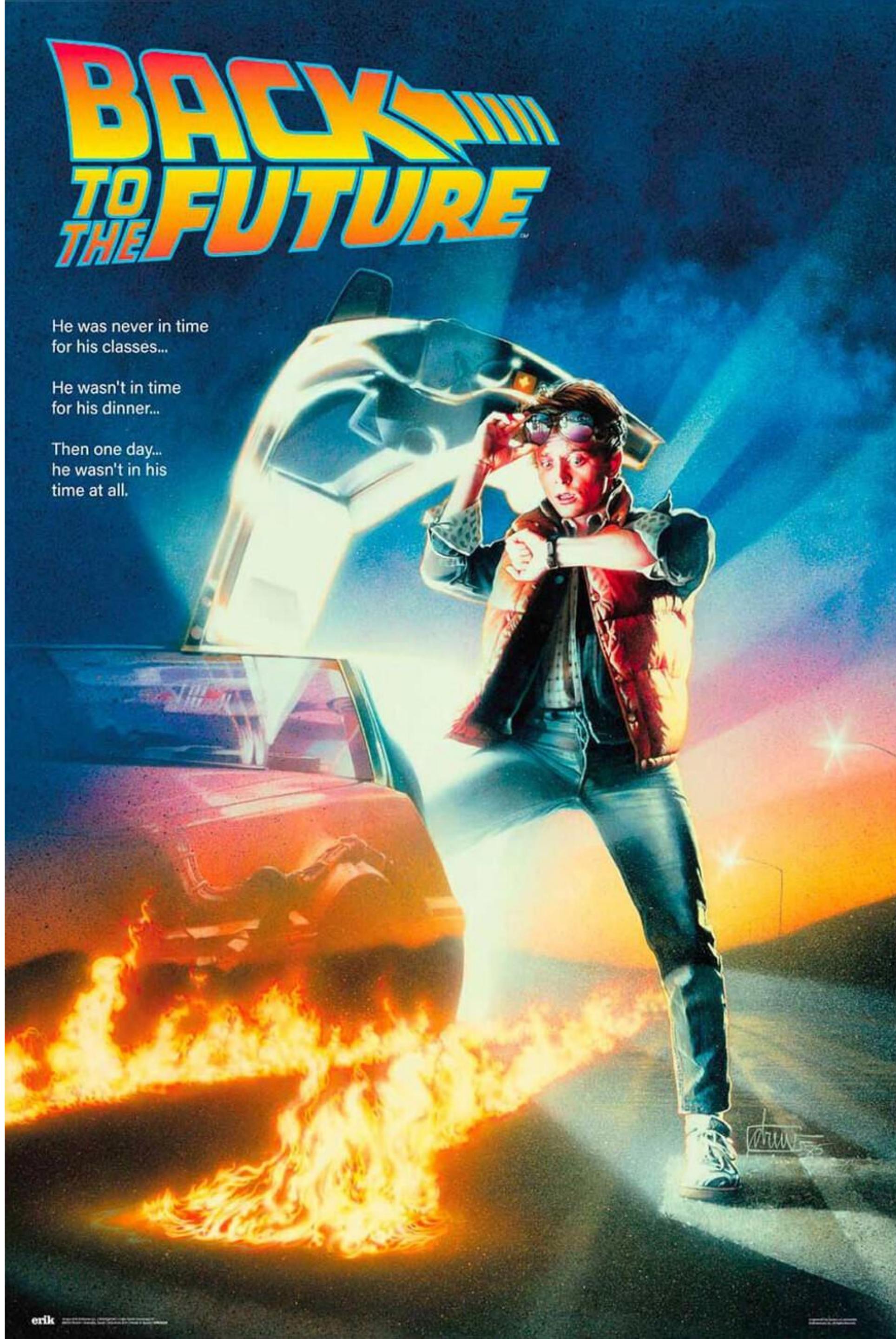
# Peeking into the Future

## MPC Resilient to Super-Rushing Adversaries


Gilad Asharov — Bar-Ilan University
Anirudh Chandramouli — Bar-Ilan University
Ran Cohen — Reichman University
Yuval Ishai — Technion Israel Institute of Technology, aws


Eurocrypt 2025


“Well, hey, Doc, what’s the harm in bringing back
a little info on the future?
You know, maybe we could place a couple bets”



# Biff’s Attack on the Timeline

1955

2015


Gives it to his past self


Biff gets rich!

Biff steals the almanac

Hill Valley Telegraph

PUBLISHED DAILY

BIFF TANNEN

# Luckiest Man On Earth

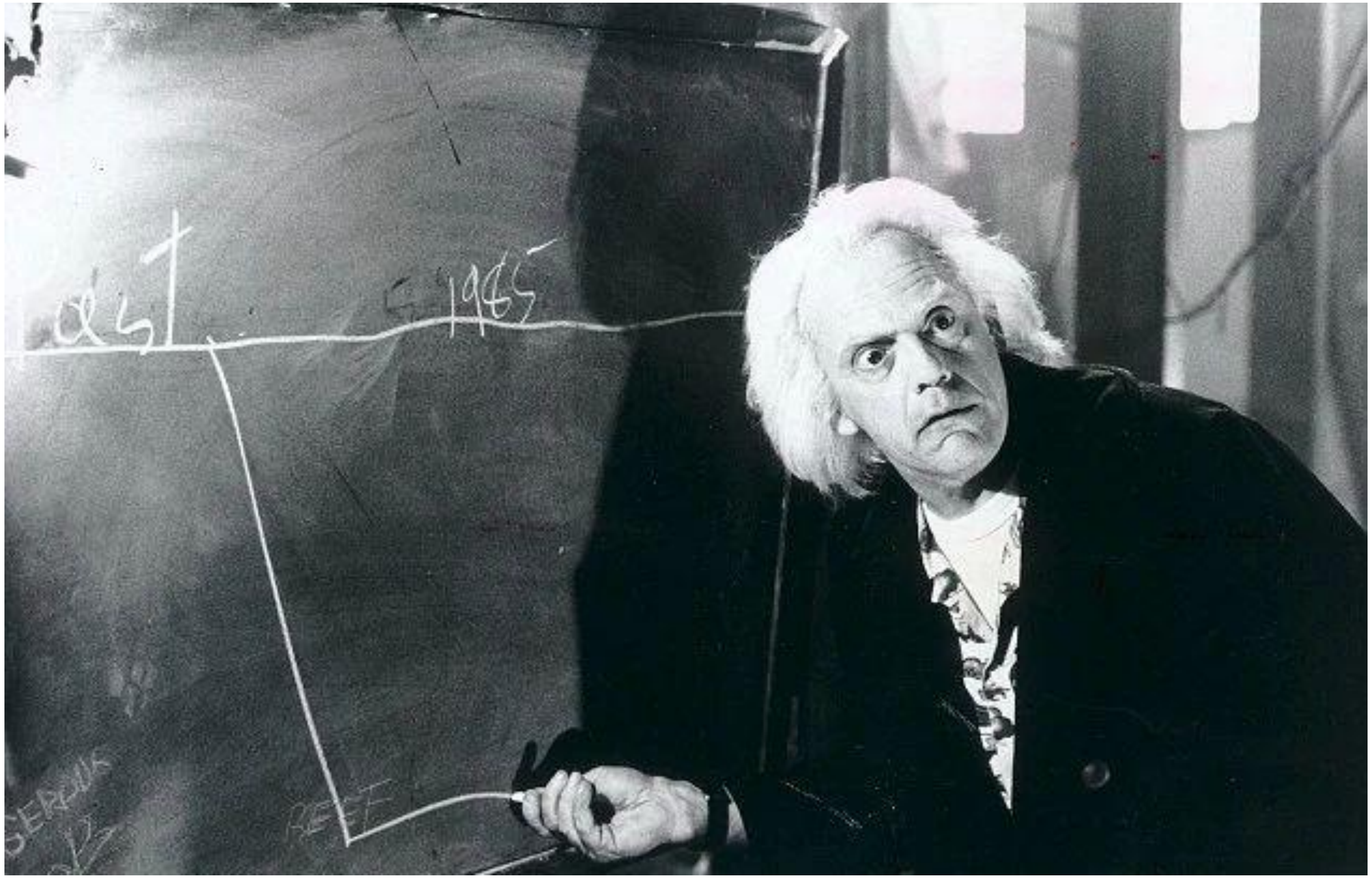
Nasser Accuses Reds of Plotting His Overthrow

State Likely to Start Payroll Tax in 1960

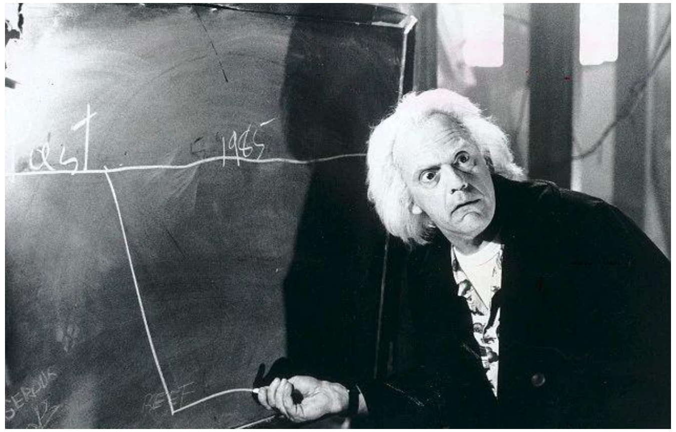
BIFF TANNEN

ac

“No, Marty, we’ve already agreed that having information about the future could be extremely dangerous!”



# This Work



- “Back to the Future” attacks on MPC
- Optimistic implementations of certain synchronous MPC protocols may be vulnerable
- Goal: understand what makes a protocol immuned to such attacks (enable optimistic implementations)

# Communication Models for MPC

## Fully asynchronous

- Adversarial message delivery (can drop messages)
- Most UC secure MPC
- No guaranteed termination

## Asynchronous with eventual delivery

- Every message eventually arrives
- Guaranteed termination
- No "input completeness"
- Inherent $t < n/3$
- Same limitations for **partial synchrony**

## Synchronous

- Round-by-round, potentially with broadcast
- Guaranteed termination
- Input completeness
- Guaranteed output delivery for $t < n/2$ (sometimes $t < n$)
- Vast majority of literature

# Communication Models for MPC

### Fully asynchronous

- Adversarial message delivery (can drop messages)
- Most UC secure MPC
- No guaranteed termination

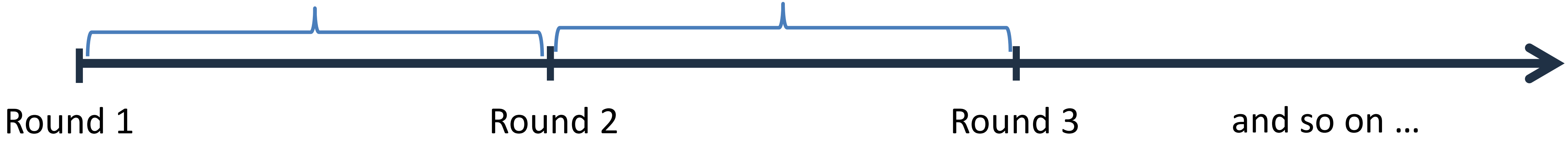
### Asynchronous with eventual delivery

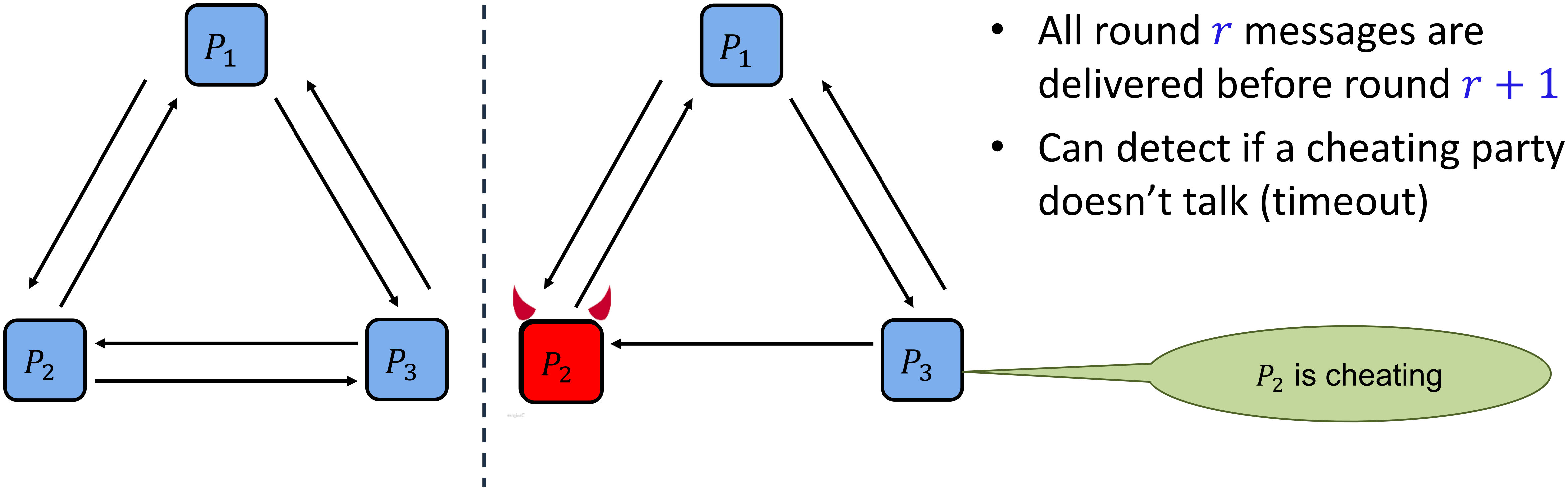
- Every message eventually arrives
- Guaranteed termination
- No “input completeness”
- Inherent $t < n/3$
- Same limitations for **partial synchrony**

### Synchronous

- Round-by-round, potentially with broadcast
- Guaranteed termination
- Input completeness
- Guaranteed output delivery for $t < n/2$ (sometimes $t < n$)
- Vast majority of literature

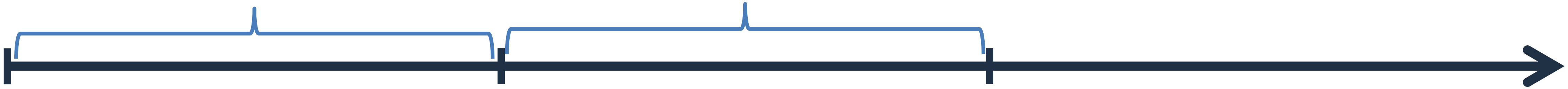
# Synchronous Protocols



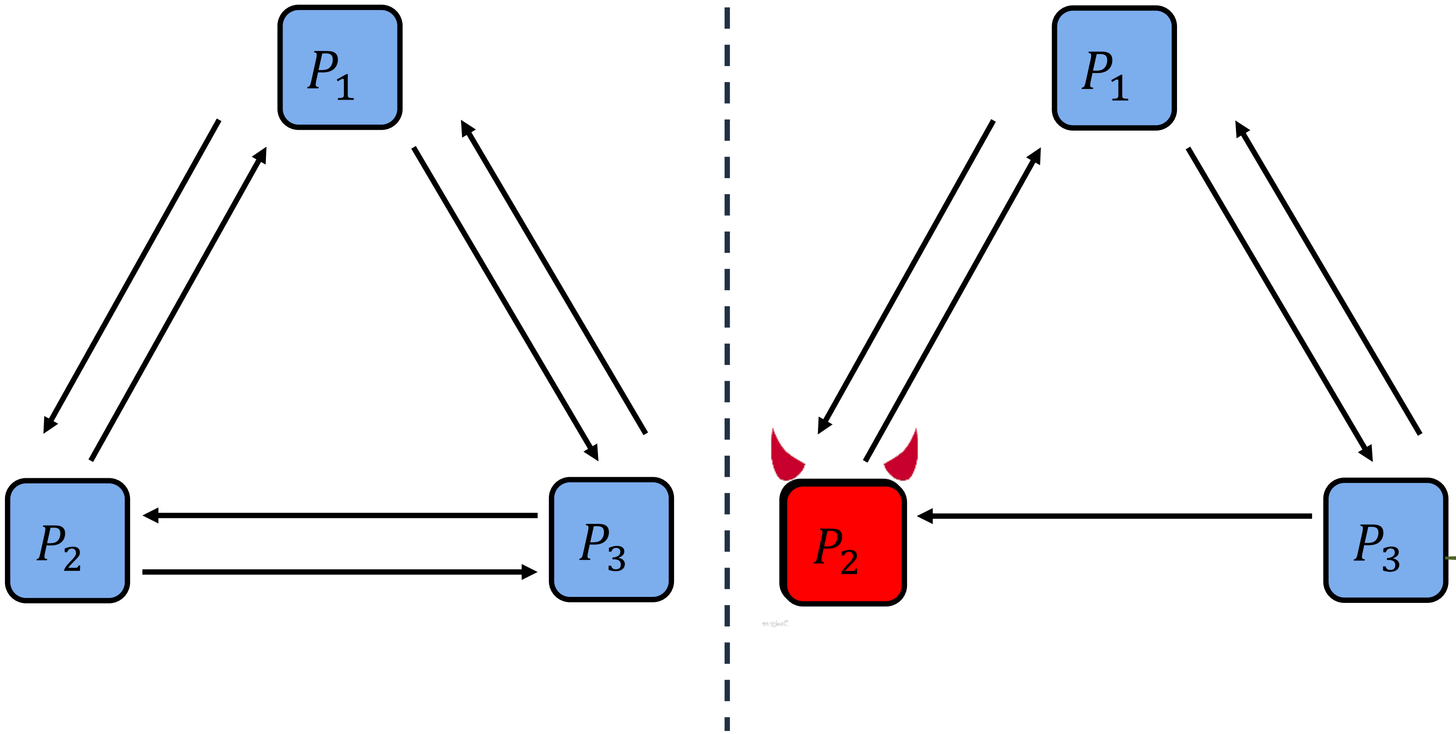




- All round $r$ messages are delivered before round $r + 1$
- Can detect if a cheating party doesn’t talk (timeout)

# Synchronous Protocols

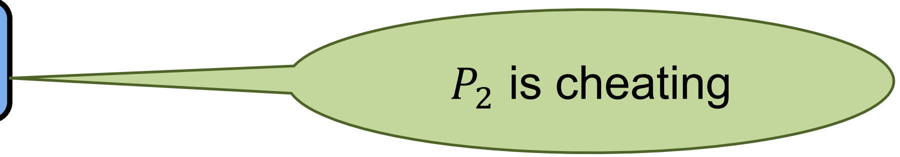


Round 1 Round 2 Round 3 and so on …

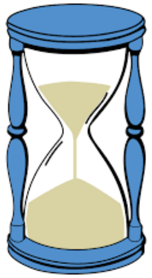



**Simplifying assumptions:**

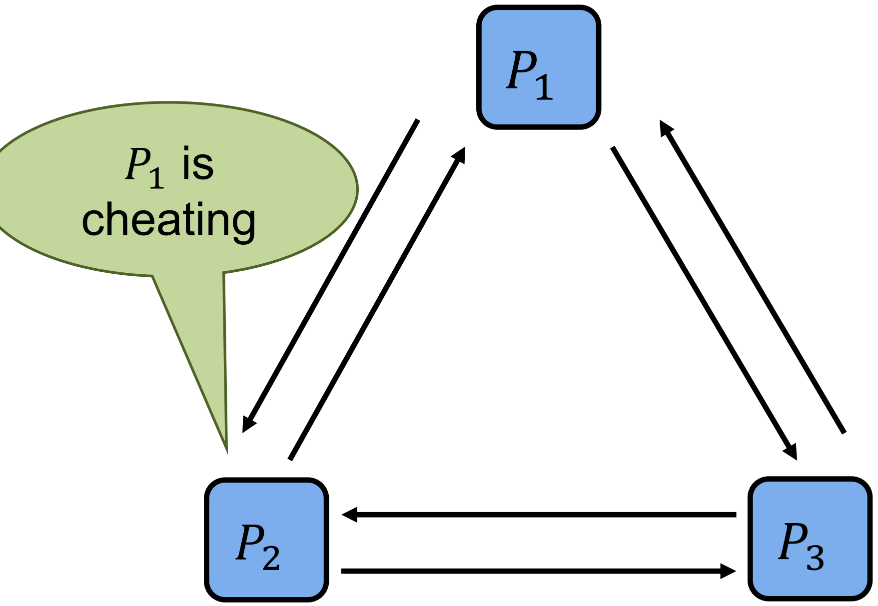
- All-to-all communication in every round (possibly dummy messages)
- Adv. also talks in every round (possibly say nothing)



# How much time should we wait?



Say the expected duration is 1 second

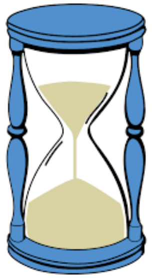




**Idea #1:**

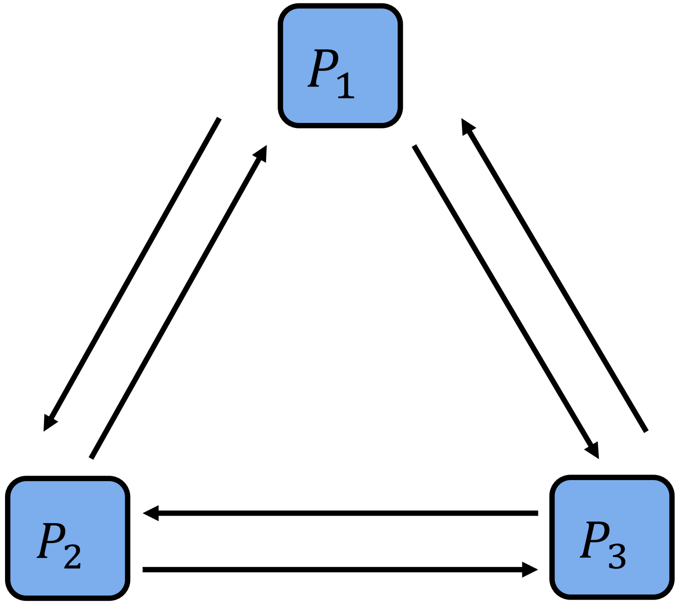
- Set round duration to 2 seconds
- But…

  honest parties might be falsely detected as cheaters

# How much time should we wait?



Say the expected duration is 1 second

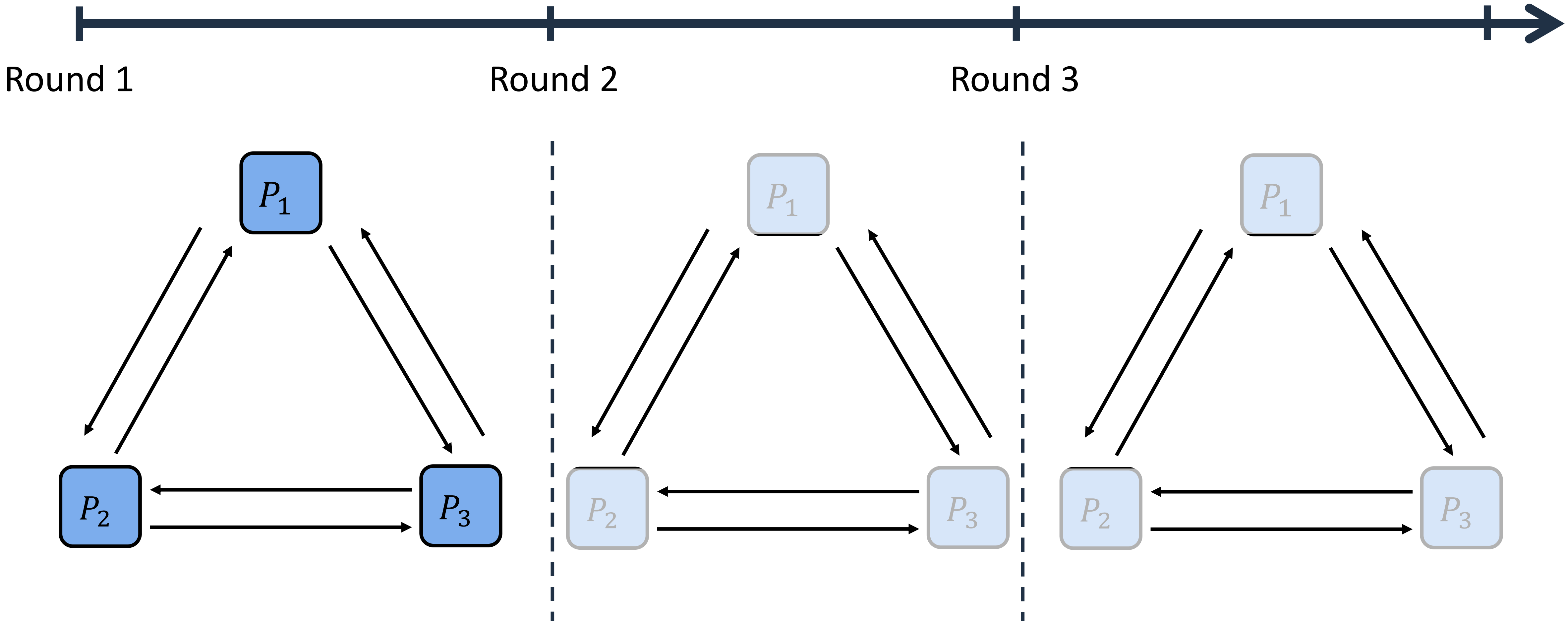




**Idea #2:**

- Set round duration to 1 hour
- No party falsely accused
- But…
  who's gonna use my protocol

# ~~How much time should we wait?~~

Proceed optimistically: once all round $r$ message arrive

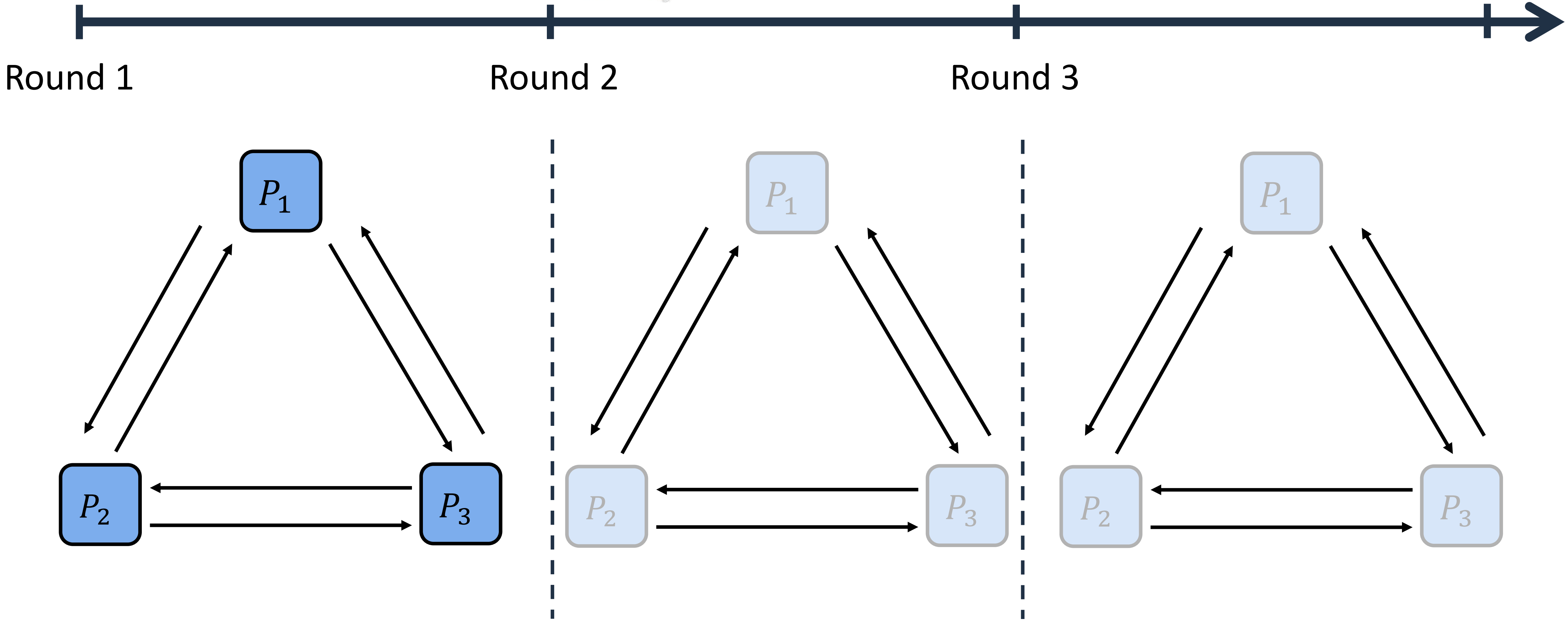


# ~~How much time should we wait?~~

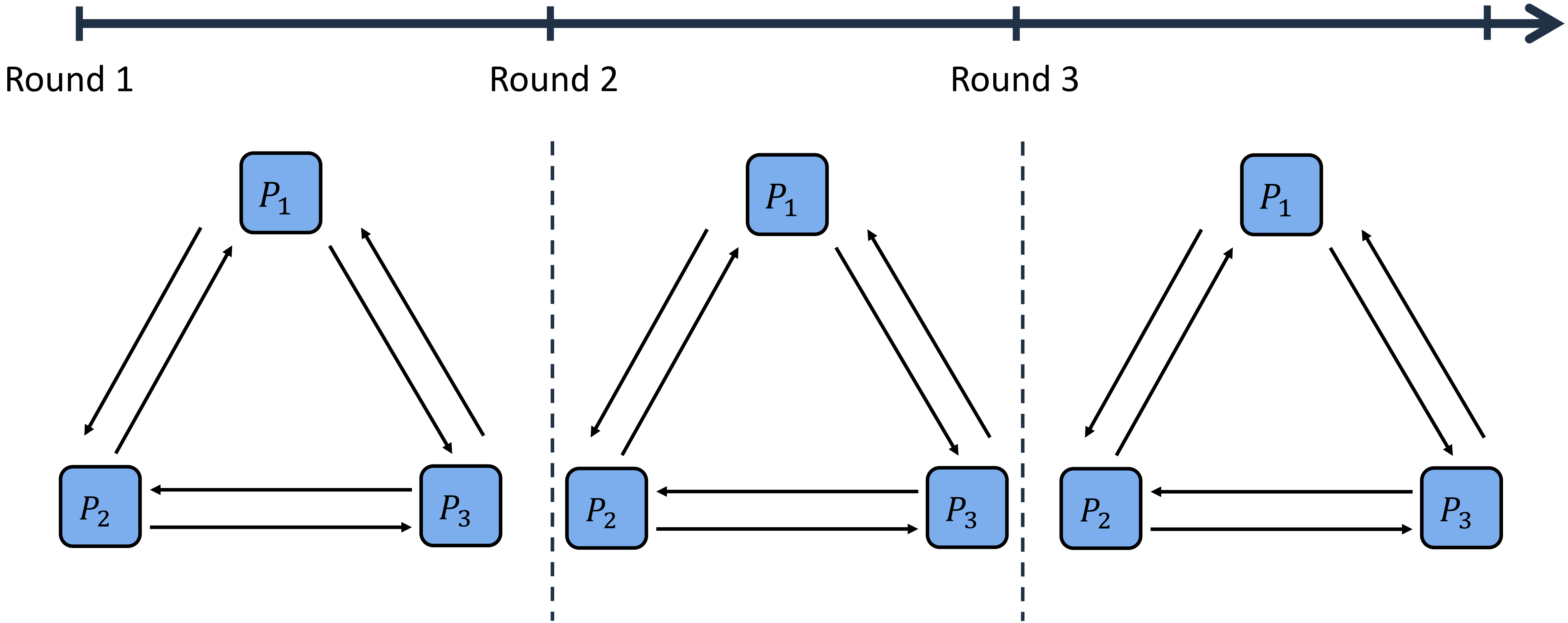
Proceed optimistically: once all round $r$ message arrive

Proceed at network speed! 

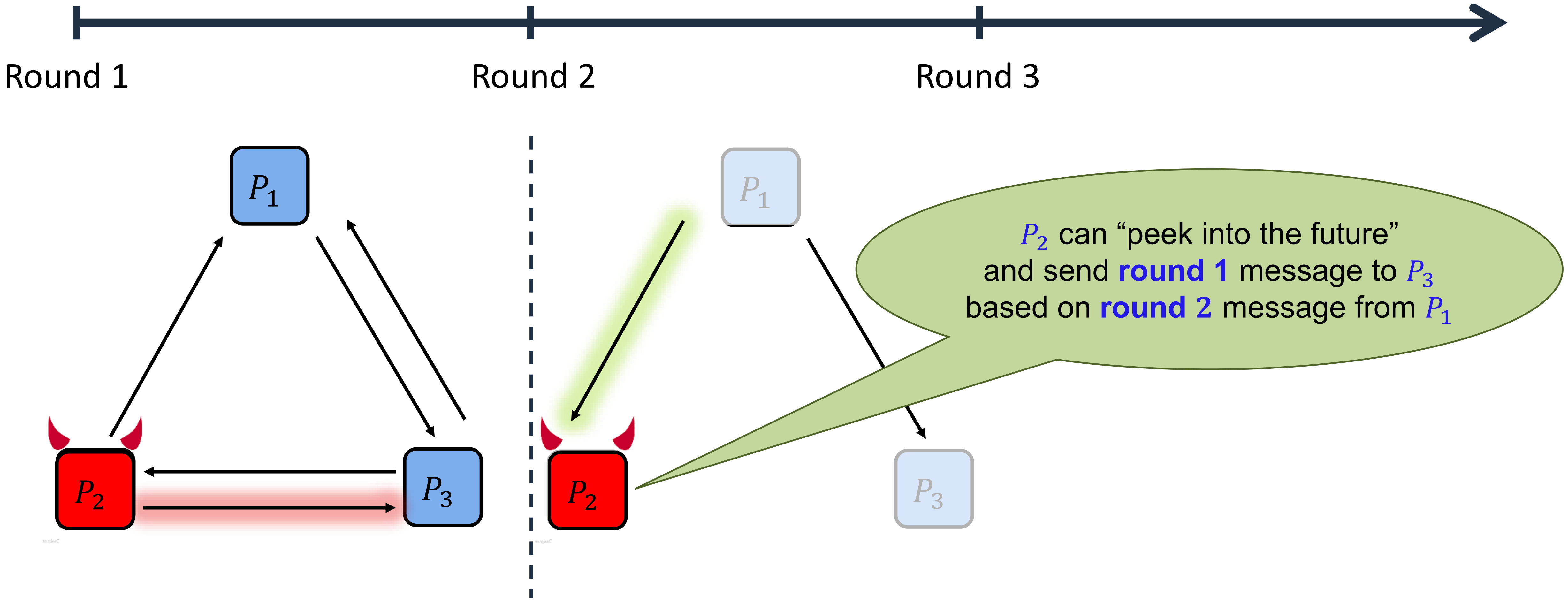
Assume we can detect parties who don’t talk



# Wait… What???



# Wait… What???



Round 1
Round 2
Round 3
$P_1$
$P_2$
$P_3$
$P_1$
$P_2$
$P_3$
$P_2$ can "peek into the future" and send **round 1** message to $P_3$ based on **round 2** message from $P_1$

# Peeking $\Rightarrow$ Super-Rushing

**Non-Rushing**

Adversary sends round-$r$ messages **before** receiving the honest parties' round-$r$ messages

**Rushing**

Adversary can send round-$r$ messages **after** receiving the honest parties' round-$r$ messages

**Super-Rushing**

Adversary can send round-$r$ messages **after** receiving some round-$(r+1)$ messages

# A Gap in the Security Analysis

Theory
(ideal synchrony)

Applied research
~~Practice~~
(optimistic implementations)

Rushing

Super-Rushing

Is it really
a meaningful attack?

Are existing synchronous
MPC protocols vulnerable
to super-rushing attacks?

# A Gap in the Security Analysis

Theory
(ideal synchrony)

**Applied research**
~~Practice~~
(optimistic implementations)

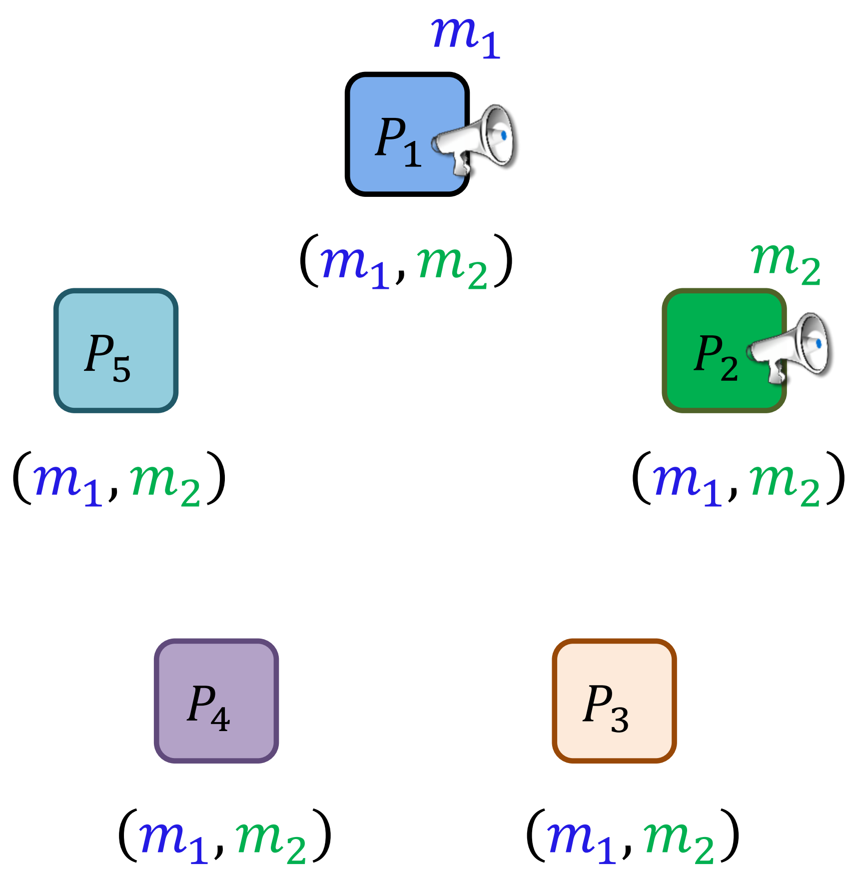
Rushing

Super-Rushing

Yes!
Some protocols are insecure against super-rushing adversaries

# Simultaneous Broadcast [CGMA85]

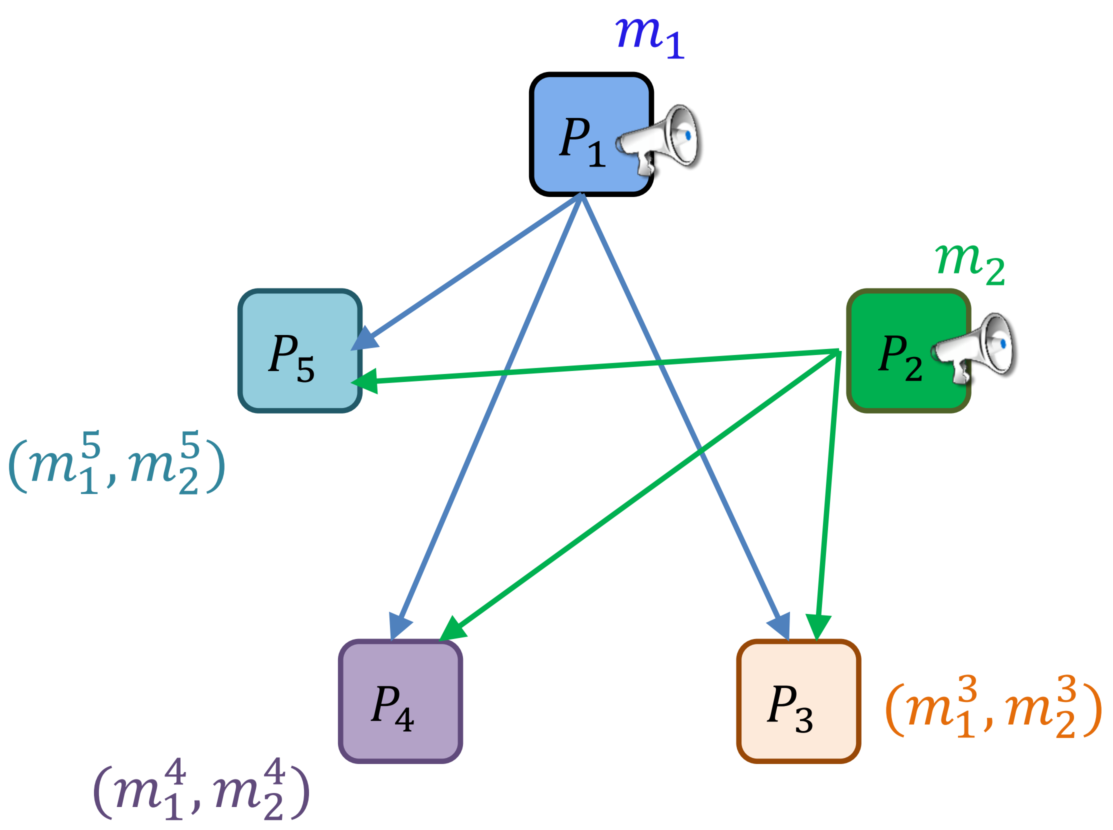
- 5 parties, 2 senders
- $P_1$ holds $m_1$ and $P_2$ holds $m_2$
- Everyone outputs $(m_1, m_2)$
- Security against 1 corruption
- $P_1$ cannot choose $m_1$ as a function of $m_2$ (and vice versa)



# A Simple Simultaneous Broadcast Protocol [GIKR02]

5 parties, 2 senders, 1 corruption

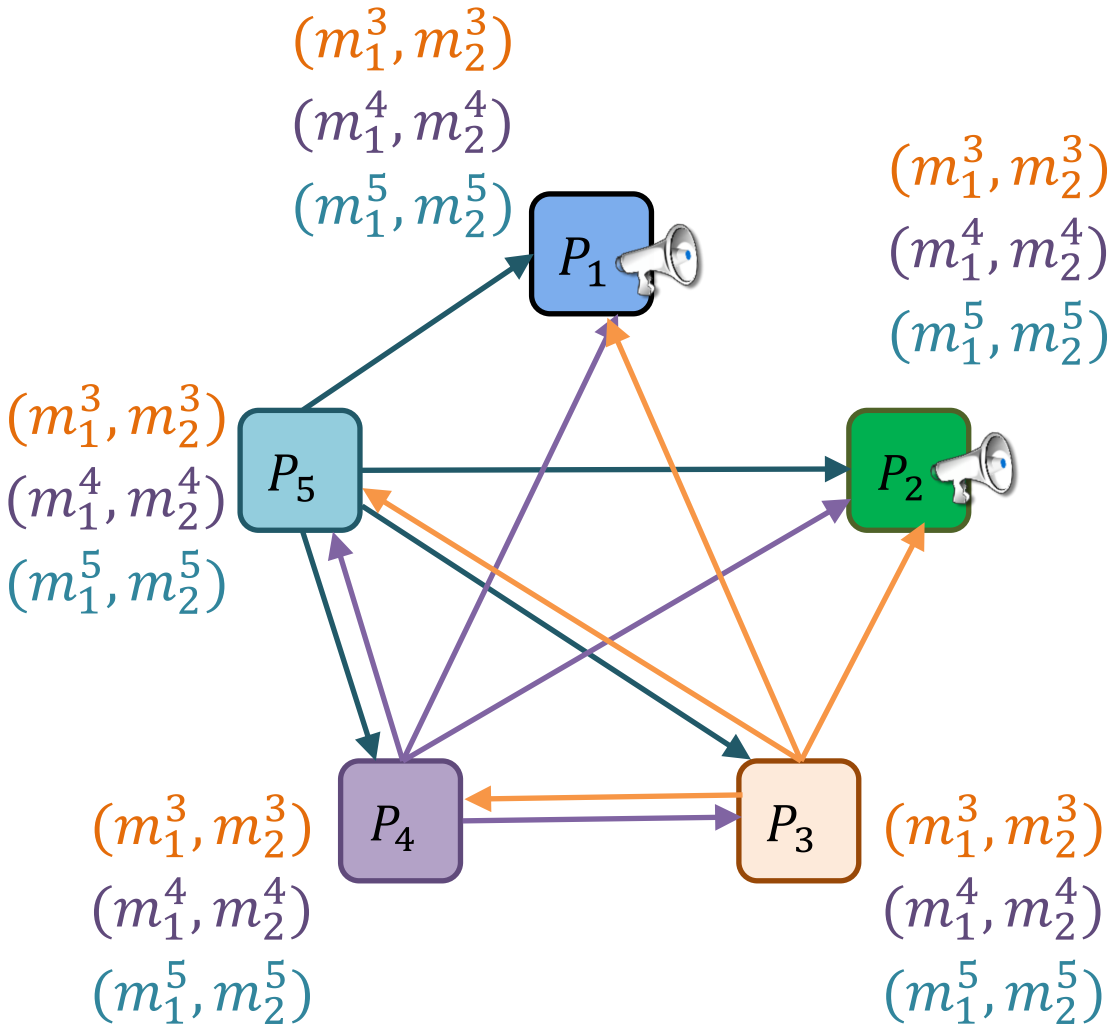
- Round 1:
  $P_1$ and $P_2$ send input message to $P_3, P_4, P_5$



# A Simple Simultaneous Broadcast Protocol [GIKR02]

5 parties, 2 senders, 1 corruption

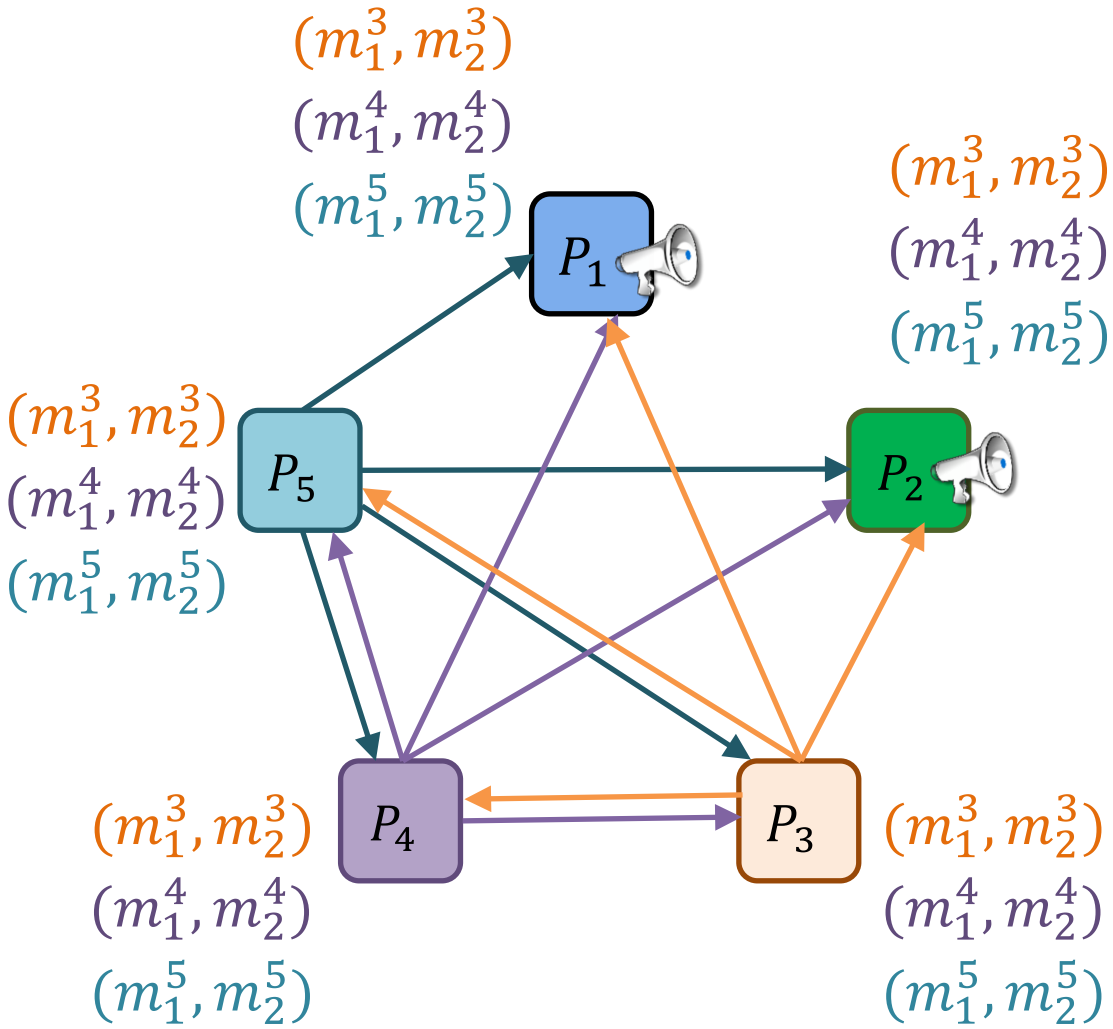
- Round 1:
  $P_1$ and $P_2$ send input message to $P_3, P_4, P_5$
- Round 2:
  $P_3, P_4, P_5$ echo message to everyone



# A Simple Simultaneous Broadcast Protocol [GIKR02]

5 parties, 2 senders, 1 corruption

- Round 1:
  $P_1$ and $P_2$ send input message to $P_3, P_4, P_5$
- Round 2:
  $P_3, P_4, P_5$ echo message to everyone
- Output:
  $(m_1', m_2')$ echoed by at least 2 parties

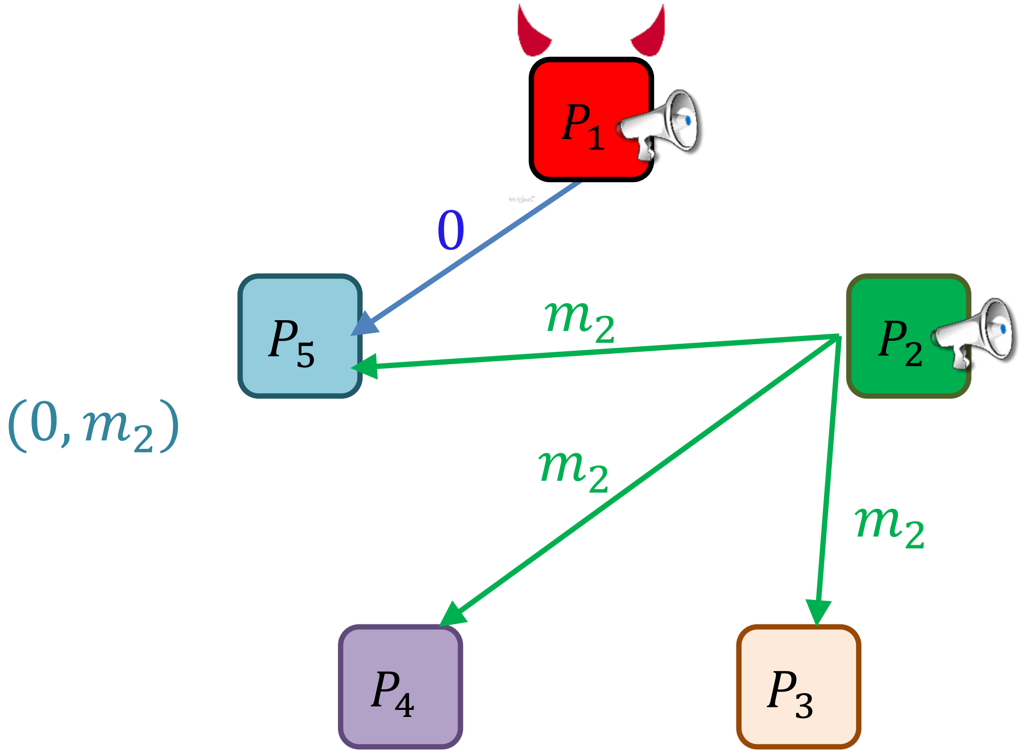



**Security against rushing adversary**

- Corrupt sender: independent message
- Corrupt non-sender: cannot affect majority

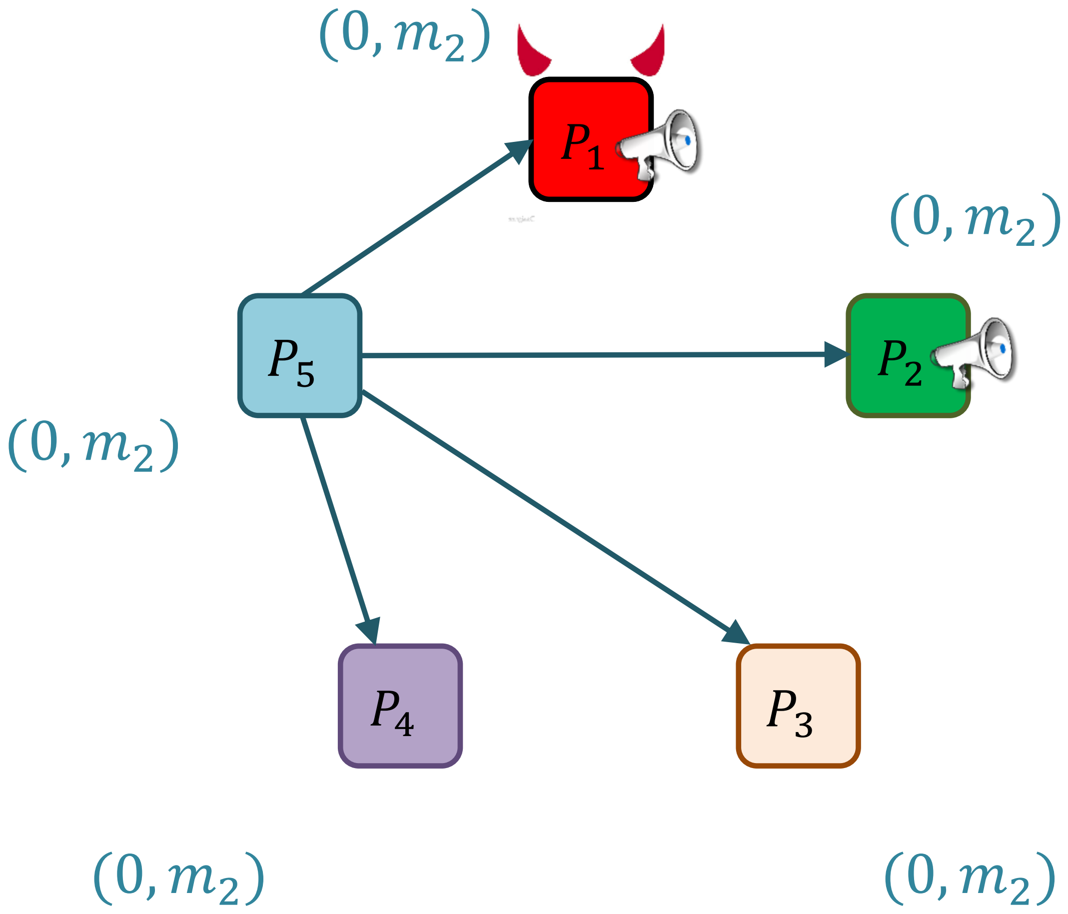
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ send $0$ only to $P_5$



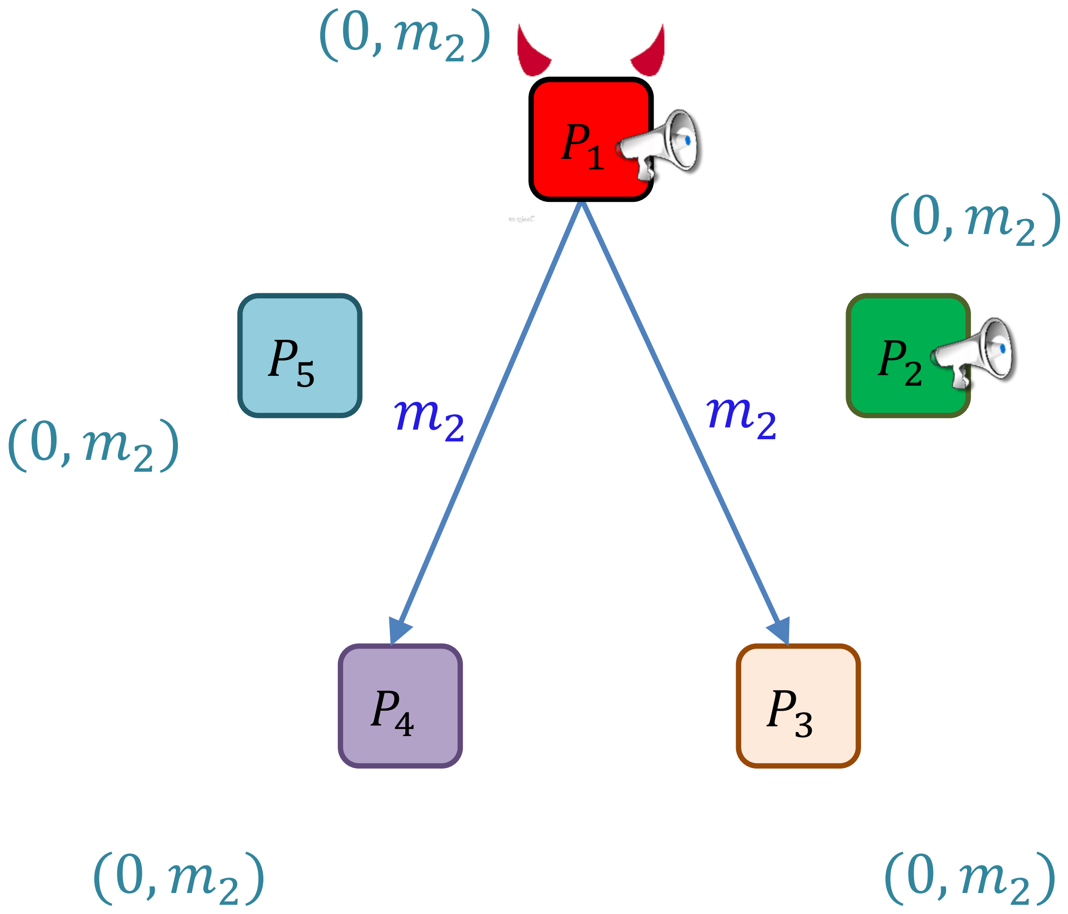
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ sends $0$ only to $P_5$
- Round 2:
  $P_5$ echos $(0, m_2)$ to $P_1, P_2, P_3, P_4$



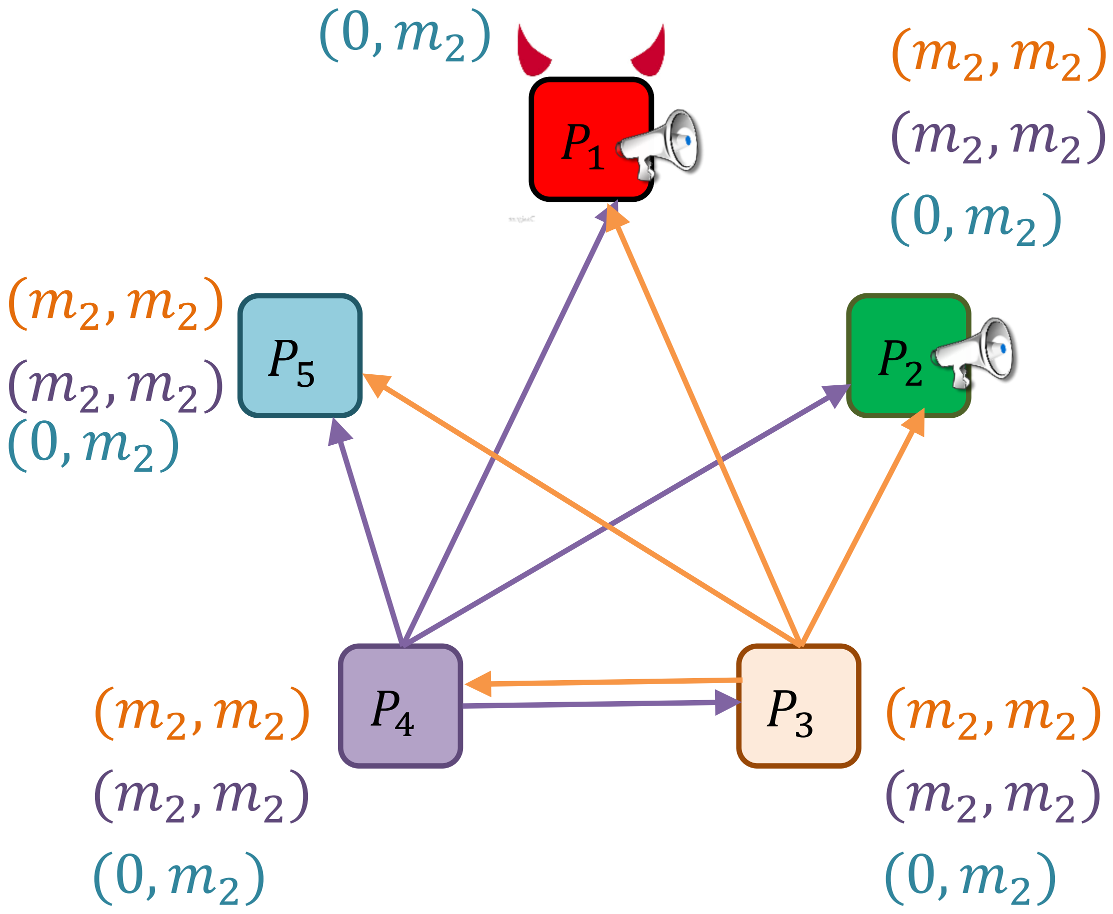
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ sends $0$ only to $P_5$
- Round 2:
  $P_5$ echos $(0, m_2)$ to $P_1, P_2, P_3, P_4$
- Round 1:
  $P_1$ sends $m_2$ to $P_3, P_4$



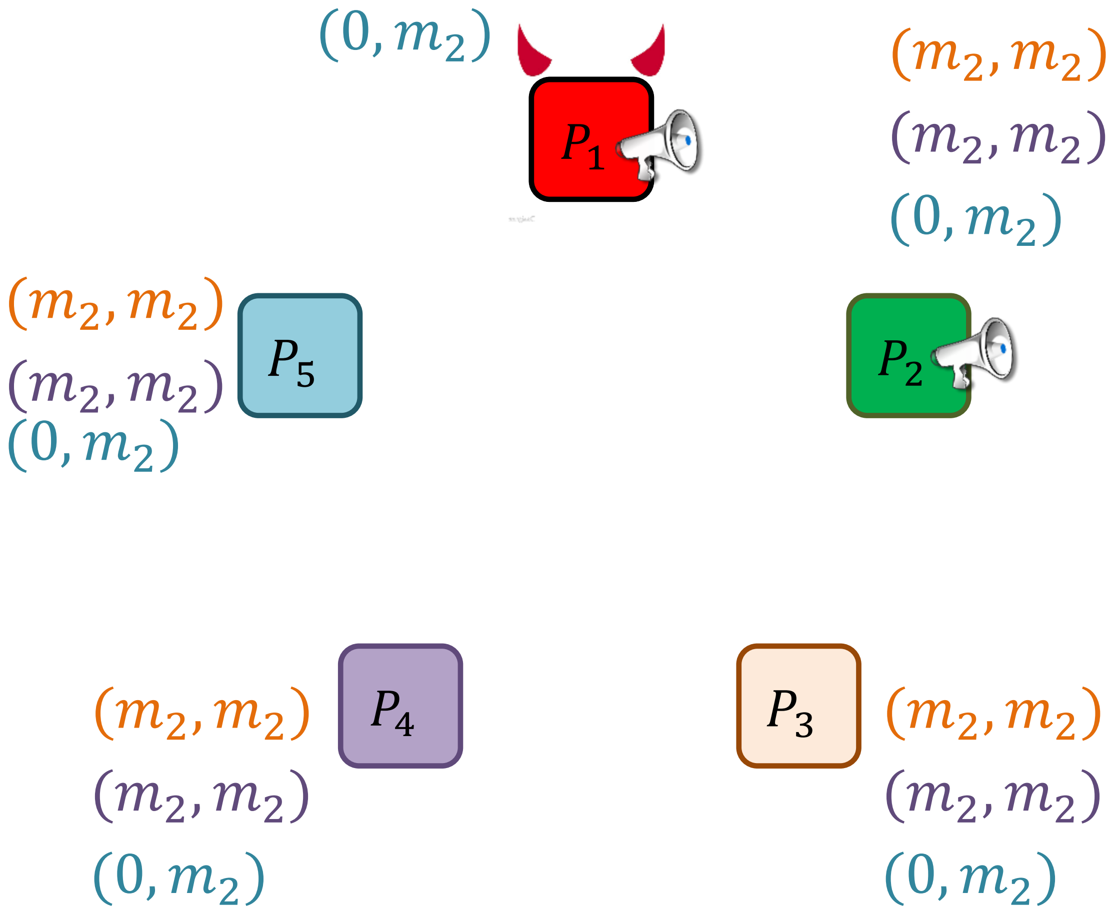
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ sends $0$ only to $P_5$
- Round 2:
  $P_5$ echos $(0, m_2)$ to $P_1, P_2, P_3, P_4$
- Round 1:
  $P_1$ sends $m_2$ to $P_3, P_4$
- Round 2:
  $P_3, P_4$ echo $(m_2, m_2)$ to everyone



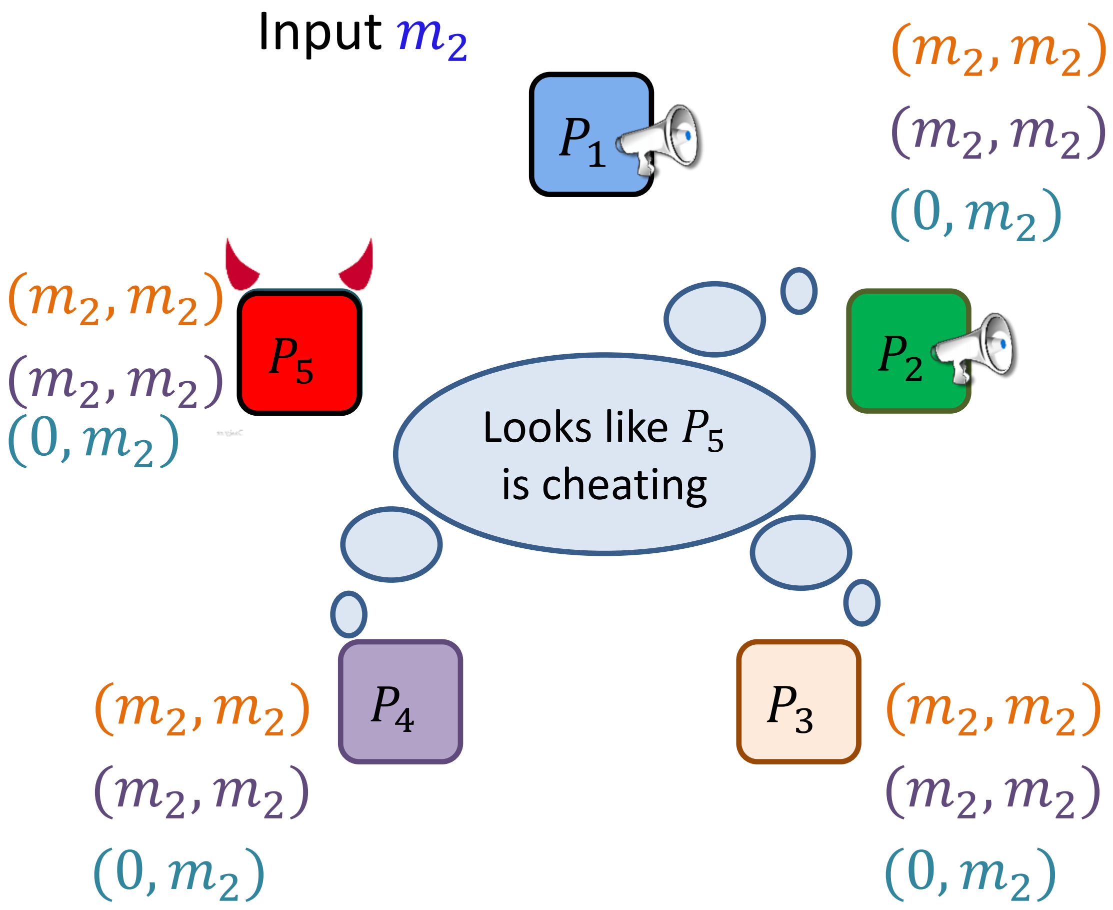
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ sends $0$ only to $P_5$
- Round 2:
  $P_5$ echos $(0, m_2)$ to $P_1, P_2, P_3, P_4$
- Round 1:
  $P_1$ sends $m_2$ to $P_3, P_4$
- Round 2:
  $P_3, P_4$ echo $(m_2, m_2)$ to everyone
- Output:
  everyone outputs $(m_2, m_2)$



# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ sends $m_2$ to $P_3, P_4, P_5$
  $P_1$ sends $0$ only to $P_5$
- Round 2:
  $P_5$ echos $(0, m_2)$ to $P_1, P_2, P_3, P_4$
- Round 1:
  $P_1$ sends $m_2$ to $P_3, P_4$
- Round 2:
  $P_3, P_4$ echo $(m_2, m_2)$ to everyone
- Output:
  everyone outputs $(m_2, m_2)$



# Our Results #1

**Theorem:** There exists a protocol (with two input providers) that is **perfectly secure** against **rushing** adversaries but is **insecure** against **super-rushing** adversaries

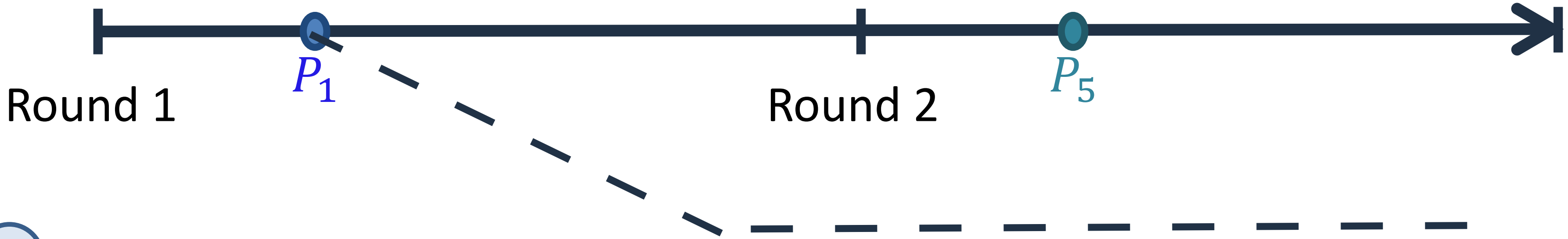
Which synchronous protocols are secure against super-rushing adversaries? (without modifications)

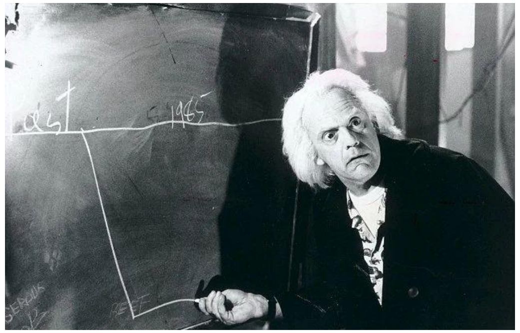
# What happened in this “Back to the Future" Attack

$P_1$ and $P_2$ provide inputs

$P_3, P_4, P_5$ learn the output

$P_3, P_4, P_5$ reveal the output



1 $P_1$ advances $P_5$ to round 2

2 $P_1$ peeks into round-2 ($P_5$’s round-2 message) & learns $P_2$’s input message

3 $P_1$ chooses input message as a function of $P_2$’s input message

Super-rushing breaks input independence

What if only one party provides input? (Broadcast, VSS, etc.)

# Our Results #2

For perfectly secure MPC with **one input provider**

**Super-Rushing** ≡ **Rushing** ≡ **Non-Rushing**

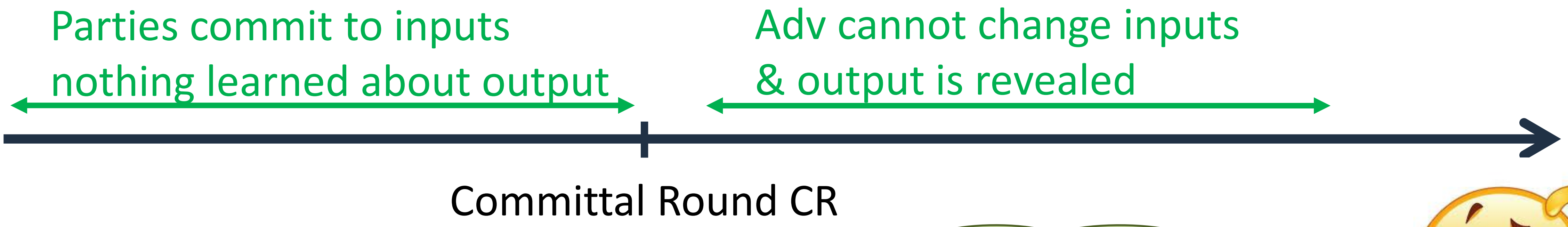
**Theorem:** every protocol with a single input provider that is perfectly secure against **non-rushing** adversaries is also perfectly secure against **super-rushing** adversaries

Till now we worked
too hard to show too little!!

# The Story So Far (Perfect Security)

- ✓ **Single Input Provider:** Super-Rushing ≡ Rushing ≡ Non-Rushing
- ✗ **Two Input Providers:** ∃ a protocol for simultaneous broadcast that is secure against rushing but not against super-rushing

The protocol feels different from MPC protocols:
no privacy in the first round

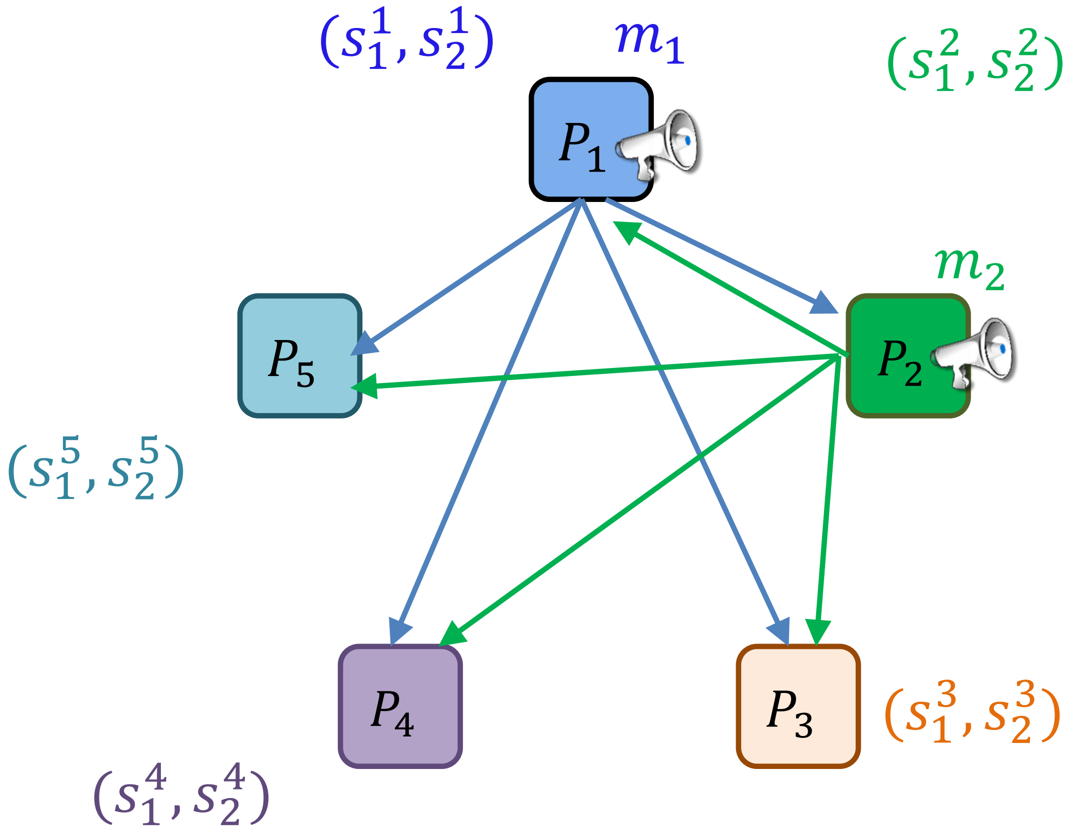



Maybe a CR prevents
super-rushing attacks?

# Simultaneous Broadcast with CR

Uses 5-party, 1-secure, 1-round VSS [GIKR01]
(2 shares suffice to reconstruct)

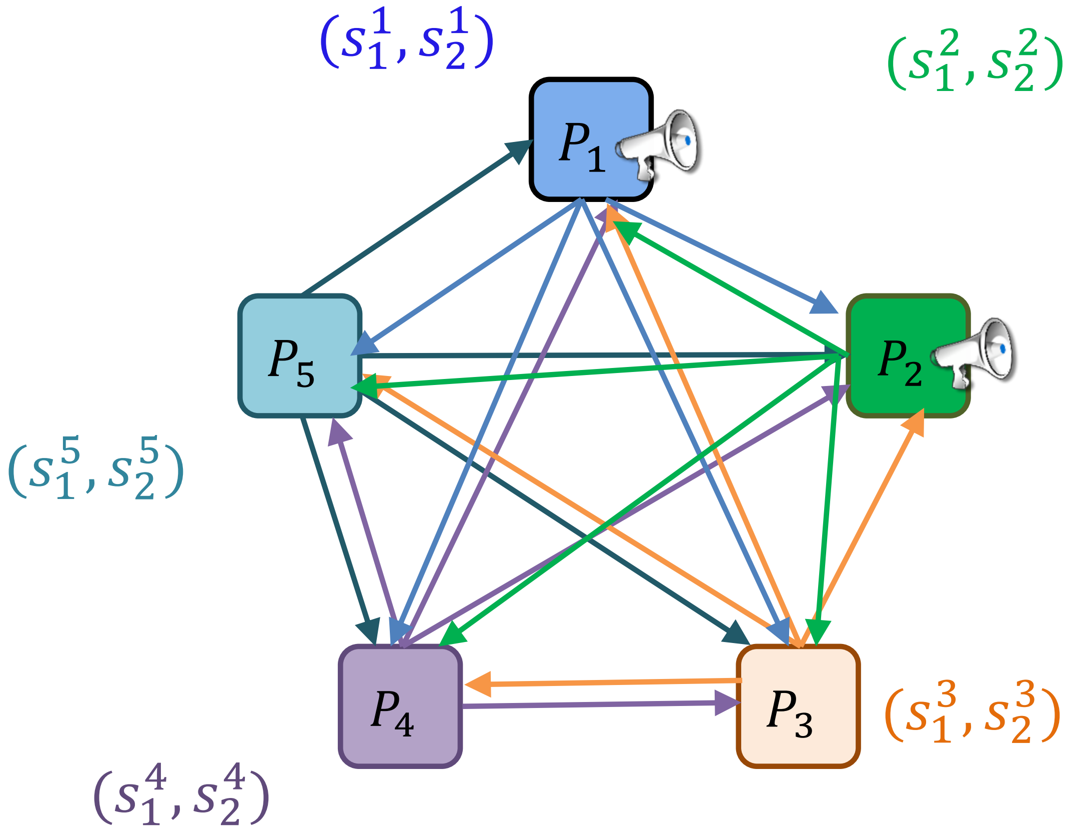
- Round 1:
  $P_1$ and $P_2$ VSS their input message



# Simultaneous Broadcast with CR

Uses 5-party, 1-secure, 1-round VSS [GIKR01]
(2 shares suffice to reconstruct)

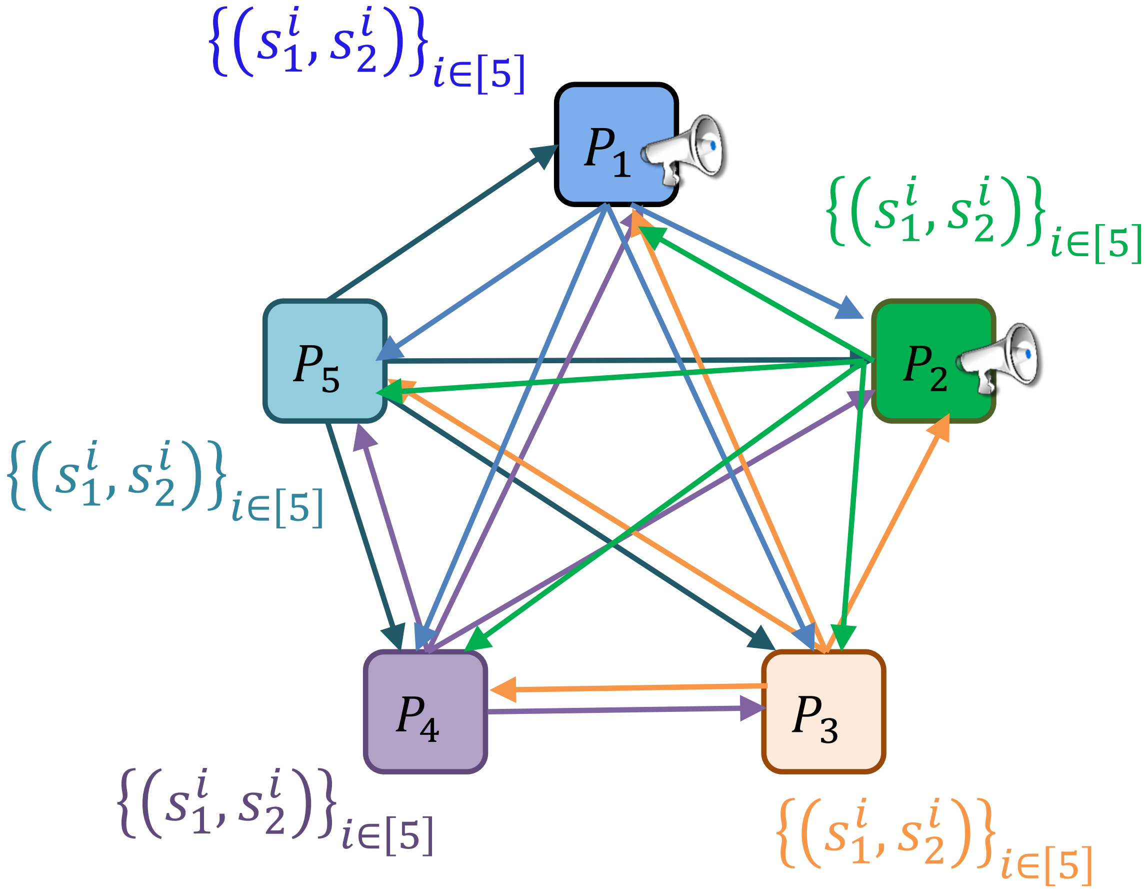
- Round 1:
  $P_1$ and $P_2$ VSS their input message
- Round 2:
  everyone echo their shares



# Simultaneous Broadcast with CR

Uses 5-party, 1-secure, 1-round VSS [GIKR01]
(2 shares suffice to reconstruct)

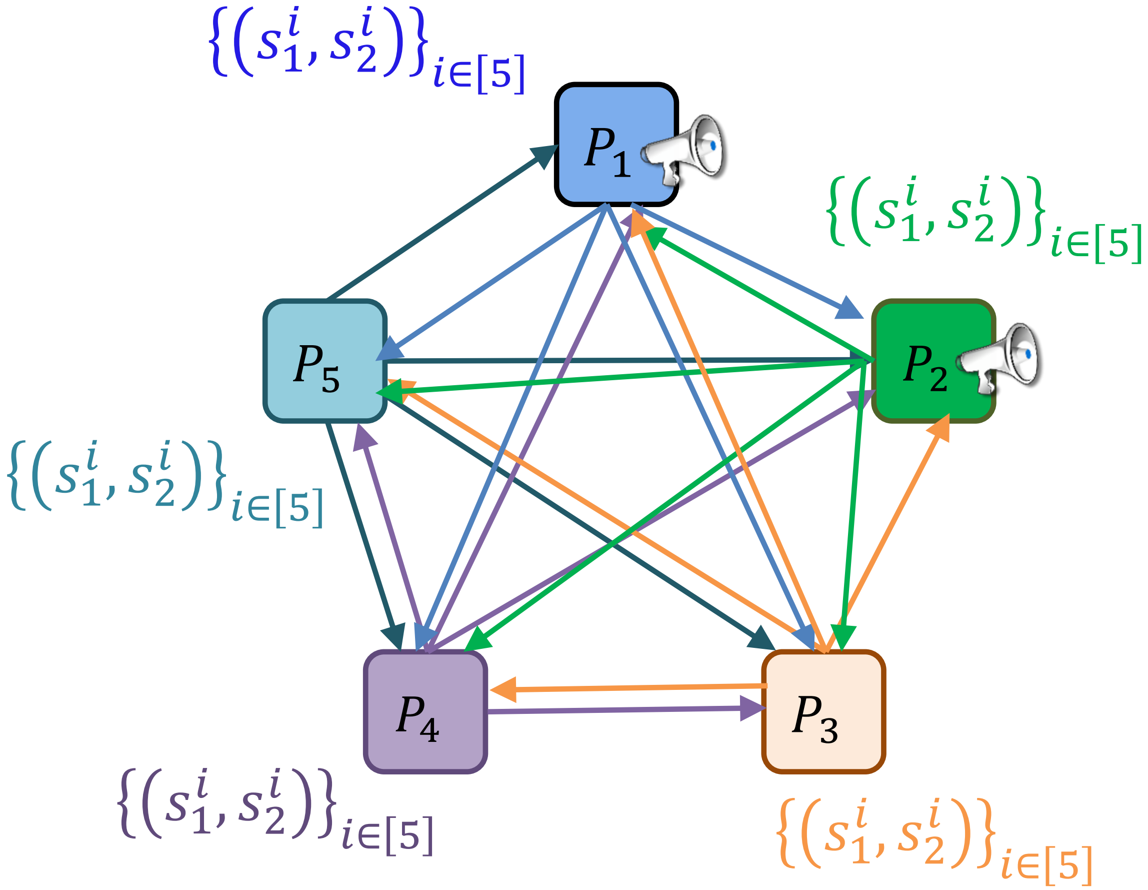
- Round 1:
  $P_1$ and $P_2$ VSS their input message
- Round 2:
  everyone echo their shares



# Simultaneous Broadcast with CR

Uses 5-party, 1-secure, 1-round VSS [GIKR01]
(2 shares suffice to reconstruct)

- Round 1:
  $P_1$ and $P_2$ VSS their input message
- Round 2:
  everyone echo their shares
- Output:
  reconstruct $(m'_1, m'_2)$

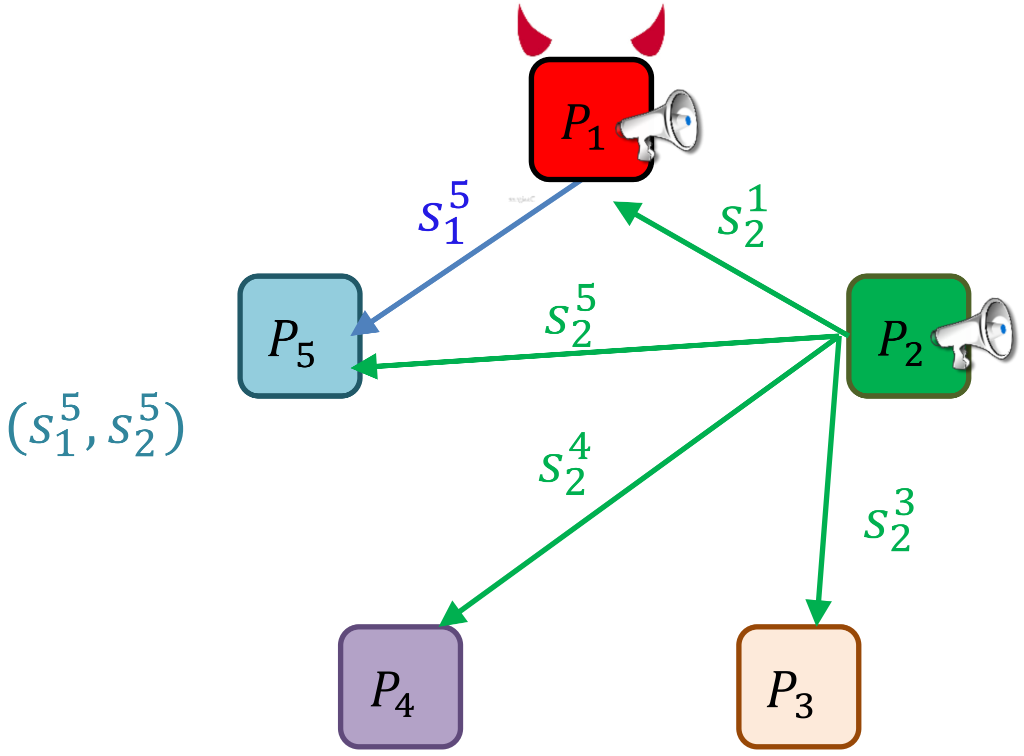



**Security against rushing adversary**

- Round 1: committal round (CR)
- Round 2: output revealing round (ORR)

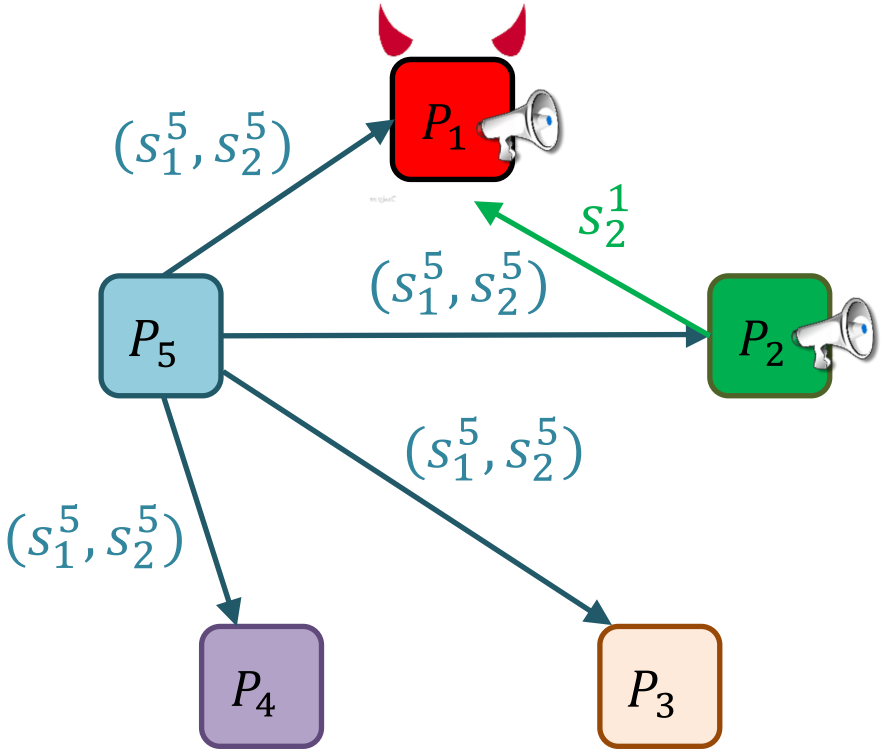
# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$



# A Super-Rushing Attack

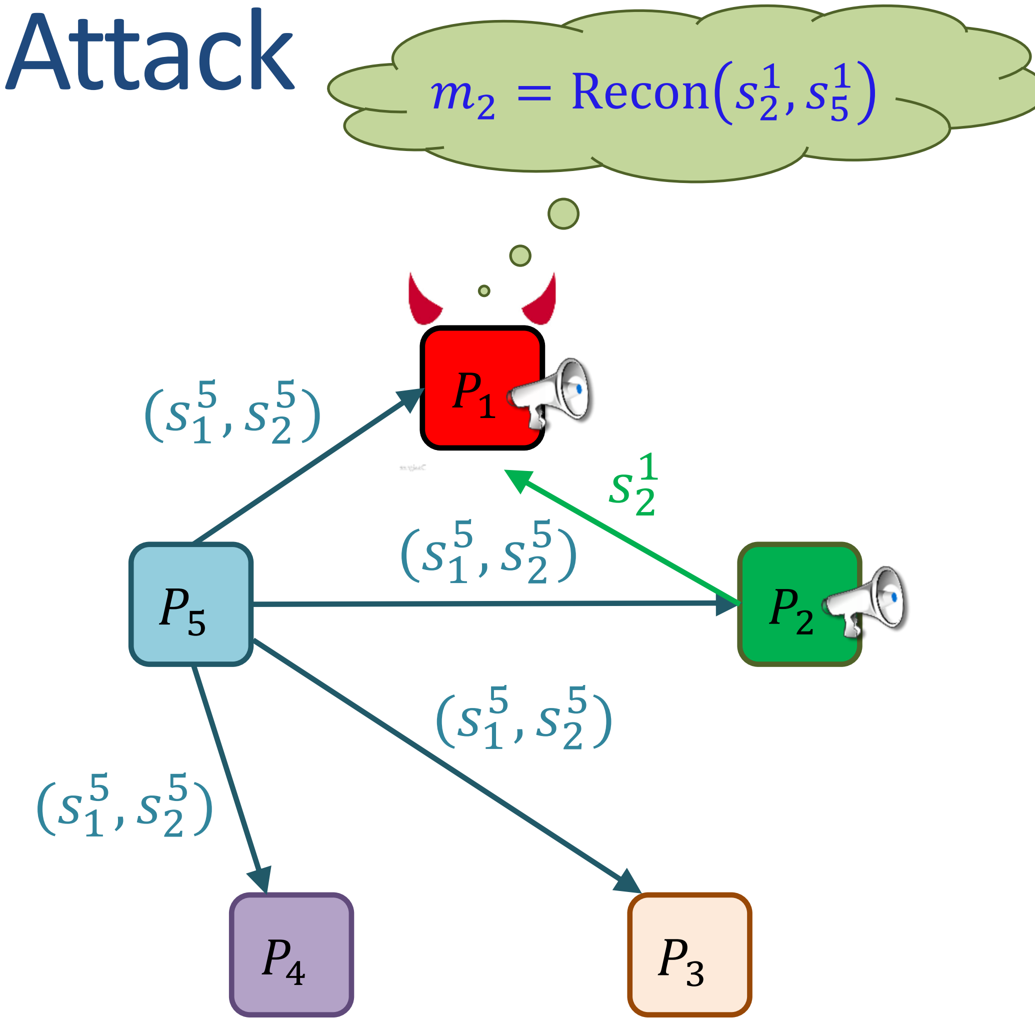
- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$
- Round 2:
  $P_5$ echos $(s_1^5, s_2^5)$ to $P_1, P_2, P_3, P_4$



# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$
- Round 2:
  $P_5$ echos $(s_1^5, s_2^5)$ to $P_1, P_2, P_3, P_4$

➢ $P_1$ reconstructs $m_2$ from $s_2^1$ and $s_2^5$

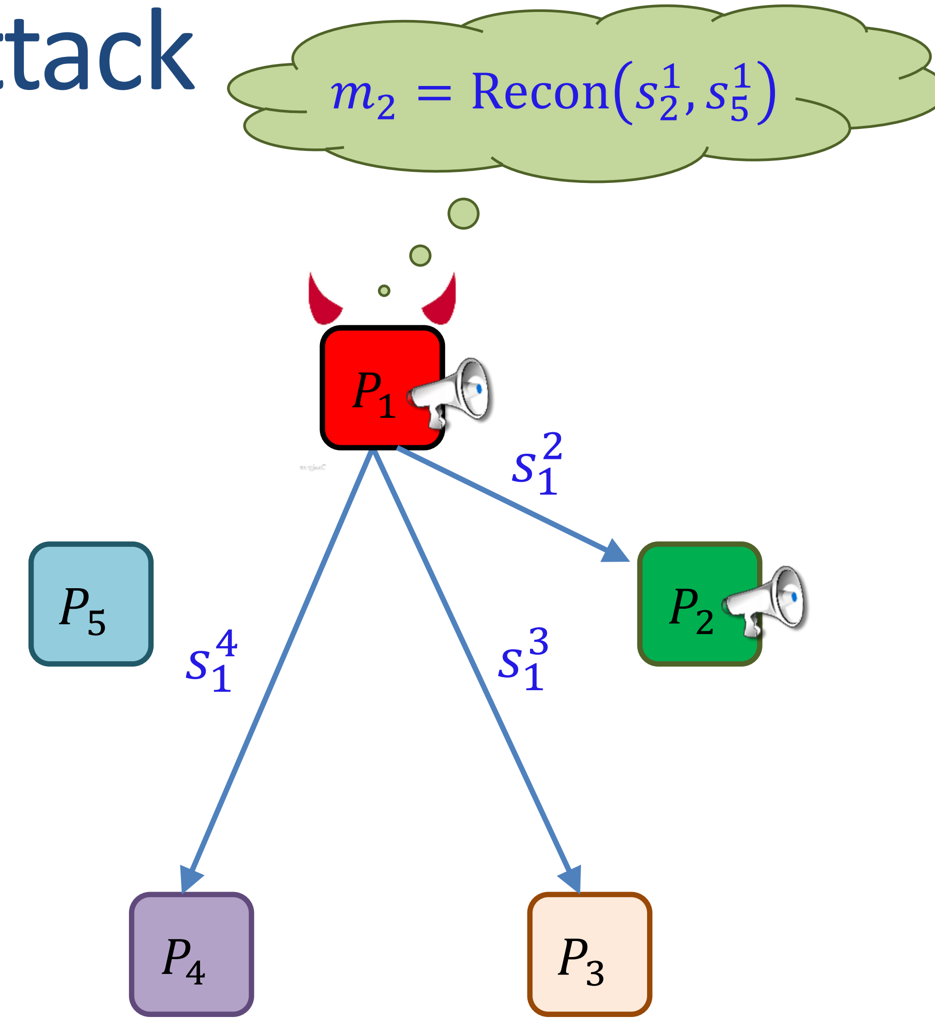


# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$
- Round 2:
  $P_5$ echos $(s_1^5, s_2^5)$ to $P_1, P_2, P_3, P_4$

➢ $P_1$ reconstructs $m_2$ from $s_2^1$ and $s_2^5$

- Round 1:
  $P_1$ VSS $m_2$ to $P_2, P_3, P_4$

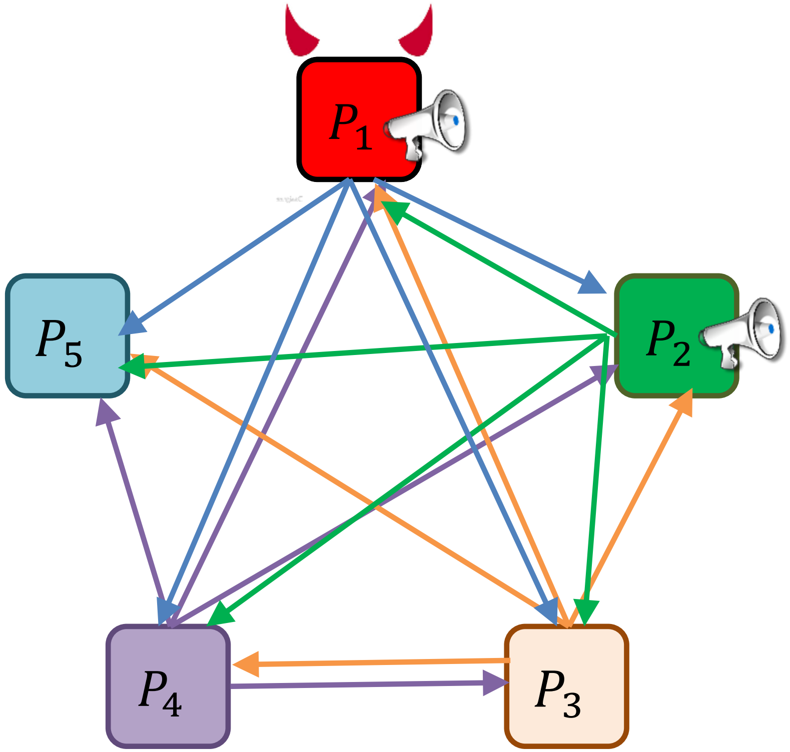


# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$
- Round 2:
  $P_5$ echos $(s_1^5, s_2^5)$ to $P_1, P_2, P_3, P_4$

➢ $P_1$ reconstructs $m_2$ from $s_2^1$ and $s_2^5$

- Round 1:
  $P_1$ VSS $m_2$ to $P_2, P_3, P_4$
- Round 2:
  $P_1, P_2, P_3, P_4$ echo their shares

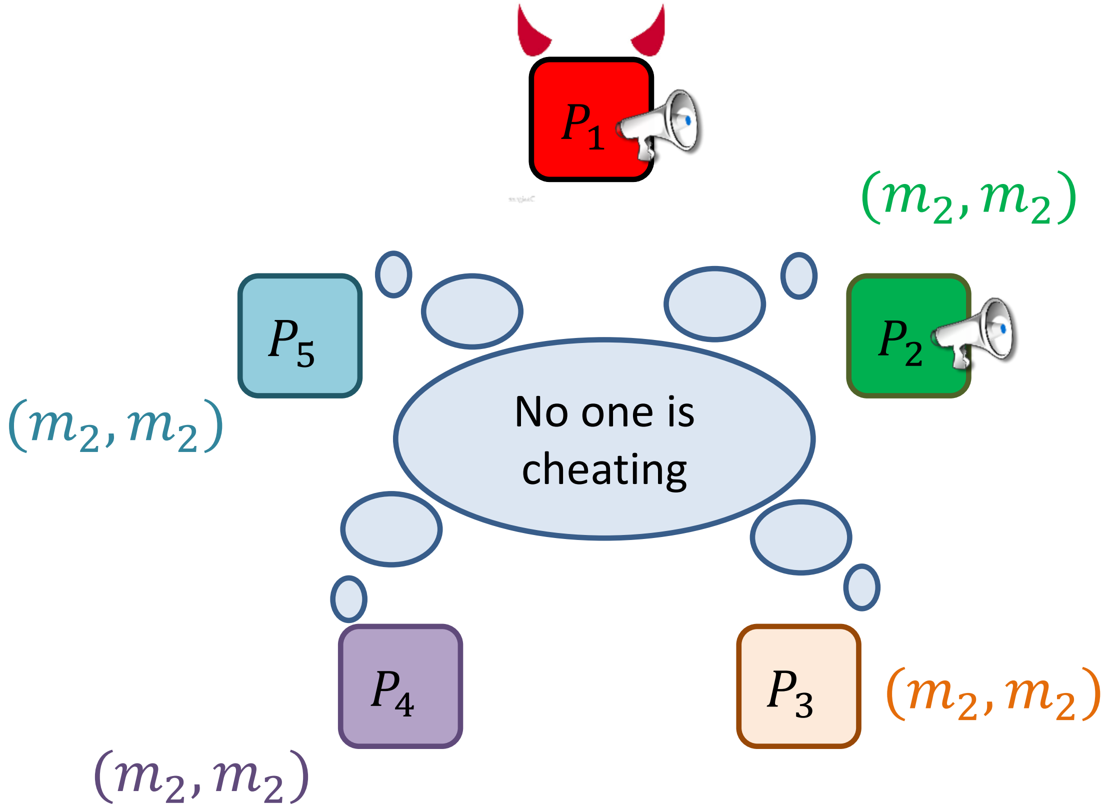


# A Super-Rushing Attack

- Attack: corrupted $P_1$
- Round 1:
  $P_2$ VSS $m_2$
  $P_1$ sends a random share only to $P_5$
- Round 2:
  $P_5$ echos $(s_1^5, s_2^5)$ to $P_1, P_2, P_3, P_4$

➢ $P_1$ reconstructs $m_2$ from $s_2^1$ and $s_2^5$

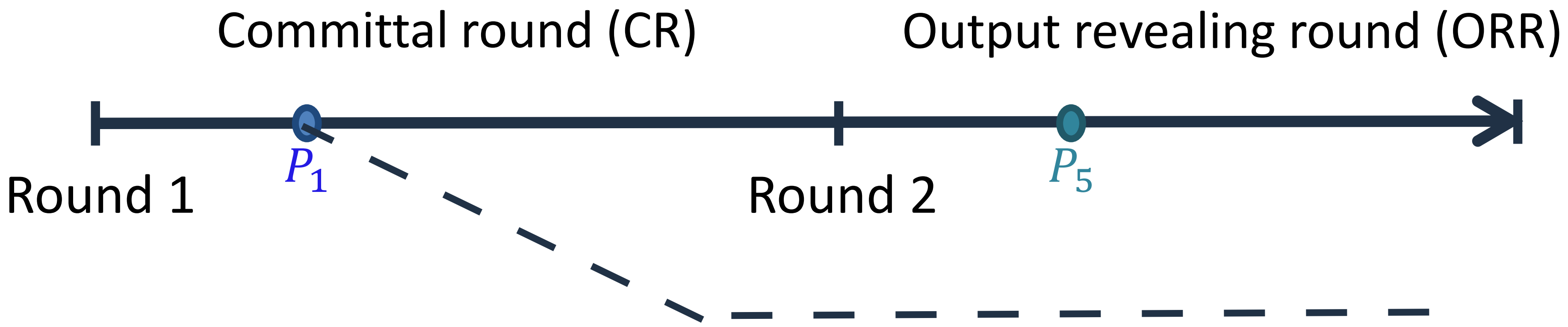
- Round 1:
  $P_1$ VSS $m_2$ to $P_2, P_3, P_4$
- Round 2:
  $P_1, P_2, P_3, P_4$ echo their shares
- Output:
  everyone outputs $(m_2, m_2)$



CR doesn't help!
Super-rushing still breaks
independence of inputs

So, what are the sufficient conditions
for tolerating super-rushing attacks?

# What happened in this “Back to the Future" Attack

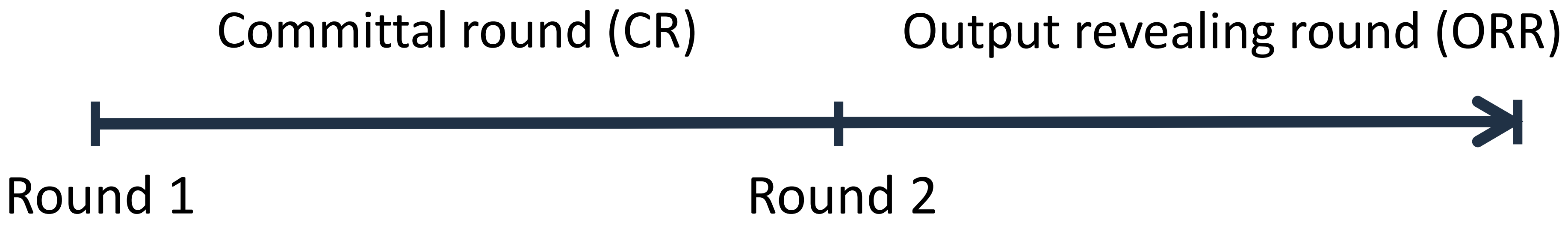



① $P_1$ advances $P_5$ to round 2

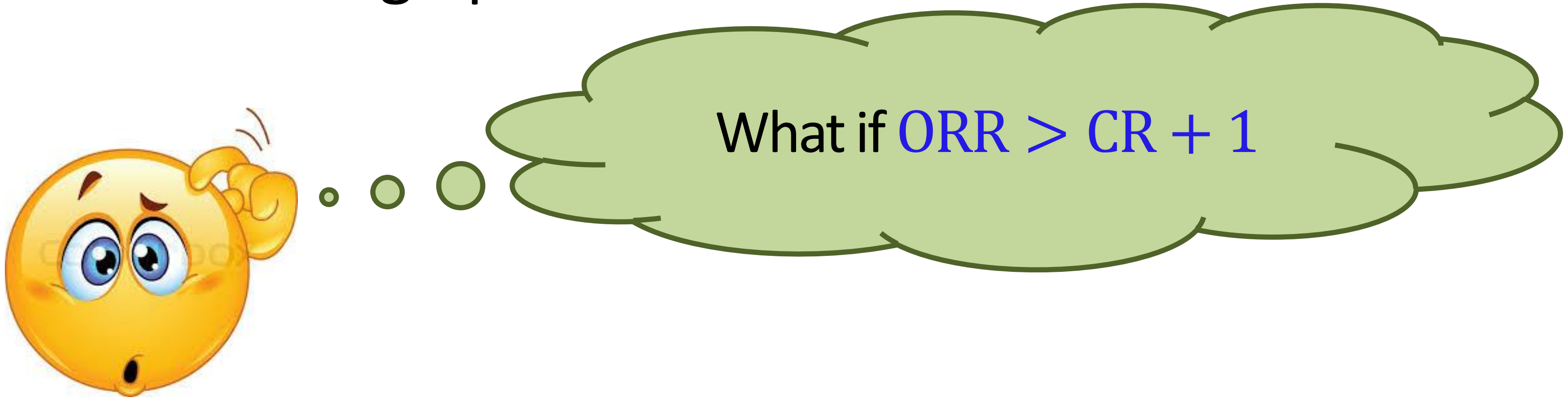
② $P_1$ peeks into round-2 ($P_5$’s round-2 message) & learns $P_2$’s input message

③ $P_1$ chooses input message as a function of $P_2$’s input message

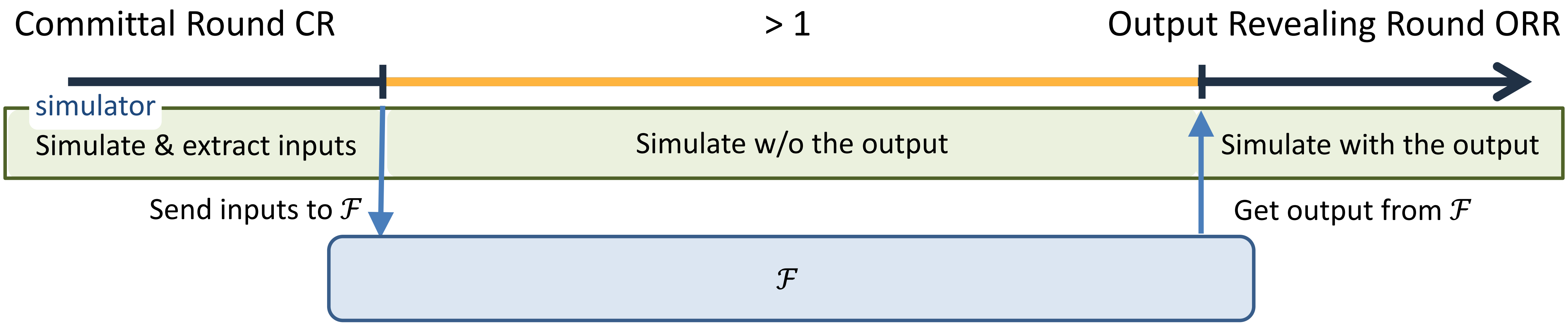
# What happened in this “Back to the Future" Attack




- Here $\mathrm{CR} = 1$ and $\mathrm{ORR} = 2$
- That is, $\mathrm{ORR} = \mathrm{CR} + 1$
- All-to-all communication $\Longrightarrow$ Peeking up to 1 round



# Our Sufficient Condition




**Mitigates Super-Rushing:** can peek into ORR only after everyone have completed CR

# Our Results #3

**Theorem:** every protocol that is

1) Perfectly secure against **rushing** adversaries *
2) Has all-to-all communication
3) $\mathrm{ORR} > \mathrm{CR} + 1$

is also perfectly secure against **super-rushing** adversaries

Can we still support
$\mathrm{ORR} = \mathrm{CR} + 1$?

* security is via "compatible simulation" (see the paper)

# Our Results #3.5

**Theorem:** every protocol that is

1) Perfectly secure against **rushing** adversaries *
2) Has all-to-all communication
3) **$\mathbf{ORR = CR + 1}$, but CR is over broadcast**

is also perfectly secure against **super-rushing** adversaries

Adv cannot change its message

Can we still support $\mathrm{ORR} = \mathrm{CR} + 1$?

* security is via "compatible simulation" (see the paper)

# Corollary

BGW is secure against super-rushing attacks!


- We have all-to-all & $\mathrm{ORR} > \mathrm{CR} + 1$

# Corollary

BGW is secure against super-rushing attacks!


- We have all-to-all & $\mathrm{ORR} > \mathrm{CR} + 1$
- What about linear functions with $\mathrm{ORR} = \mathrm{CR} + 1$?
- The VSS ends with a broadcast round
- Same for round-efficient variants [ABT19,AKP20]

# Our Main Result

**Corollary:** BGW is secure against super-rushing attacks!

Stronger
security for free!

BGW be executed optimistically:

- Parties advance upon receiving messages
- Everyone talk ⇒ no need for continuous synchronization & long delays
- Timeouts only needed to detect parties who don’t talk

# Our Results #4

What about statistical security?

**Theorem:** ∃ a protocol that is

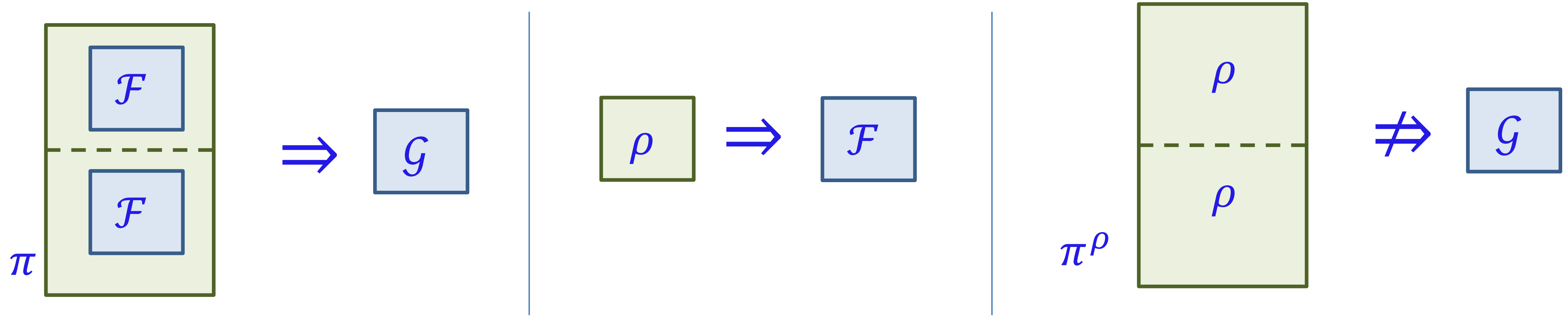
1) **Statistically** secure against **rushing** adversaries
2) Has all-to-all communication
3) $\mathrm{ORR} > \mathrm{CR} + 1$

But is **not** statistically secure against **super-rushing** adversaries

# Our Results #5

**Theorem:** super-rushing security is not sequentially composable

- $\exists$ functionalities $\mathcal{F}$ and $\mathcal{G}$
- $\exists$ a protocol $\pi$ that realizes $\mathcal{G}$ against super-rushing in the $\mathcal{F}$-hybrid model
- $\exists$ a protocol $\rho$ that realizes $\mathcal{F}$ against super-rushing
- But $\pi^{\rho}$ does not realize $\mathcal{G}$ against super-rushing



# The Story So Far (Perfect Security)

✓ **Single Input Provider:** Super-Rushing ≡ Rushing ≡ Non-Rushing

✗ **Two Input Providers:** Super-Rushing ≢ Rushing ≢ Non-Rushing

✗
- Committal round does not help (on its own)

✗ Modular analysis is tricky (no sequential composition)

✓ **Sufficient conditions:** Rushing ⇒ super-rushing if

- All-to-all communication
- $\mathrm{ORR} > \mathrm{CR} + 1$, or $\mathrm{ORR} = \mathrm{CR} + 1$ and CR over broadcast

✗ This result doesn't extend to statistical security

# An Alternate Strategy

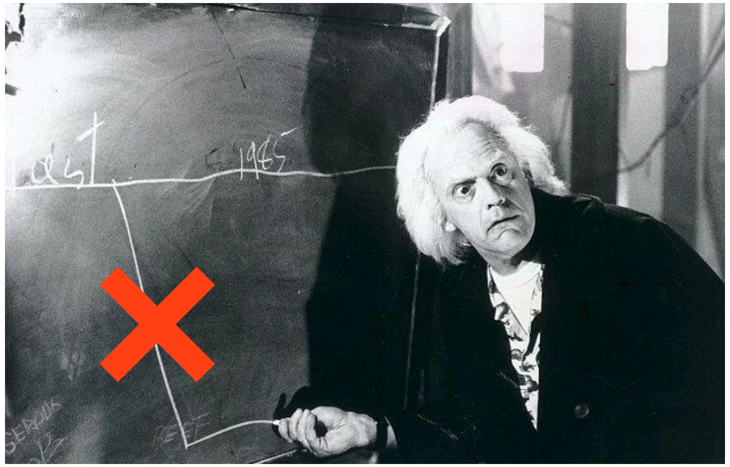
- Kushilevitz, Lindell, and Rabin [STOC ’06]
  - A generic compiler of synchronous MPC to asynchronous UC
  - In each round:
    1) Each party waits for all messages
    2) Sends OK to all
    3) Once receiving OK from all, advances to the next round
  - Can be used for optimistically execute synchronous MPC
  - But $\times 2$ round complexity and $+ O(n^2)$ communicaiton
- This work: analyze unmodified synchronous protocols

# Coming soon

- New sufficient conditions for perfect MPC with $\mathrm{ORR} = \mathrm{CR} + 1$ (capture [IKP10] and alike)
- Sequential composition theorem
- Capture protocols w/o all-to-all communication
  - Where communication pattern is fixed and known before each round
  - À la [DN07, GLS19]

# Conclusion

- Optimistic implementations may be vulnerable to “Back to the Future" attacks
- All-to-all & $\mathrm{ORR} > \mathrm{CR} + 1$ sufficient for Rushing $\Longrightarrow$ Super-Rushing



**Conjecture:** most (if not all) general purpose MPC remain secure

Thank You