Practically Efficient Secure Small Party Computation over the Internet

Megha Byali

Under the guidance of ***Dr. Arpita Patra***

Indian Institute of Science, Bangalore, India.

Publications Based on this Thesis

- *Fast Actively-Secure Five Party Computation with Security Beyond Abort*. **Megha Byali**, Carmit Hazay, Arpita Patra, Swati Singla. ***ACM CCS 2019***.

- *Beyond Honest Majority: On the Efficiency of 4-Party Computation in High-Latency Networks*. **Megha Byali**, Arpita Patra, Divya Ravi, Swati Singla. *Under Submission*.

Other Publications in the Area

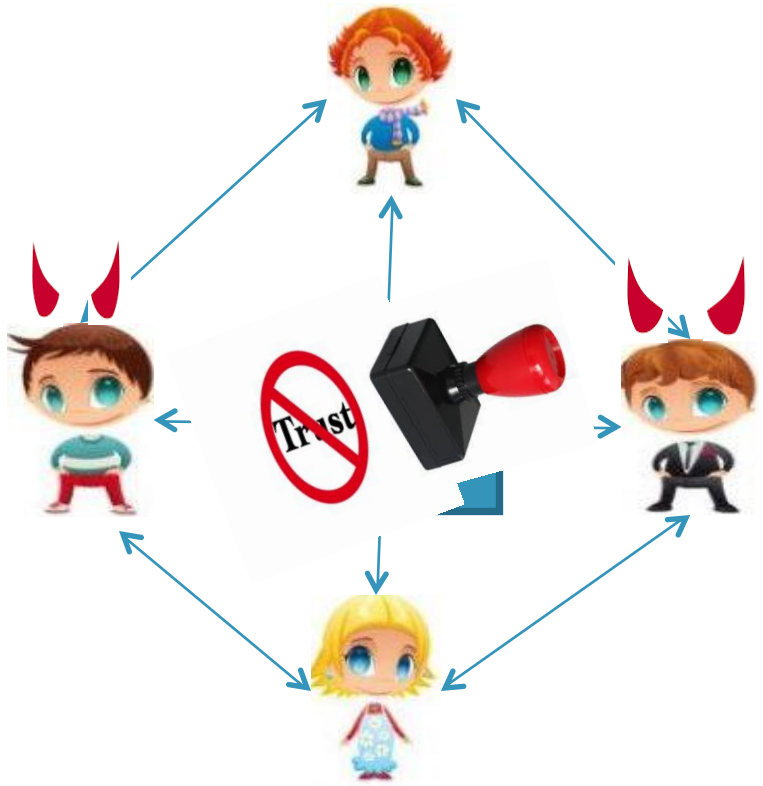
- *Fast Secure Computation for Small Population over the Internet*. **Megha Byali**, Arun Joseph, Arpita Patra, Divya Ravi. **ACM CCS 2018**.
- *Trusted B2B Market Platforms using Permissioned Blockchains and Game Theory*. **Megha Byali**, Pankaj Dayama, Shivika Narang, Yadatti Narahari and Vinayaka Pandit. **ICBC 2019**.
- *Speedo4: High-Speed Secure 4-Party Computation over the Internet.* **Megha Byali**, Nishat Koti, Arpita Patra, Divya Ravi, Swati Singla. *Under Submission.*
- *FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning*. **Megha Byali**, Harsh Chaudhari, Arpita Patra, Ajith Suresh. PETS 2020.
- *Efficient, Round-optimal, Composable Oblivious Transfer and Commitment Scheme with Adaptive Security*. **Megha Byali**, Arpita Patra, Divya Ravi, Pratik Sarkar.

Roadmap

- Secure MultiParty Computation (MPC)
- Adversarial Models
- Motivation
- Security Model and Security Notions
- Results
- Five Party Computation with Fairness
- Efficiency
- Future Scope

Secure Multi-Party Computation (MPC)

A set of *n* parties wish to compute a joint function $f(x_1, x_2, ..., x_n)$ on their inputs $(x_1, x_2, ..., x_n)$.



Goals:

- **Correctness:** Compute $f(x_1,x_2,..x_n)$.
- **Privacy:** Nothing more than function output should be revealed.

MPC: Real World emulation of TTP

Adversarial model

Based on adversarial behaviour:

- **Semi-honest :** Follows the steps of the protocol, but tries to glean extra information from the messages received.
- **Malicious :** Arbitrarily deviates from the protocol.

Based on number of corruptions (**t**) :

- **Honest Majority :** In the presence of ***n*** *parties*, at most ***t<n/2*** are corrupt.
- **Dishonest Majority :** In the presence of ***n*** *parties*, at most ***t<n*** are corrupt.

Our model :

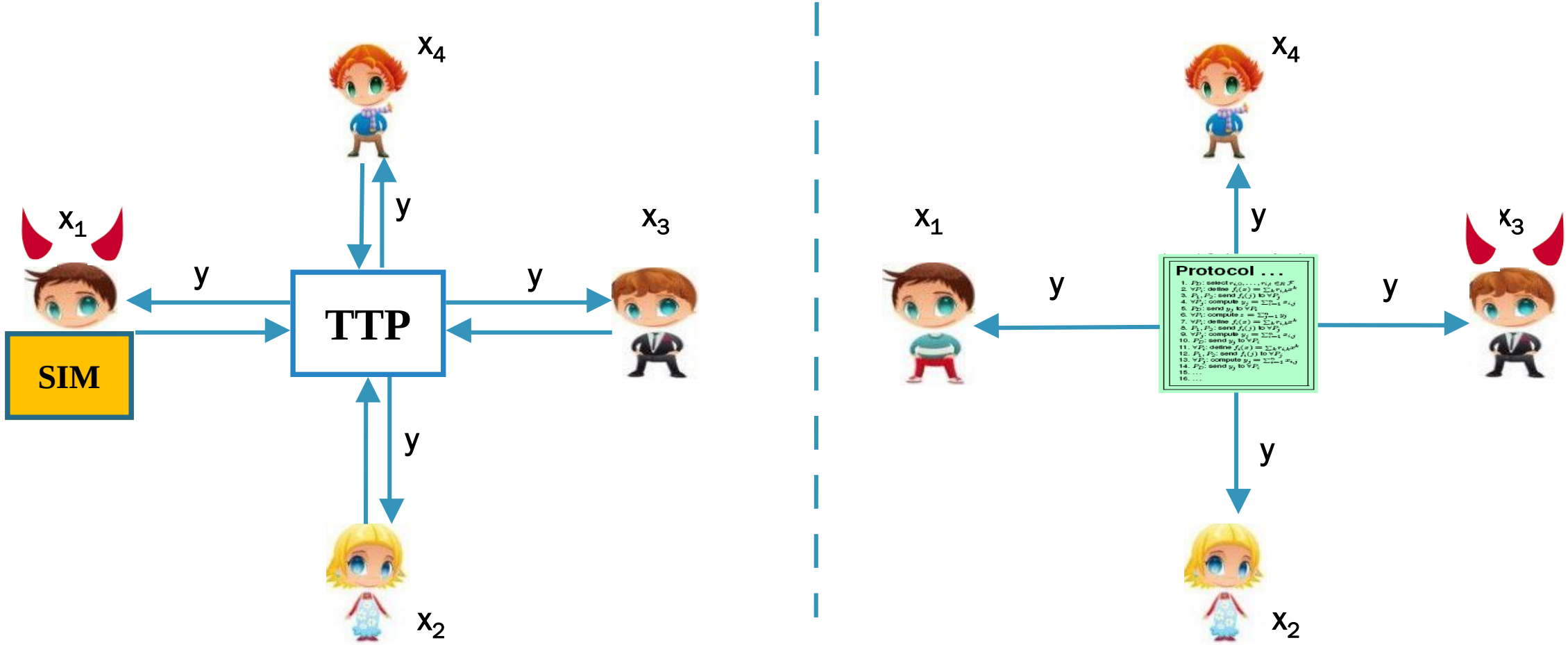
1. **Malicious adversary with honest majority for Five Party Computation.**
2. **Adversary with 1 malicious, 1 semi-honest corruption for Four Party Computation.**

Why Small Population?

- **Real world applications:** Secure ML, Danish Sugar Beet Auction, Fair Auctions.
- **Weaker Assumptions:** Eliminate PK primitives like OT altogether as symmetric-key functions are sufficient.
- **Stronger Security:** The properties, fairness and guaranteed output delivery can be achieved only in the case of honest majority [Cleve86].
- **Light Weight Tools and Efficiency:**
  - *Customized Secret Sharing schemes.*
  - *Use of passively secure tools.*
  - *Customized OT.*
- **1 corruption → 2 corruptions:** Elevating the challenges to achieve stronger security notions while maintaining the efficiency goal , as the adversary has a co-conspirator.

[Cleve86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In ACM STOC, 1986.

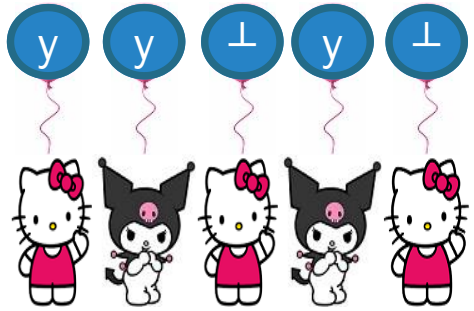
Security Model



$\{View^{Ideal}\}_{i,Pi \text{ in } C} \approx \{View^{Real}\}_{i,Pi \text{ in } C}$

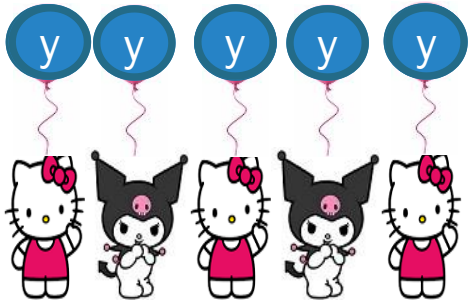
Ideal World **Real World**

Security Notions : Degree of Robustness

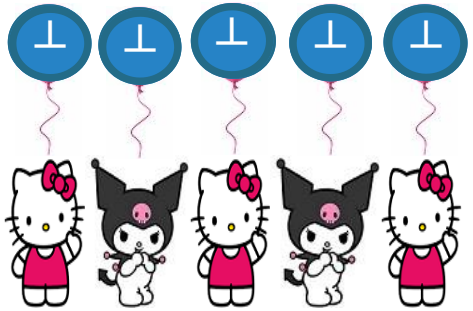


Security with Abort

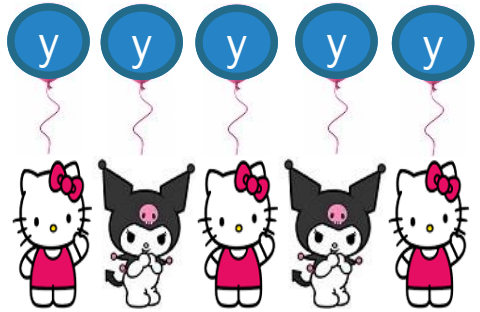
Weakest



Unanimous Abort



Fairness



Guaranteed Output Delivery

Strongest

Our Results

Efficient 5-Party (5PC) Protocols with honest majority:

- Unanimous Abort (8 rounds).
- Fairness (8 rounds).
- **G**uaranteed **O**utput **D**elivery (god).
  -- 6 rounds (best case).
  -- 12 rounds (worst case).

Assumptions:

- One Way Permutations.
- Minimalistic network of point-to-point channels.
- Necessary Broadcast for 5PC god [CohenHOR16].

Efficient 4-Party (4PC) Protocols with Mixed Adversary (1 Active, 1Passive):

- Fairness.
- **G**uaranteed **O**utput **D**elivery (god).

Implementation:

- **Highly Efficient for practical systems**.
- **First robust Broadcast Implementation in 5PC**.

[CohenHOR16] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of Secure Multiparty Computation Without Broadcast. In TCC. 2016.

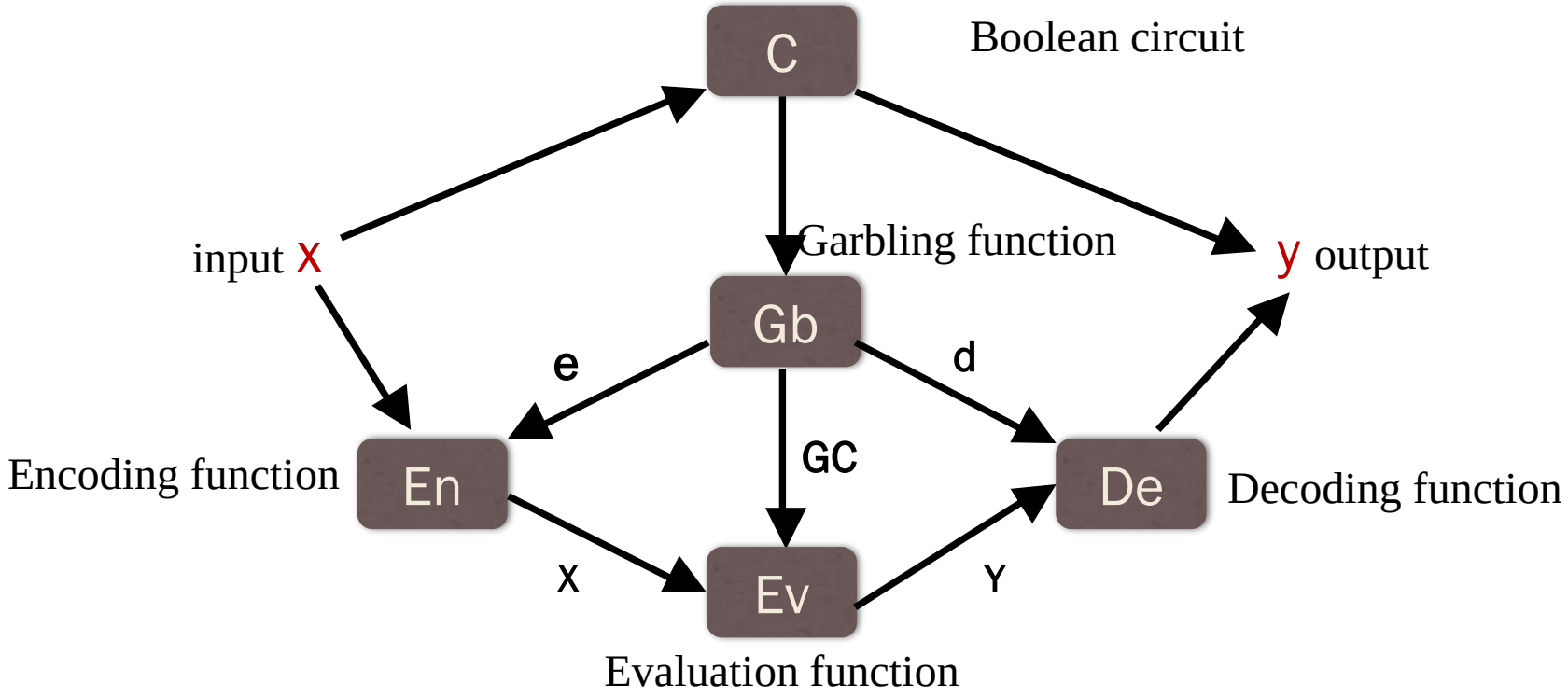
Comparison

Reference	Security	Corruption	Broadcast
[ChandranGMV17]	Selective Abort	2 active	✗
This work 5PC	Unanimous Abort	2 active	✗
This work 5PC	Fairness	2 active	✗
This work 5PC	GOD	2 active	✓ **[CohenHOR16]**
This work 4PC	Fairness	1 active, 1 passive	✗
This work 4PC	GOD	1 active, 1 passive	✗

[CohenHOR16] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of Secure Multiparty Computation Without Broadcast. In TCC. 2016.

[ChandranGMV17] Nishanth Chandran, Juan Garay, Payman Mohassel and Satyanarayana Vusirikala. Efficient, Constant-Round and Actively Secure MPC: Beyond the Three-Party Case. In CCS 2017.

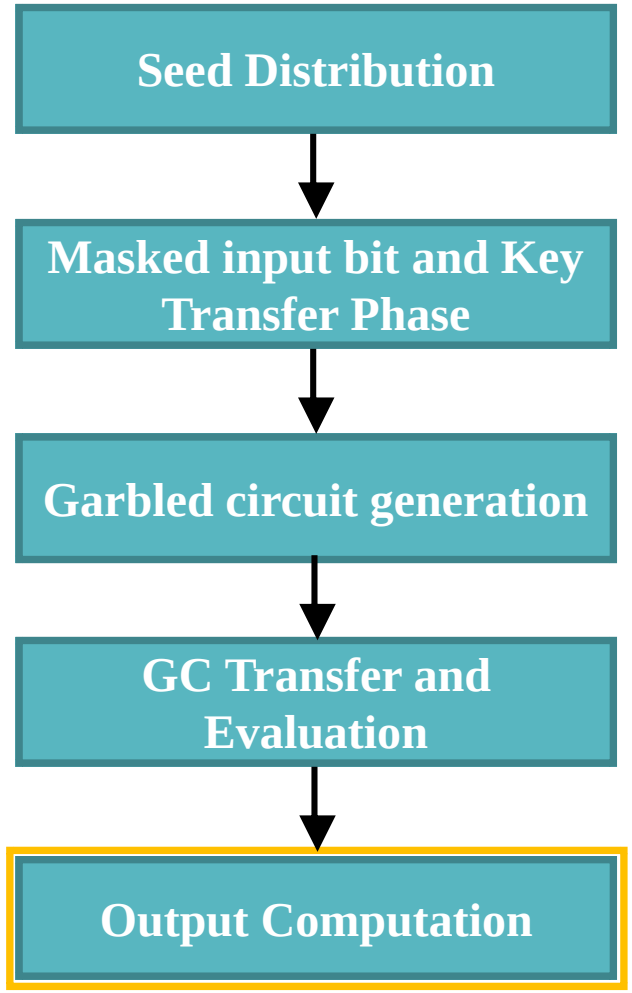
Garbled Circuit (GC) [BellareHR12]



$GC = GC^1 \,||\, GC^2 \,||\, \ldots \,||\, GC^n$

[BellareHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In CCS, 2012.

5PC with Fairness



$y = f(x_1, x_2, x_3, x_4, x_5)$
is the function to be computed.

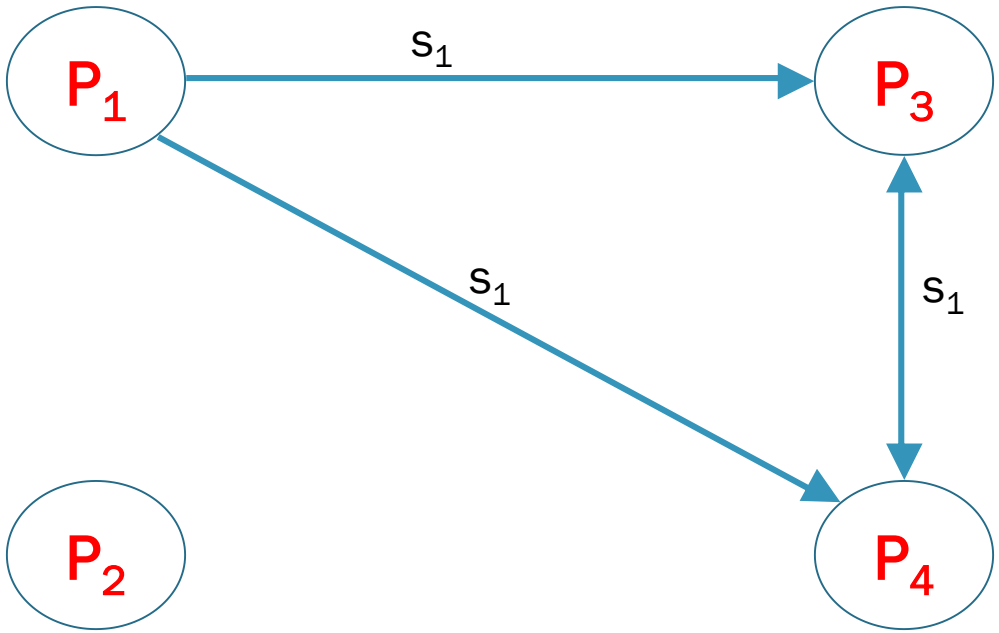
Garblers - P_1, P_2, P_3, P_4
Evaluator - P_5

n=5, *t*=2

5PC with Fairness

Seed Distribution

Pick s_1

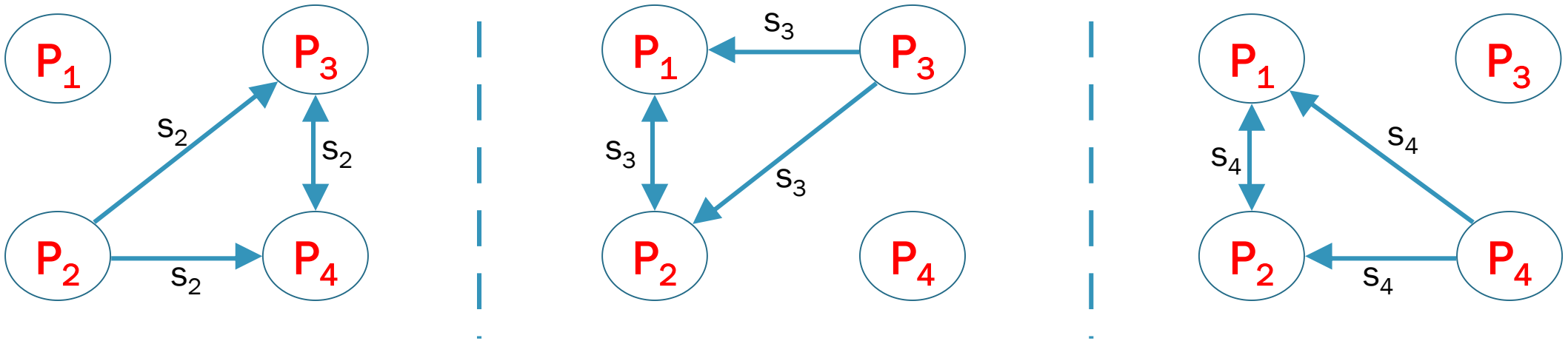


Has no knowledge of s_1

n=5, t=2

5PC with Fairness

Seed Distribution

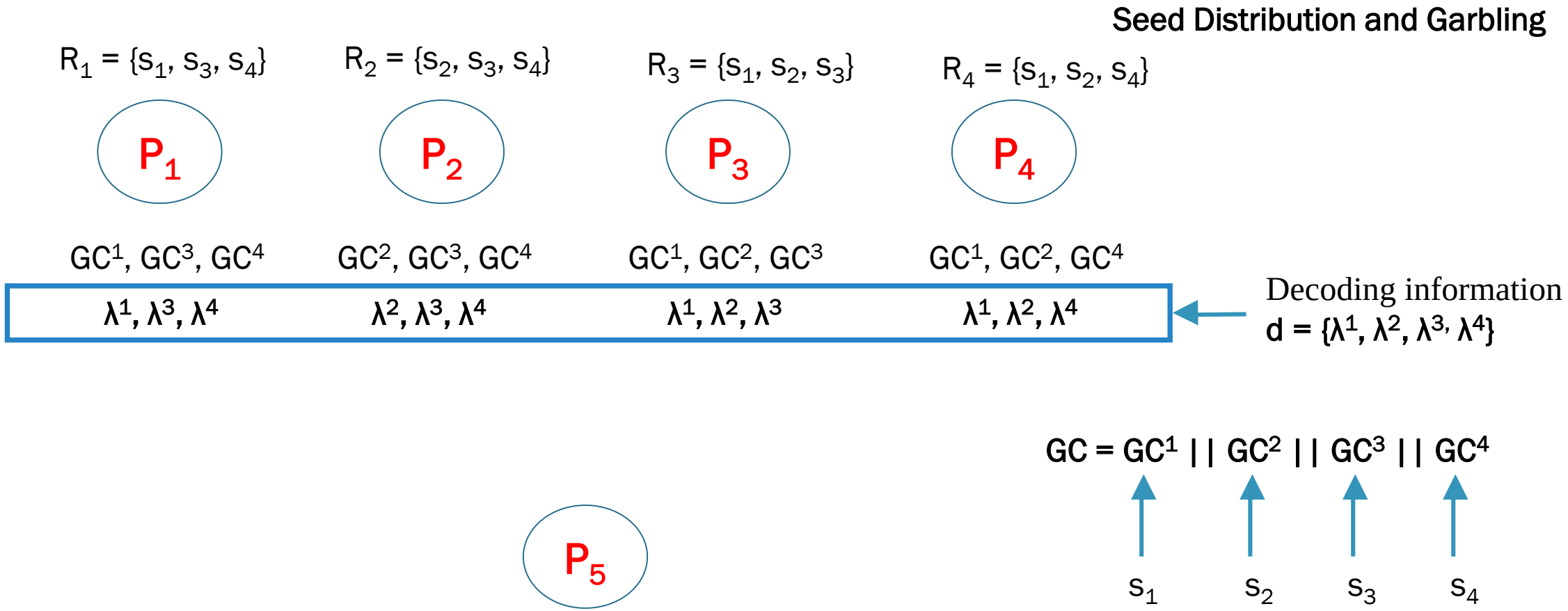


$R_1 = \{s_1, s_3, s_4\}$
$R_2 = \{s_2, s_3, s_4\}$
$R_3 = \{s_1, s_2, s_3\}$
$R_4 = \{s_1, s_2, s_4\}$

For $i \in [4]$,
R_i *indicates the seeds held by a party* P_i
R_i *indicates the parties who hold* s_i

n=5, t=2

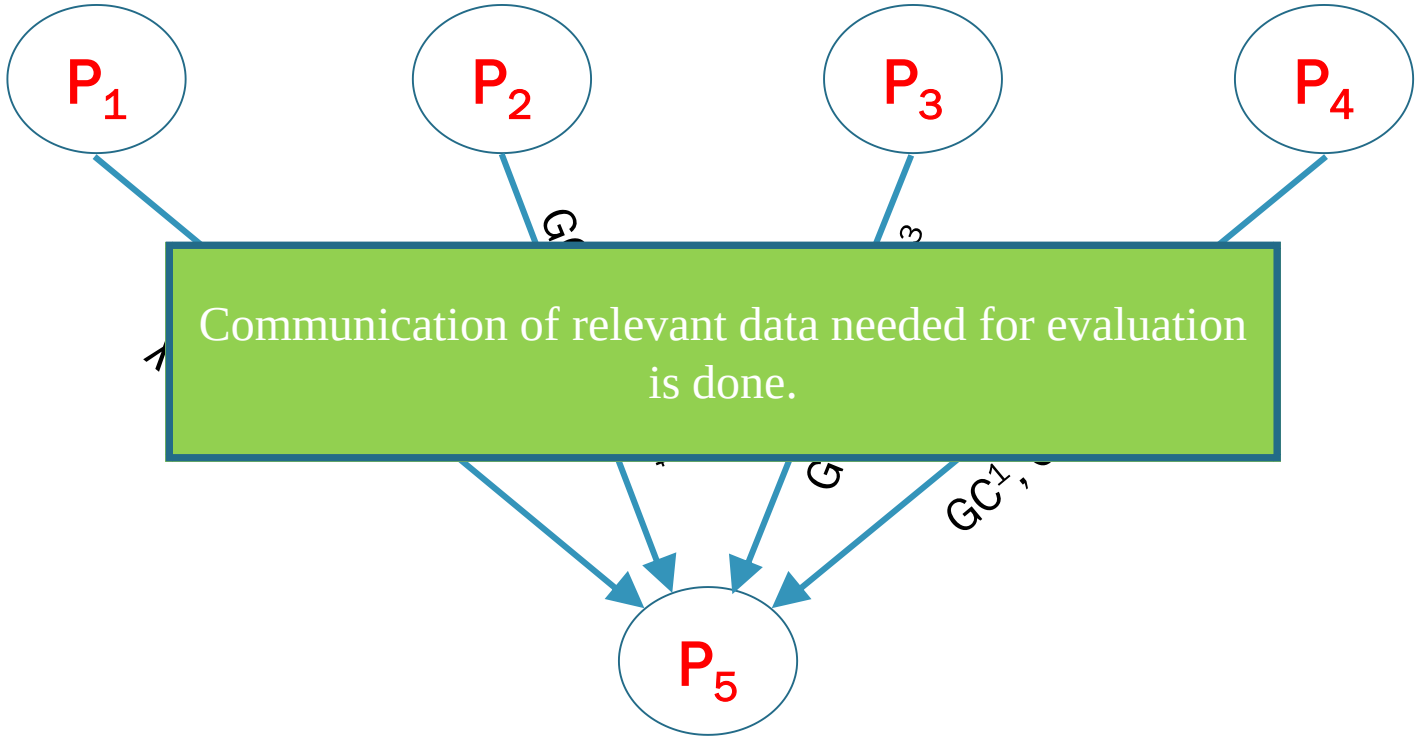
5PC with Fairness



n=5, t=2

5PC with Fairness

GC Transfer and Evaluation



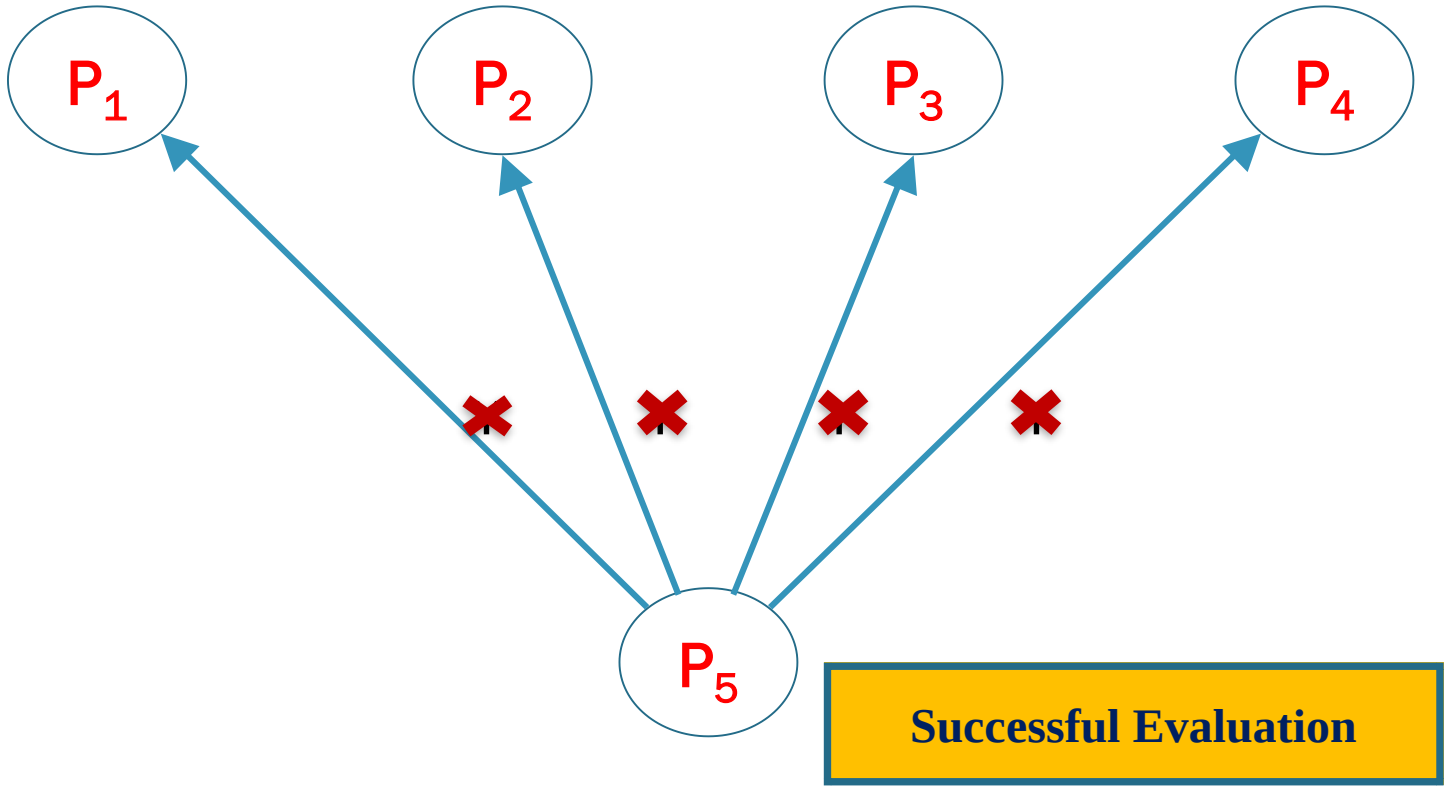
Accept only if all copies of each value match. Else abort.

n=5, t=2

5PC with Fairness

Evaluation and Output Computation

Check if **Y** is valid. If so, use **Y** to output **y**.



Decode the output **y**

Problem?

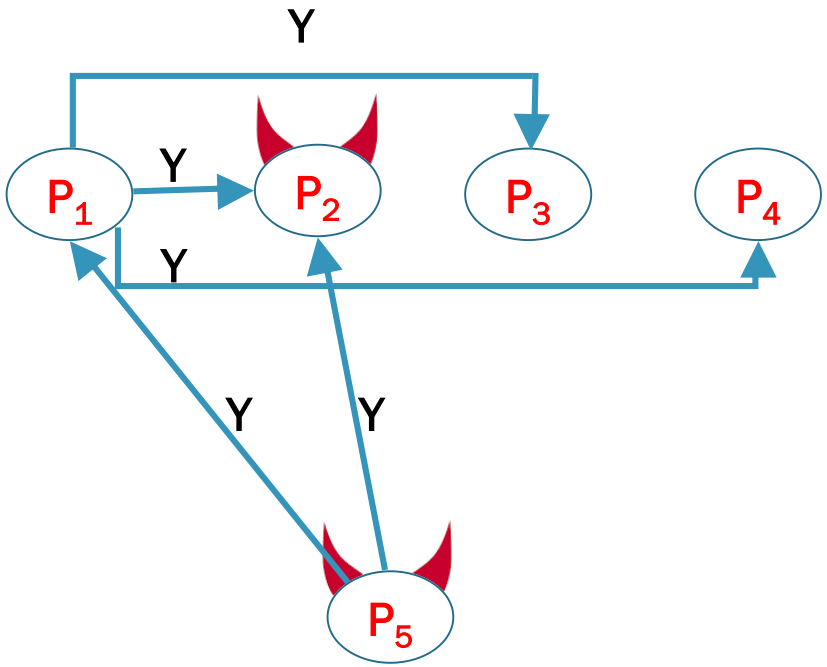
1. P_5 selectively sends **Y**.
2. P_5 sends no **Y**.

Solution :

1. Allow garblers to exchange **Y**.
2. Delay exchange of λ–values (decoding) of output wires until **Y** is received from P_5.

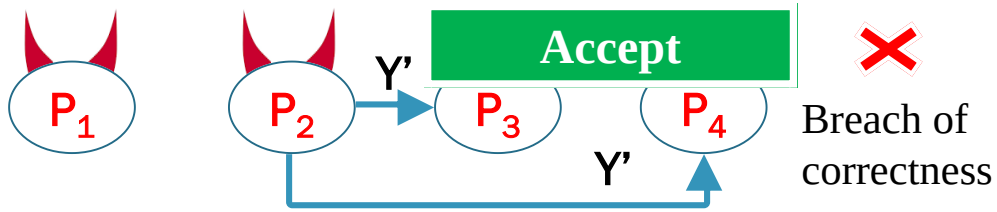
n=5, t=2

5PC with Fairness



Output Computation

$R_1 = \{s_1, s_3, s_4\}$ $R_2 = \{s_2, s_3, s_4\}$



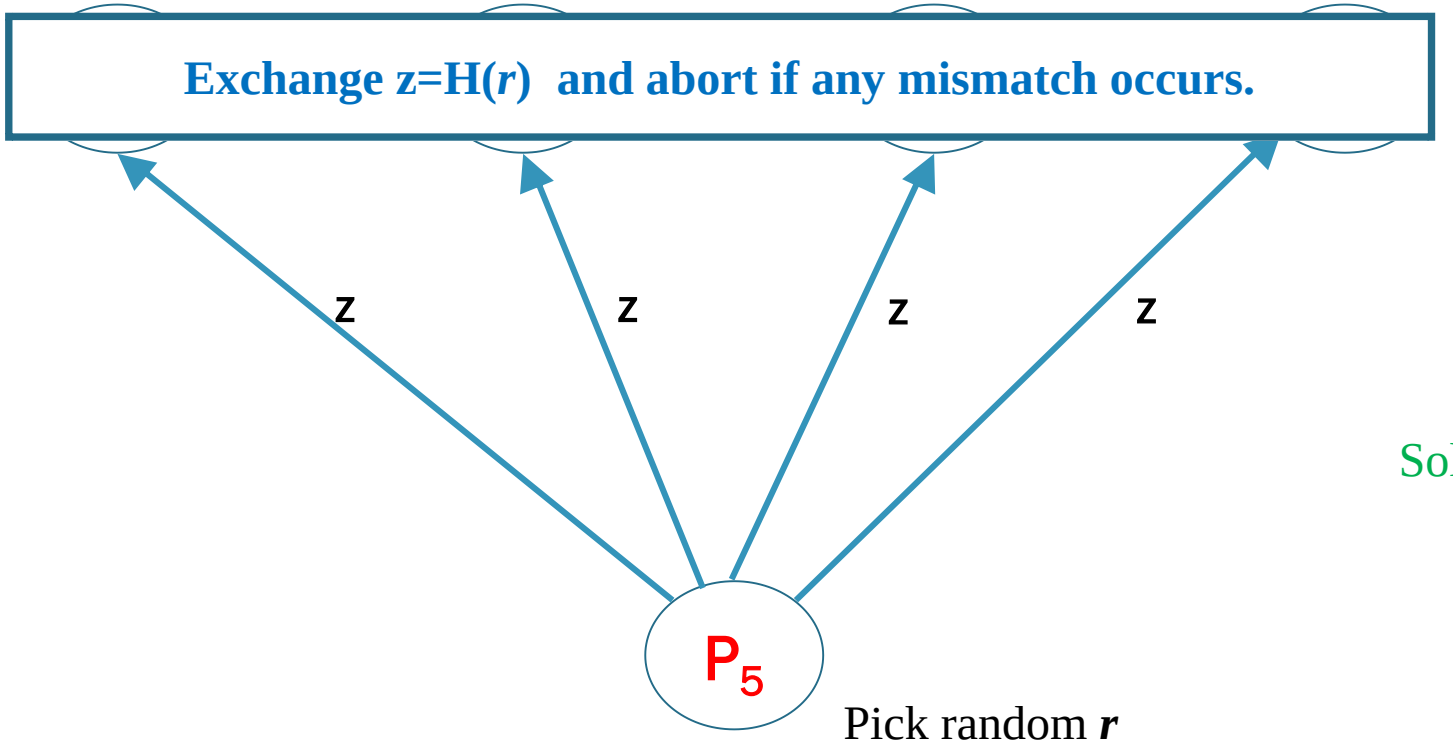
Problems in Solution 1?

Solution: Need of proof that Y originated from P_5.

5PC with Fairness

Proof Establishment Phase

Run before key transfer

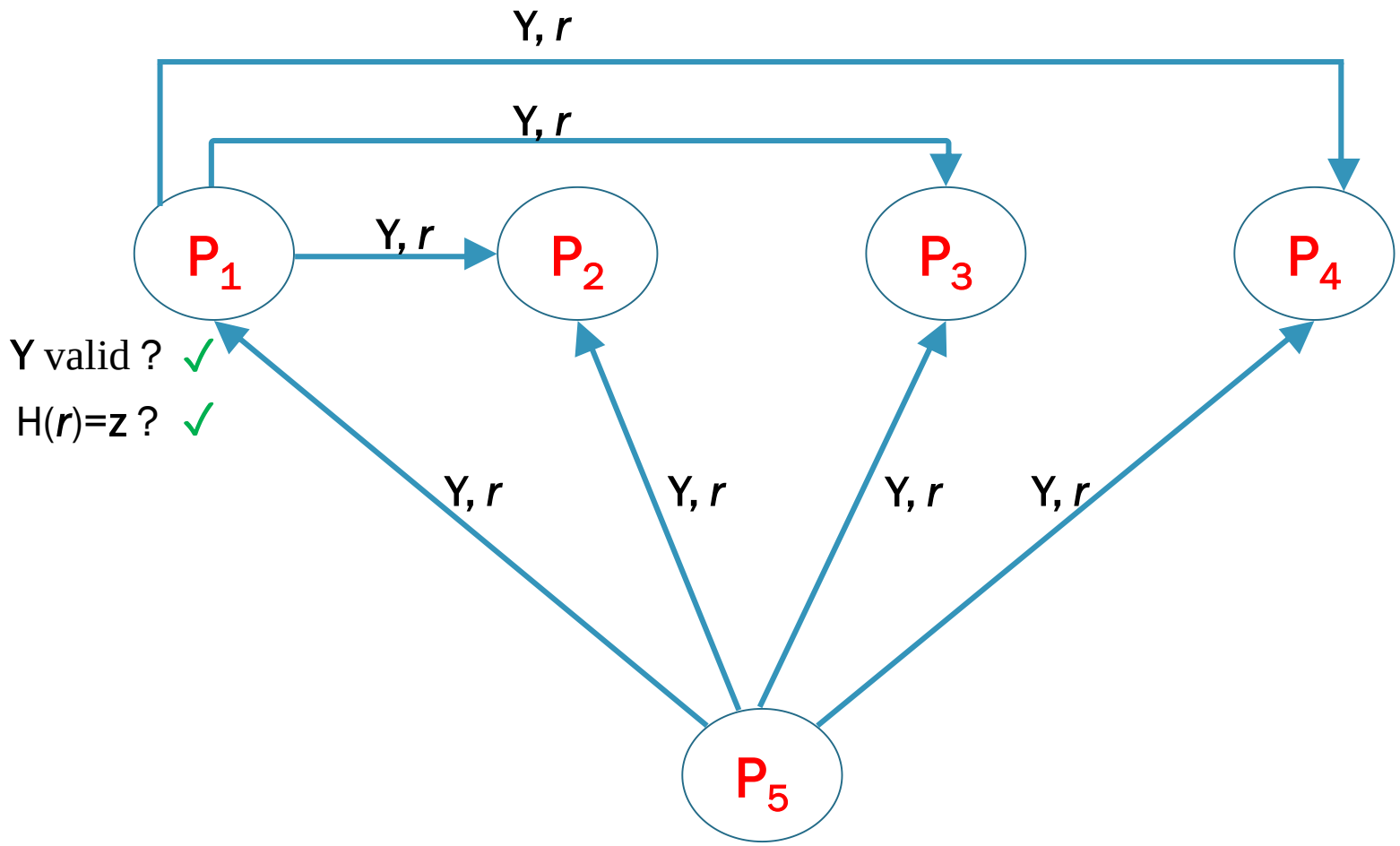


Pick random ***r***
Compute **z**=H(***r***)

Solution 1 + Proof solves problem 1.

n=5, *t*=2

5PC with Fairness



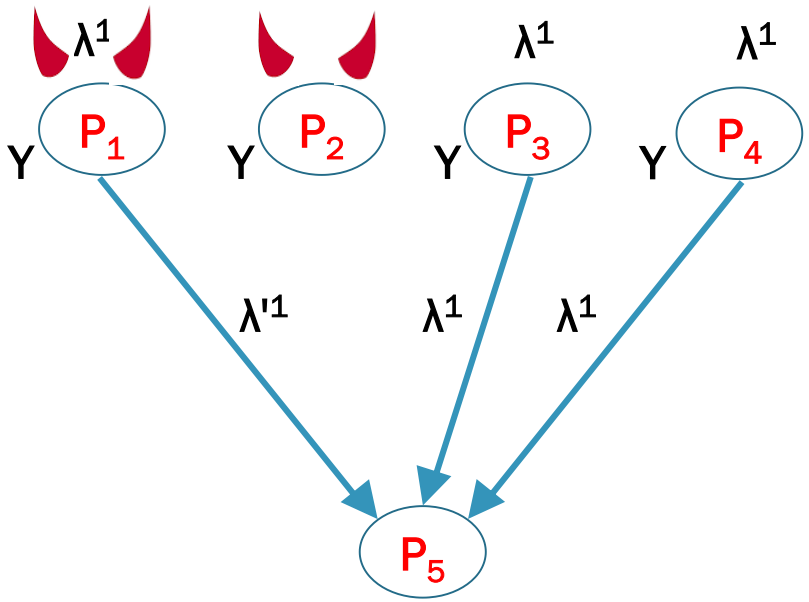
Output Computation

Accept if Y is valid and z= H(*r*)

n=5, *t*=2

5PC with Fairness

Communicate the λ-values for each output wire



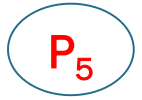
Problems in Solution 2?

Commitment Establishment

P_i computes $c^i = com(\lambda^i)$, $s_i \in R_i$

Exchange three copies of each c^i, $i \in [4]$

Else commitment on λ-values for each output wire are in agreement

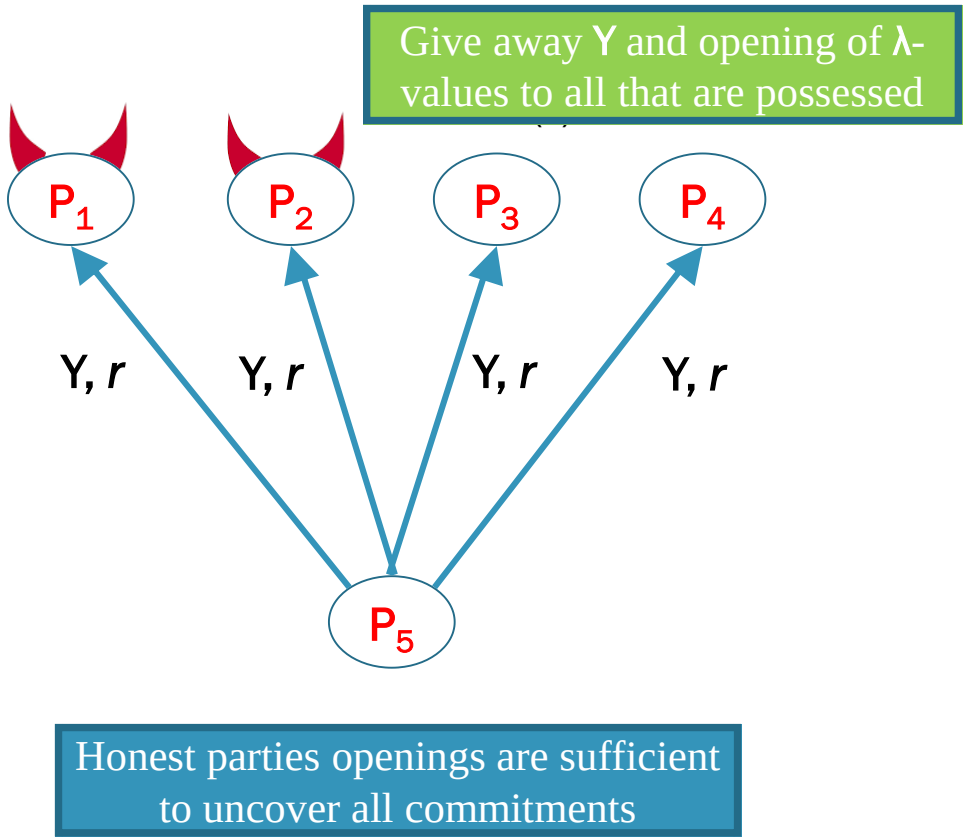


At least one party will give a valid opening for each c^i in output computation

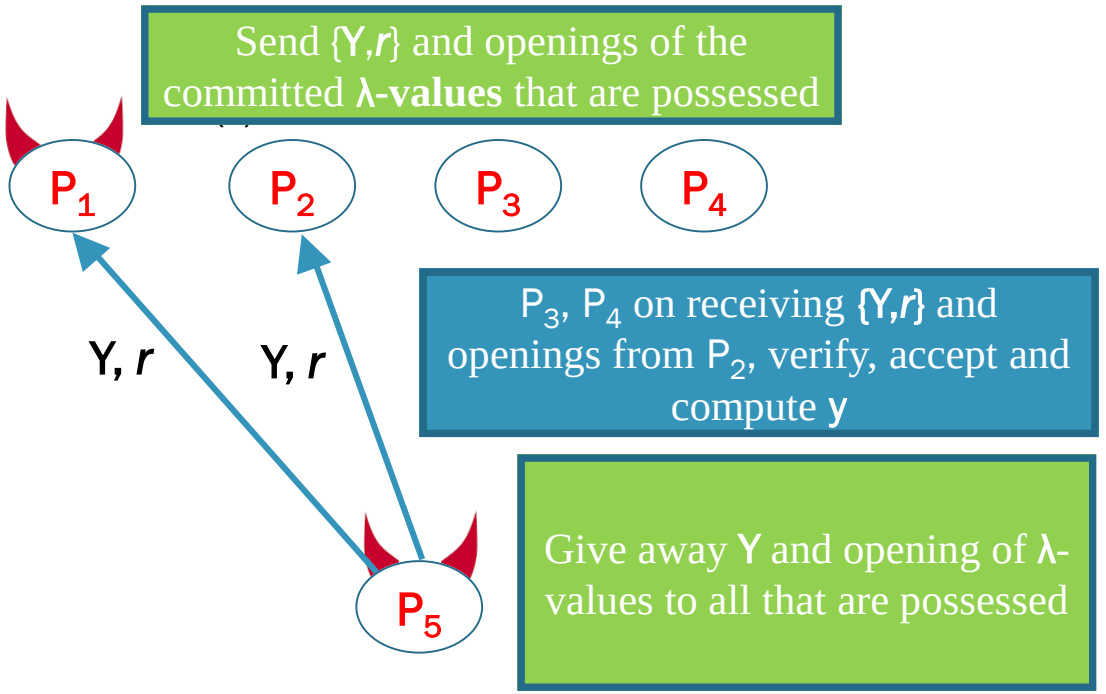
Solution : Commit-then-open

n=5, t=2

5PC with Fairness

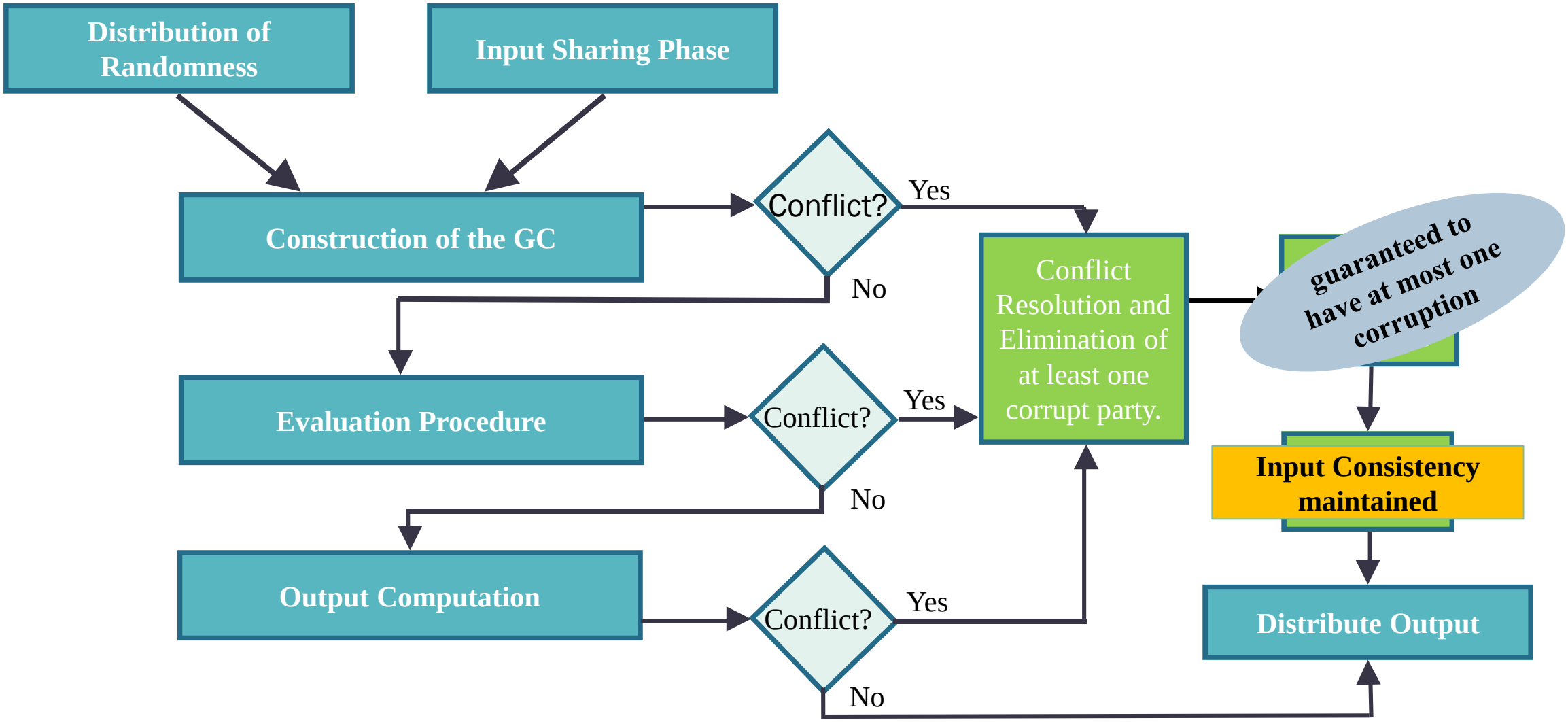


Output Computation



n=5, t=2

5PC with Guaranteed Output Delivery



Efficiency Comparison

Overheads in comparison to [ChandranGMV17] for AES-128 and SHA-256 circuits (given in the range):

Protocol	LAN (ms)	WAN (s)	Communication (MB)
5PC with Unanimous Abort	0.65-2.87	0.01-0.2	0.09-0.16
5PC with Fairness	1.05-10.95	0.03-0.28	0.13-0.2
5PC with GOD (Honest Run)	3.94-4.92	0.82-1.16	0.17-0.18
(Worst Case)	6.33-16.82	2.26-2.33	0.49-6.34
4PC with Fairness	2.93-23.14 **(g)**	0.37-0.99 **(g)**	12.83-132.36 **(g)**
4PC with GOD (Honest Run)	2.54-17.38 **(g)**	0.01-0.54 **(g)**	12.77-132.24 **(g)**
(Worst Case)	1.14-1.9 **(g)**	-0.23-0.29 **(g)**	12.73-129.24 **(g)**

GOD - guaranteed output delivery, **(g)**- gain over [ChandranGMV17] .

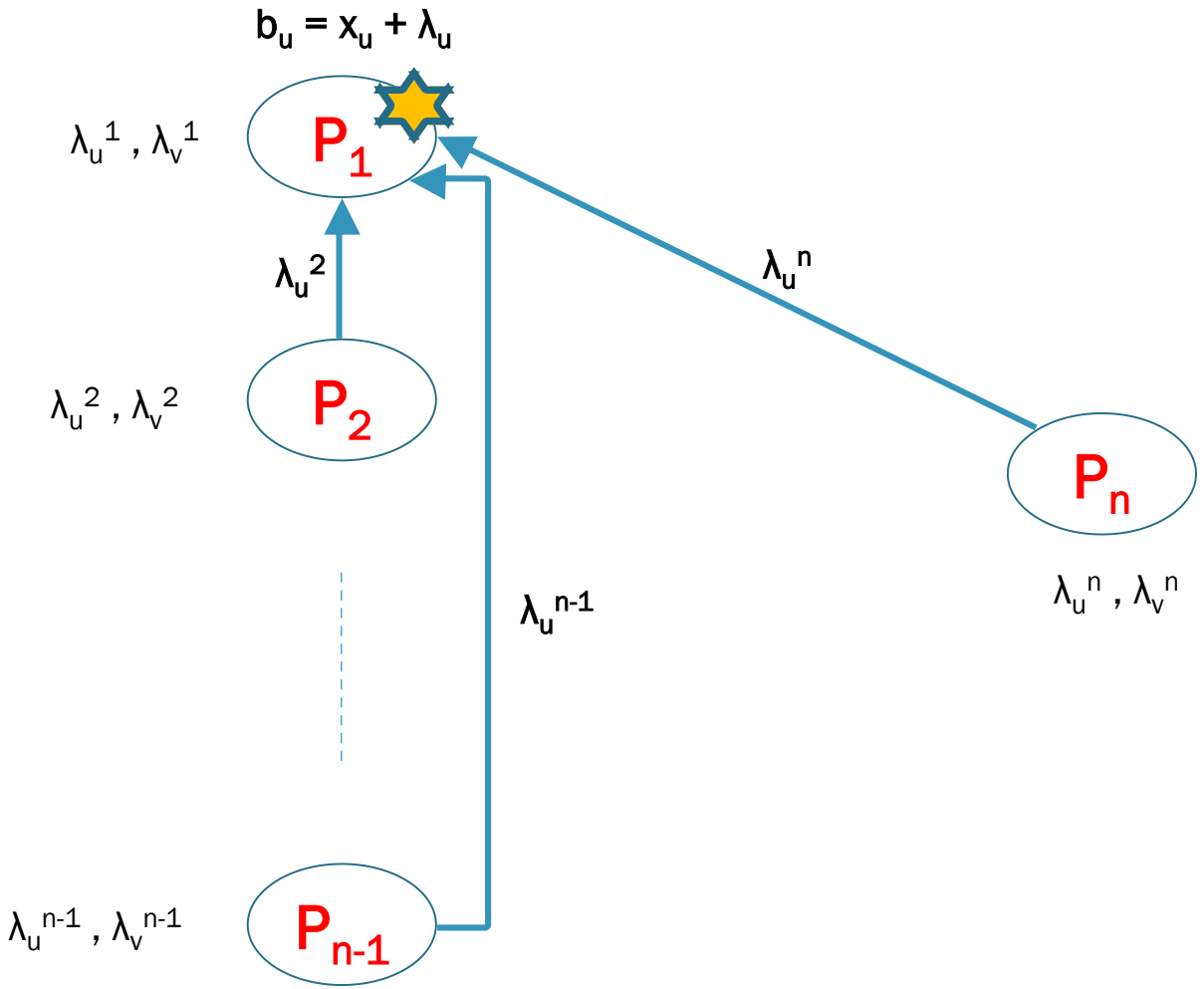
[ChandranGMV17] Nishanth Chandran, Juan Garay, Payman Mohassel and Satyanarayana Vusirikala. Efficient, Constant-Round and Actively Secure MPC: Beyond the Three-Party Case. In CCS 2017.

Future Work

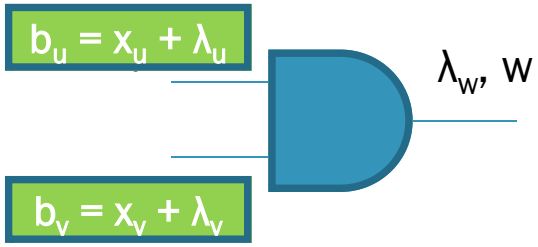
- Improving the round complexity of our protocols while guaranteeing stronger security notions and maintaining similar efficiency.

Thank You!

Distributed GC [BMR90]

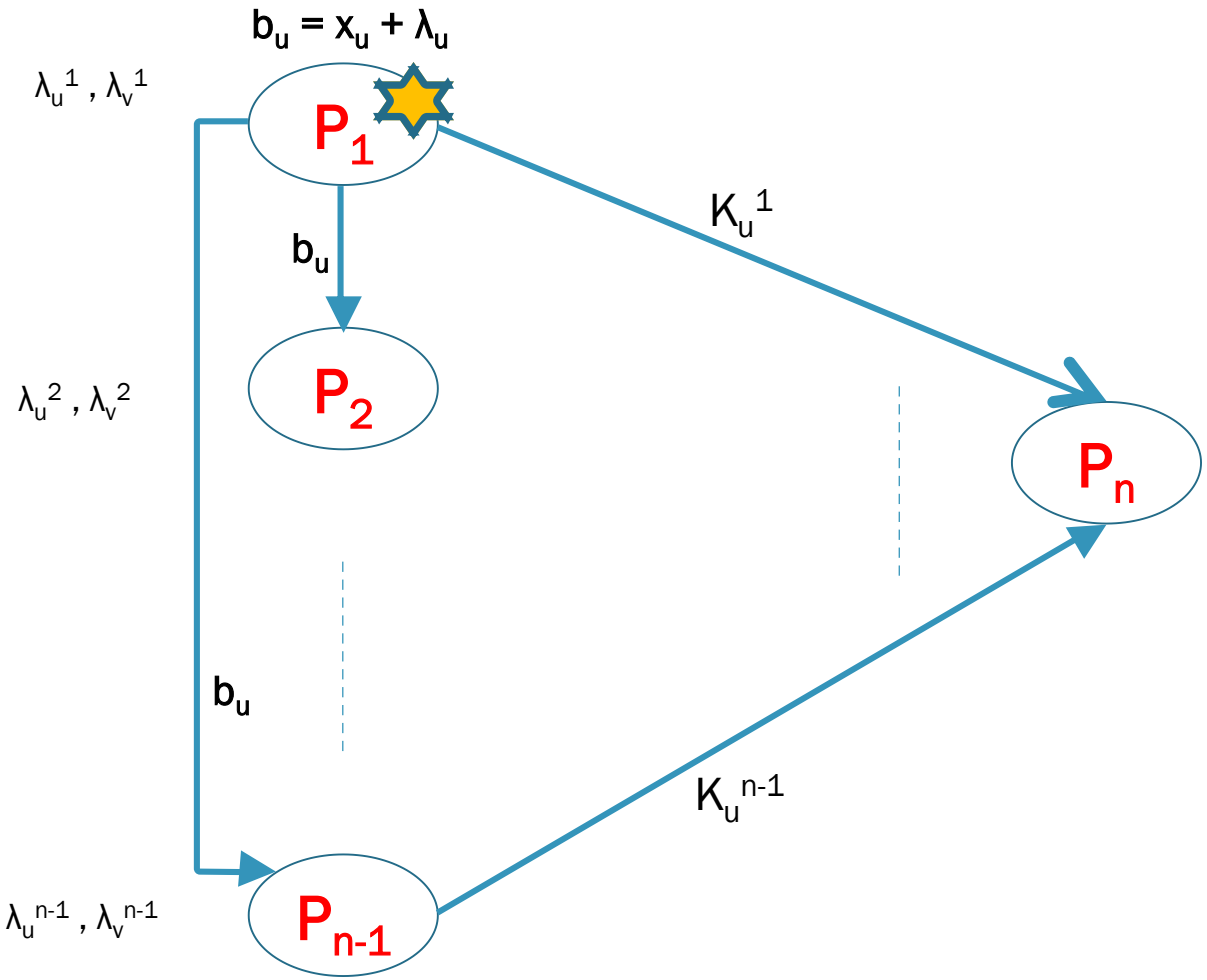


Masked Evaluation

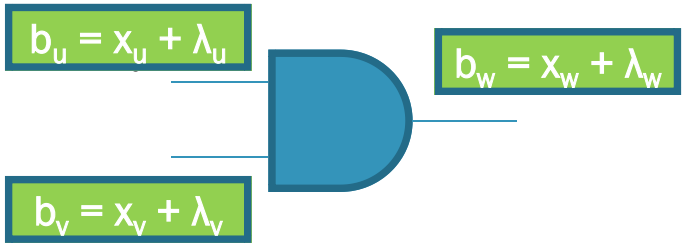


$$\lambda_u = \lambda_u^1 + \lambda_u^2 + ... + \lambda_u^n$$

Distributed GC [BMR90]

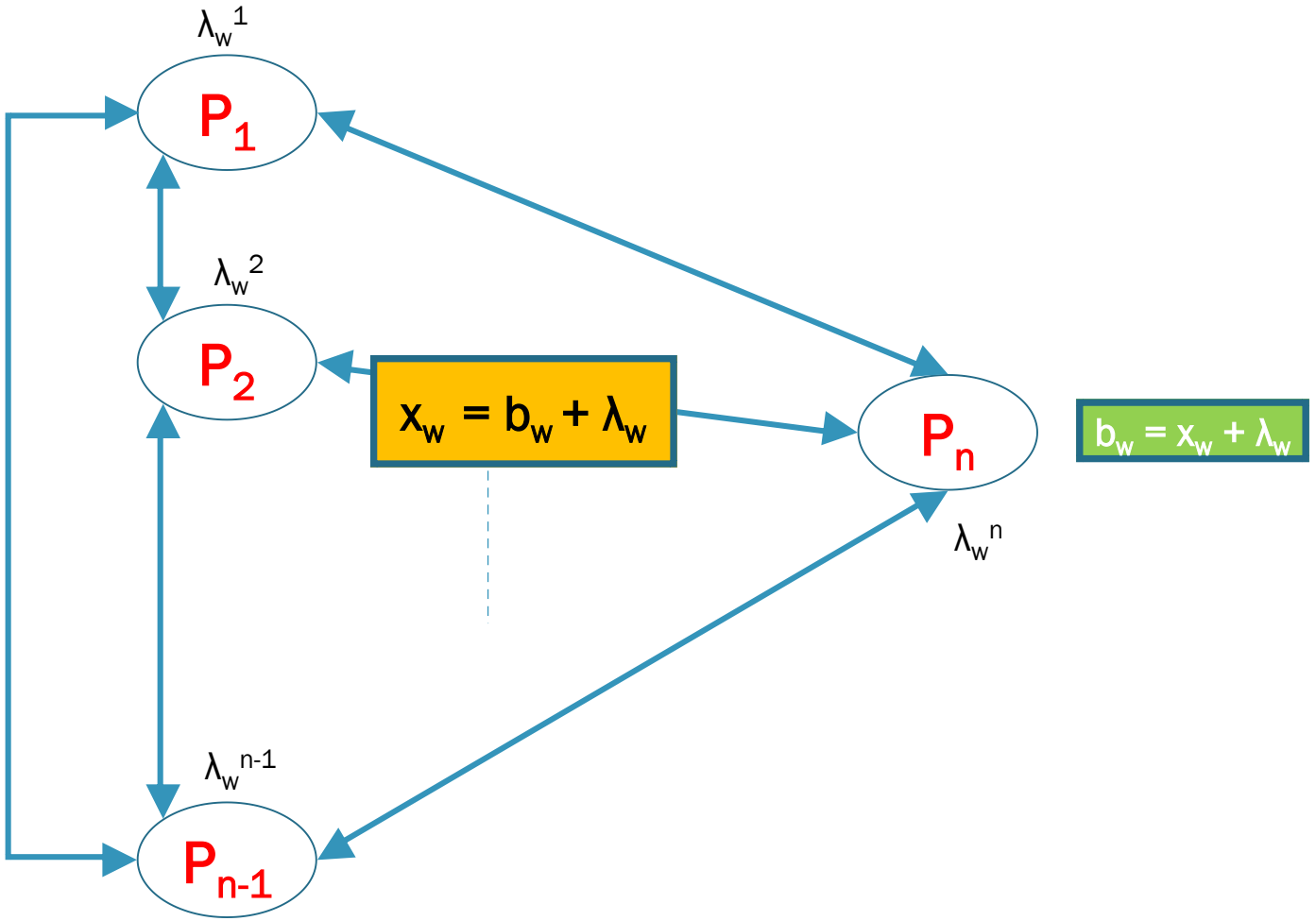


Masked Evaluation



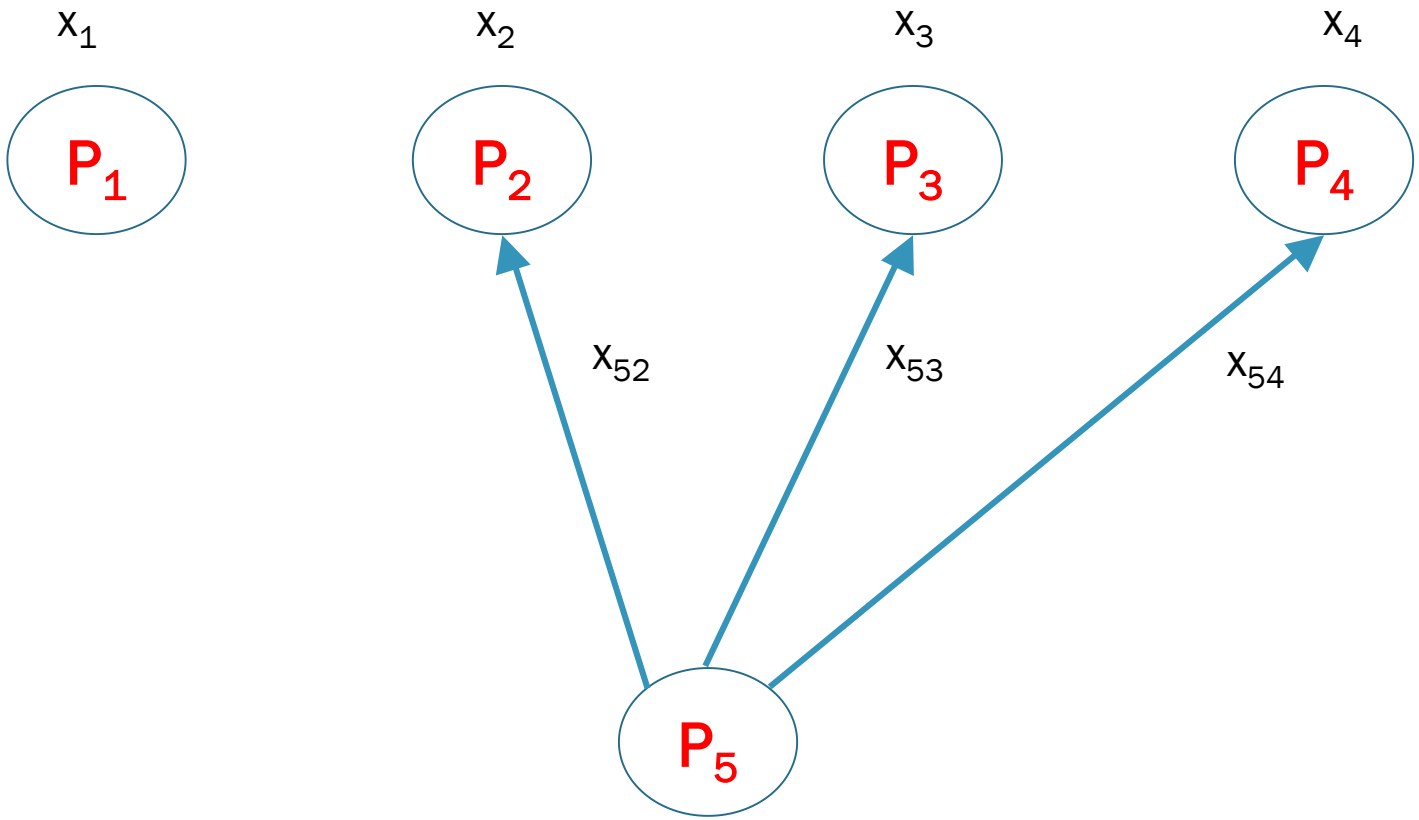
$$\lambda_u = \lambda_u^1 + \lambda_u^2 + ... + \lambda_u^n$$

Distributed GC [BMR90]



5PC with Fairness

Masked Input and Key Transfer

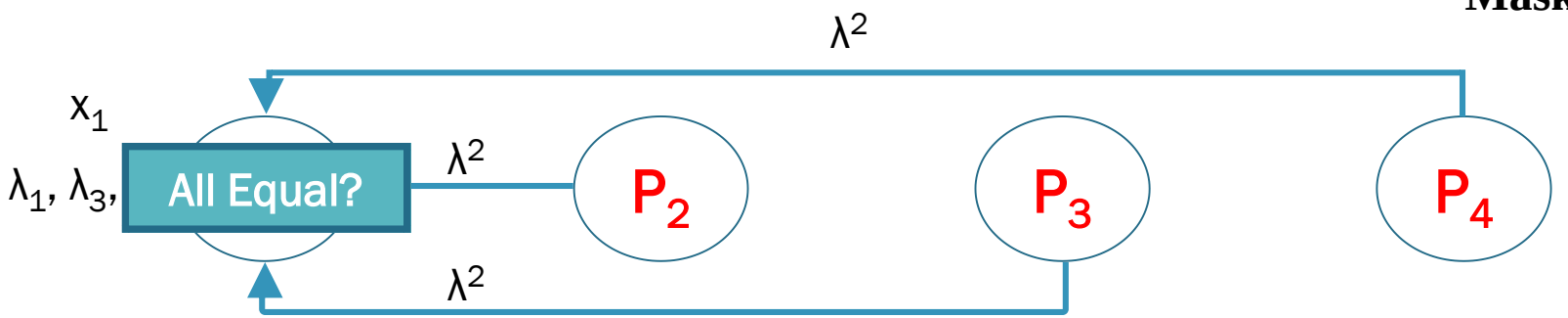


$x_5 = x_{52} + x_{53} + x_{54}$

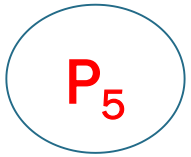
n=5, *t*=2

5PC with Fairness

Masked Input and Key Transfer



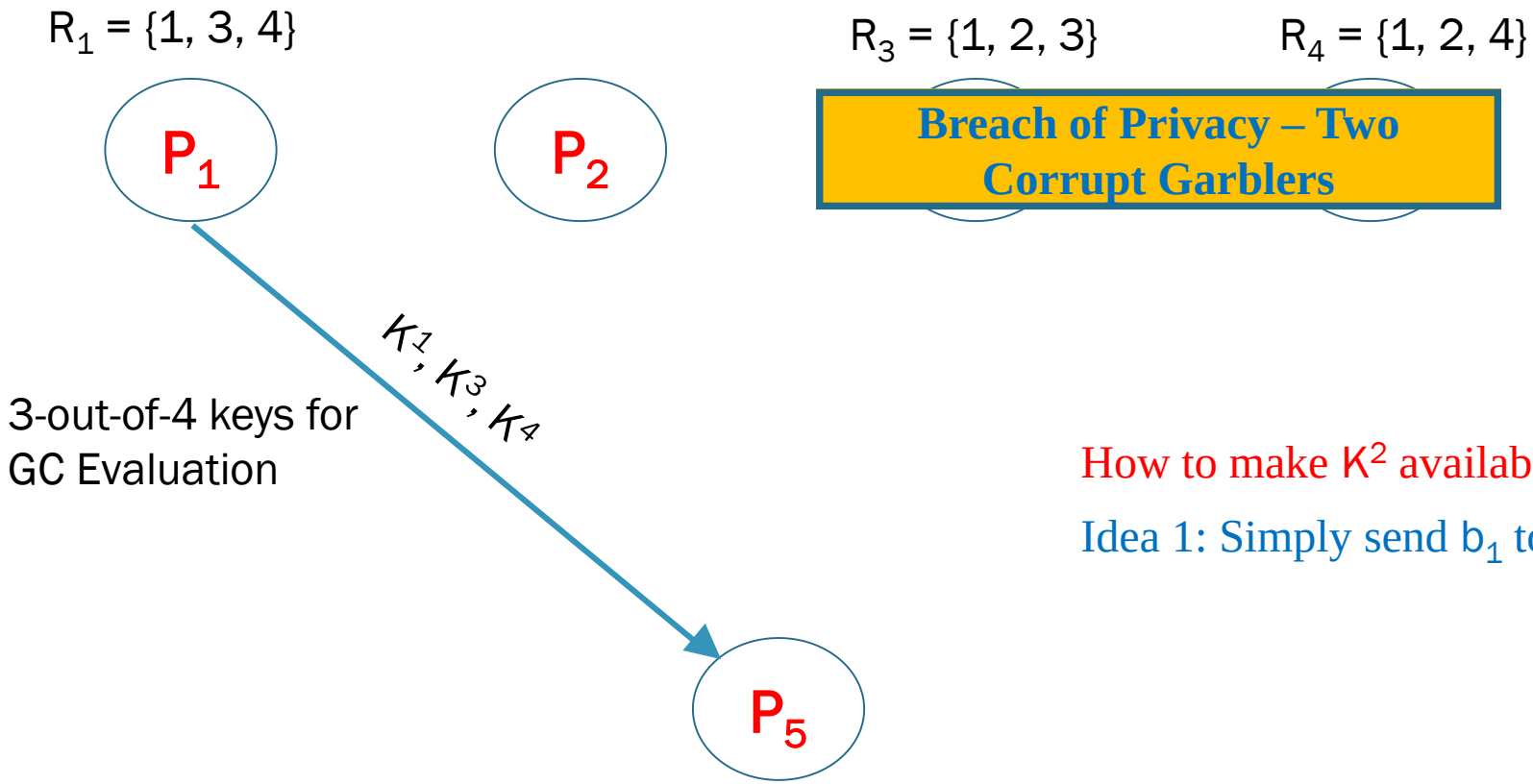
$b_1 = x_1 + \lambda = x_1 + (\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4)$



n=5, t=2

5PC with Fairness

Masked Input and Key Transfer



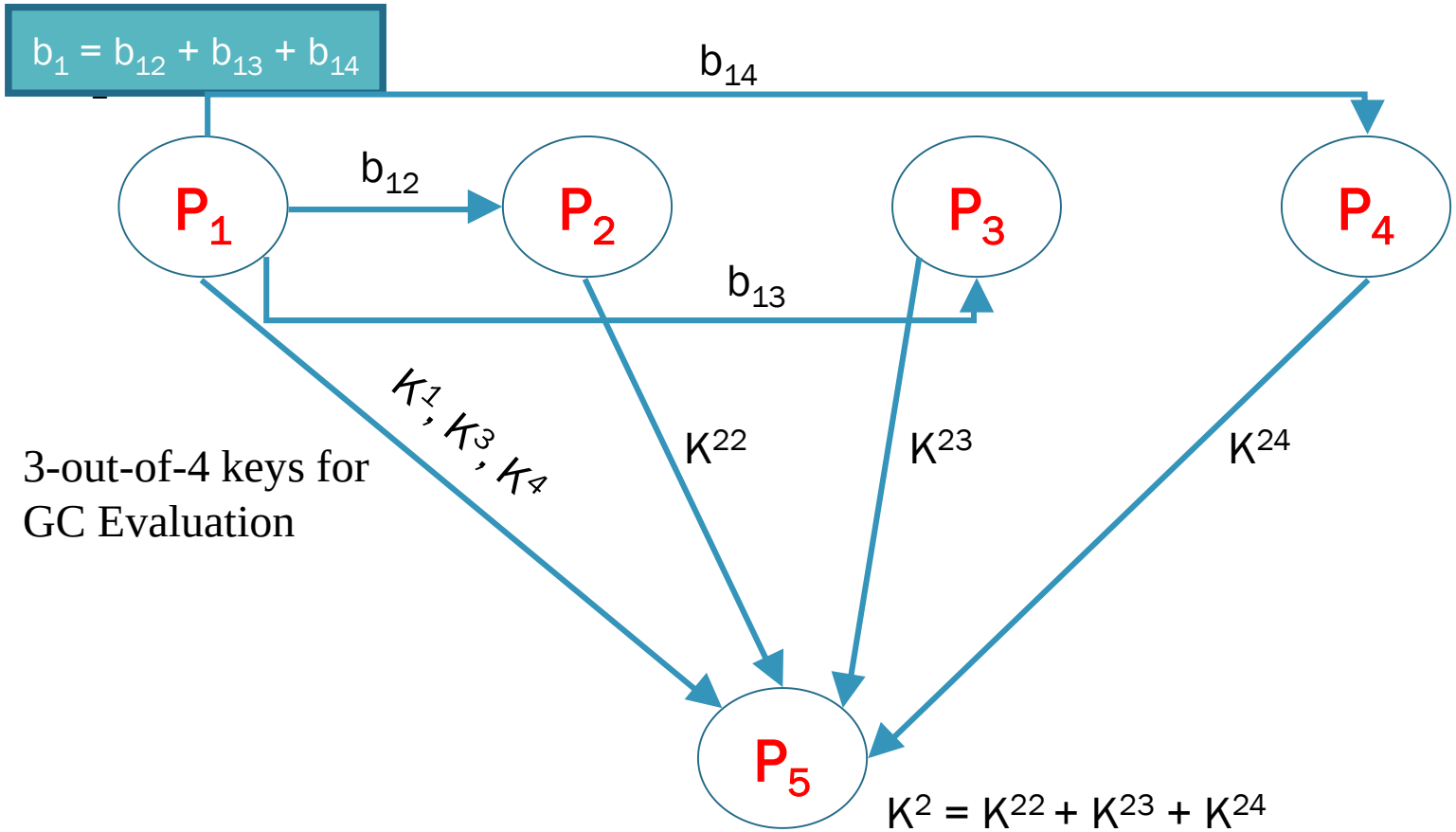
How to make K^2 available to P_5?

Idea 1: Simply send b_1 to the rest of the garblers.

K^1, K^2, K^3, K^4 are keys corresponding to bit b_1

n=5, t=2

5PC with Fairness



Masked Input and Key Transfer

$K^{22} = K^2$ (for bit b_{12}) + random pad
$K^{23} = K^2$ (for bit b_{13}) + random pad
$K^{24} = K^2$ (for bit b_{14}) + random pad

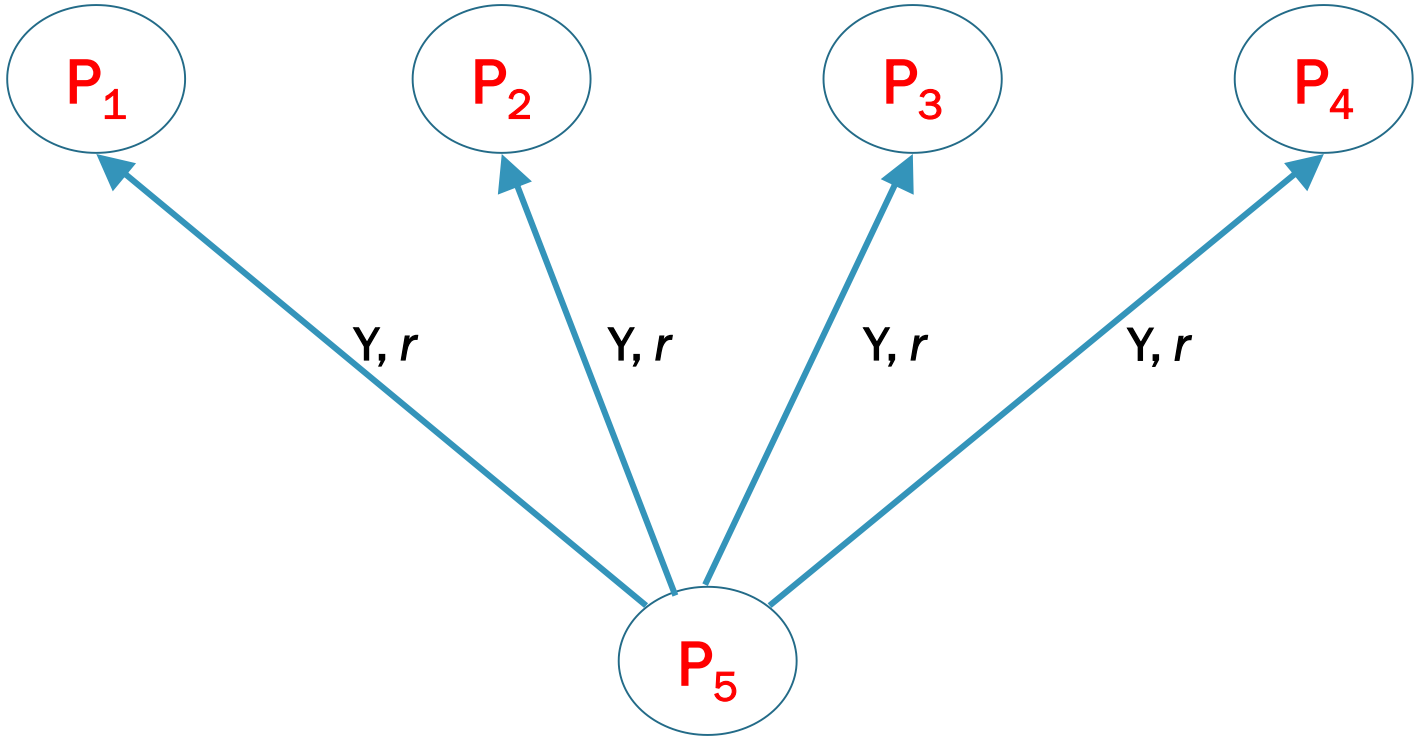
K^1, K^2, K^3, K^4 are keys corresponding to bit b_1

n=5, t=2

5PC with Fairness

Output Computation

To Summarize:



n=5, *t*=2

Efficiency

Protocol	LAN (ms)		WAN (s)		Total Communication (MB)	
	AES-128	**SHA-256**	**AES-128**	**SHA-256**	**AES-128**	**SHA-256**
[CGMV17]	25.01	290.38	2.54	4.78	29.55	389.12
5PC with Unanimous Abort	25.66	293.25	2.74	4.79	29.71	389.2
5PC with Fairness	26.06	301.33	2.82	4.81	29.75	389.24
5PC with GOD	26.03 (+2.62)	317.35 (+16.25)	3.7 (+1.1)	5.6 (+1.51)	29.67 (+0.31)	389.16 (+6.15)
4PC with Fairness	22.08	267.24	2.17	3.79	16.72	256.76
4PC with GOD	22.47 (+1.4)	273.0 (+15.48)	2.53 (+0.24)	4.24 (+0.25)	16.78 (+0.3)	256.88 (+3.0)

The bracket values indicate the worst case run of our guaranteed output delivery (GOD) protocol.