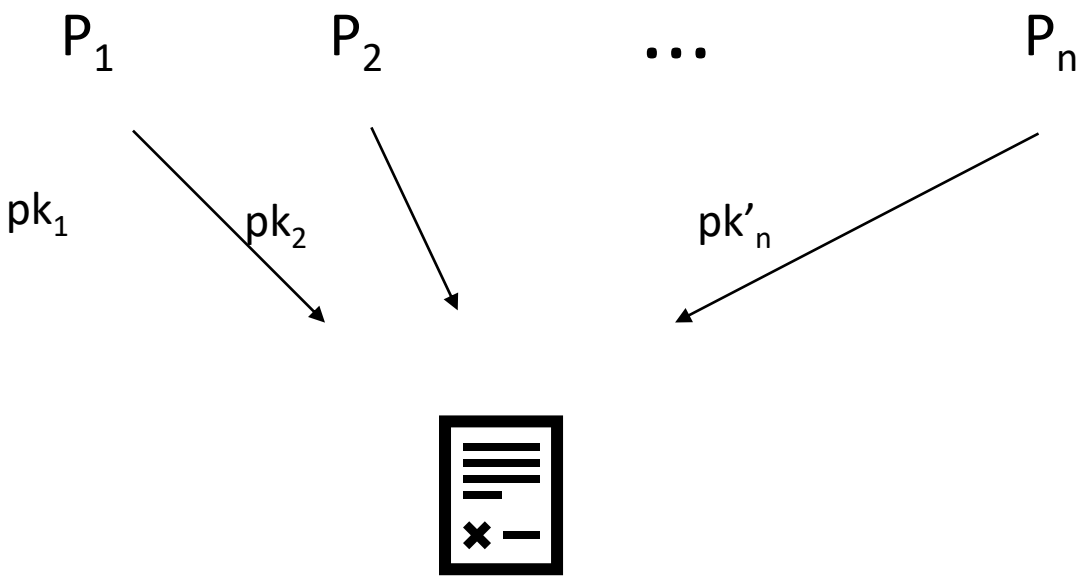
Our Solution

Step 1:



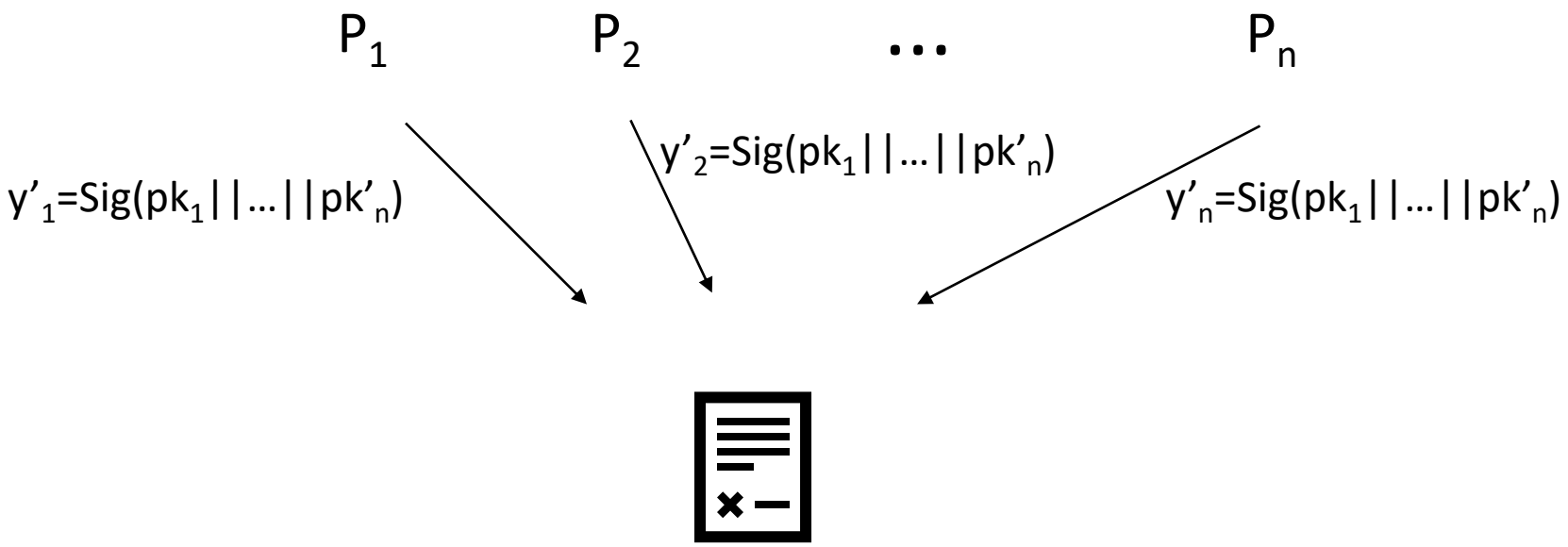
Our Solution

Step 1’:



Our Solution

Step 2’:



Our Solution

Step 3:

Computes Hash($y'_1||\ldots||y'_n$)

Questions?