On the Exact Round Complexity of Secure Three-Party Computation [CRYPTO 2018]

Arpita Patra, Divya Ravi

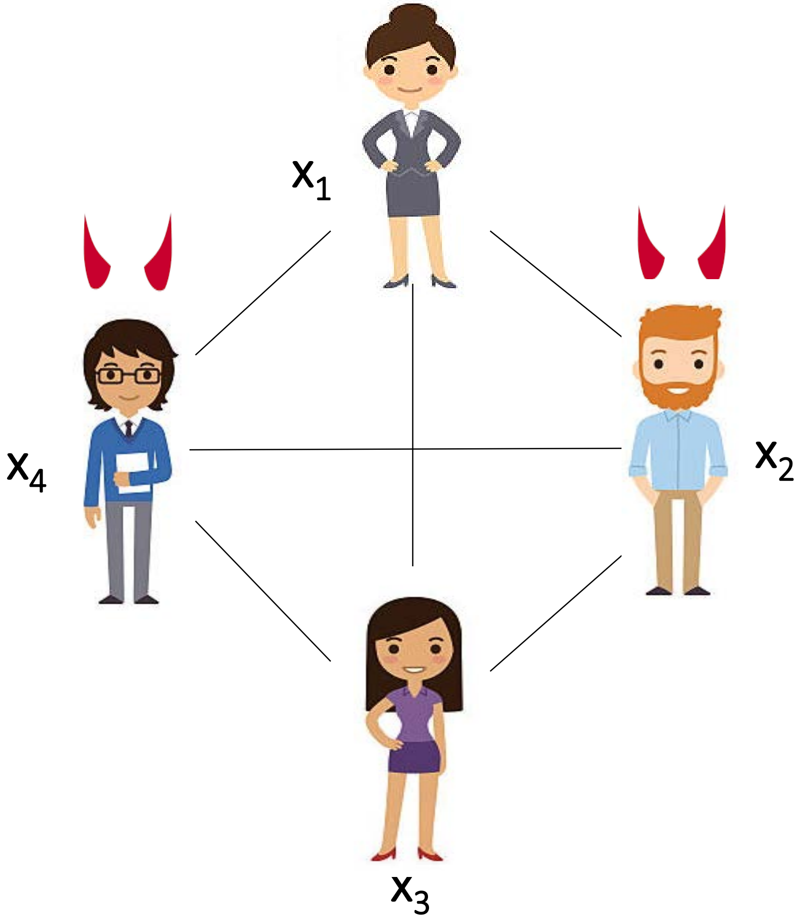
Indian Institute of Science

TPMPC 2018

Roadmap

- MPC
- Security notions
  - **g**uaranteed **o**utput **d**elivery (**god**),
  - **f**air**n**ess (**fn**),
  - **u**nanimous **a**bort (**ua**) and
  - **s**elective **a**bort (**sa**)
- 3PC with one malicious corruption- special case of honest majority
- Our results (2 lower bounds and 3 upper bounds) settling all questions on exact round complexity
  - point-to-point channels
  - above + broadcast
- 3-rounds are sufficient for 3PC protocol with fairness in [- broadcast]
- 3 rounds are necessary for nPC protocol with fairness in [+broadcast]; 3t > n>2t

MPC



Setup:

- n parties $P_1,....,P_n$; t are corrupted by a centralized adv
- P_i has **private** input x_i
- A common n-input function $f(x_1,x_2,..x_n)$

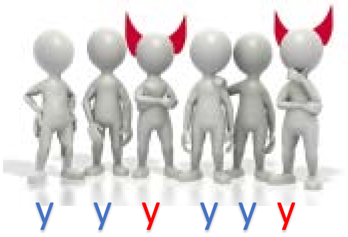
Goals:

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

Security Notions: Degree of Robustness

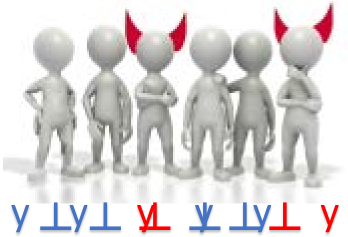
- **Guaranteed output delivery** (**god**) - Strongest

  Adversary cannot prevent honest parties from getting output



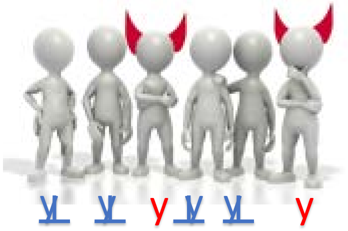
- **Fairness (fn)**

  If adversary gets output, all get the output



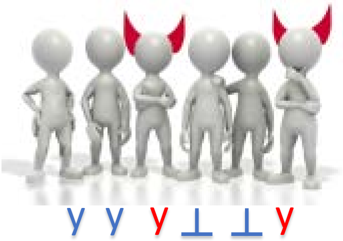
- **Security with unanimous abort (ua)**

  Either all or none of the honest parties get output (may be unfair)



- **Security with selective abort (sa)** - weakest

  Adversary selectively deprives some honest parties of the output



3PC with One Corruption: Why?

1st: **Popular setting for MPC in practice:** First Large-Scale Deployment of Danish Sugar Beet Auction, ShareMind, Secure ML

2nd: **Improved fault tolerance:** recovery of secrets is possible with 3 as opposed to 2

3rd: **Strong security goals:** god and fairness only achievable in honest majority setting [Cleve86]

4th: **Leveraging one corruption to circumvent lower bounds:**

+ 2-round 4PC of [IKPP15] circumvents the lower-bound 3 rounds for fair MPC with t > 1 [GIKR02]!
+ VSS with one corruption is possible in one round!

5th: **Weak assumptions:** possible from OWF/P shunning PK primitives such as OT altogether

6th: **Lightweight constructions and better round guarantee:**

+ No cut-and-choose
+ 2 vs 4 in plain model with point-to-point channels

The Exact Round Complexity of 3PC

	- Broadcast			+ Broadcast		
		Lower	Upper		Lower	Upper
selective abort (**sa**)	2	[HLP11]	[IKKP15]	2	[HLP11]	[IKKP15]
unanimous abort (**ua**)	3	Our Work	Our Work	2	[HLP11]	Our Work
fairness (**fn**)	3	Our Work	Our Work	3	Our Work	Our Work
Guaranteed (**god**)	Impossible	[CHOR16]	--	3	Our Work	Our Work

LB1: 3 rounds are necessary for **ua** in [- broadcast]

- Implies optimality of 3PC with **sa** in terms of security

UB1: 3 rounds are sufficient for **fn** in [- broadcast]

Lower bounds can be extended for any n, t; 3t > n > 2t

Upper bounds rely on (injective) OWF (garbled circuits)

LB2: 3-rounds are necessary for **fn** in [+ broadcast]

- Broadcast does **not** improve round complexity
- Complements a result that fairness requires 3 rounds for t>1 and any n;
- n=4 is necessary implying known 4PC optimal

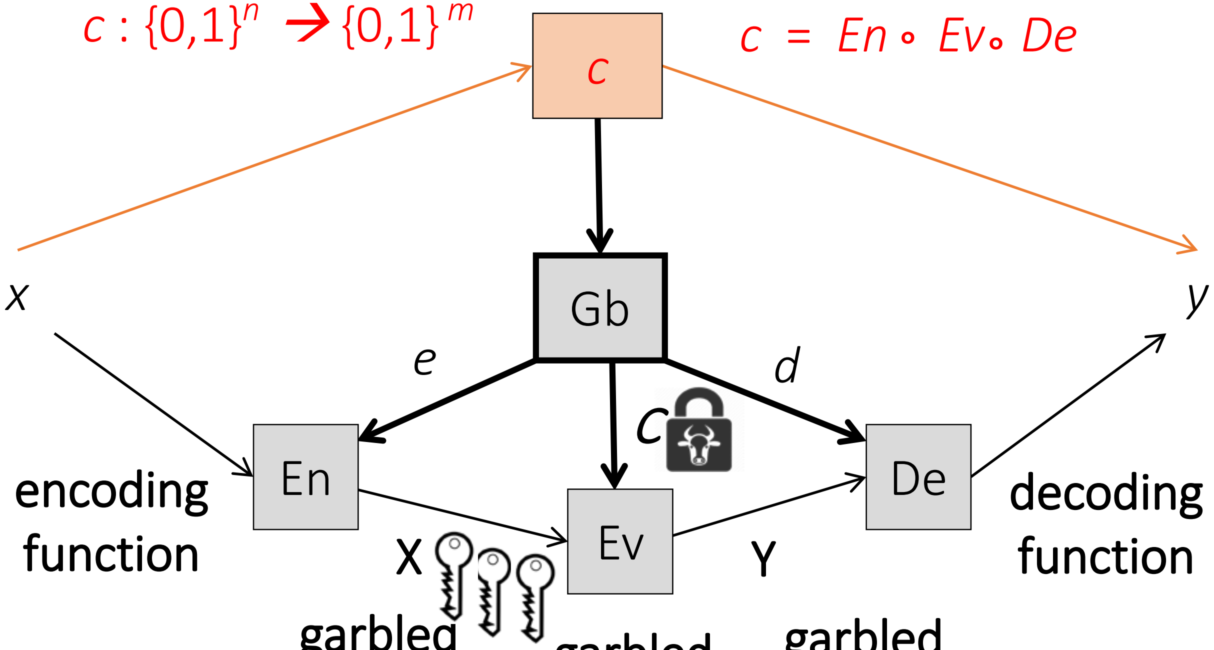
UB2: 2-rounds are sufficient for **ua** in [+ broadcast]

- Broadcast improves round complexity

UB3: 3-rounds are sufficient for **god** in [+ broadcast]

Circuit Garbling

Evaluates a circuit in encoded domain



Boolean circuit

$c : \{0,1\}^n \rightarrow \{0,1\}^m$

c

$c = En \circ Ev \circ De$

input x

Gb

e

d

C

y output

En

Ev

De

encoding function

X

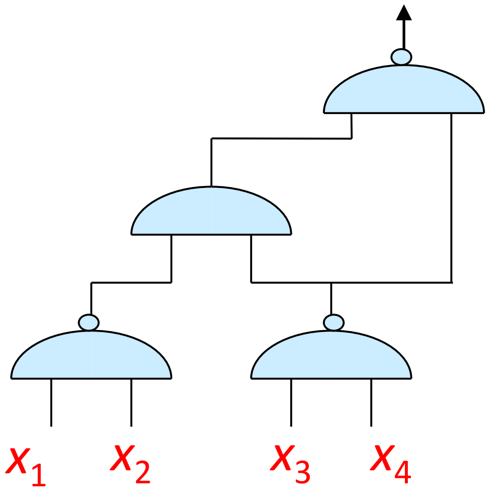
garbled input

garbled Circuit evaluation

Y

garbled output

decoding function



$c(x_1,x_2,x_3,x_4)$

Privacy: Input privacy

Privacy-free

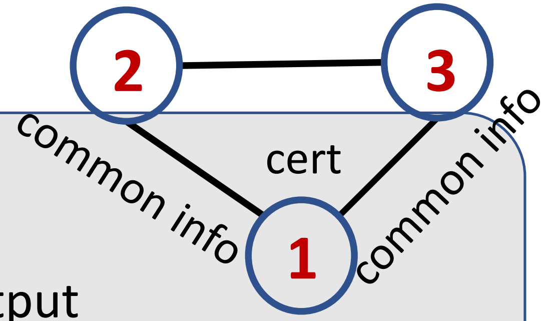
Obliviousness: Output privacy when decoding info is withheld

Authenticity: Unforgeability of **Y**

Upper Bounds: Overview and Challenges

3–round Fair protocol [-Broadcast]

- No broadcast : Conflict and confusion
- Novel mechanism : Reward honesty with certificate used to unlock output
- New primitive : Authenticated conditional disclosure of secret (Authenticated- CDS) via privacy-free garbled circuits



2–round unanimous abort [+Broadcast]

R2 private communication: Soft spot

R1 private (detect early and report in R2)

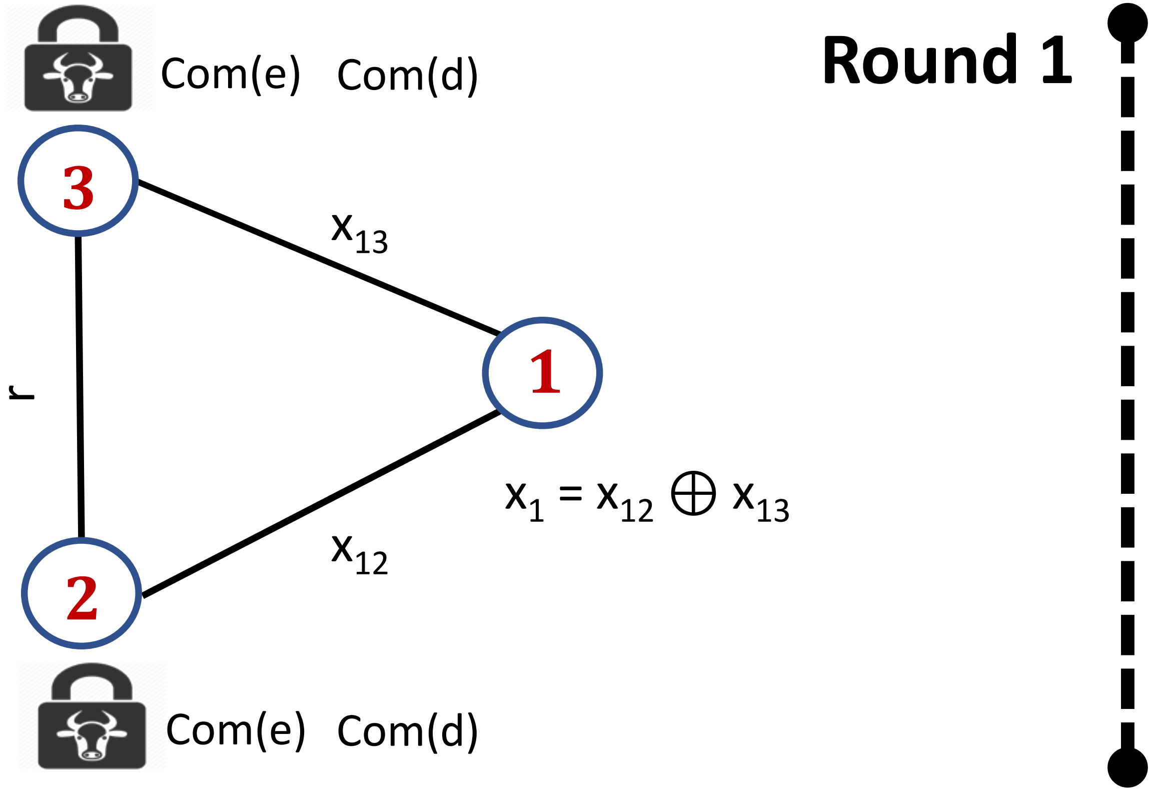
Two-part release mechanism for encoded inputs of the parties

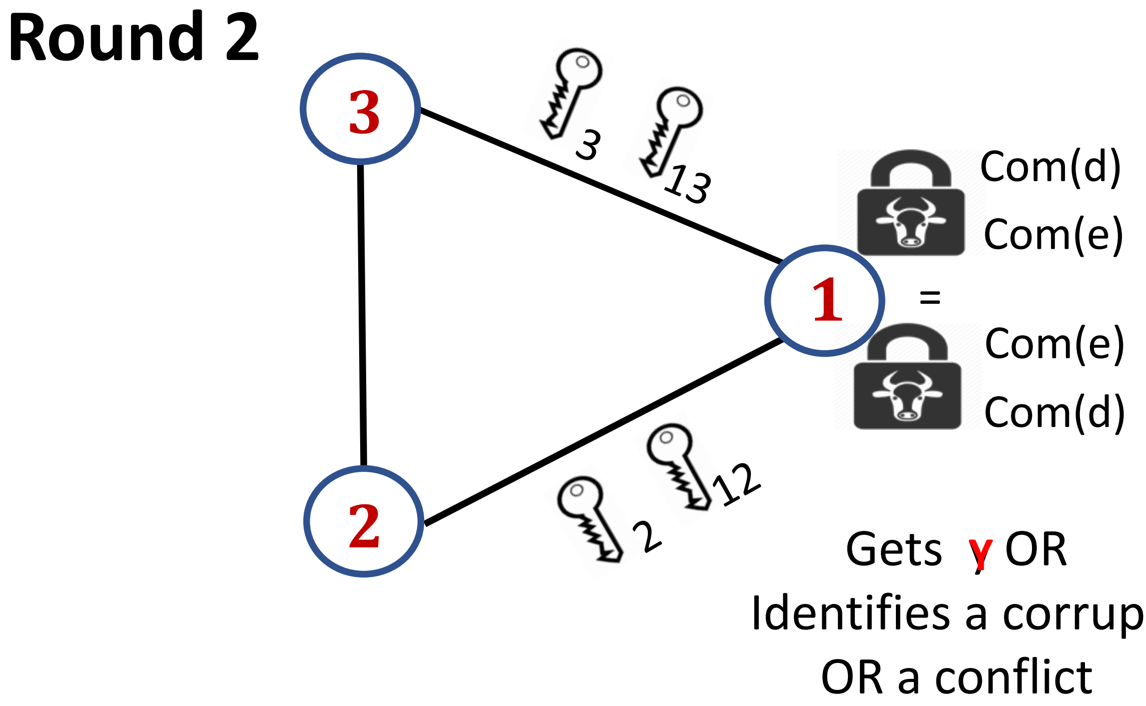
R2 broadcast (publicly detectable)

3–round Guaranteed Output Delivery [+Broadcast]

Strong identifiability : either get output / identify corrupt by second round

Fair 3PC in 3 rounds [- Broadcast]





Issue1: Revealing y can violate fairness

Sol: Use oblivious garbling and commit to d, open when well-behaved

A1: No cut-and-choose

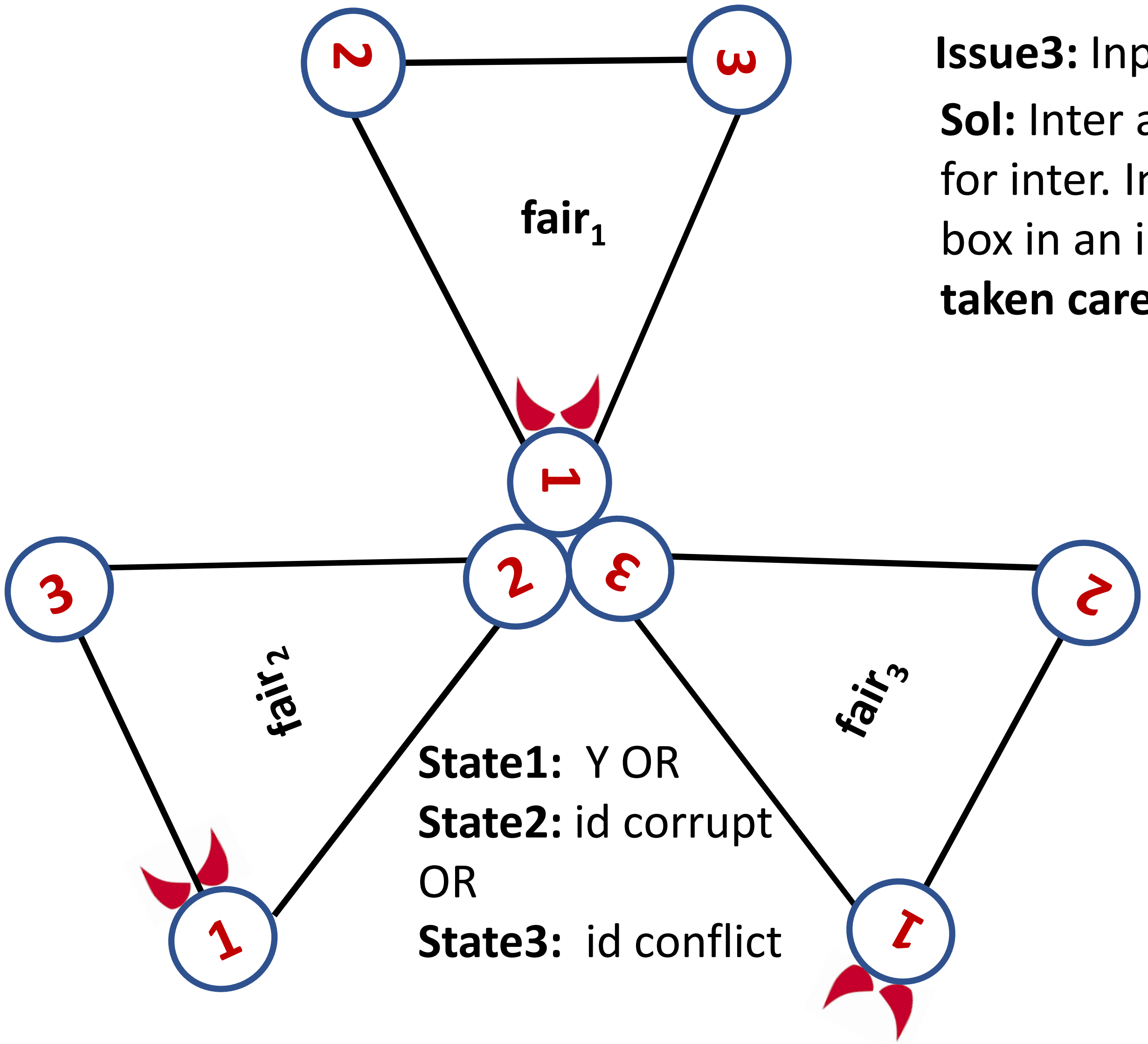
A2: No OT

Issue2: Cannot rely on the evaluator to send Y to others

Sol: Repeat this BB three times, one for each party

[MRZ15, IKKP10]

Fair 3PC in 3 rounds [- Broadcast]



State1: Y OR
State2: id corrupt
OR
State3: id conflict

Issue3: Input consistency

Sol: Inter and intra execution. Free for inter. Intra uses cheat recovery box in an intricate way. **Assume taken care!**

Fair 3PC in 3 rounds [- Broadcast]

Issue4: Corrupt always gets Y. Can keep one happy and other **confused**. Get decoding info from happy and get output. How to get fairness ?

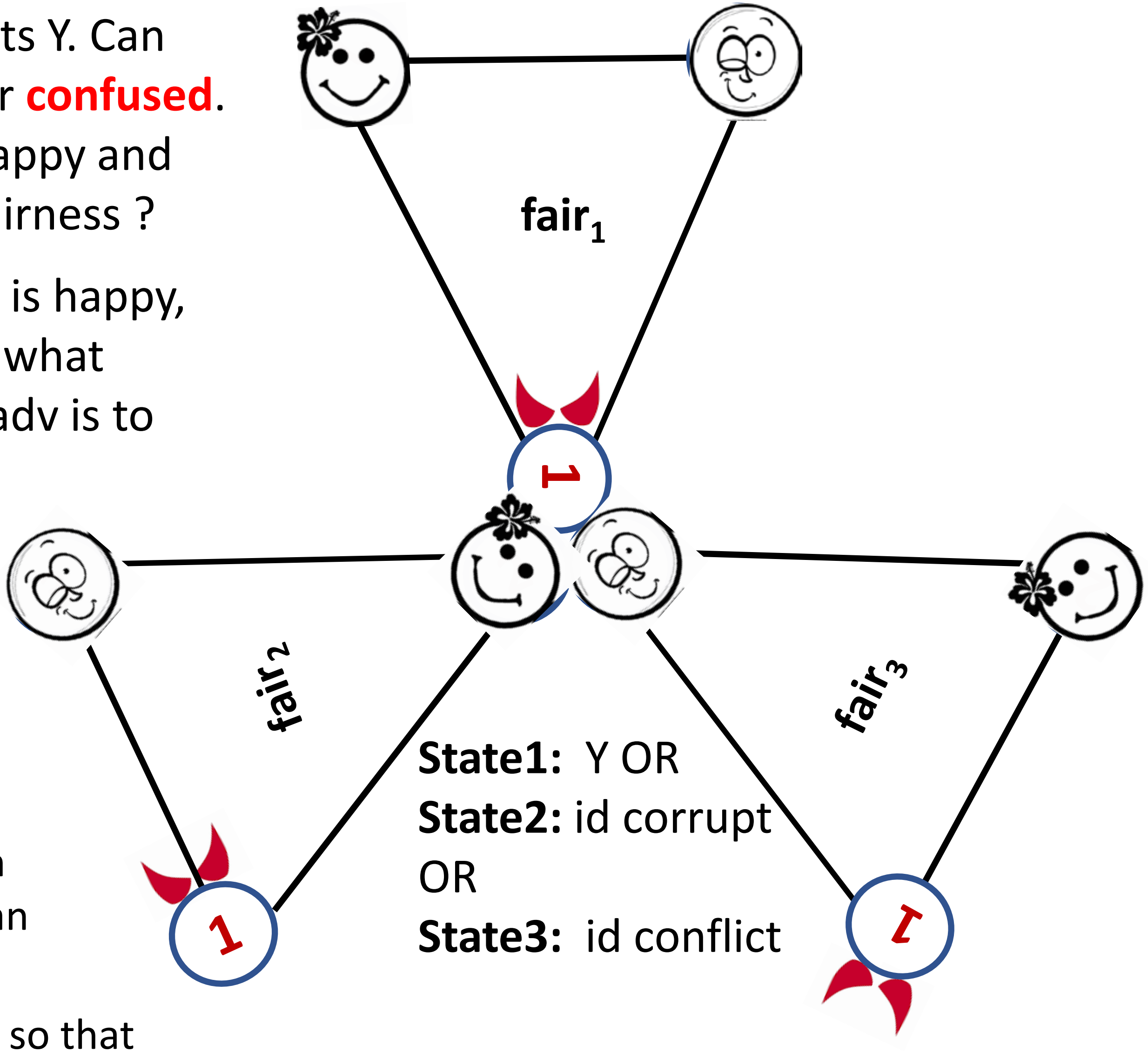
Sol: (1) If an honest party is happy, all gets output no matter what
(2) Only way to get **d** for adv is to keep an honest happy

A **confused honest** party can identify the honest and use her Y to compute y

Certificate proves honesty

A **confused honest** party can deliver d in a way that only an honest happy party decrypt.

Certificate carries d securely so that only legitimate holder can open

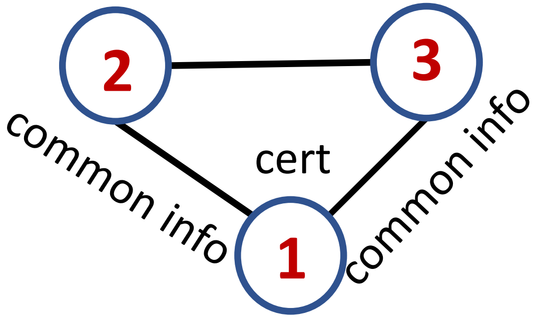


State1: Y OR
State2: id corrupt
OR
State3: id conflict

Confusion because of disagreement on common message such as

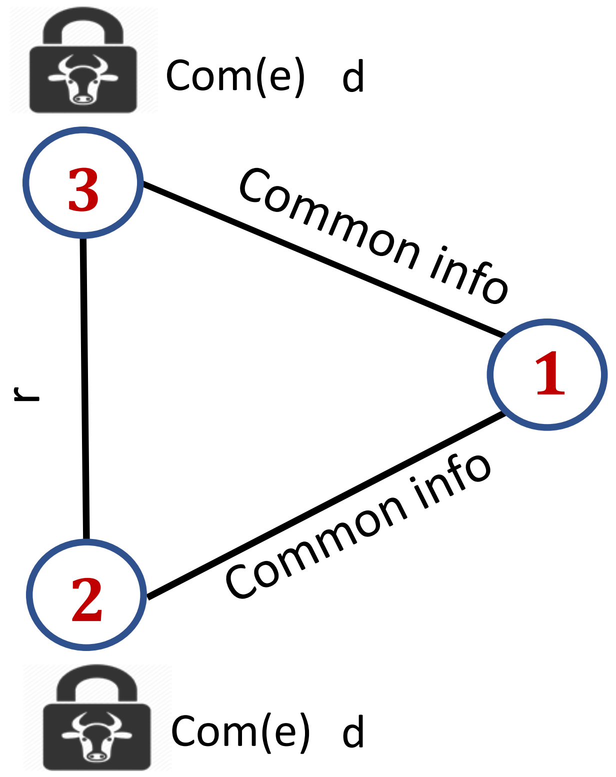
Sol: Reward a **certificate** for emulating a correct broadcast for common message as a sender.

Via authenticated 3-party CDS for equality!



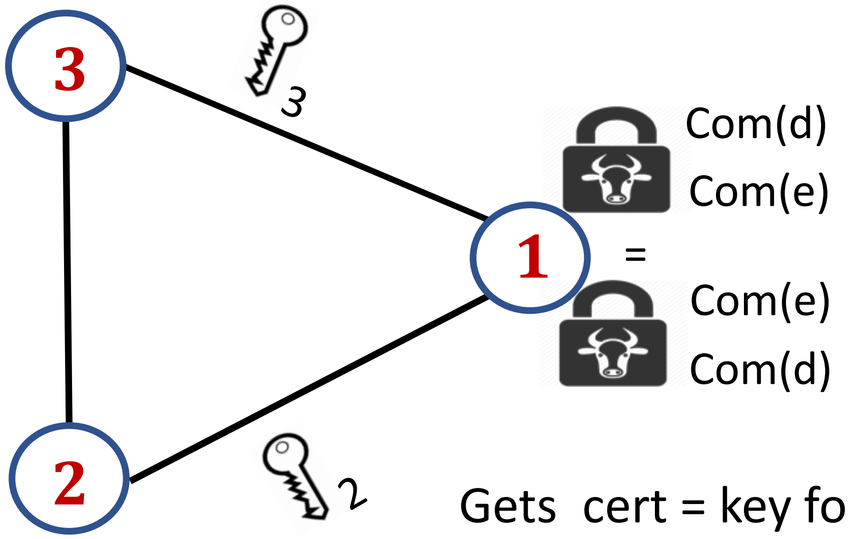
receiver receives a correct certificate or identifies a corrupt or conflict

Fair 3PC in 3 rounds [- Broadcast]



Round 1

Round 2

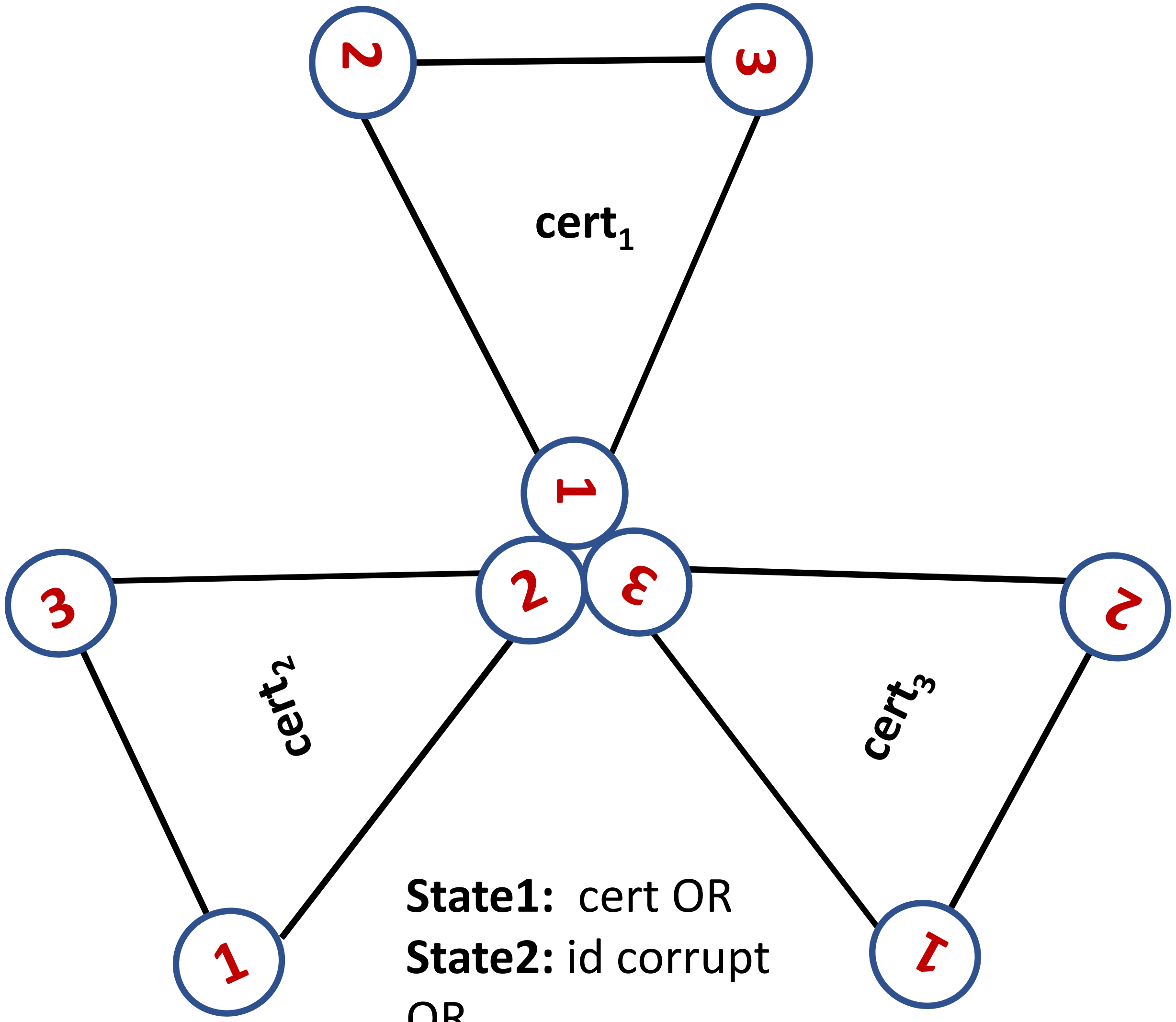


Gets cert = key for
1 OR
Identifies a corrupt
OR a conflict

Equality checking circuit

Privacy-free garbling

Fair 3PC in 3 rounds [- Broadcast]

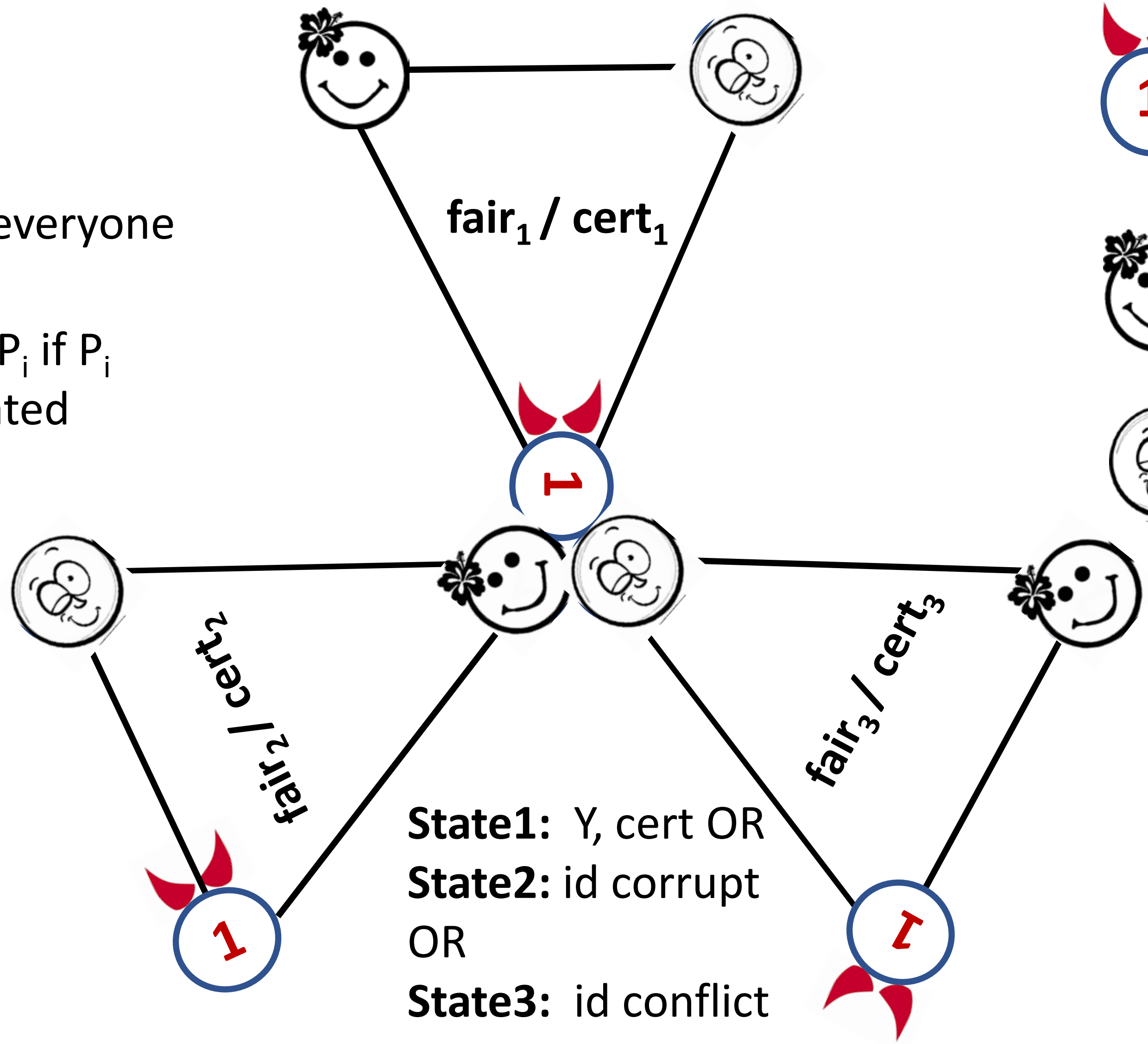


State1: cert OR
State2: id corrupt
OR
State3: id conflict

Fair 3PC in 3 rounds [- Broadcast]

Send Y, cert, d to everyone

Send Enc_{cert}(d) to P_i if P_i common info created confusion



fair_1 / cert_1
1
fair_2 / cert_2
1
fair_3 / cert_3
1

State1: Y, cert OR
State2: id corrupt
OR
State3: id conflict

1 Can get output only by keeping a party happy

Recovers d via cert and gets y

Cert proves 2's honesty, takes his Y and compute y

Lower Bounds (3 rounds necessary for **ua [-broadcast]** and for **fn [+broadcast]**)

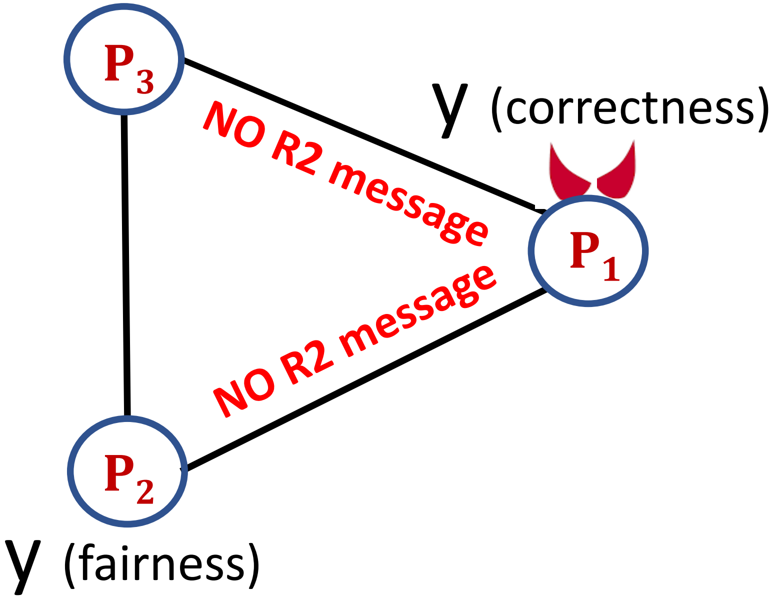
Pick a special function
Assume 2-round protocol exists

$$f(x_1, x_2, x_3) = \begin{cases} 1 & \text{if } x_2 = x_3 = 1 \\ 0 & \text{otherwise} \end{cases}$$

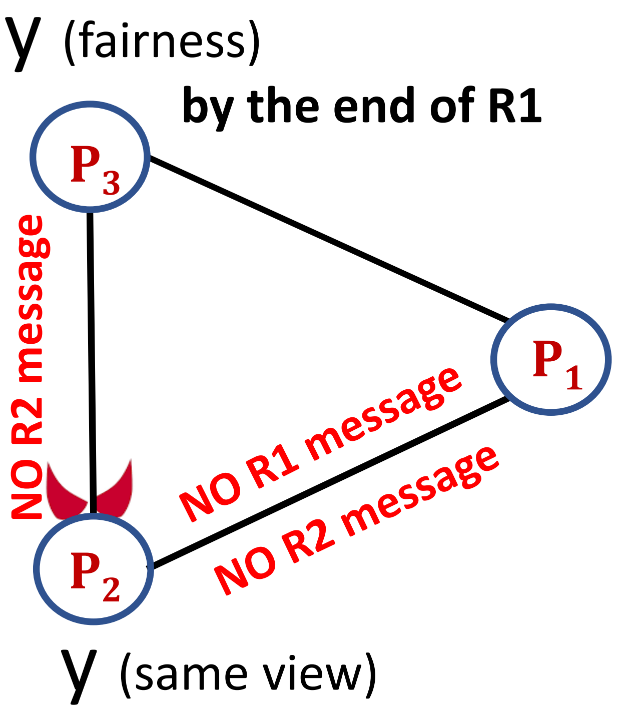
→ Define a sequence of hybrids (under diff adv strategies)

- within hybrid use fn/ua to conclude why a party should output
- Across hybrids use view equality

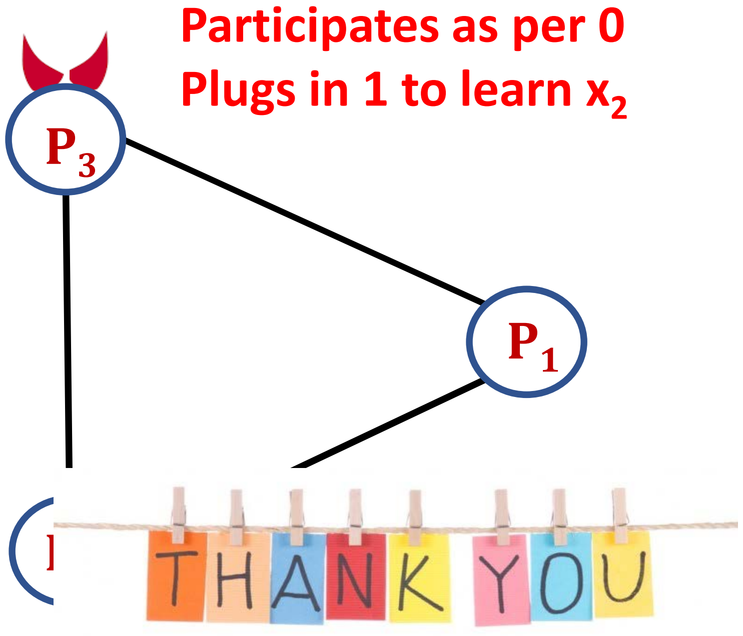
→ No privacy!



Σ_1

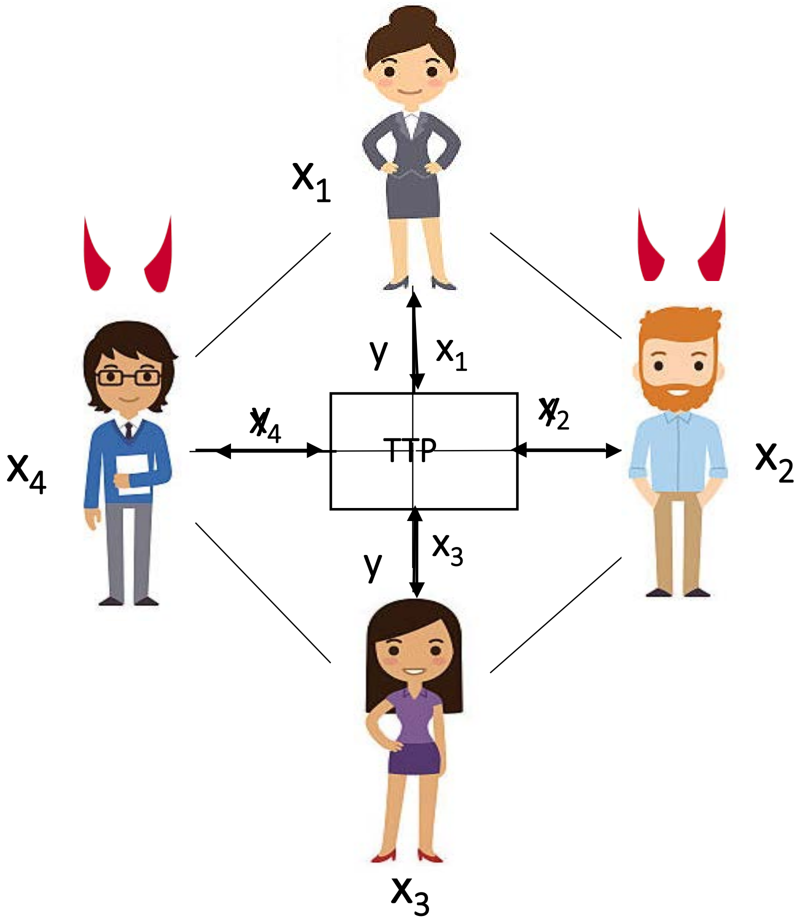


Σ_2



Σ_3

MPC



Setup:

- n parties $P_1,....,P_n$; t are corrupted by a centralized adv
- P_i has **private** input x_i
- A common n-input function $f(x_1,x_2,..x_n)$

Goals:

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

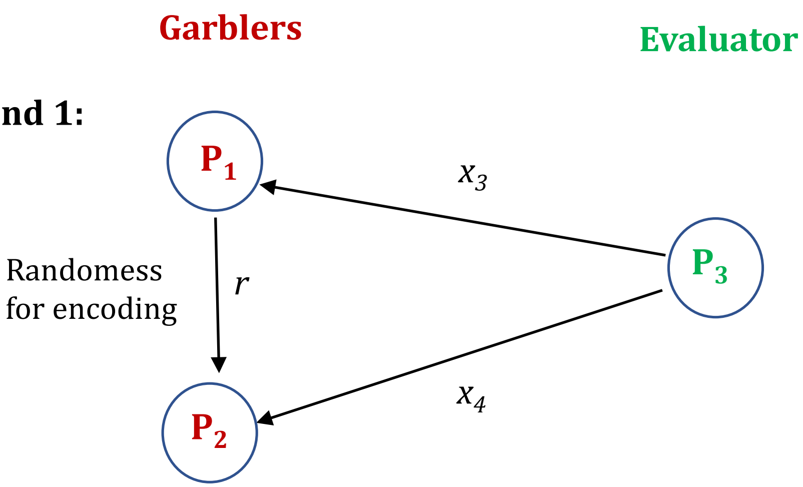
Challenge:

NO TTP

MPC: interactive protocol that emulates TTP

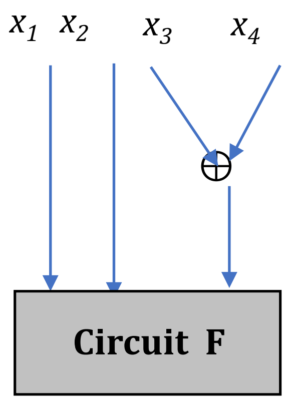
Extension of garbling for 3 PC

Round 1:

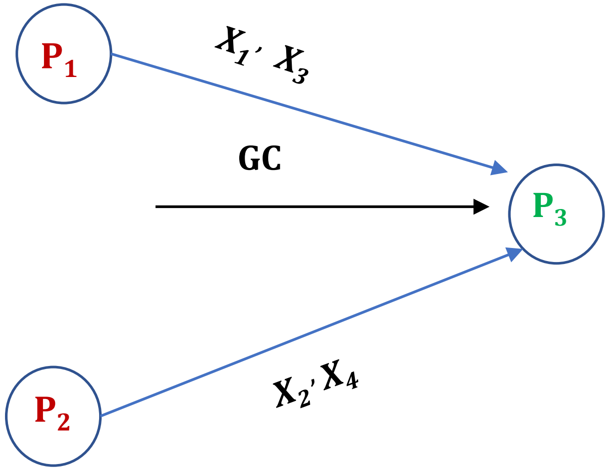


$x'_3 = x_3 \oplus x_4$

$$F(x_1, x_2, x'_3) = f(x_1, x_2, x_3 \oplus x_4)$$



Round 2 :

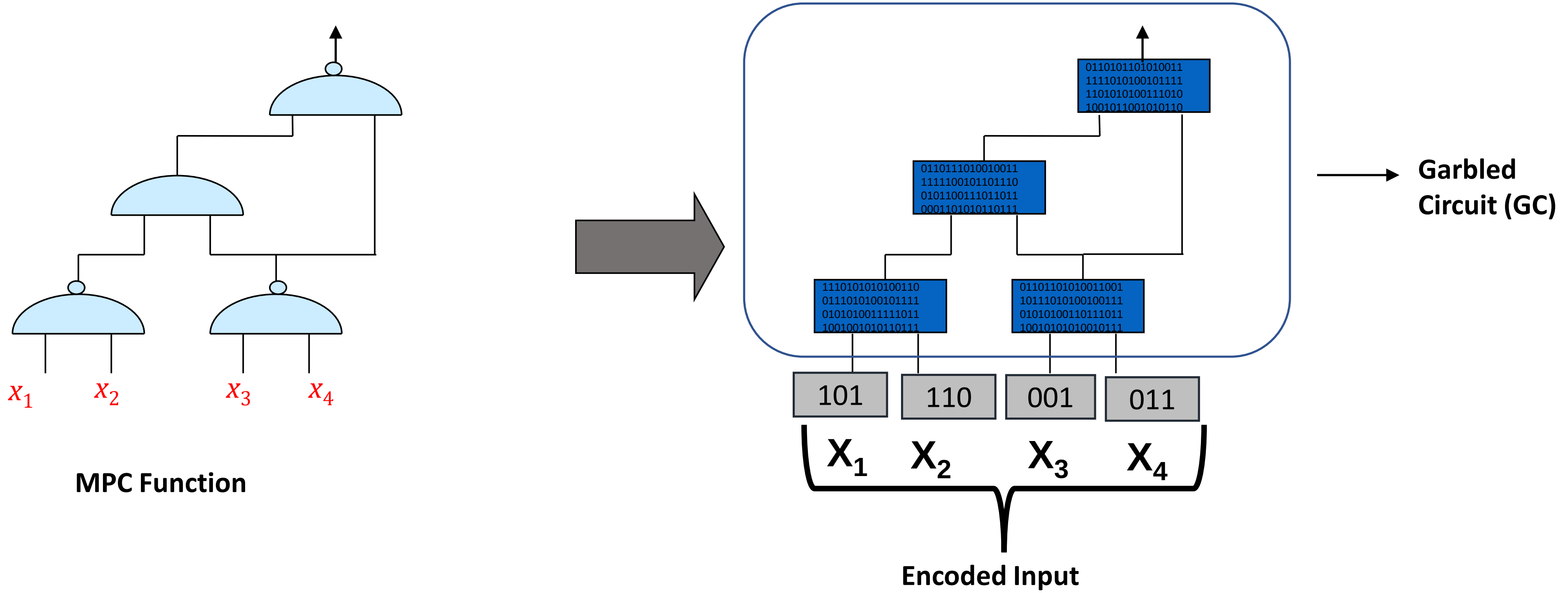


Honest Majority: avoided public-key

Only P_3 gets output.

How to design 2-round protocol?

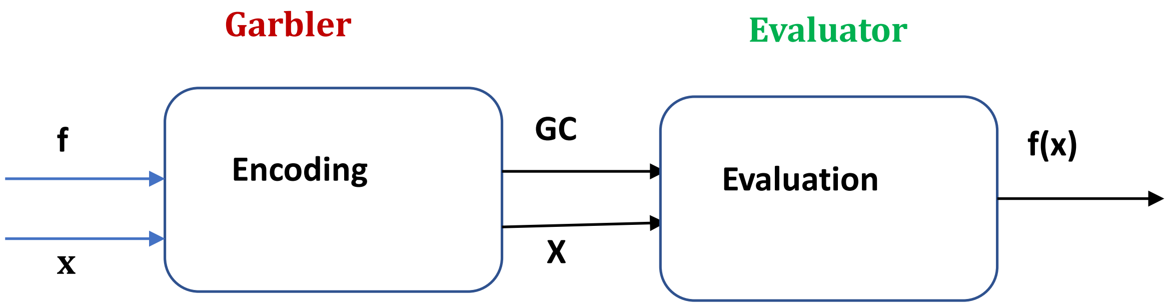
Garbling : Randomized Encoding



MPC Function

Garbled Circuit (GC)

Encoded Input



Garbler

Evaluator

f

x

Encoding

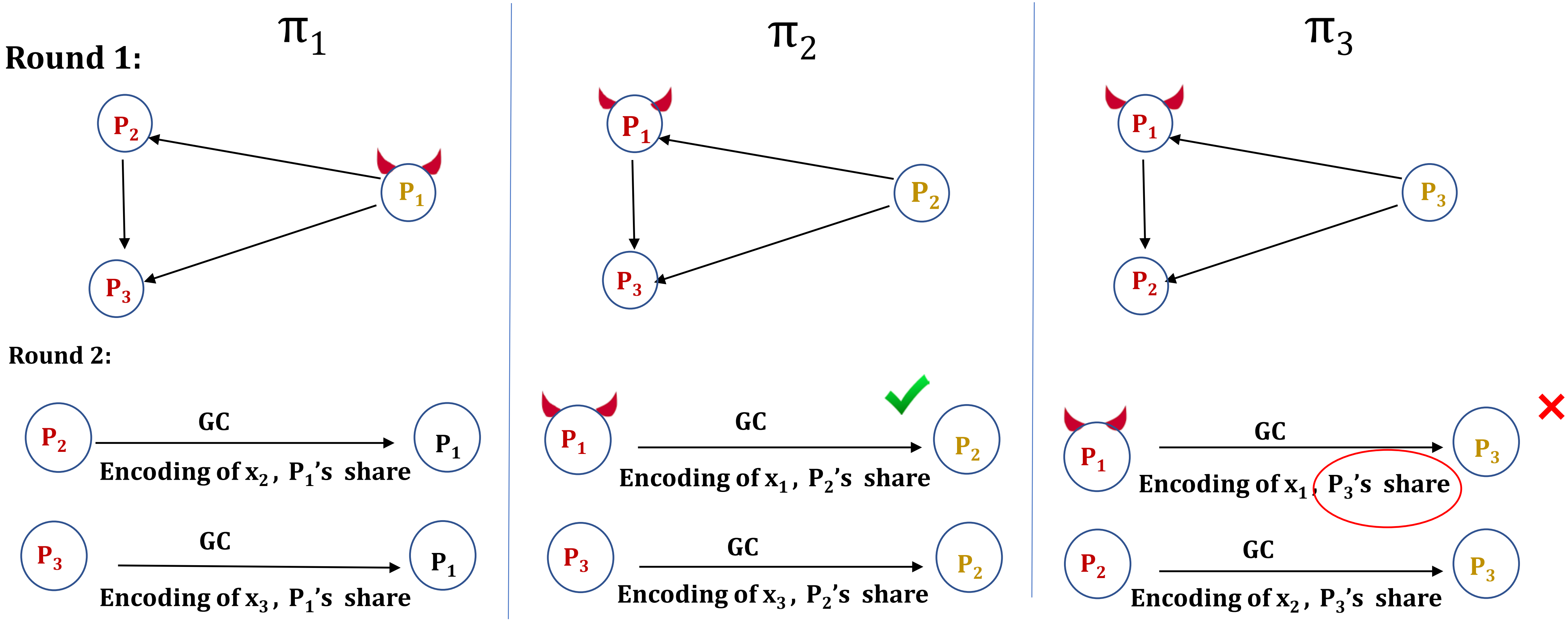
GC

X

Evaluation

f(x)

Attempt : 2-round 3PC with unanimous abort



π_1

Round 1:

P_2 P_1 P_3

Round 2:

P_2 → P_1: GC, Encoding of x_2 , P_1's share

P_3 → P_1: GC, Encoding of x_3 , P_1's share

π_2

P_1 P_2 P_3

P_1 → P_2: GC, Encoding of x_1 , P_2's share ✓

P_3 → P_2: GC, Encoding of x_3 , P_2's share

π_3

P_1 P_3 P_2

P_1 → P_3: GC, Encoding of x_1 , P_3's share ✗

P_2 → P_3: GC, Encoding of x_2 , P_3's share

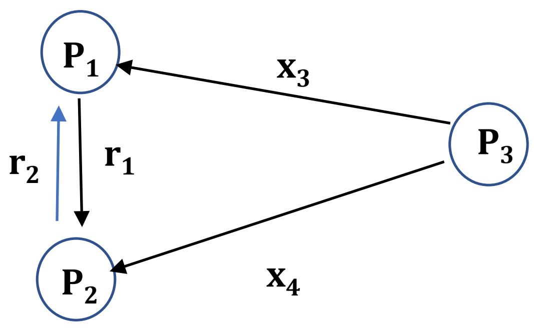
Honest P_2 gets output but P_3 does not.

Unanimous abort violated!

Takeaway: Honest garbler must be informed if honest evaluator unable to get output.

Partial Solution

Round 1: `



Wrong GC_1?

P_1 — **Broadcast GC_1** ✓

Send X_1 for GC_1 — Wrong X_1 ?

P_2 — **Broadcast GC_2**

Send X_2 for GC_2

P_3

Round 2 :

P_1 — Broadcasts **abort / agree**

If agree : Send X_3 for GC_1 and X_1 , X_3 for GC_2

No way to handle!

P_2 — Broadcasts **abort / agree**

If agree : Send X_4 for GC_2 and X_2 , X_4 for GC_1 ✓

P_3

Broadcast **abort** if X_1 of **GC_1** / X_2 of **GC_2** invalid

No Abort =>

- **GC_1, X_1 for GC_1 correct**
- **Got X_2, X_4 for GC_1**

Only X_3 missing!

Cannot send before Round 2

Cannot use broadcast

Rule : If any party broadcasts "abort", all honest parties abort

Private communication in Round 2 – only option to send X_3??

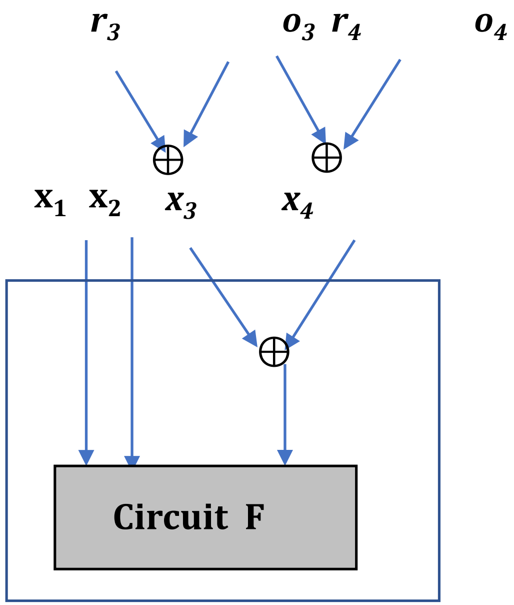
Building the solution

- **What we know:** Handle misbehavior
  - Type 1: Private info sent in Round 1
  - Type 2: Broadcast info sent in Round 2

Idea : Combine both!

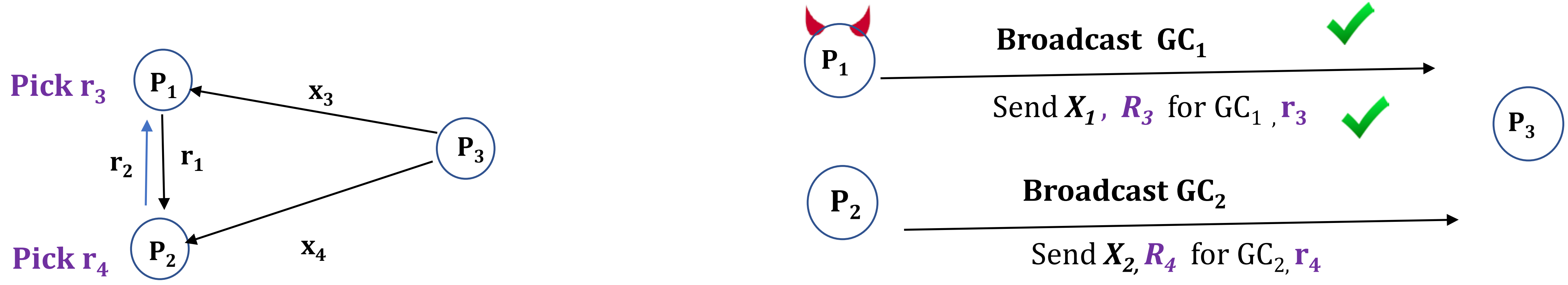
- **Idea :** Evaluator's share broken down as :
  - random input picked by garbler
  - offset of actual share and random input

- **Solution: Two – part release mechanism**
  - **Private** release of encoding of random inputs
  - **Public** release of encoding of offset



Completing the picture

Round 1: **(Private Release of encoded random input)**



Broadcast GC_1

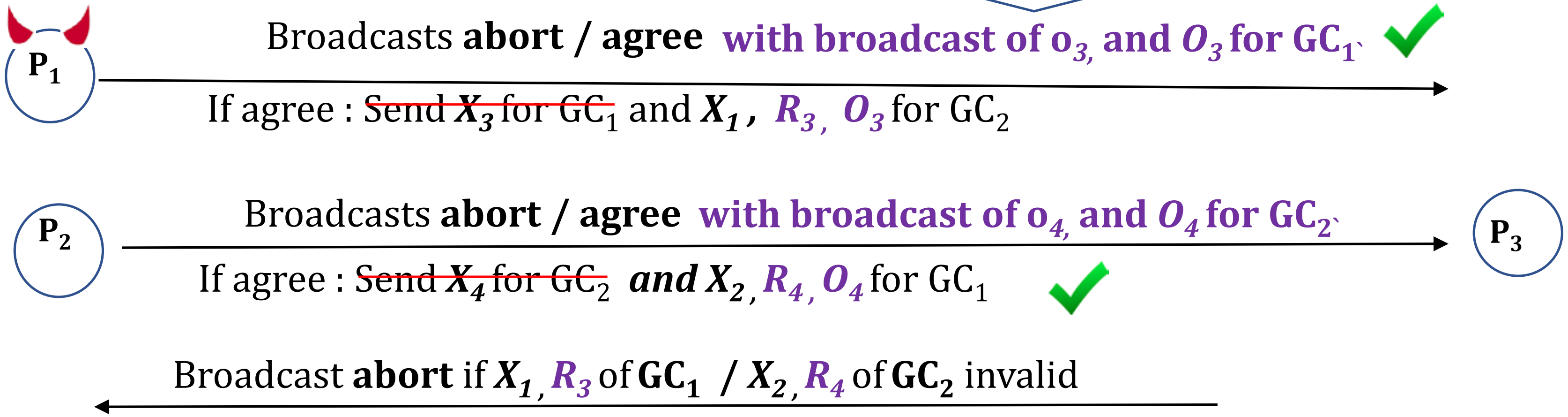
Send X_1, R_3 for GC_1, r_3

Broadcast GC_2

Send X_2, R_4 for GC_2, r_4

Round 2 : **(Public release of encoded offset)**

Safe! x_3 protected by r_3



Broadcasts **abort / agree** **with broadcast of o_3, and O_3 for GC_1`**

If agree : ~~Send X_3 for GC_1~~ and X_1, R_3, O_3 for GC_2

Broadcasts **abort / agree** **with broadcast of o_4, and O_4 for GC_2`**

If agree : ~~Send X_4 for GC_2~~ *and* X_2, R_4, O_4 for GC_1

Broadcast **abort** if X_1, R_3 of **GC_1** / X_2, R_4 of **GC_2** invalid

$o_3 = r_3 \oplus x_3$

$o_4 = r_4 \oplus x_4$

No Abort => correctness of

- **GC_1, X_1, R_3 for GC_1**
- **O_3 for GC_1**
- **X_2, R_4, O_4 for GC_1**

Claim: No Abort => P_3 gets output!