Towards Practical MPC for Re-staking:

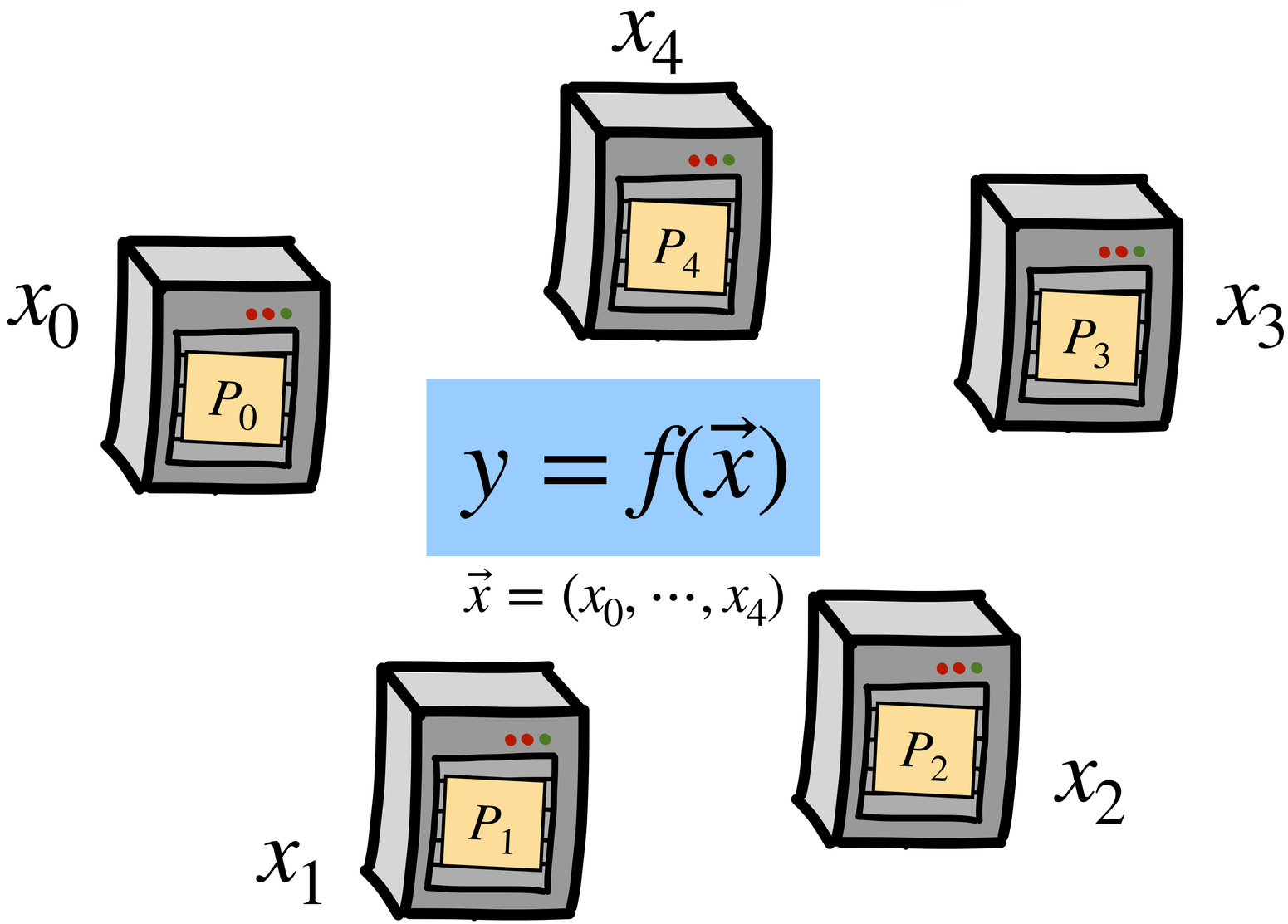
Separating Broadcast from Cheater Identification

Yashvanth Kondi

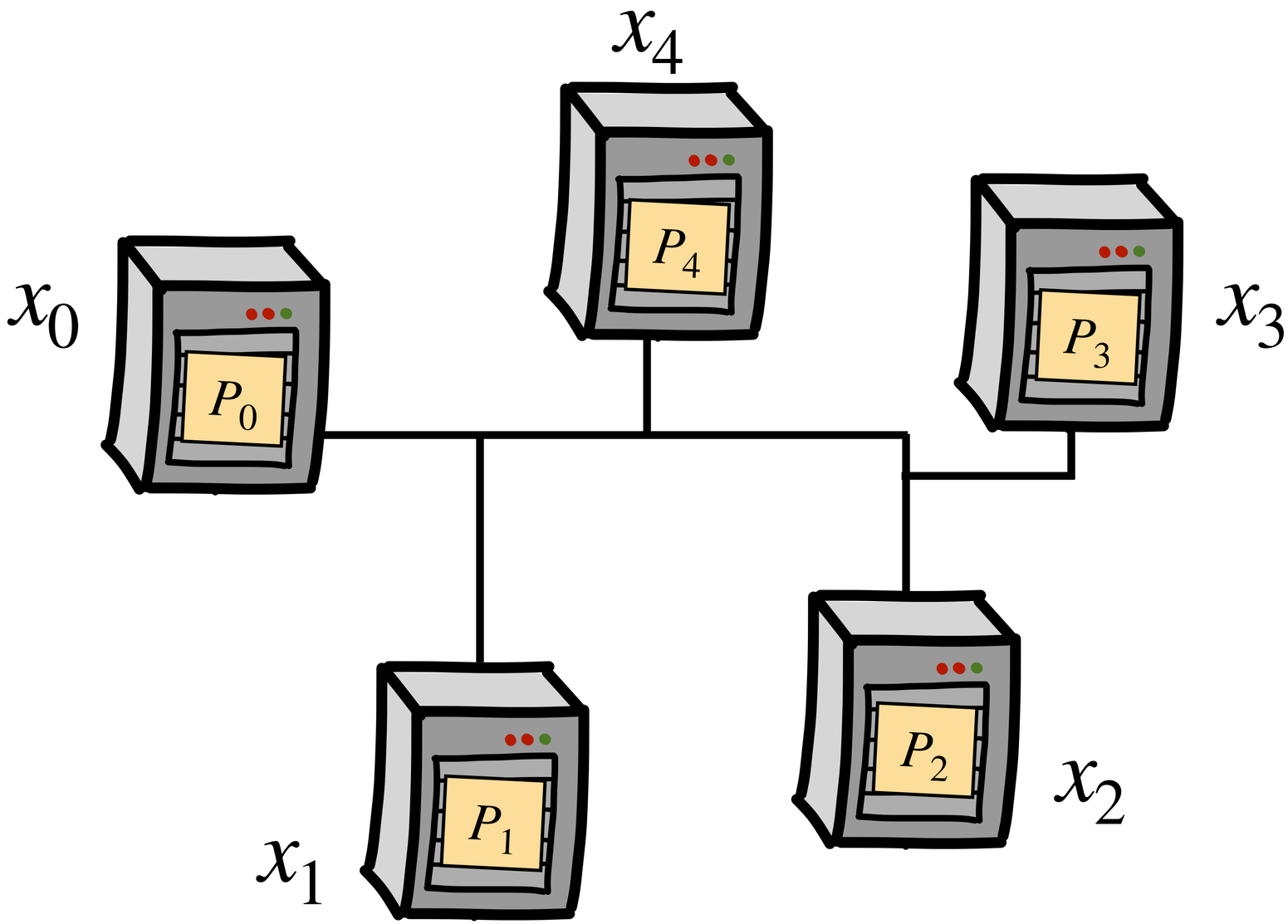
Divya Ravi

UNIVERSITY OF AMSTERDAM

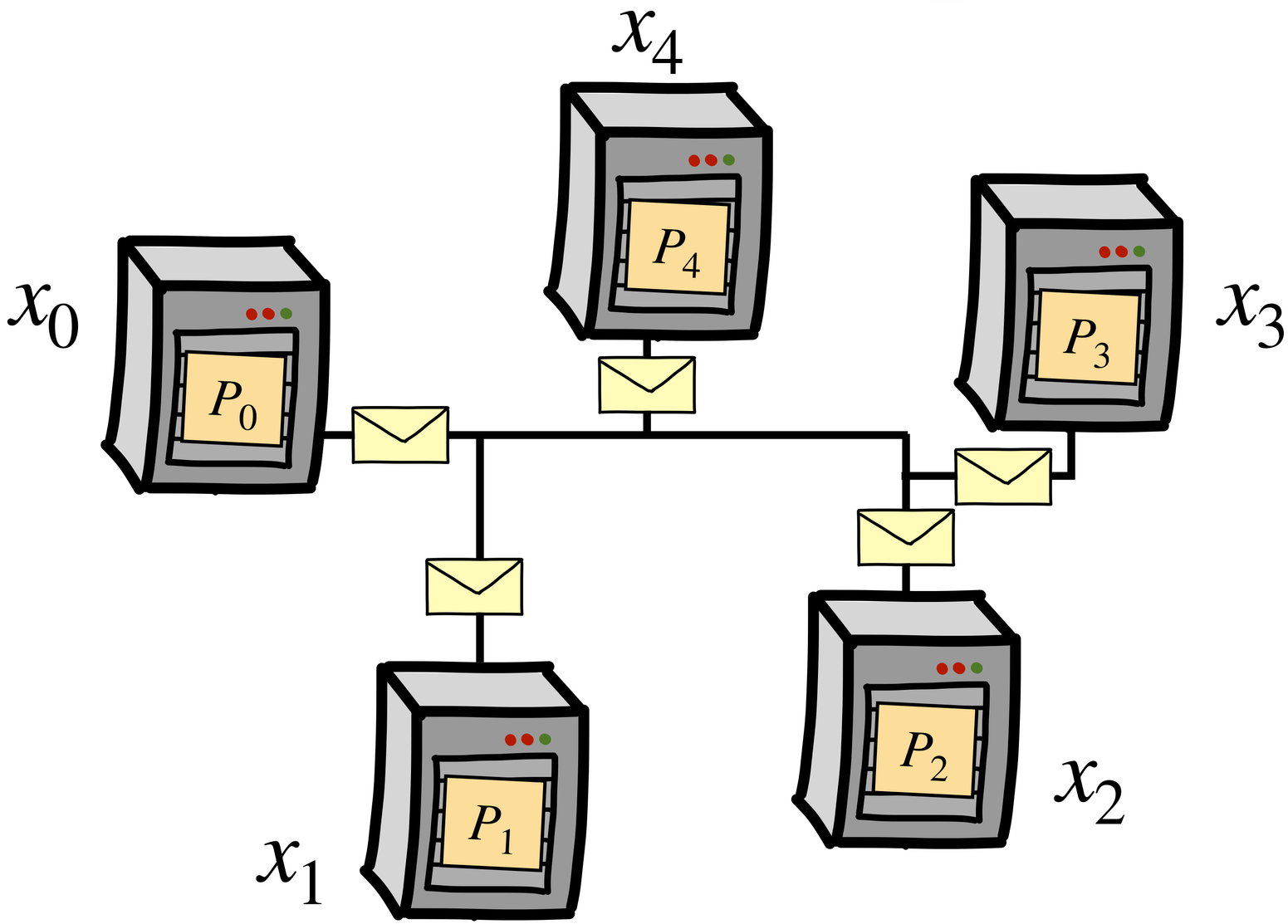
MPC: Recap



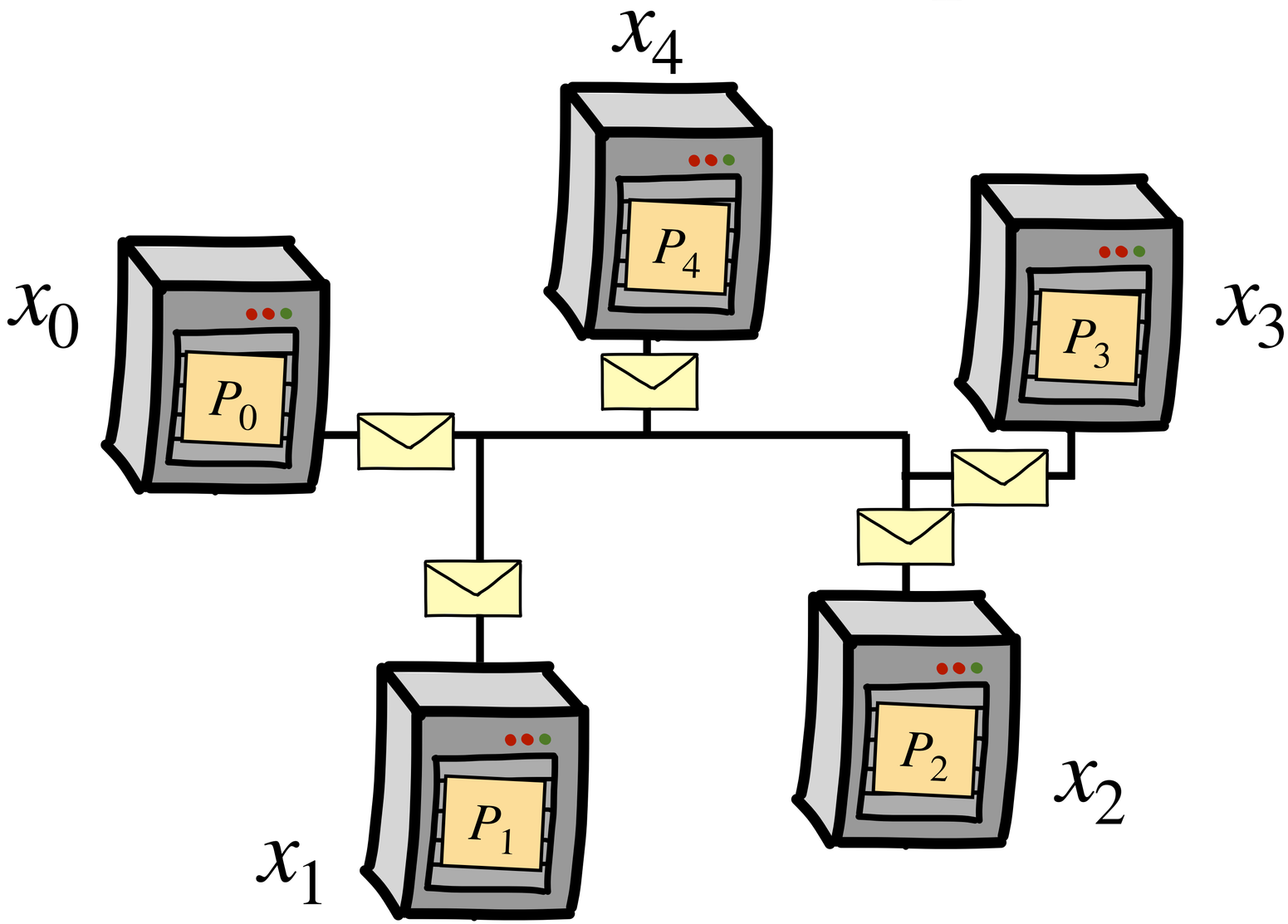
MPC: Recap



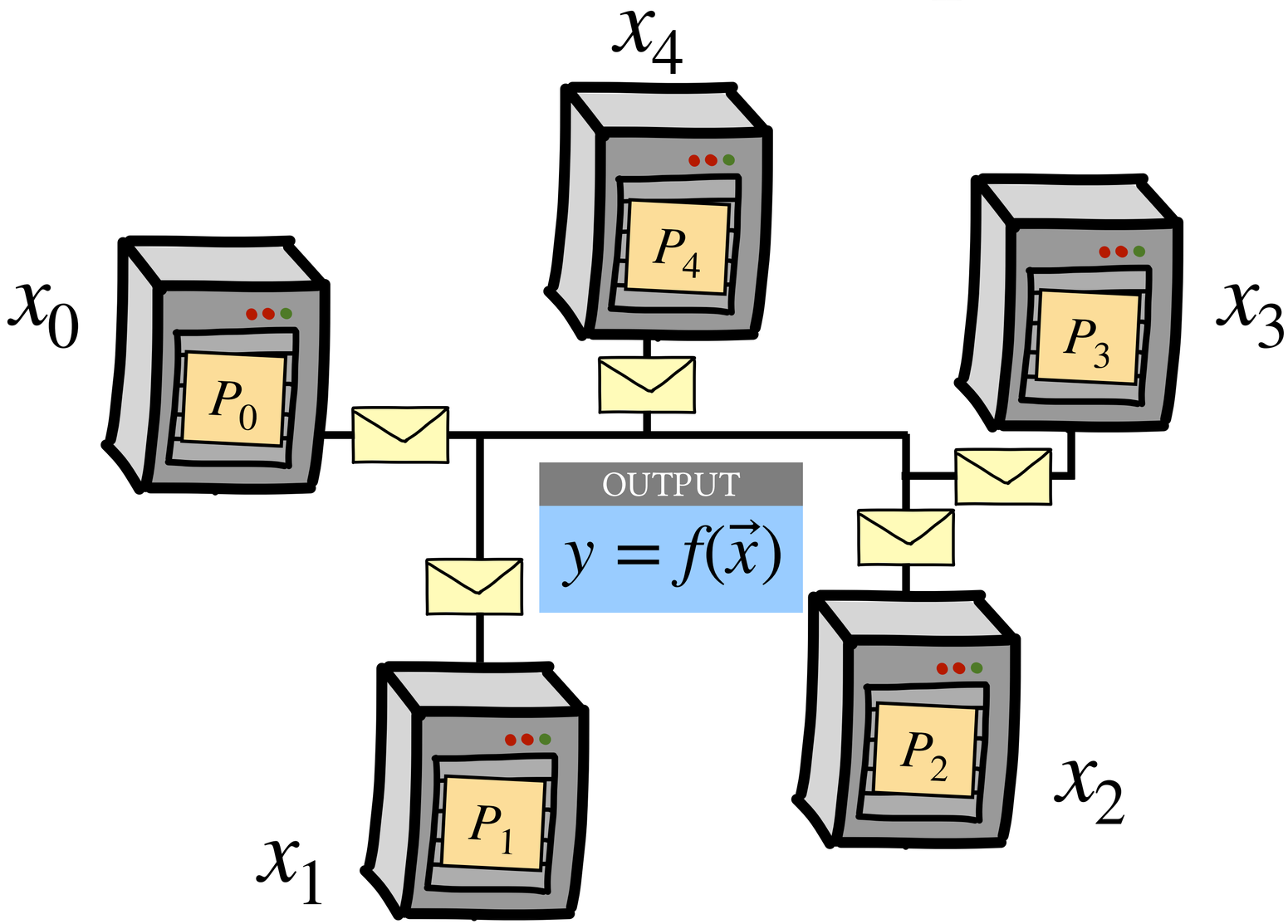
MPC: Recap



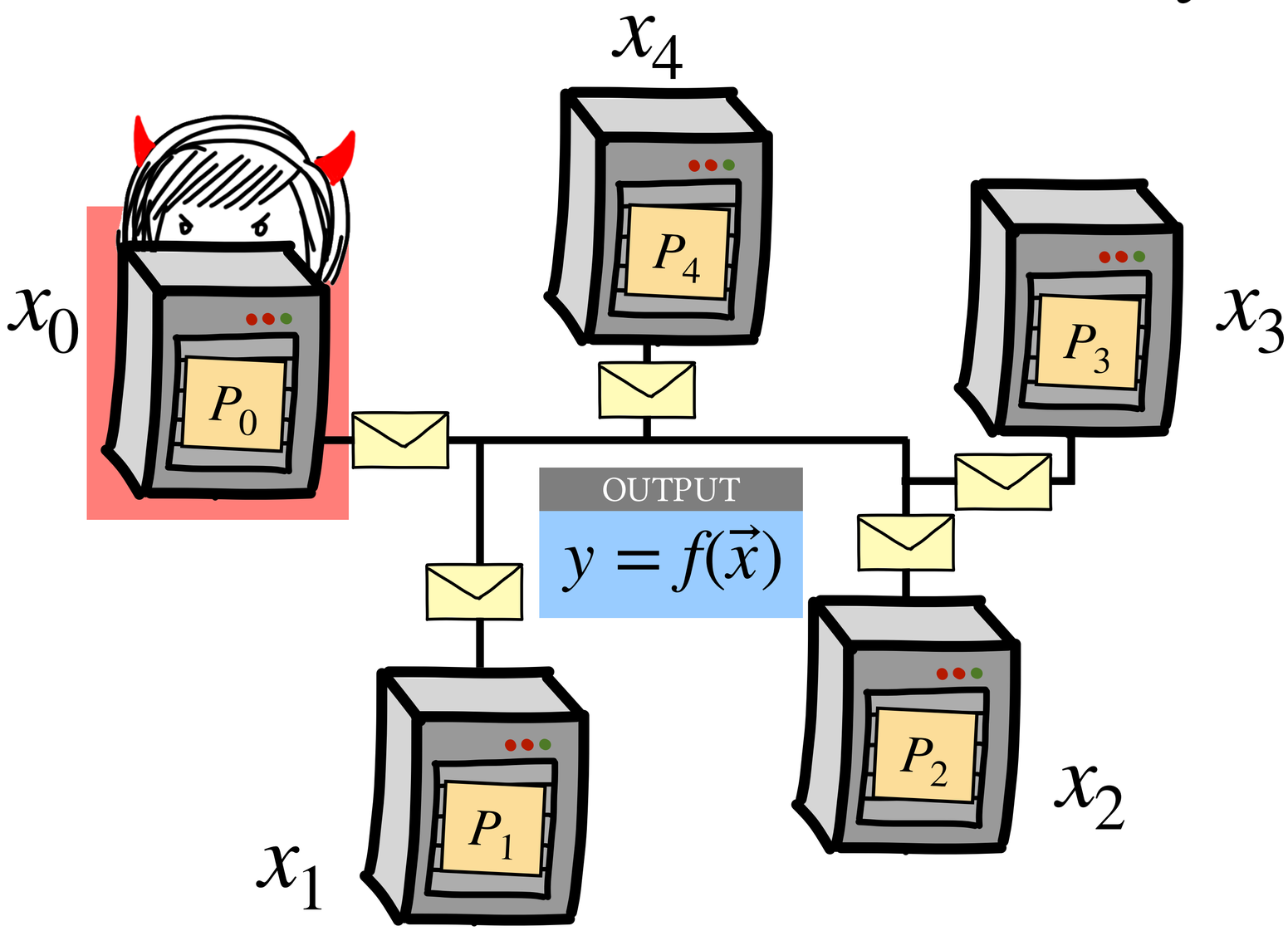
MPC: Recap



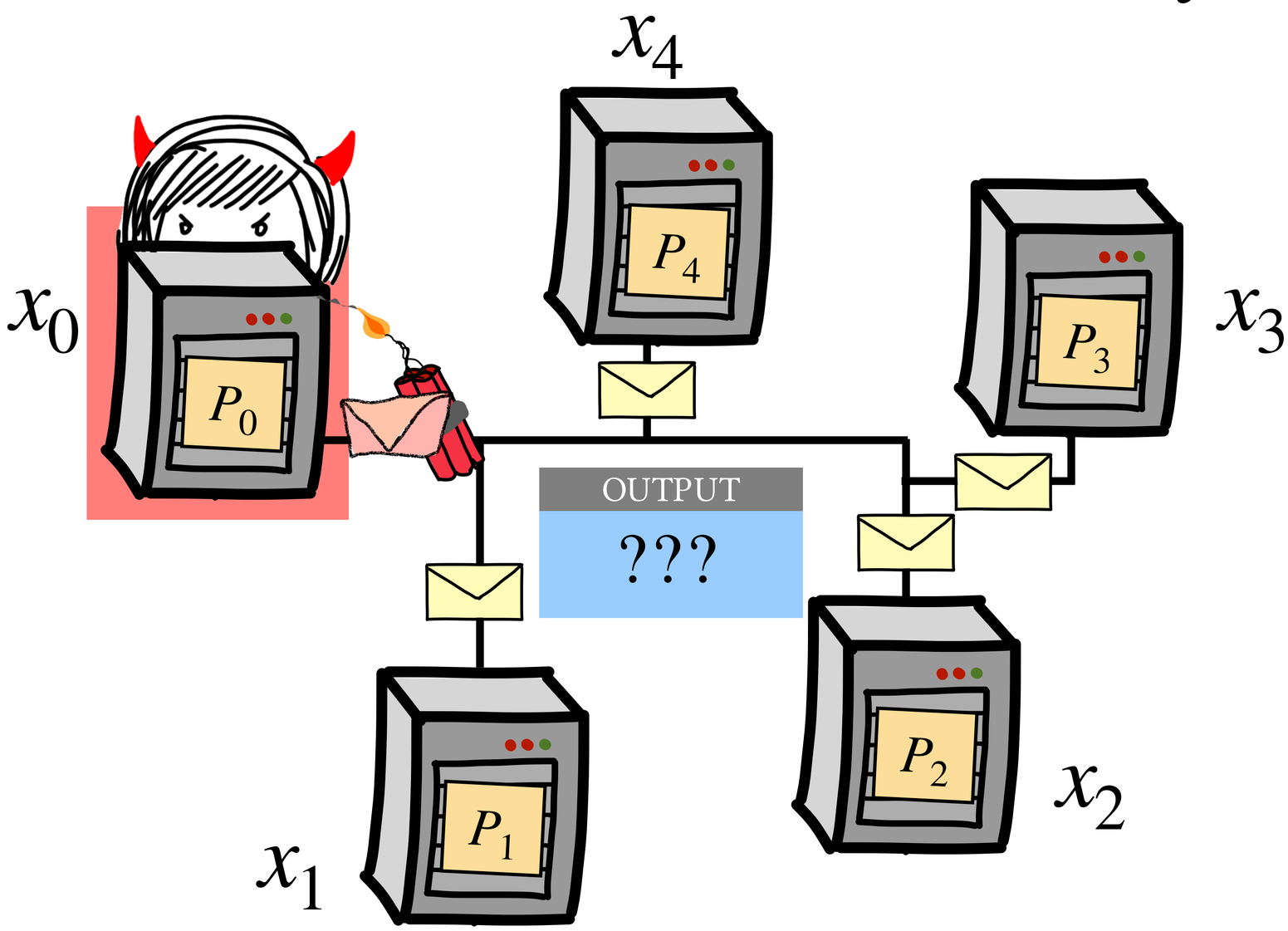
MPC: Recap



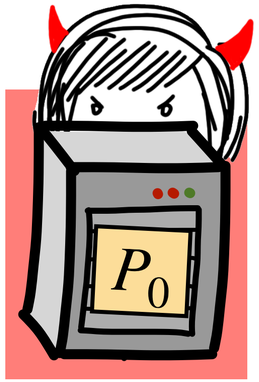
MPC: Active Security

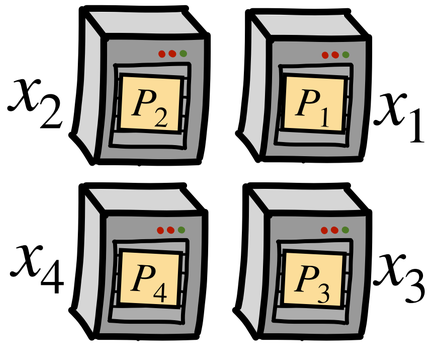


MPC: Active Security



Grades of Active Security



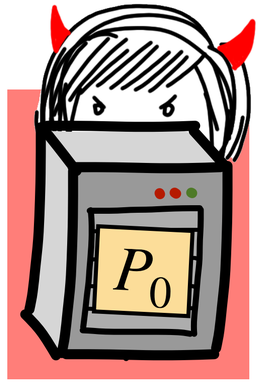


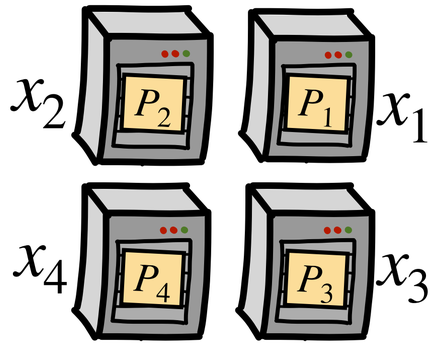
Security with Abort

Fairness

Guaranteed Output

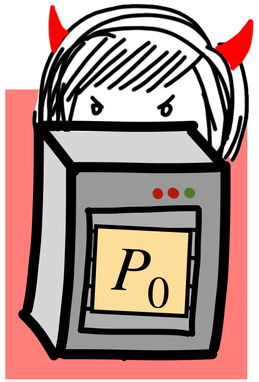
Grades of Active Security

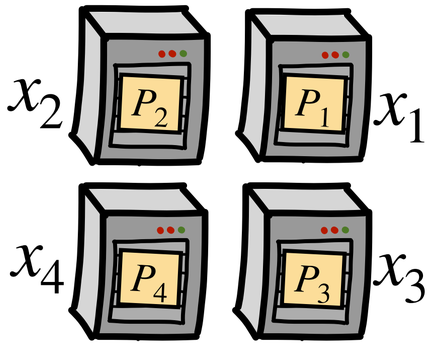




	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

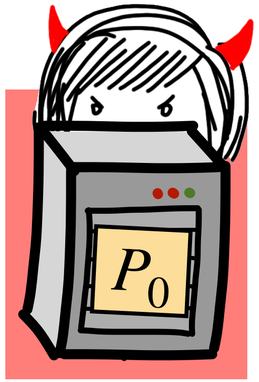
Grades of Active Security

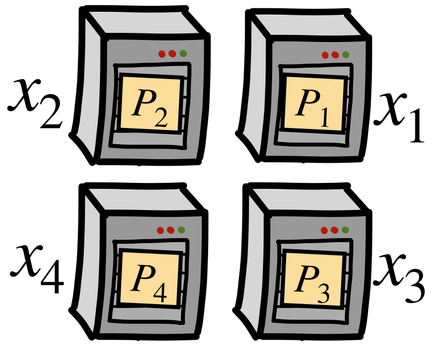




	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness	Case 1 : $y = f(\vec{x})$ Case 2 : $\perp$	$y = f(\vec{x})$ $\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

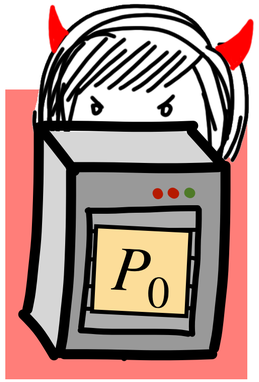
Grades of Active Security

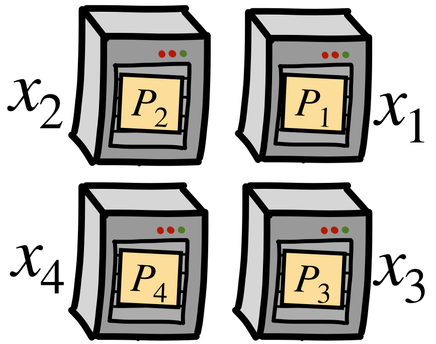




	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

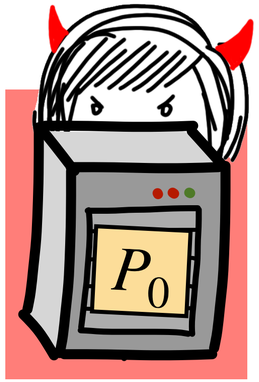
Grades of Active Security

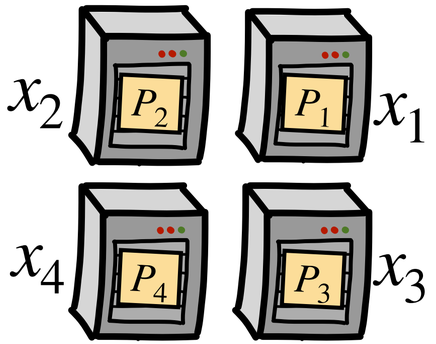




	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

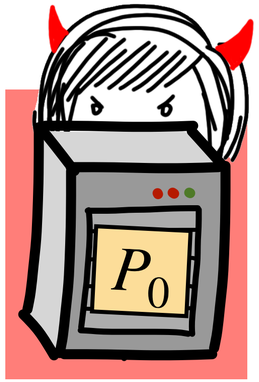
Grades of Active Security

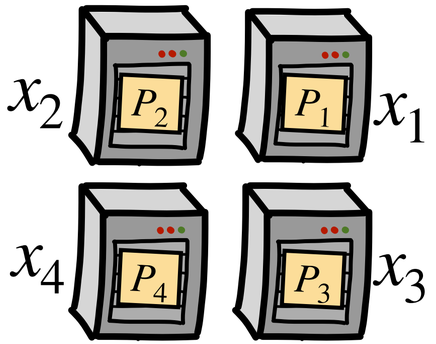




	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output DoS-resistant	$y = f(\vec{x})$	$y = f(\vec{x})$

Grades of Active Security





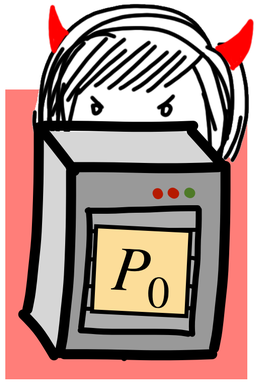
	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Fairness — Case 1 :	$y = f(\vec{x})$	$y = f(\vec{x})$
Fairness — Case 2 :	$\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

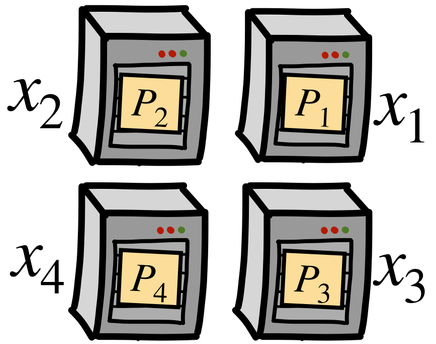
Privacy

x_1, x_2, x_3, x_4 always protected

DoS-resistant

Grades of Active Security





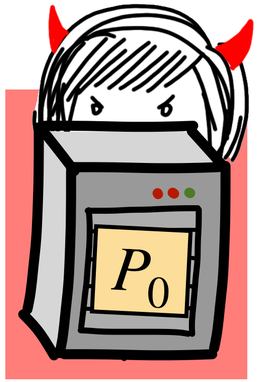
	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Identifiable Abort	$y = f(\vec{x})$	! CHEATER FOUND ! 
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

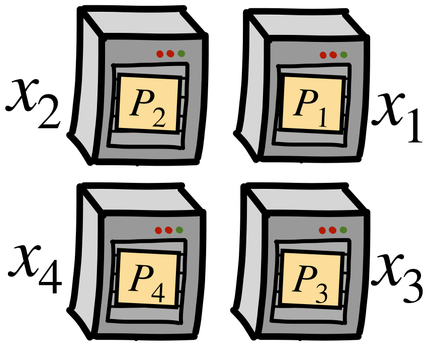
Privacy

x_1, x_2, x_3, x_4 always protected

DoS-resistant

Grades of Active Security





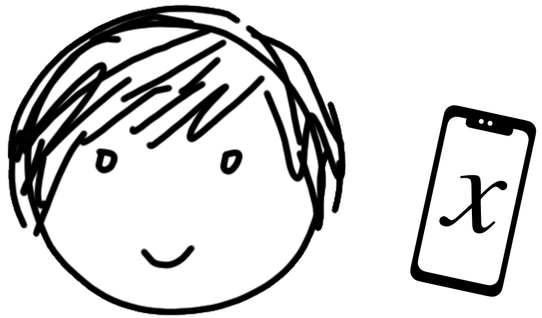
	OUTPUT	
Security with Abort	$y = f(\vec{x})$	$\perp$
Identifiable Abort	$y = f(\vec{x})$	! CHEATER FOUND ! 
Fairness	Case 1 : $y = f(\vec{x})$	$y = f(\vec{x})$
	Case 2 : $\perp$	$\perp$
Guaranteed Output	$y = f(\vec{x})$	$y = f(\vec{x})$

Privacy

x_1, x_2, x_3, x_4 always protected

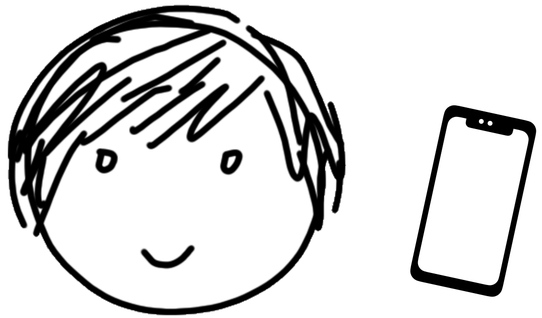
DoS-resistant

Guaranteed Output *vs.* Identifiable Abort



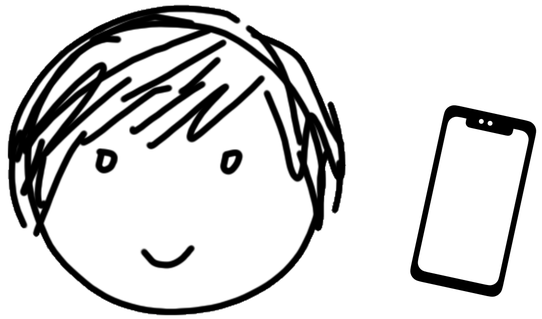
Guaranteed Output *vs.* Identifiable Abort

$\mathsf{SecretShare}(x) \mapsto$

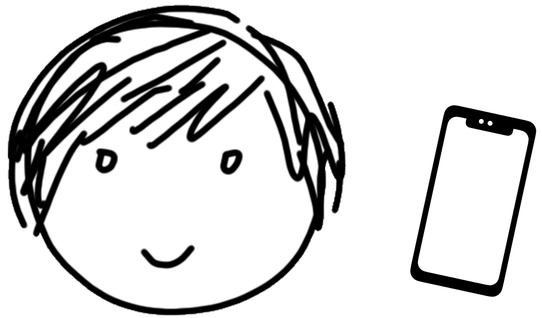


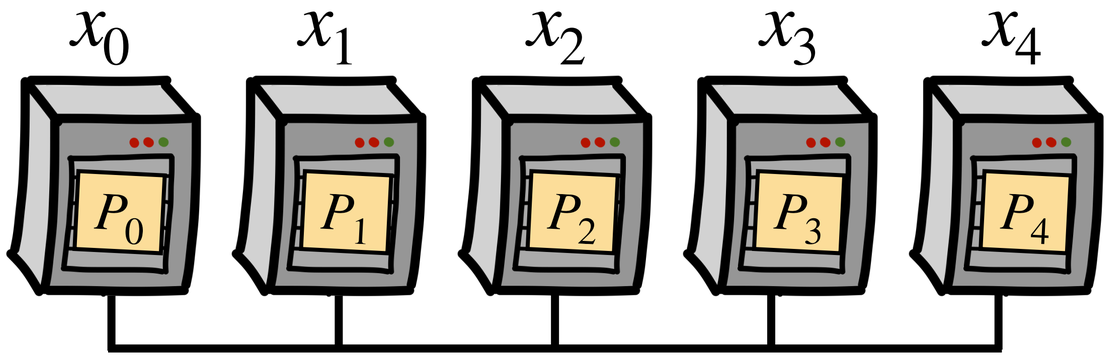
Guaranteed Output *vs.* Identifiable Abort

$\mathsf{SecretShare}(x) \mapsto \quad x_0 \qquad x_1 \qquad x_2 \qquad x_3 \qquad x_4$

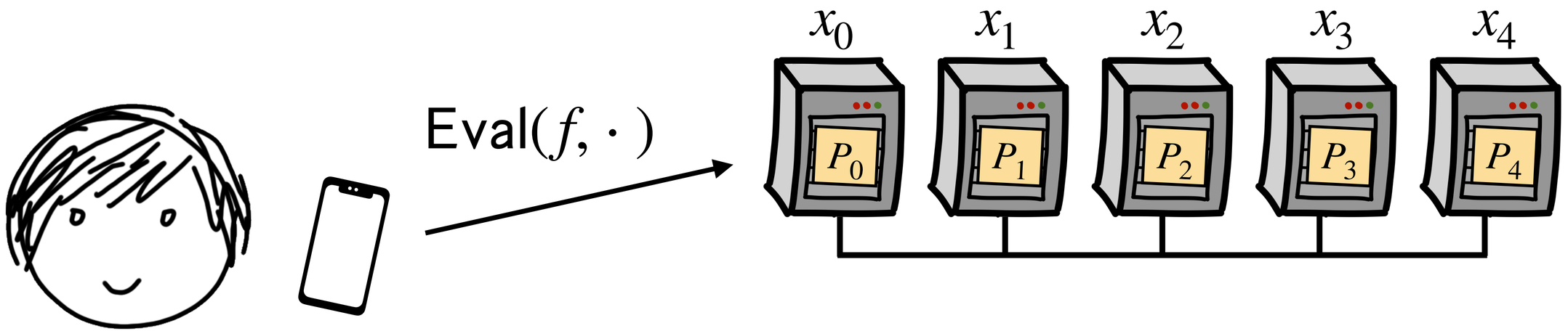


Guaranteed Output *vs.* Identifiable Abort

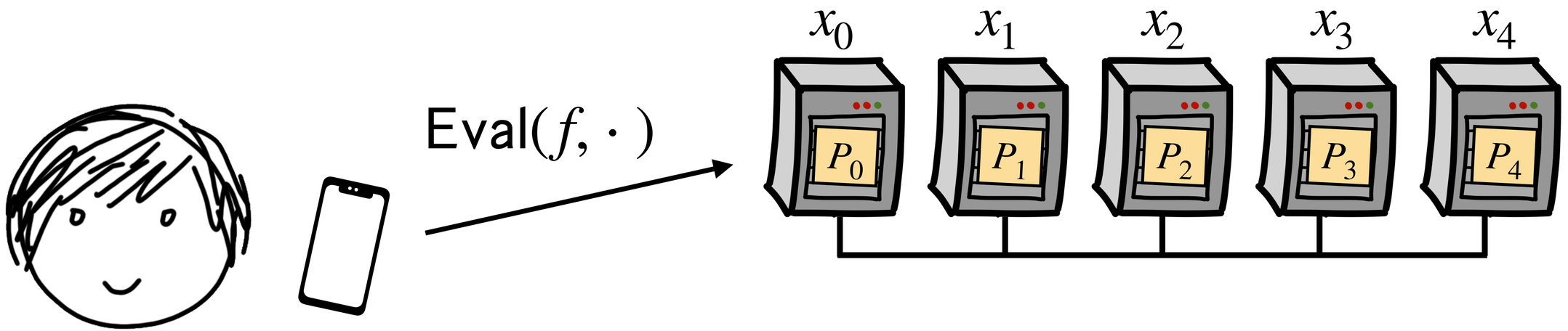




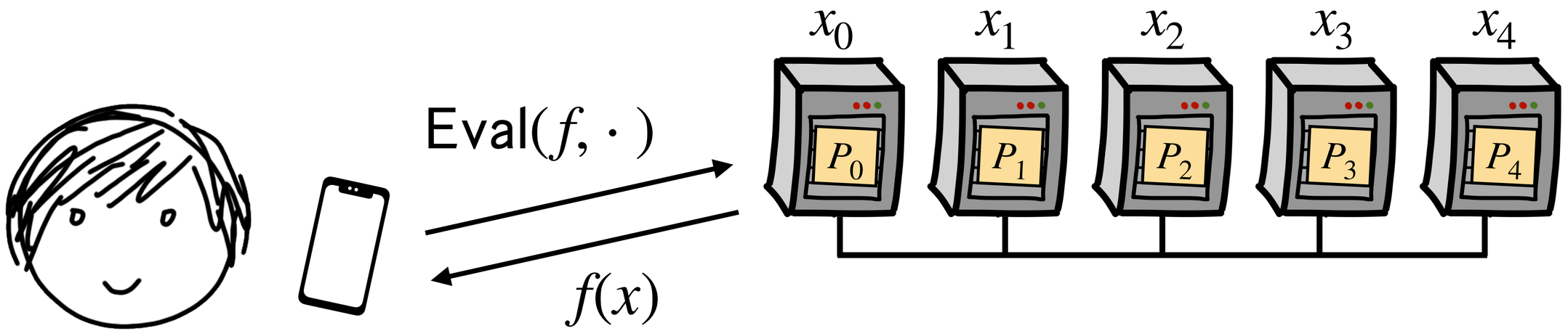
Guaranteed Output *vs.* Identifiable Abort



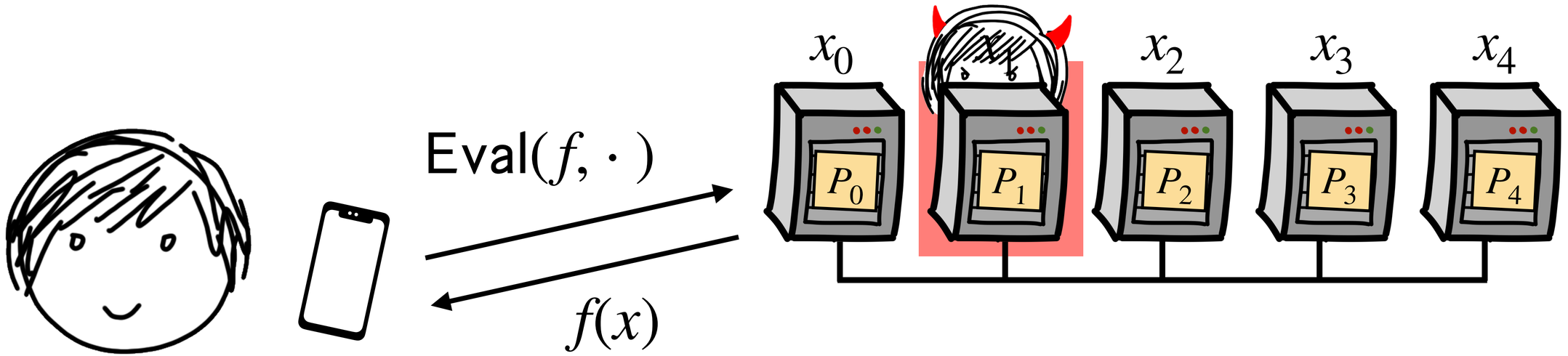
Guaranteed Output *vs.* Identifiable Abort



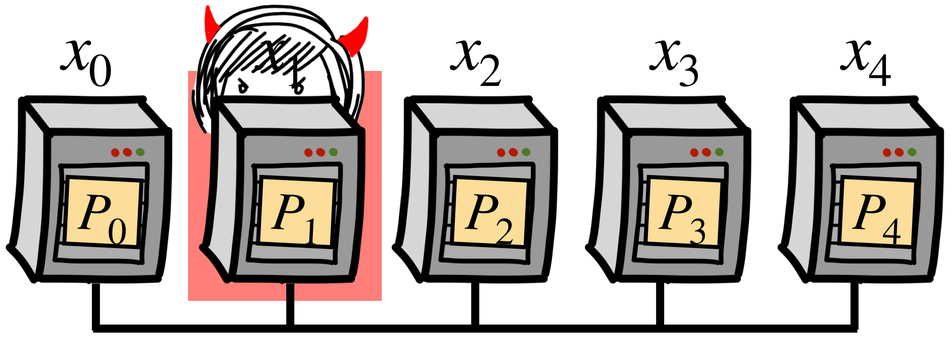
Guaranteed Output *vs.* Identifiable Abort

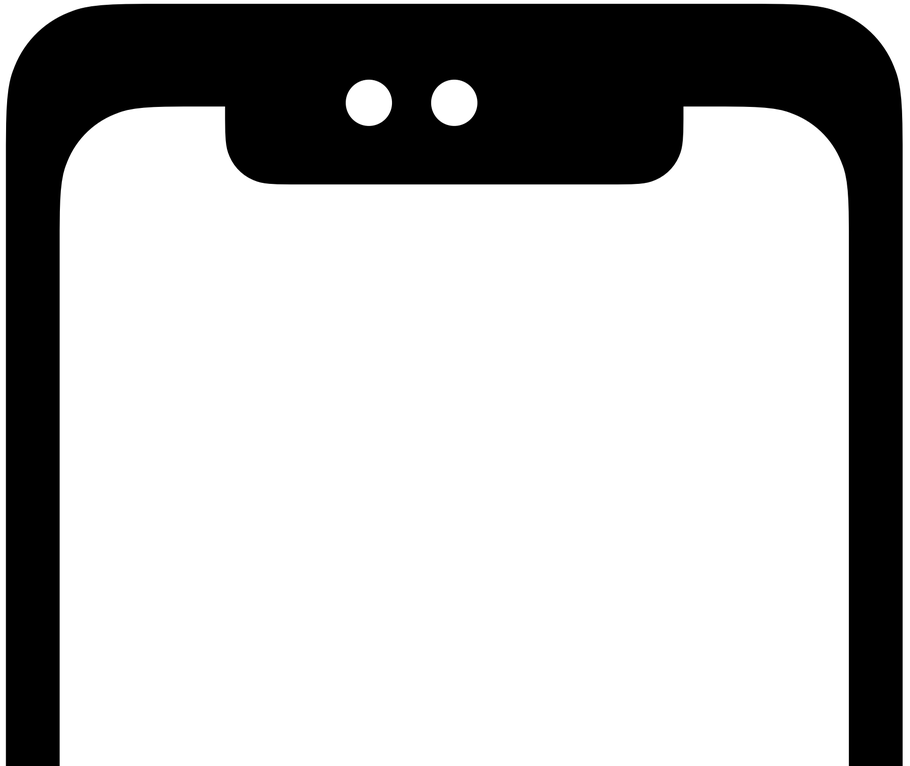


Guaranteed Output *vs.* Identifiable Abort

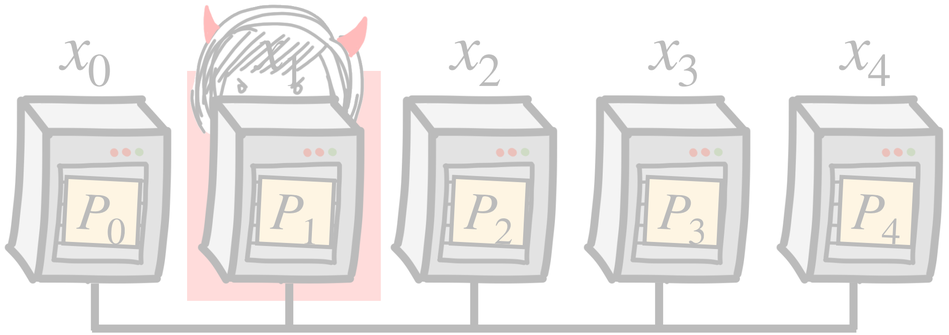


Guaranteed Output *vs.* Identifiable Abort

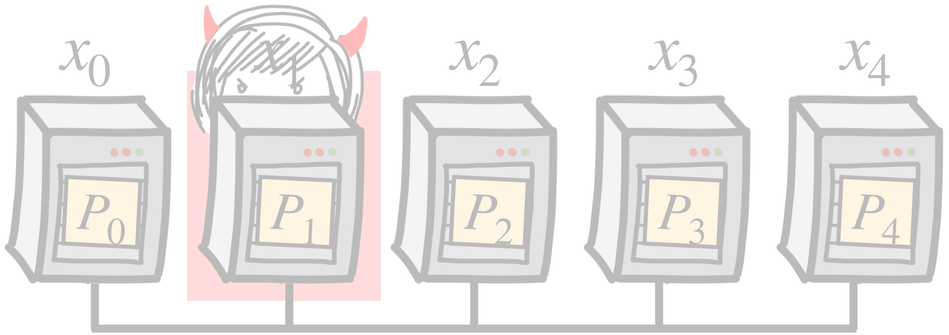


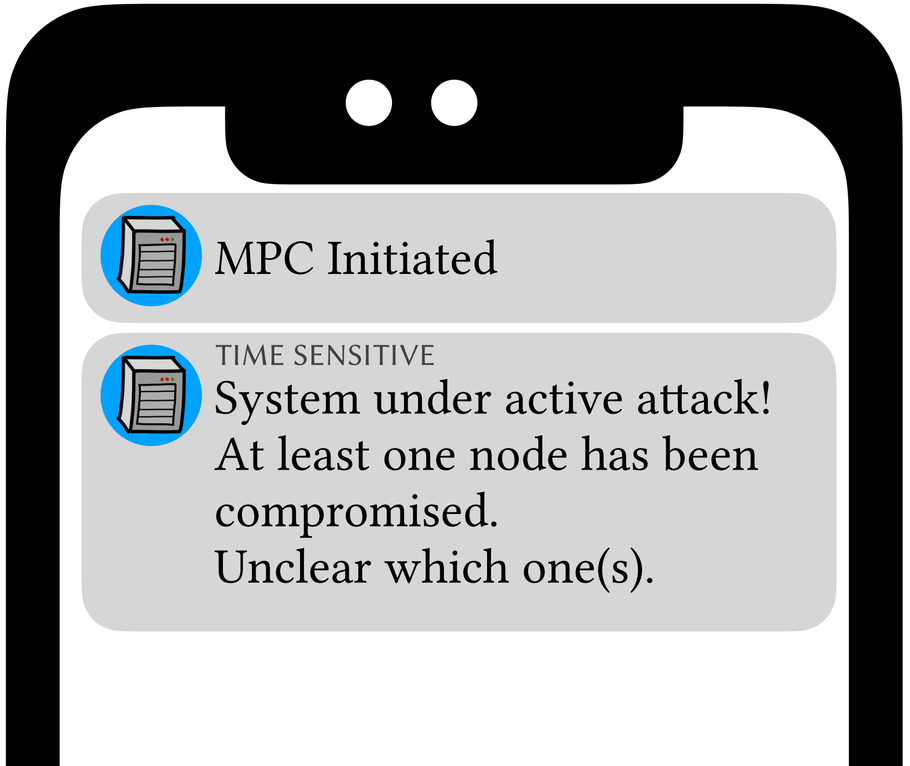


Guaranteed Output *vs.* Identifiable Abort

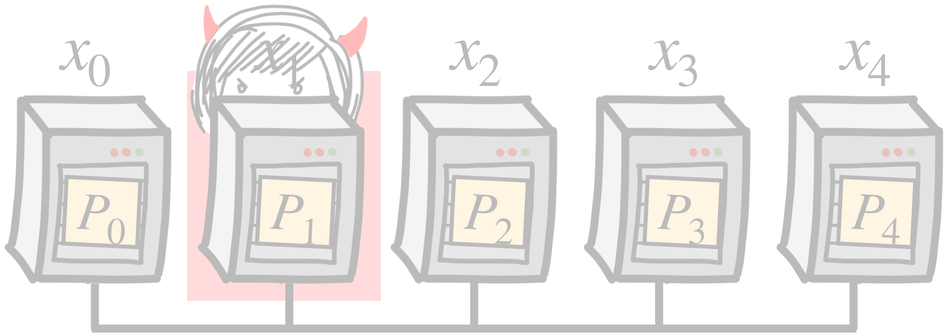


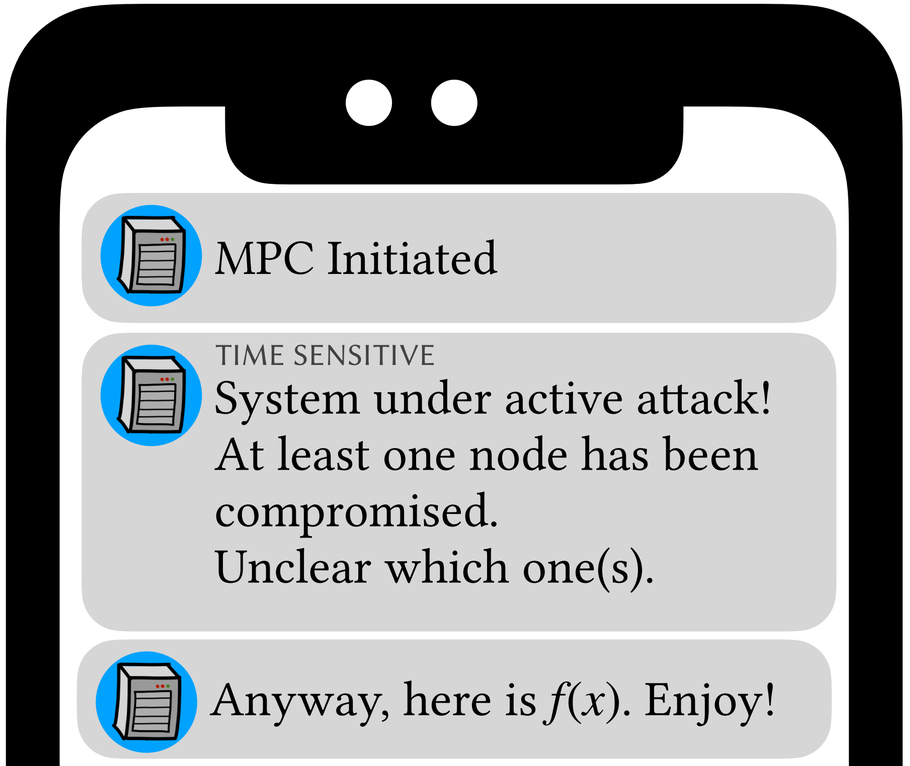
Guaranteed Output *vs.* Identifiable Abort



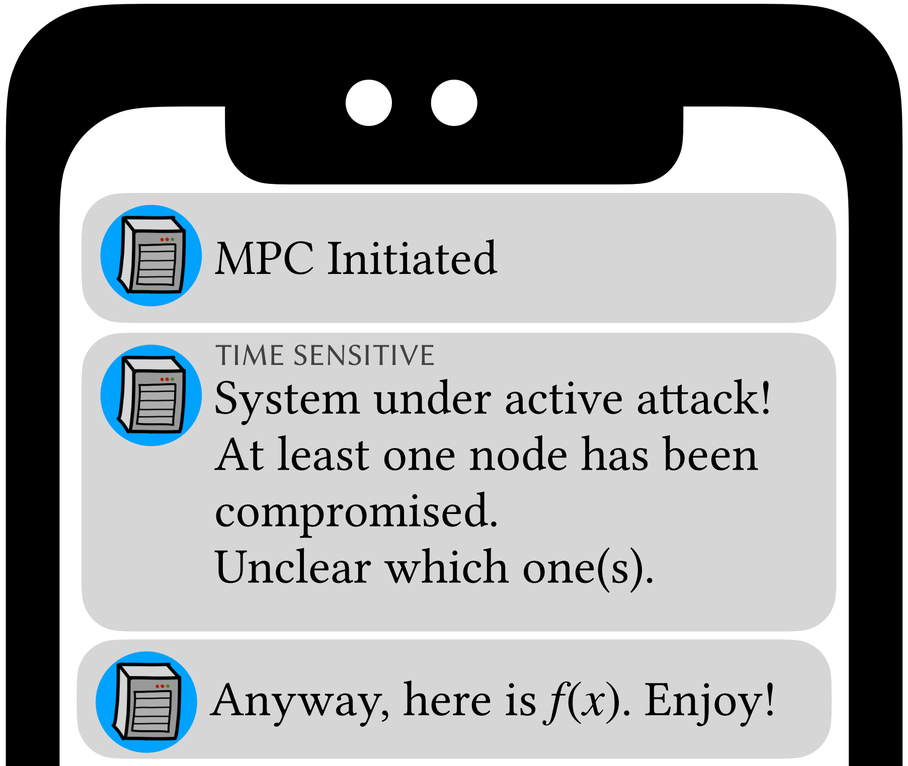


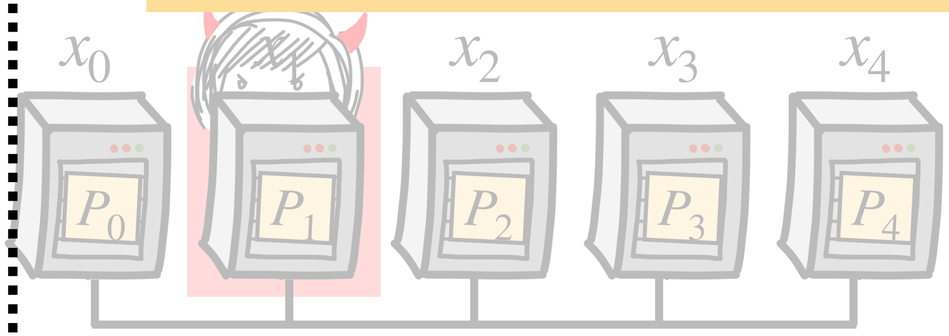
Guaranteed Output *vs.* Identifiable Abort



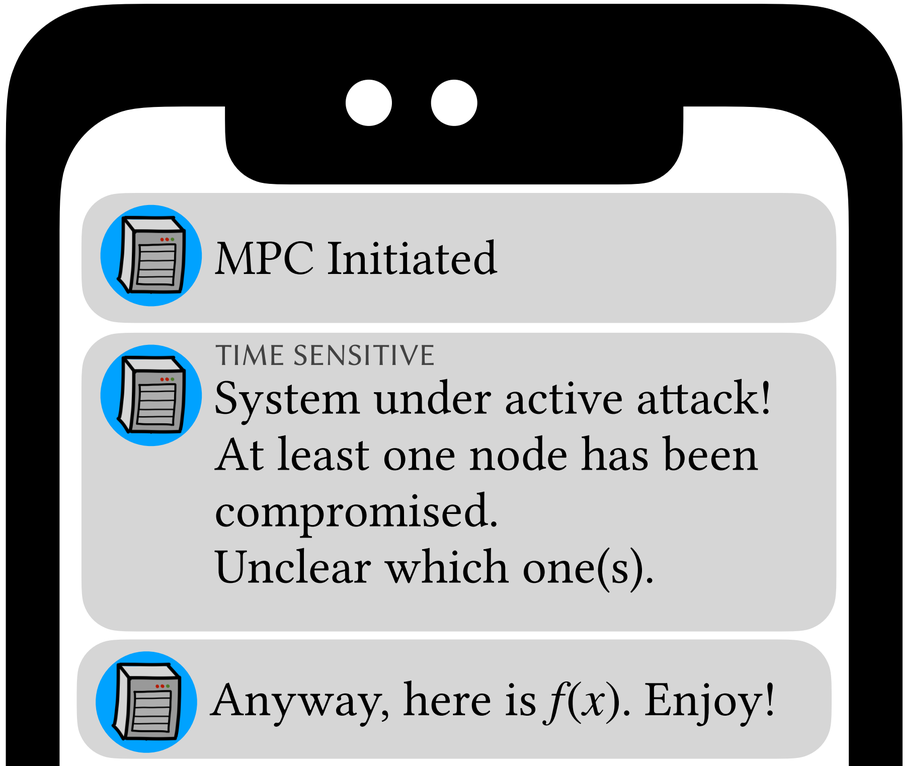


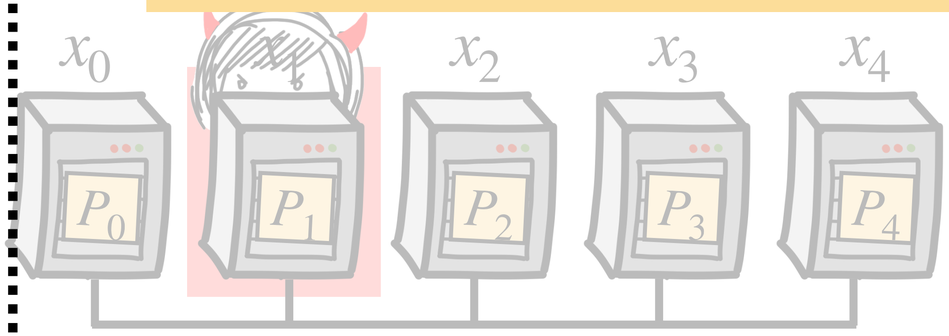
Guaranteed Output *vs.* Identifiable Abort

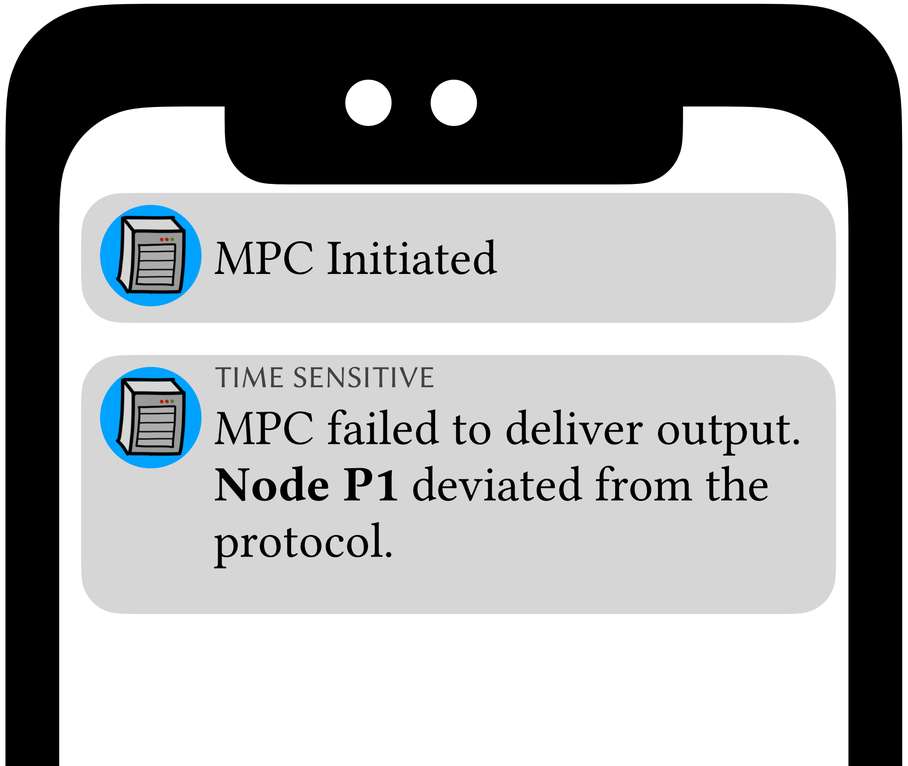




Guaranteed Output *vs.* Identifiable Abort







Identifiability in the Context of Re-staking

- Re-staking TLDR:
  - Operators buy into the protocol (service/AVS) with “re-staked” assets
  - In case of malicious behaviour, this stake can be “slashed”
  - Economic security: protocol deviations are disincentivized

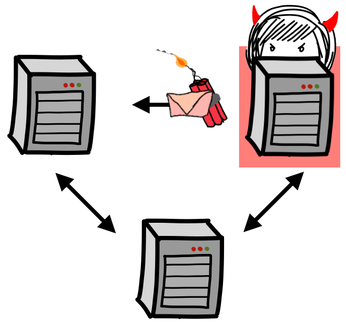
- Identifiable Abort is a natural fit for this setting
  - Cheating parties can be identified and slashed
  - DoS resistant MPC via economic incentives

- **Hope**: complexity of IA closer to Sec w. Abort than Guaranteed Output Delivery

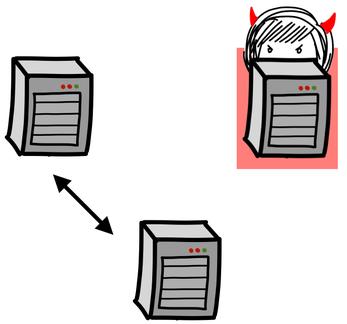
Identification Mechanisms

- Cheater *could* be found through out of band methods.
- We want **certifiable** protocol mechanism to identify who crashed the protocol
  ⇒ each party either gets output, or identity of cheating party + cert. of cheat

  Note: no consensus on identity
- Two ways to crash protocol:



1. **Malformed protocol message**



2. **No message at all**

Anatomy of MPC-IA

Baseline security-with-abort protocol

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Anatomy of MPC-IA

ZK proofs, carefully open secrets

Mechanism to guarantee wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee each party sends *some* message every round

Anatomy of MPC-IA

ZK proofs, carefully open secrets

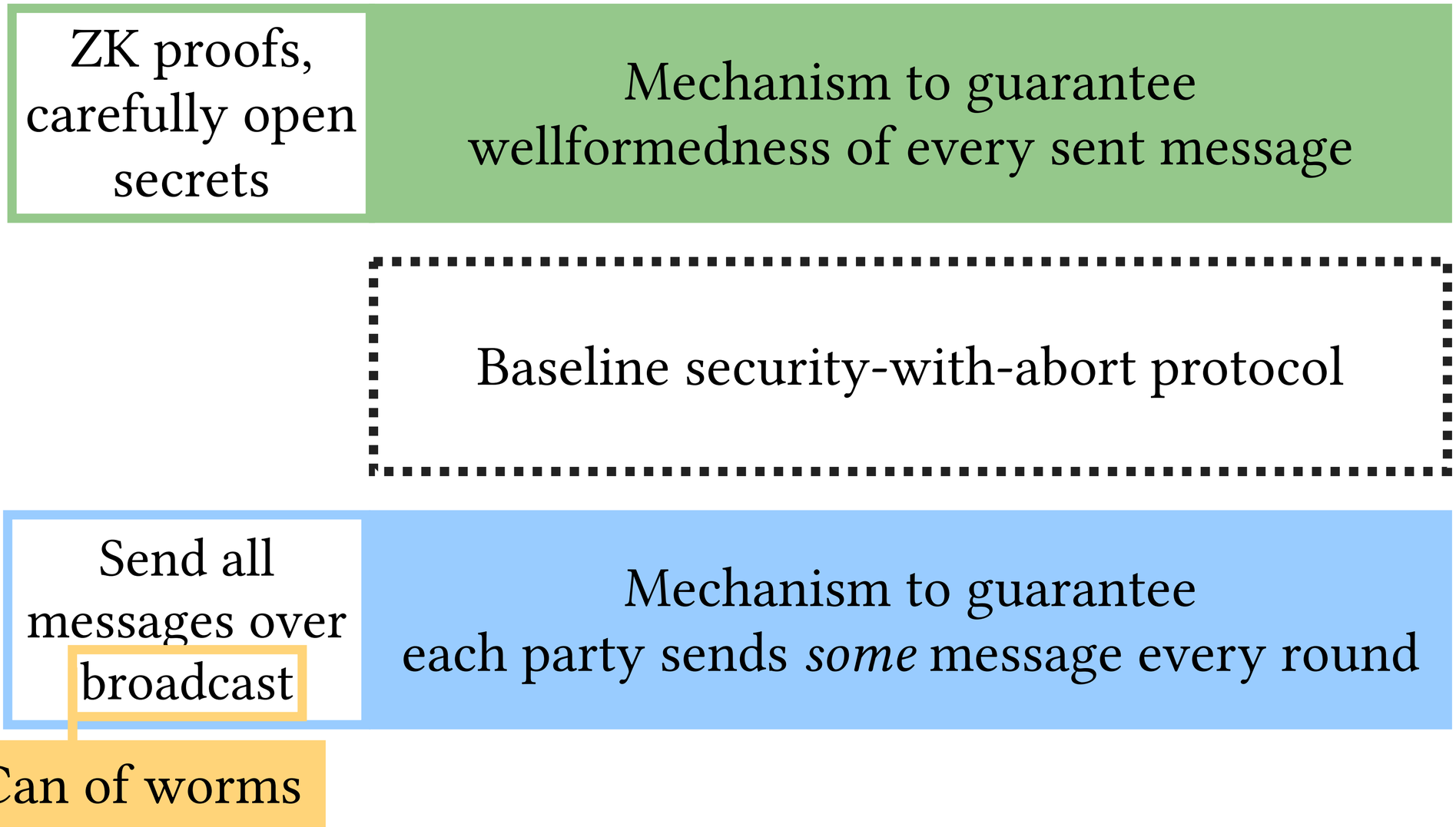
Mechanism to guarantee wellformedness of every sent message

Baseline security-with-abort protocol

Send all messages over broadcast

Mechanism to guarantee each party sends *some* message every round

Anatomy of MPC-IA



“Broadcast”?

- Engineering Anecdata:
  “Do I really need to implement broadcast?”
  “*yes*”
  “Is it just for some theoretical proof nonsense?”
  “*no, it’s to catch parties that don’t send messages for example*”
  “That seems unnecessary, I can just <insert heuristic>”

- In some settings: coordinator routes all messages
  ⇒ implicit single point of failure

- Other settings: use external broadcast channel like a blockchain
  ⇒ expensive, slow, introduces external dependencies

Broadcast Protocols

- [Cohen Lindell 14] MPC-IA implies broadcast: compute $\mathcal{F}_{\mathsf{PKI}}$ with IA

- PKI already available (+synchrony), broadcast is *feasible* [Dolev Strong 83]
  …but **round complexity is an issue**: $O(t)$ deterministic, or expected $O(1)$ randomized with large constants
  [Katz Koo 06][Abraham Devadas Dolev Nayak Ren 19]

- This is straightforward in the security with abort setting, via simple echo broadcast [Goldwasser Lindell 02]

- Can we construct a simple instantiation of BC as suitable for IA?
  Goal: an MPC-IA protocols that are easy to deploy over p2p channels

BC-IA Properties

- **Consistency**: All honest parties that output a valid (dealer signed) message will be in agreement

- If the sender is corrupt, an honest party alternatively obtains a certificate:
  - (An attempt to) violate consistency, yields a **certificate of cheating** Ω
  - If the sender sends nothing, yields a **certificate of non-responsiveness** ω

- Ω vs. ω: Definite misbehaviour vs. potential network fault—different penalties

- **Defamation-freeness**: Honest party can't be framed with Ω or ω

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

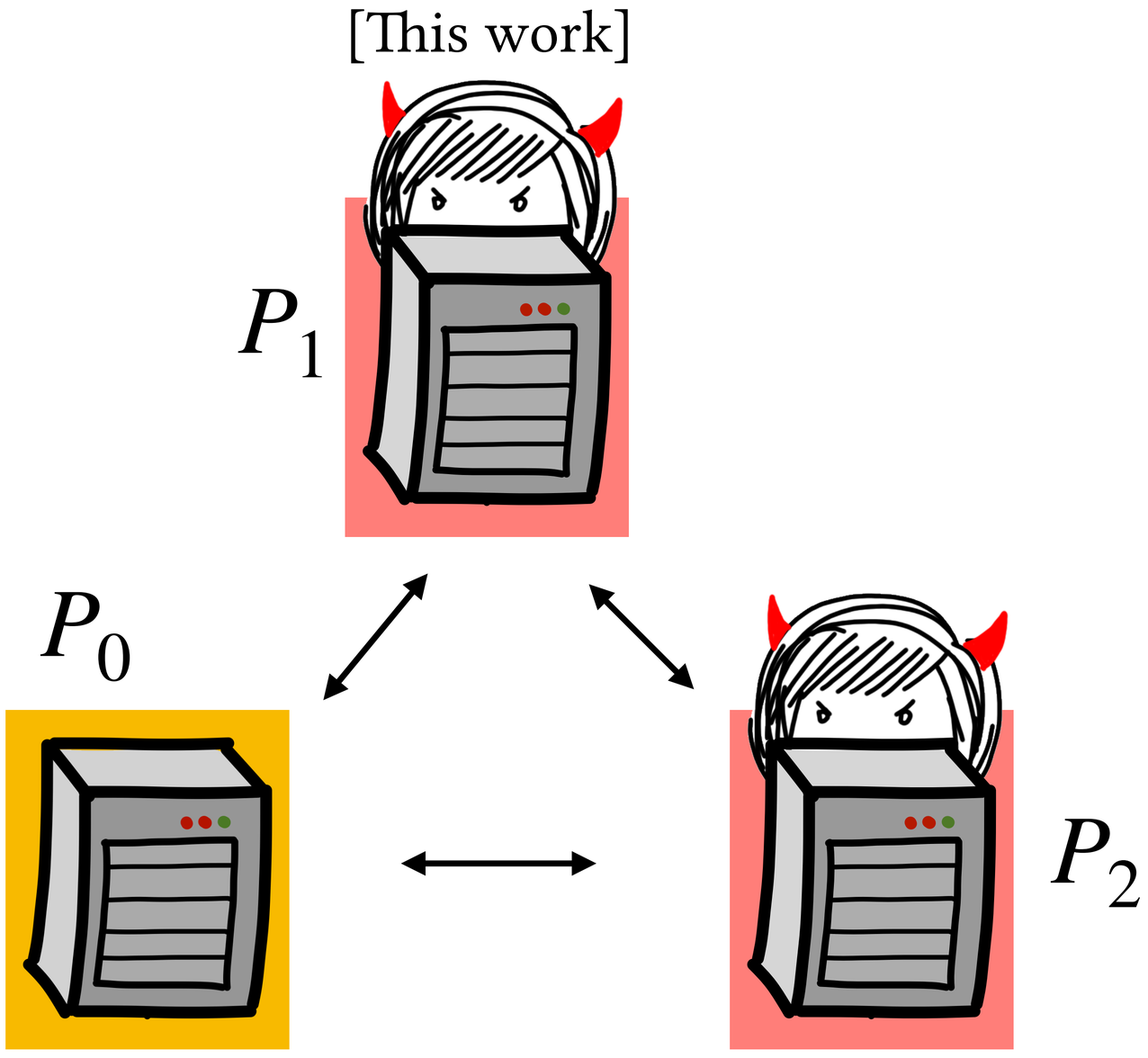
Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

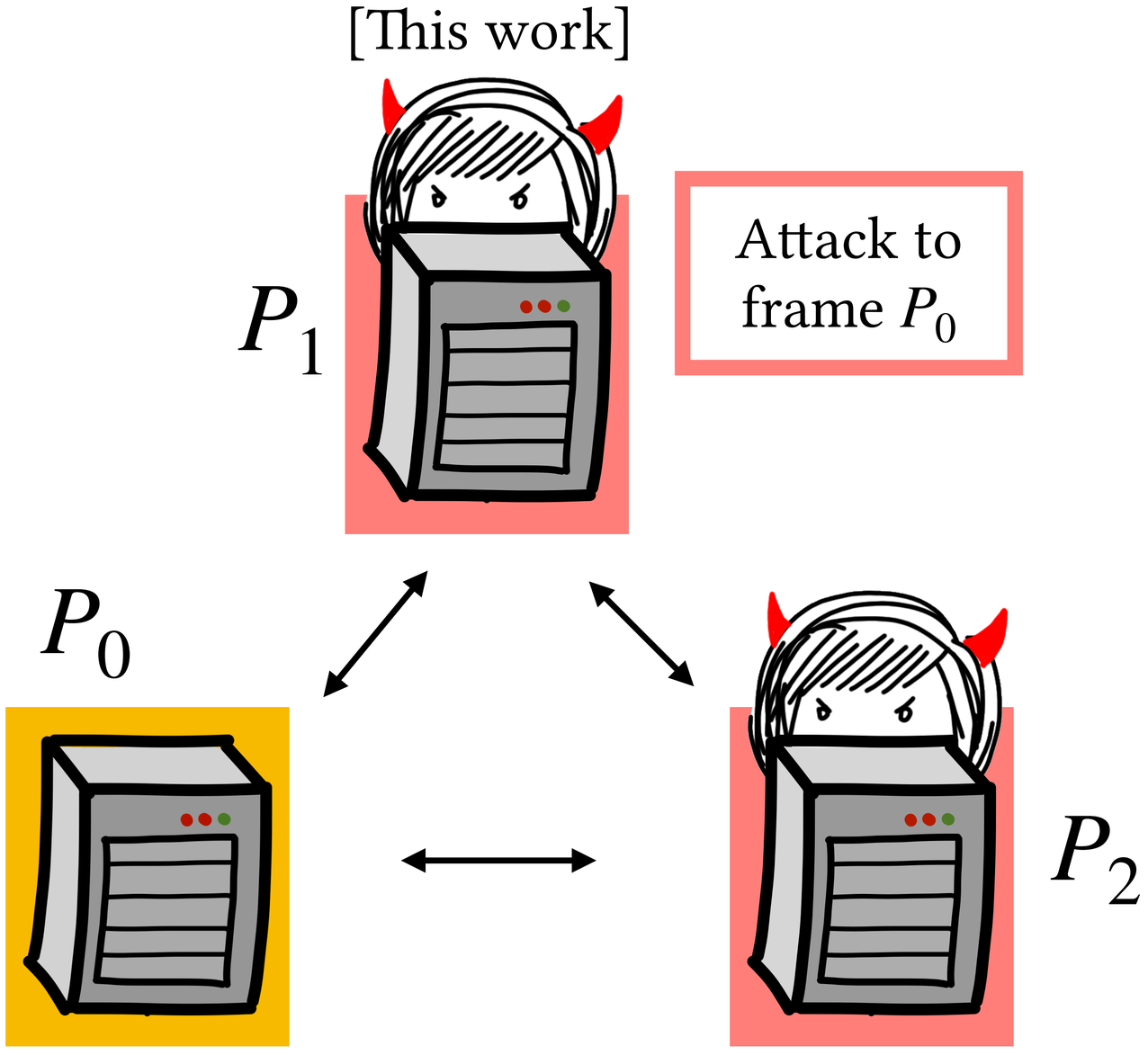
This work: define “Broadcast-IA”

- Impossible w. dishonest majority
- 2-round honest-majority protocol

Broadcast-IA is Impossible with Dishonest Majority

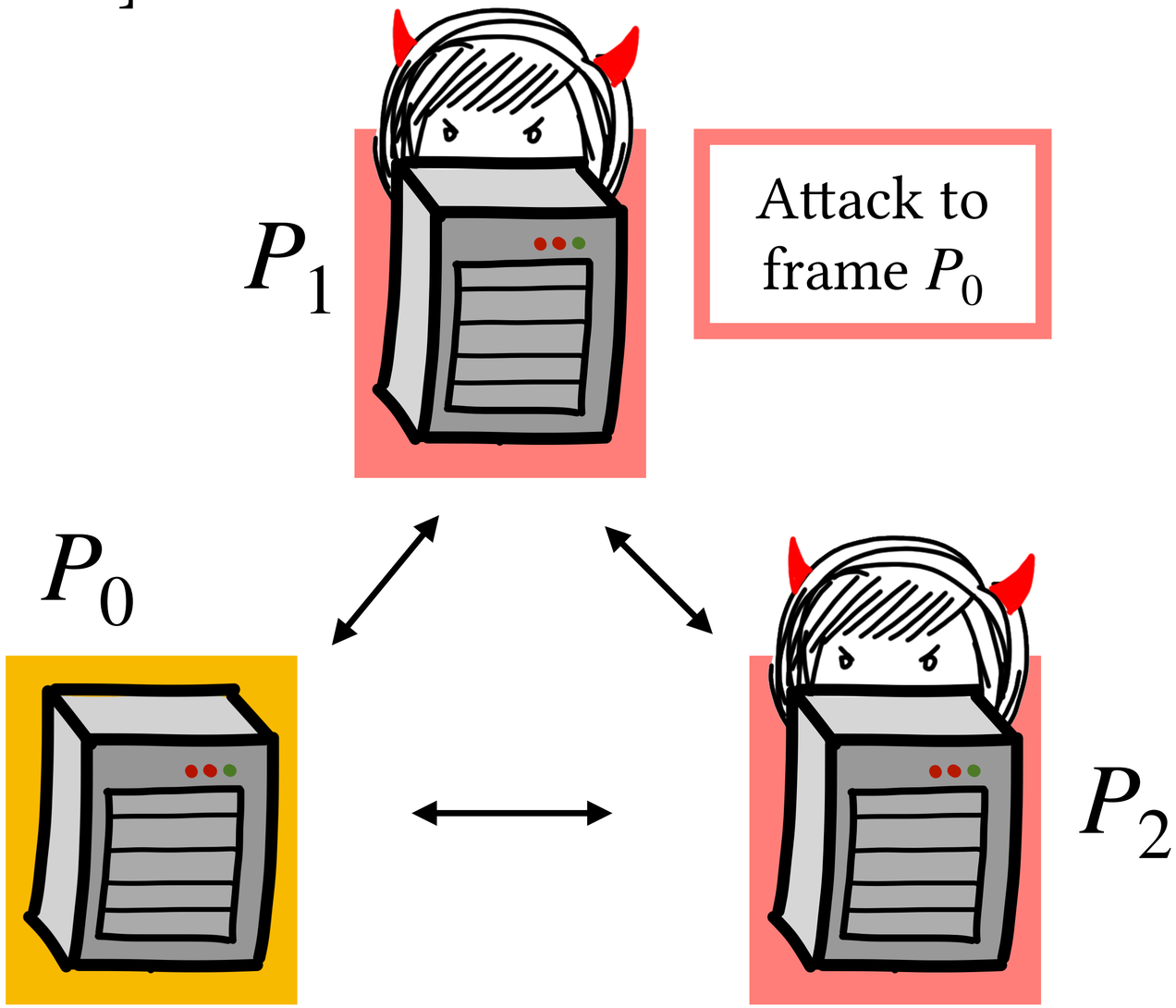


Broadcast-IA is Impossible with Dishonest Majority



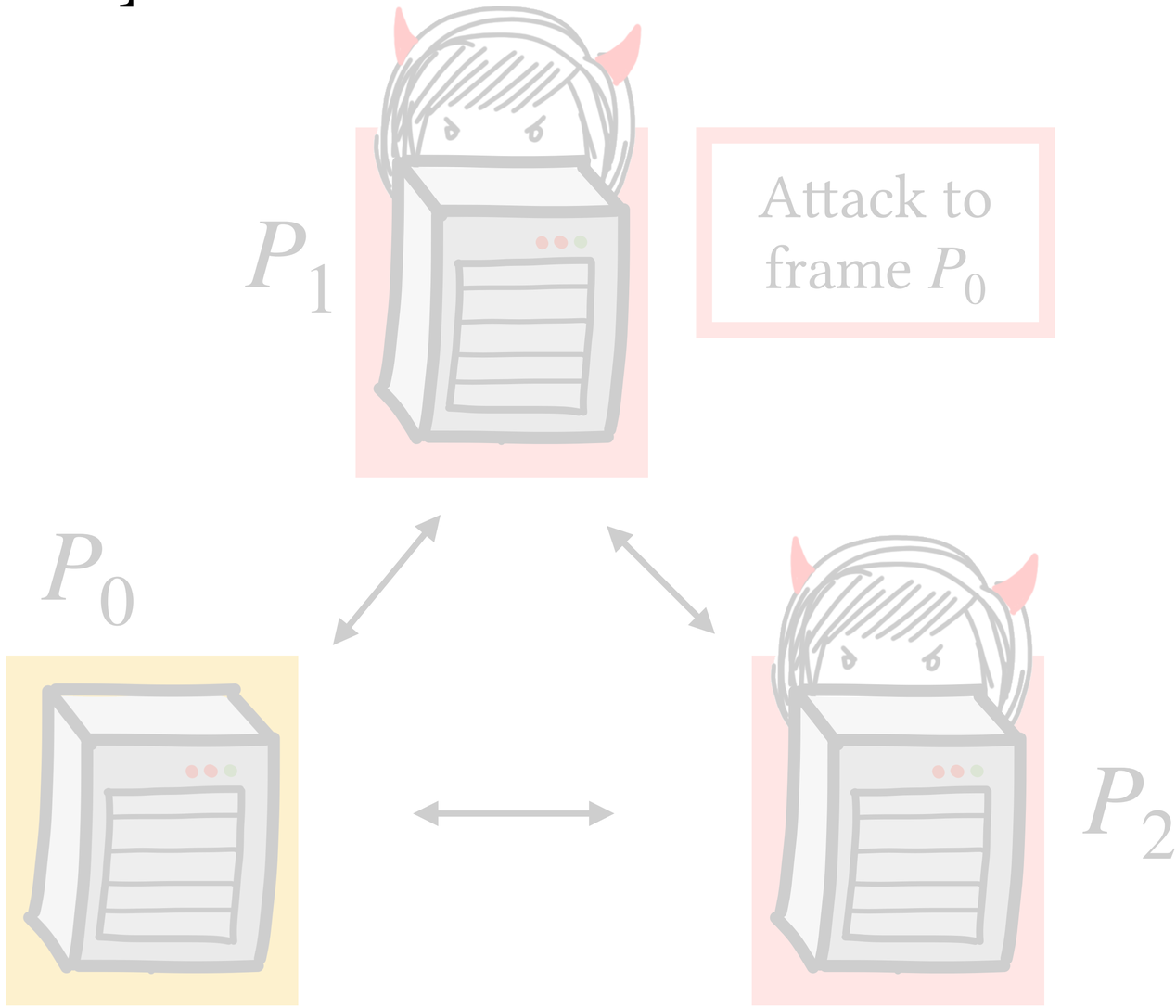
Broadcast-IA is Impossible with Dishonest Majority

[This work]



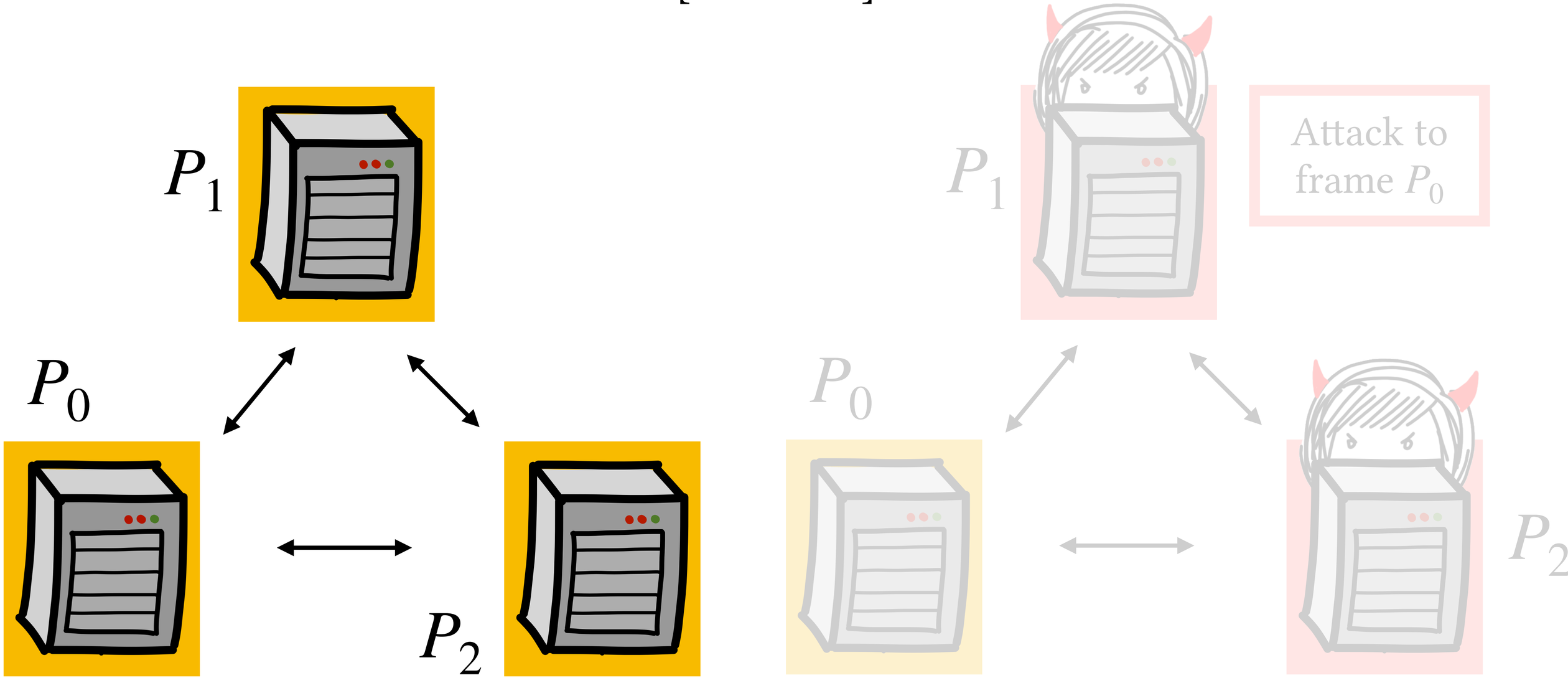
Broadcast-IA is Impossible with Dishonest Majority

[This work]



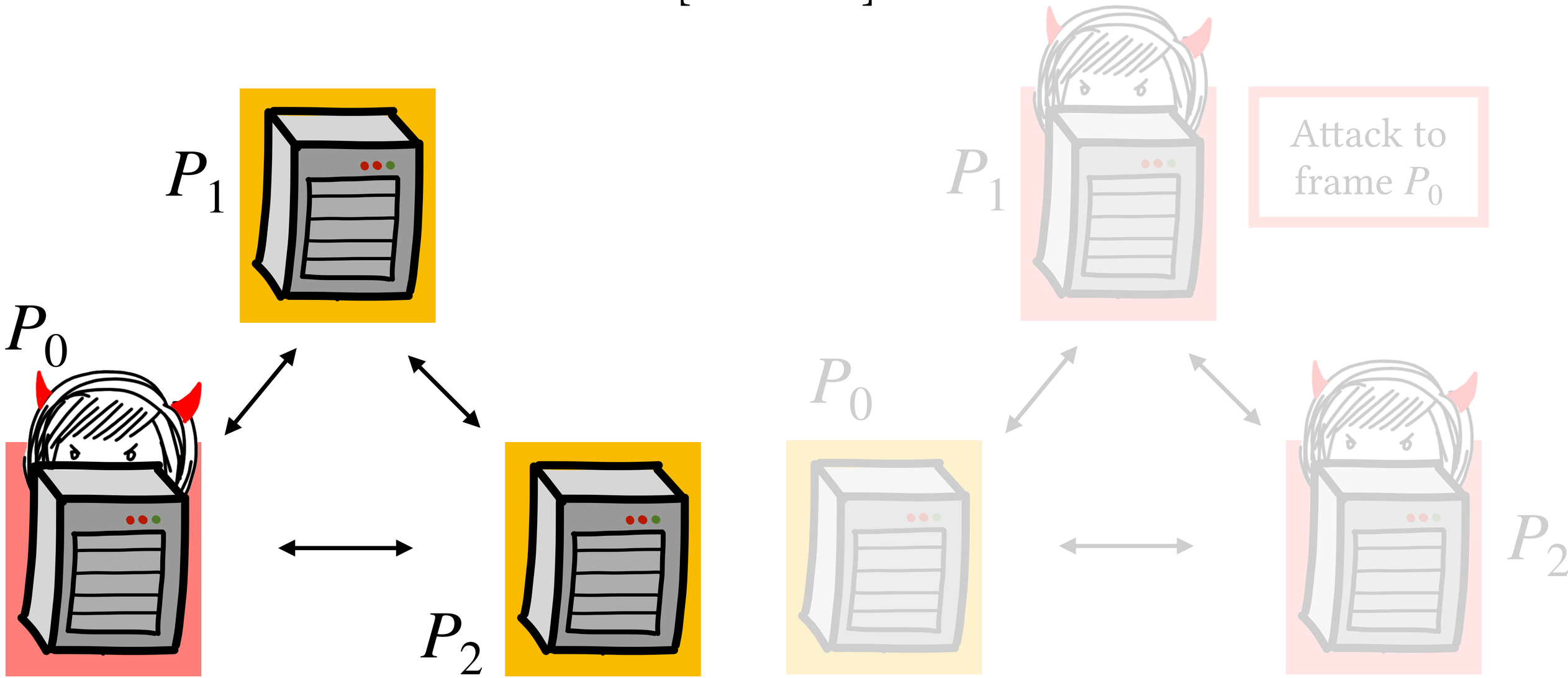
Broadcast-IA is Impossible with Dishonest Majority

[This work]



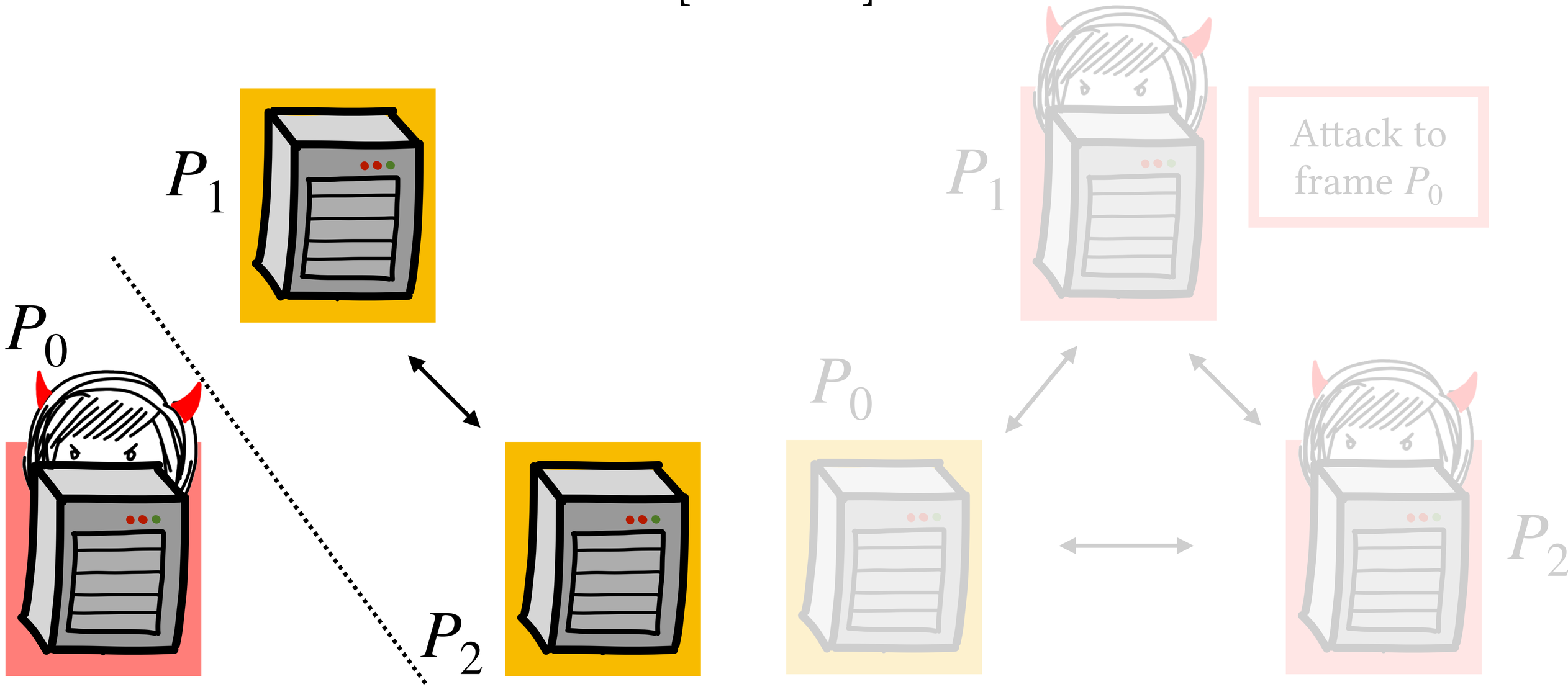
Broadcast-IA is Impossible with Dishonest Majority

[This work]



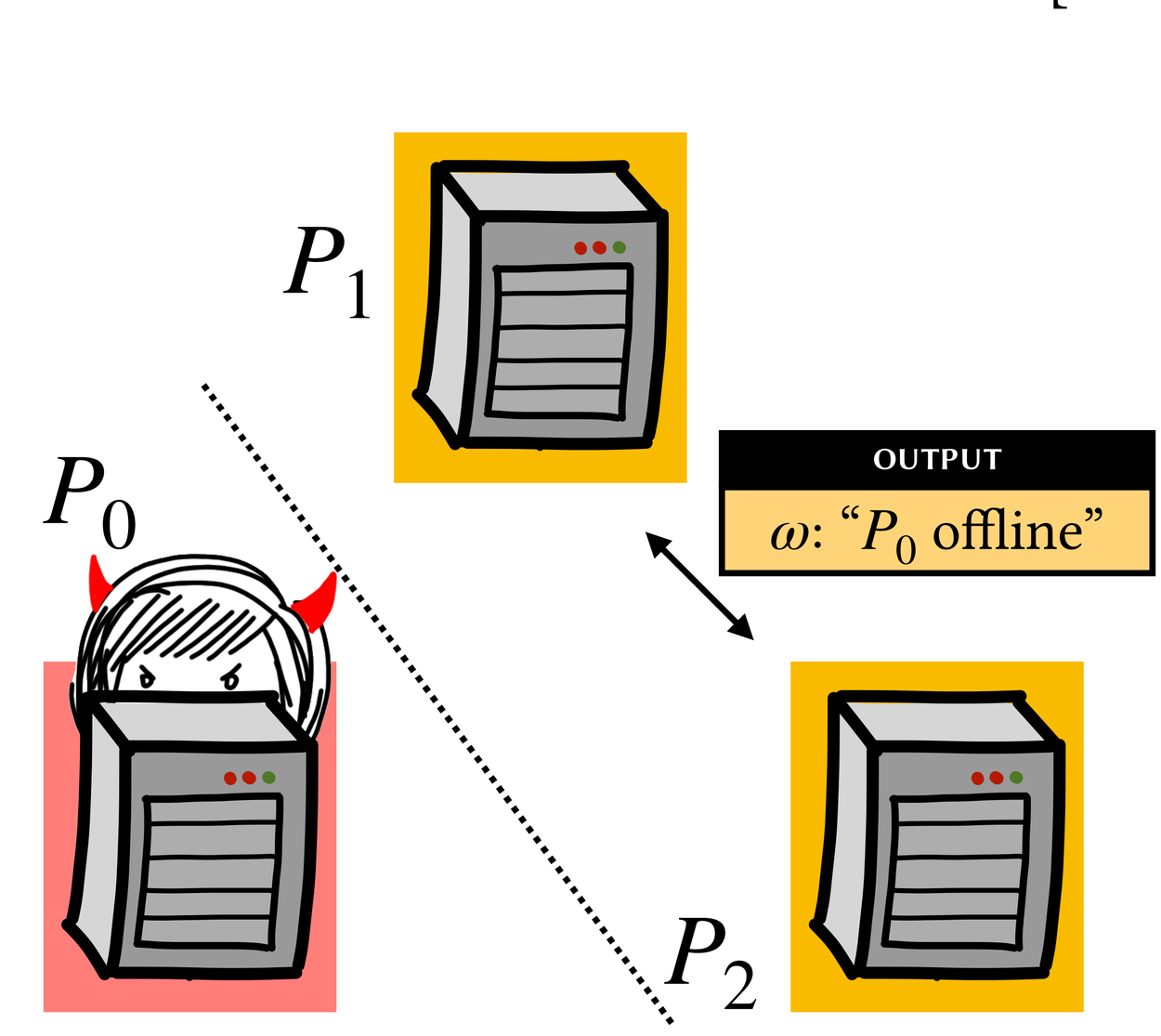
Broadcast-IA is Impossible with Dishonest Majority

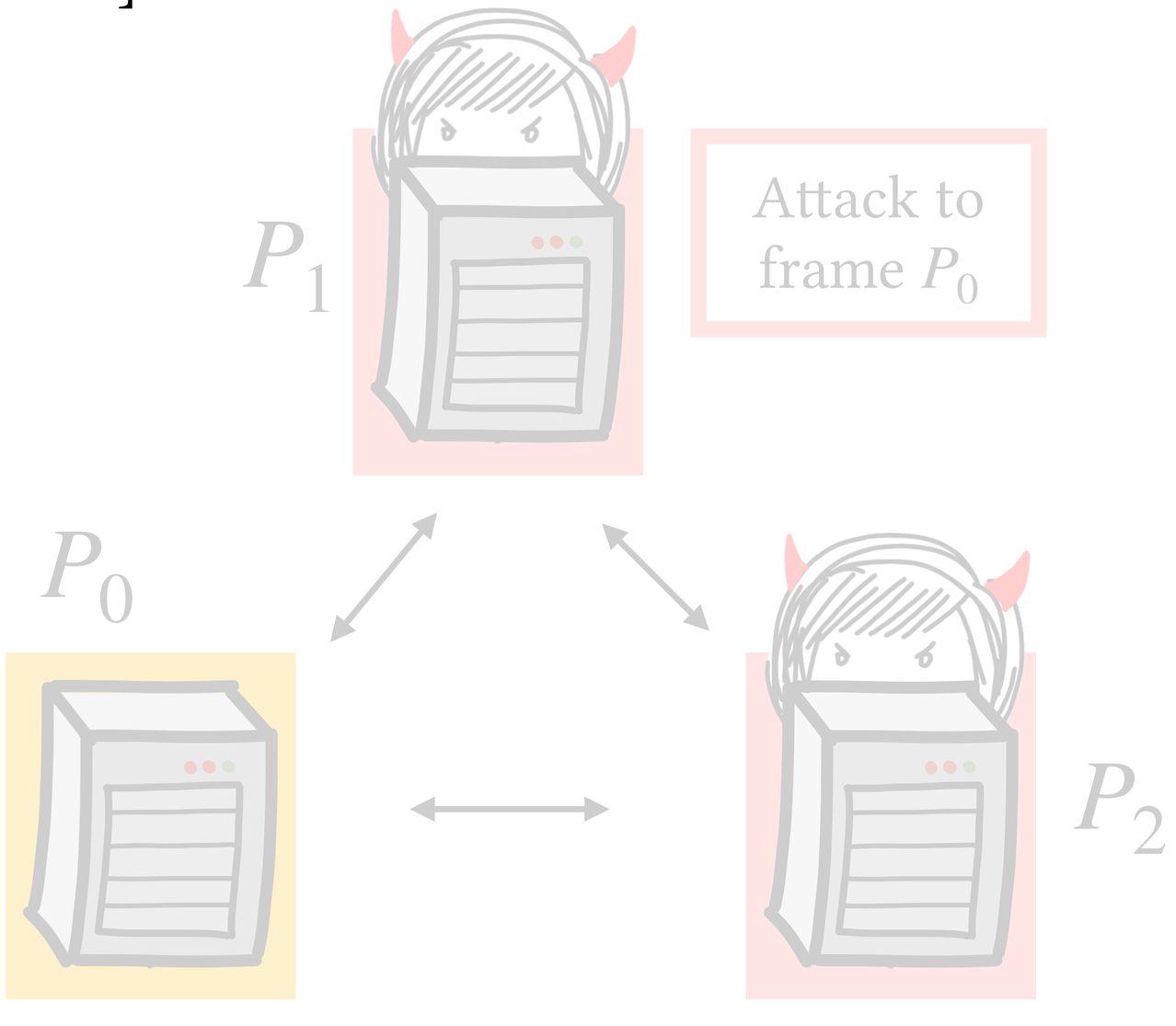
[This work]



Broadcast-IA is Impossible with Dishonest Majority

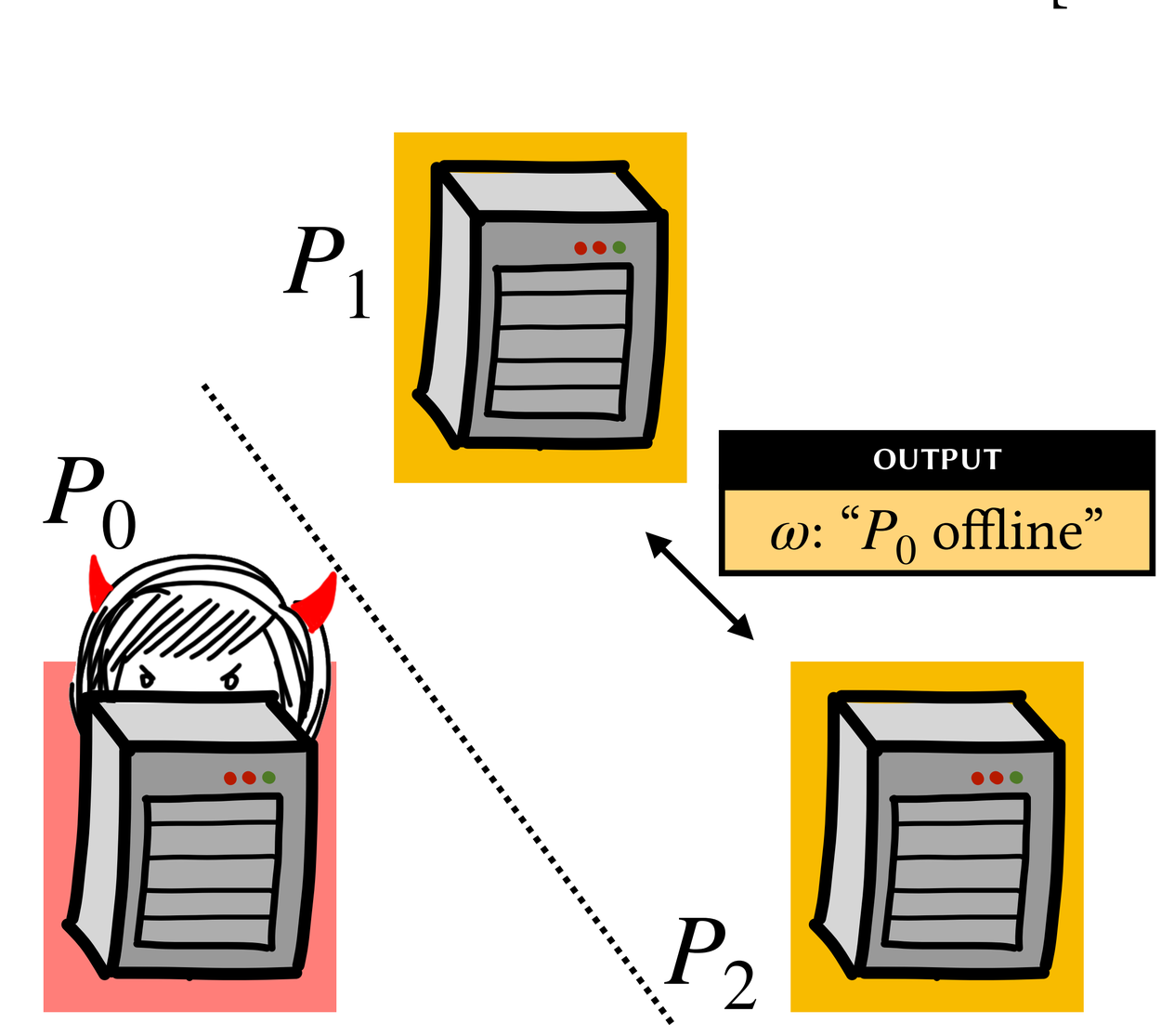
[This work]

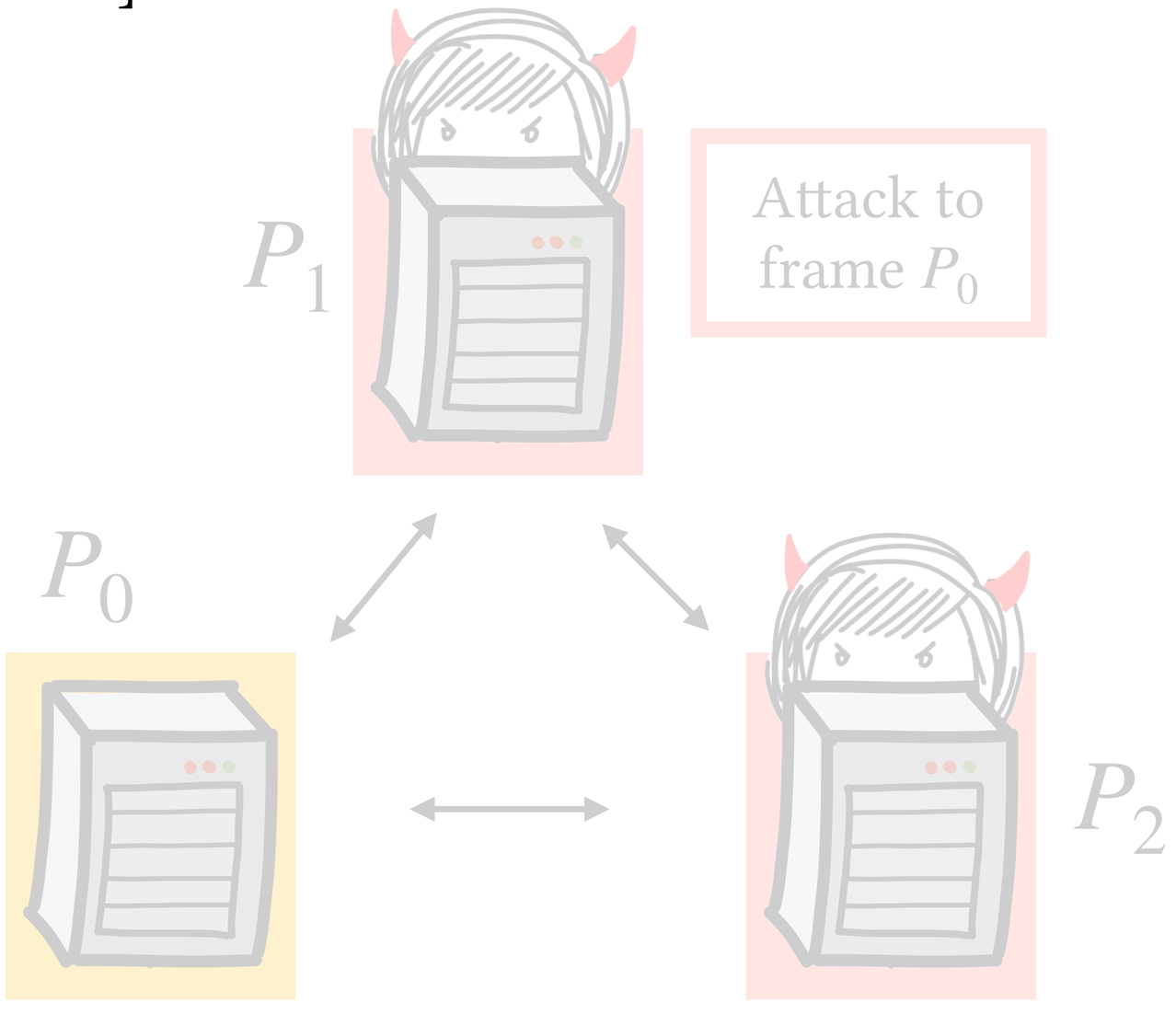




Broadcast-IA is Impossible with Dishonest Majority

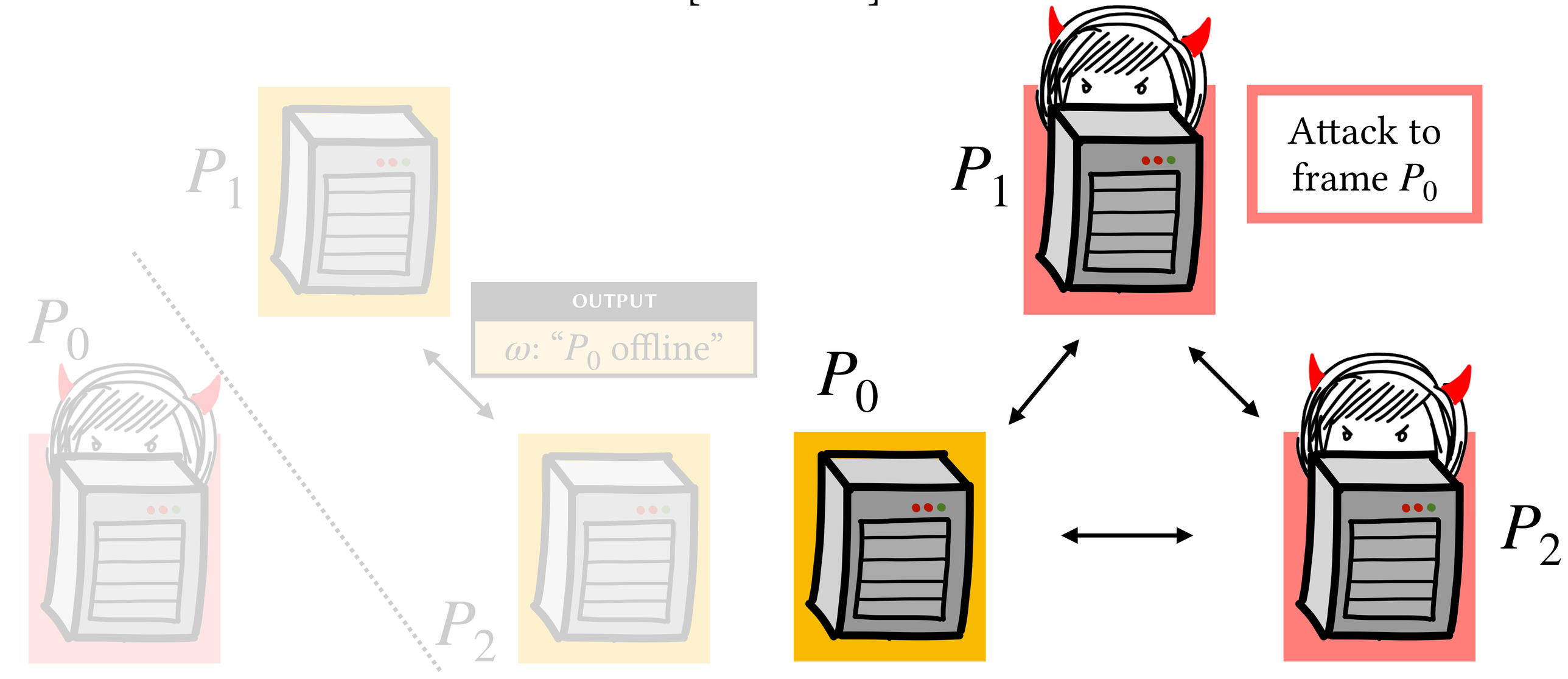
[This work]





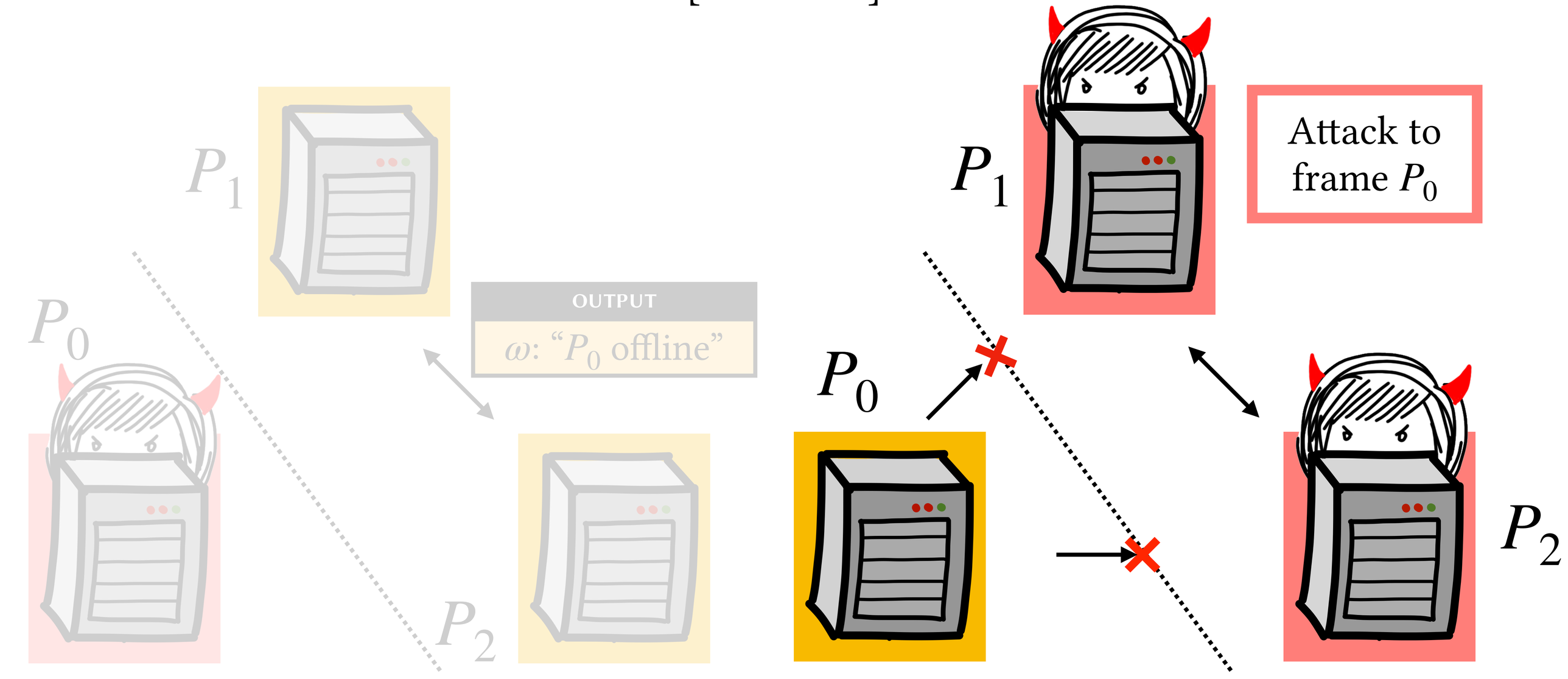
Broadcast-IA is Impossible with Dishonest Majority

[This work]



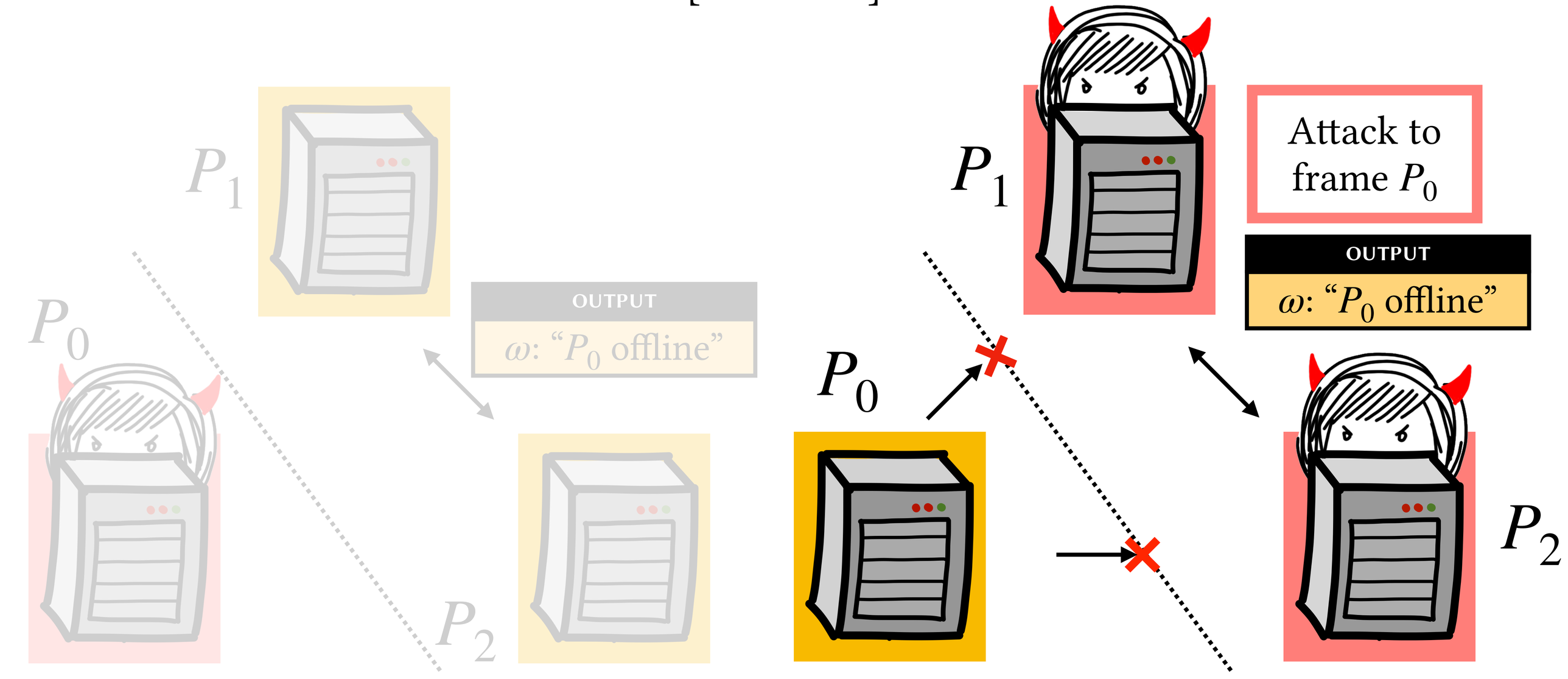
Broadcast-IA is Impossible with Dishonest Majority

[This work]



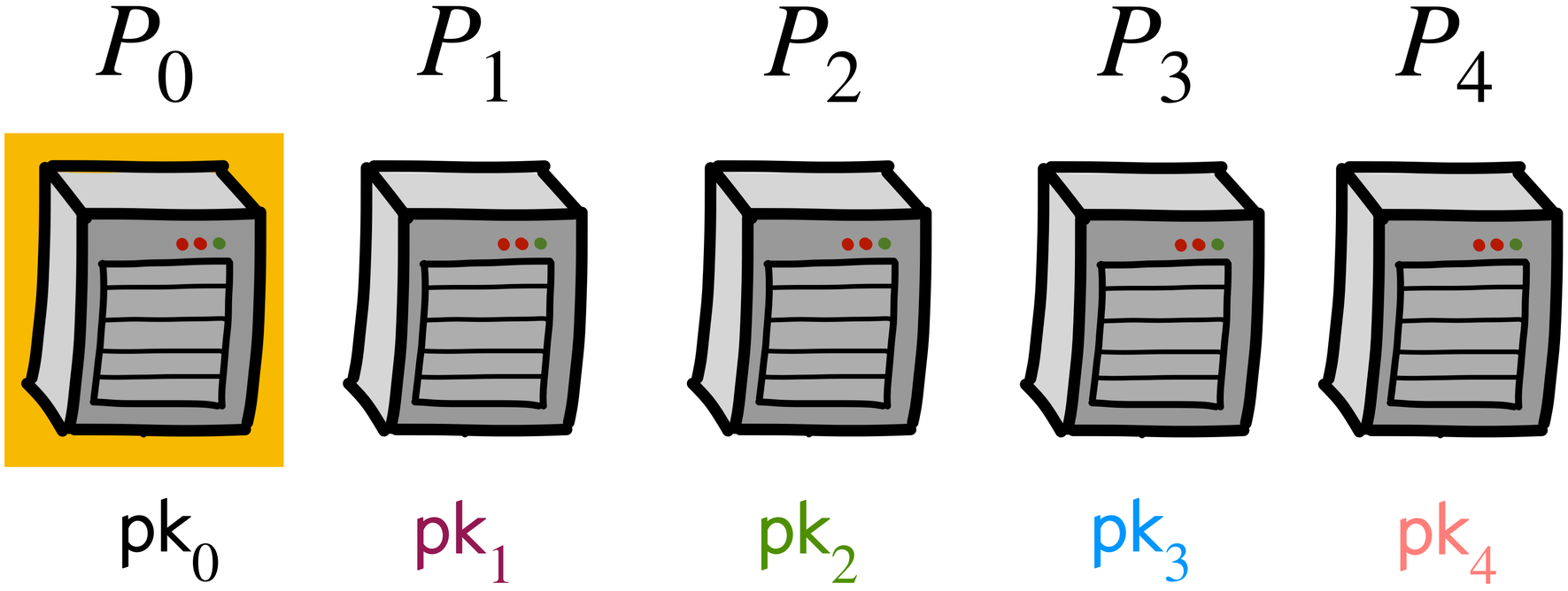
Broadcast-IA is Impossible with Dishonest Majority

[This work]



Broadcast-IA with Honest Majority

[This work]



P_0 wishes to broadcast m

Broadcast-IA with Honest Majority

[This work]

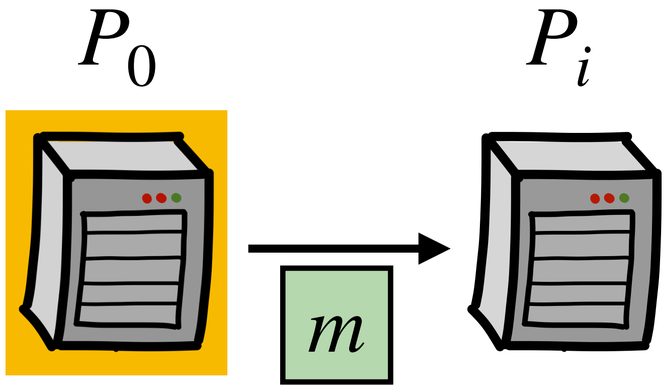
Round 1 | Round 2 | Output

Broadcast-IA with Honest Majority

[This work]

Round 1

Sign m,
Send to all



Round 2

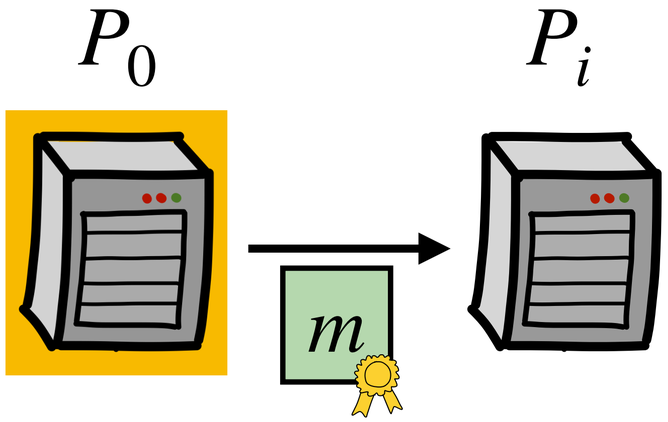
Output

Broadcast-IA with Honest Majority

[This work]

Round 1

Sign m,
Send to all

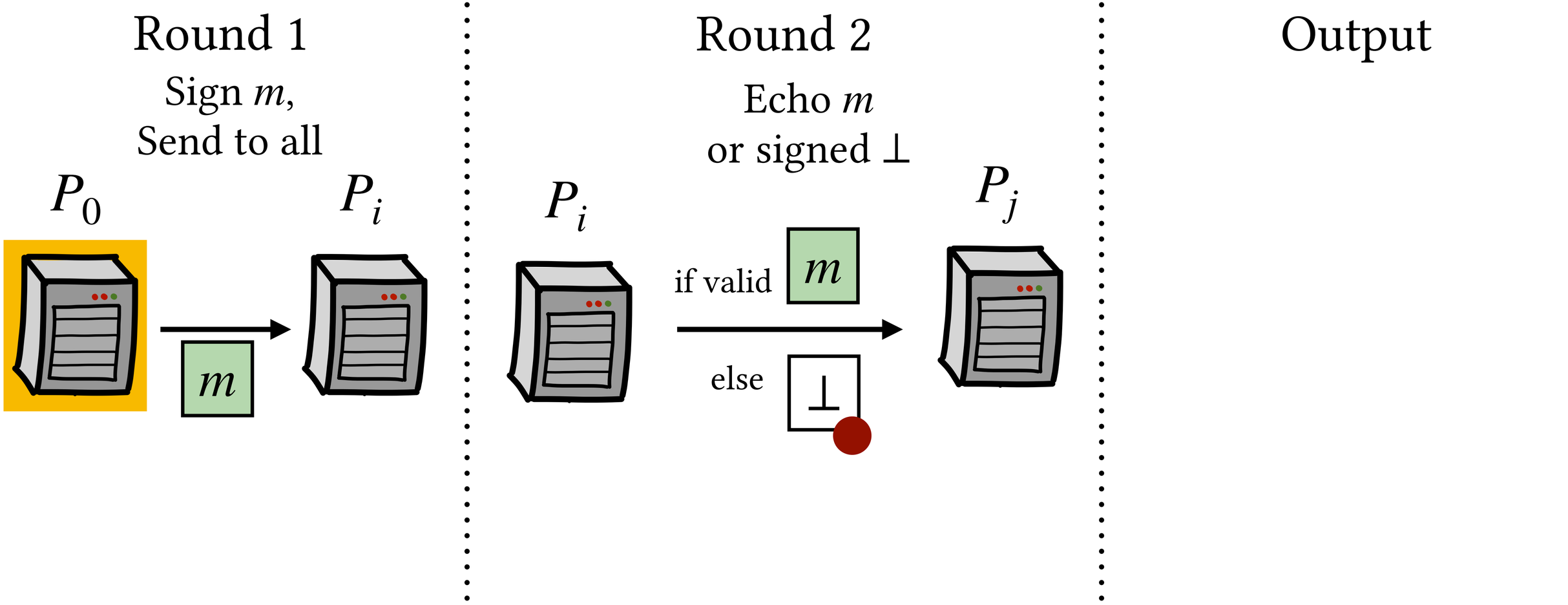


Round 2

Output

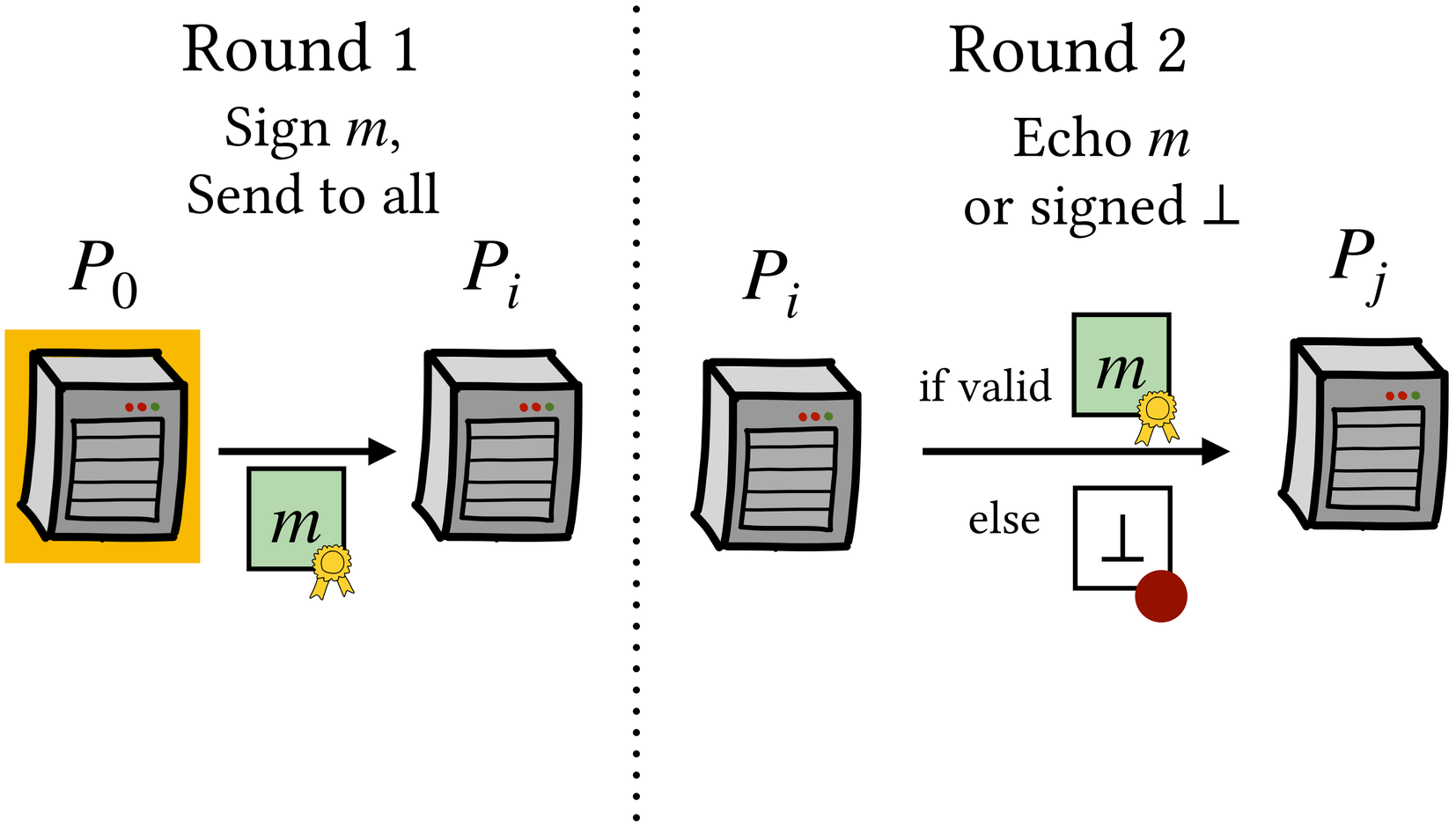
Broadcast-IA with Honest Majority

[This work]



Broadcast-IA with Honest Majority

[This work]

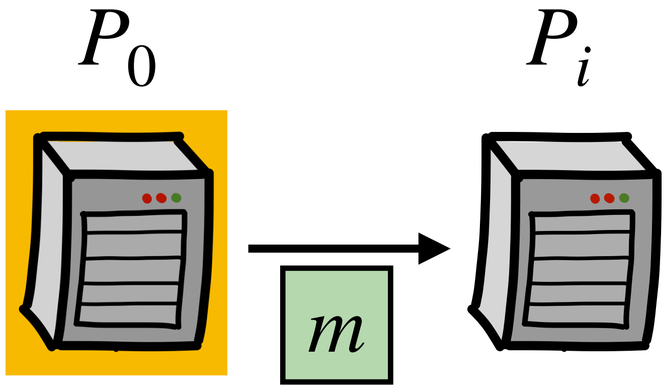


Broadcast-IA with Honest Majority

[This work]

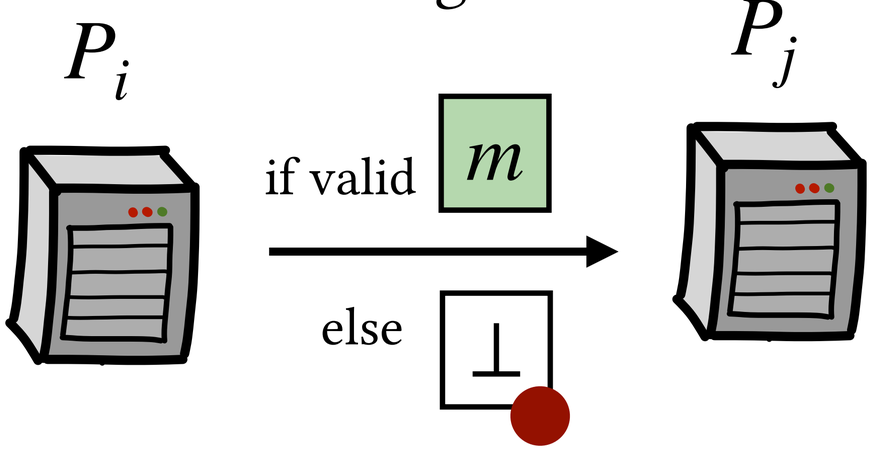
Round 1

Sign m,
Send to all



Round 2

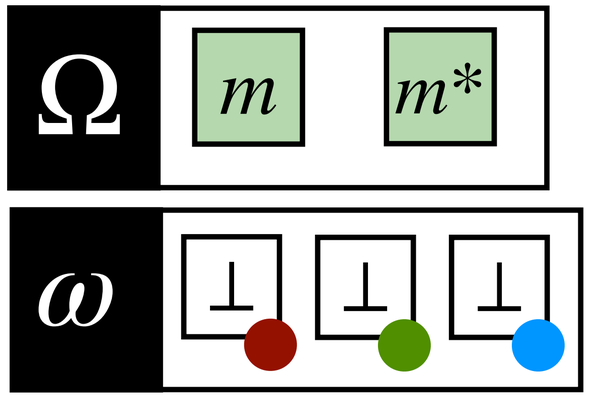
Echo m
or signed $\perp$



Output

Each P_i  :

1. Check for potential certificates of cheating:



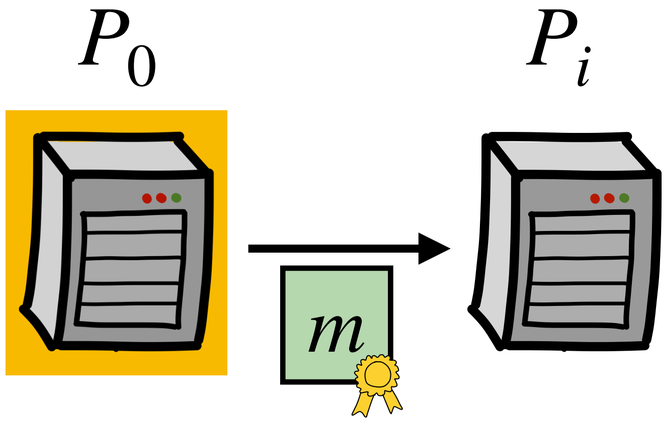
2. If no Ω, ω found, output 

Broadcast-IA with Honest Majority

[This work]

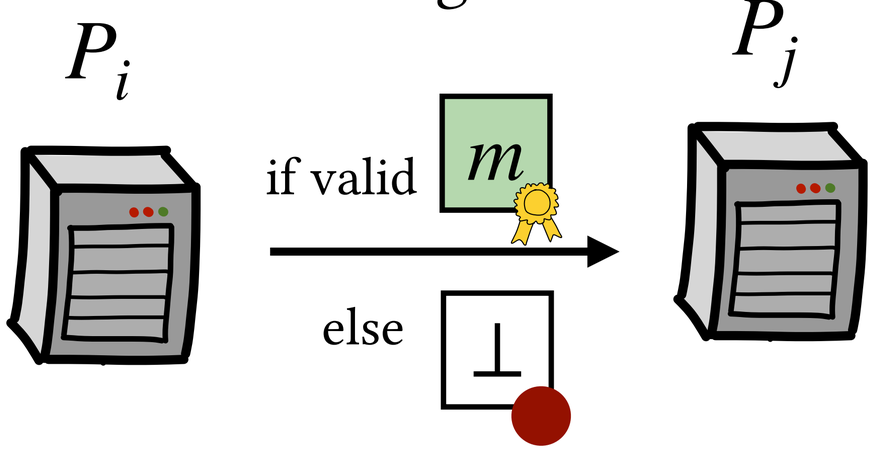
Round 1

Sign m,
Send to all



Round 2

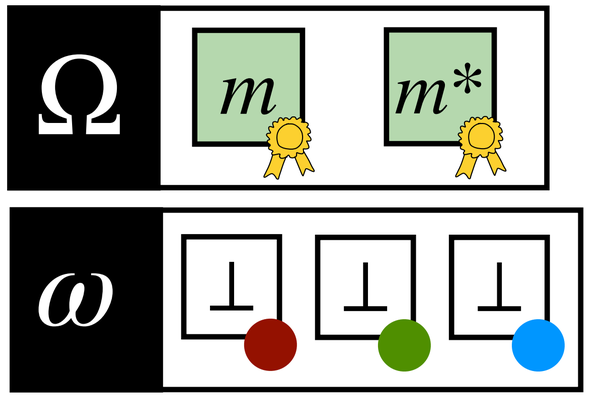
Echo m
or signed $\perp$



Output

Each P_i  :

1. Check for potential certificates of cheating:



2. If no Ω, ω found, output 

Broadcast-IA: Analysis

- **Honest** P_0: Complete, defamation-free
  - No Ω: Will not sending conflicting m, m^*
  - No ω: At most t corrupt parties will echo $\perp \Rightarrow$ not enough sigs

- **Corrupt** P_0: Consistent
  - If any honest parties receive $m, m^* \Rightarrow$ yields Ω
  - If m withheld from *all* honest parties $\Rightarrow$ yields ω
  - Send m to any honest party $\Rightarrow$ m committed as output

- **Notes on output** m:
  1. Accompanied by sig(m) from P_0: proves P_0 sent m to P_i
  2. P_i producing sig(m) DOES NOT prove that some P_j also output m

Synchrony

- Protocol assumes a well-defined network time-out (i.e. synchrony)

- Inherent: Identifiable Abort not well-defined in p2p asynchronous setting
  - Honest parties w. bad network indistinguishable from corrupt

- Important to reason about what happens when network goes bad:
  - Honest parties may be certified non-responsive (ω)

    ⇒ Very bad idea to take drastic action based on non-responsiveness alone
  - Liveness may be violated
  - Cheat (Ω) remains attributable to corrupt parties only

    ⇒ Higher level protocols can still maintain safety/privacy of secrets

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

[This work]
2-round honest majority BC-IA

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Informal Theorem
If Π^{BC} is a protocol that achieves IA* using Ideal Broadcast, then $\Pi^{\mathsf{BC-IA}}_{\mathsf{compiled}}$ achieves IA using BC-IA

[This work]
2-round honest majority BC-IA

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Which Π^{BC} to plug in?

(Increasingly) well studied in the dishonest majority ($t < n$) setting
[Ishai Ostrovsky Zikas 14][Baum Orsini Scholl Soria-Vazquez 20][Cohen Doerner K shelat 24][Baum Melissaris Rachuri Scholl 24]

[This work]
2-round honest majority BC-IA

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Which Π^{BC} to plug in?

(Increasingly) well studied in the dishonest majority ($t < n$) setting
[Ishai Ostrovsky Zikas 14][Baum Orsini Scholl Soria-Vazquez 20][Cohen Doerner Kshelat 24][Baum Melissaris Rachuri Scholl 24]

[This work]
2-round honest majority BC-IA

inherent

Anatomy of MPC-IA

Mechanism to guarantee
wellformedness of every sent message

Baseline security-with-abort protocol

Mechanism to guarantee
each party sends *some* message every round

Which Π^{BC} to plug in?

(Increasingly) well studied in the dishonest majority ($t < n$) setting
[Ishai Ostrovsky Zikas 14][Baum Orsini Scholl Soria-Vazquez 20][Cohen Doerner Kshelat 24][Baum Melissaris Rachuri Scholl 24]

Understudied in $t < n/2$ setting

[This work]
2-round honest majority BC-IA

inherent

Sample Instantiation: Threshold ECDSA

This work: Instantiate ECDSA-IA

Mechanism to guarantee
wellformedness of every sent message

Light ZK proofs in $\mathbb{G}$
+ verifiable complaints

Baseline security-with-abort protocol

3-BC-round honest-majority
ECDSA signing à la [DKLs23]

Mechanism to guarantee
each party sends *some* message every round

[This work]
2-round honest majority BC-IA

inherent

ECDSA-IA: Efficiency

- Envisioned mode of operation:
  - Run [DKLs23] (sec w. abort) by default
  - Fall back to this protocol if too many aborts observed

- Worst case execution path most relevant to measuring efficiency
  - $(t, n) = (10,21)$: ~500ms compute time on standard hardware

  Relative to dishonest majority

  noticeably slower than (s.w.a.) OT-based ECDSA [DKLs23]

  order of magnitude faster than Paillier-based ECDSA-IA [CGGMP20]

- Actual worst-case performance depends on network conditions
  - Up to $6 \times$ Network Timeout

In Conclusion

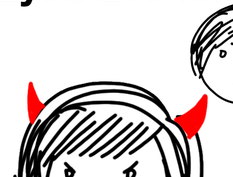
- Sometimes—e.g. re-staking setting—Identifiable Abort can offer more meaningful security than Guaranteed Output
  - IA requires some form of broadcast (tricky to instantiate)

- We define Broadcast-IA to certify cheaters: silent parties and protocol deviations
  - Prove *impossible* w. dishonest majority
  - 2-round $t < n/2$ construction over p2p channels (synchrony + PKI)

- Use this tool to instantiate Threshold ECDSA-IA over p2p channels
  - Simpler, more efficient than Guaranteed Output
  - Ongoing research: General Secure Function Evaluation with IA

Thanks!

Thanks **Eysa Lee** for



eprint coming soon, **(pre)preprint on ykondi.net**