QUORUMS QUICKEN QUERIES: EFFICIENT ASYNCHRONOUS SECURE MULTIPARTY COMPUTATION

Mahnush Movahedi

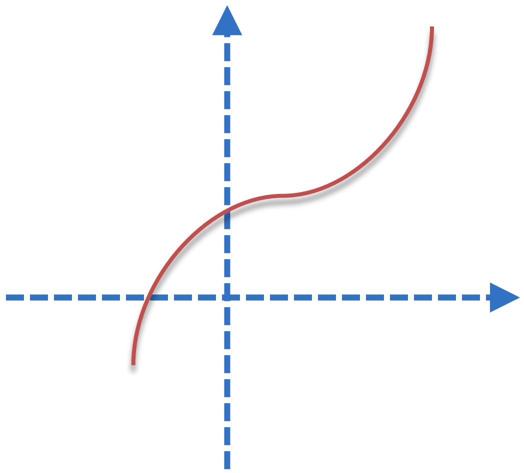
with Jared Saia, Valerie King, Varsha Dani

University of New Mexico and University of Victoria

February 2014

Multi-Party Computation

- Given
  - n parties
  - Each party has a private input
  - Function f over n inputs
- Goals
  - Correctness
  - Privacy



Our Model

- Communication model
  - Pairwise private channels
  - Asynchronous
- Adversary
  - Static
  - Unbounded
  - t $< n/8$ are malicious (bad)

Asynchronous Model

- The adversary can control latency of channels
  - Can arbitrarily delay messages
  - Cannot delete messages
- MPC simulates a trusted third party
  - Waits for $n - t$ input,
  - Computes f,
  - Sends the output to every party.

What is the difference?

- Count inputs that are received
- Parties cannot wait for all messages
  - Decrease number of bad parties
  - Wait until sufficient number of the same message is received.

First Step

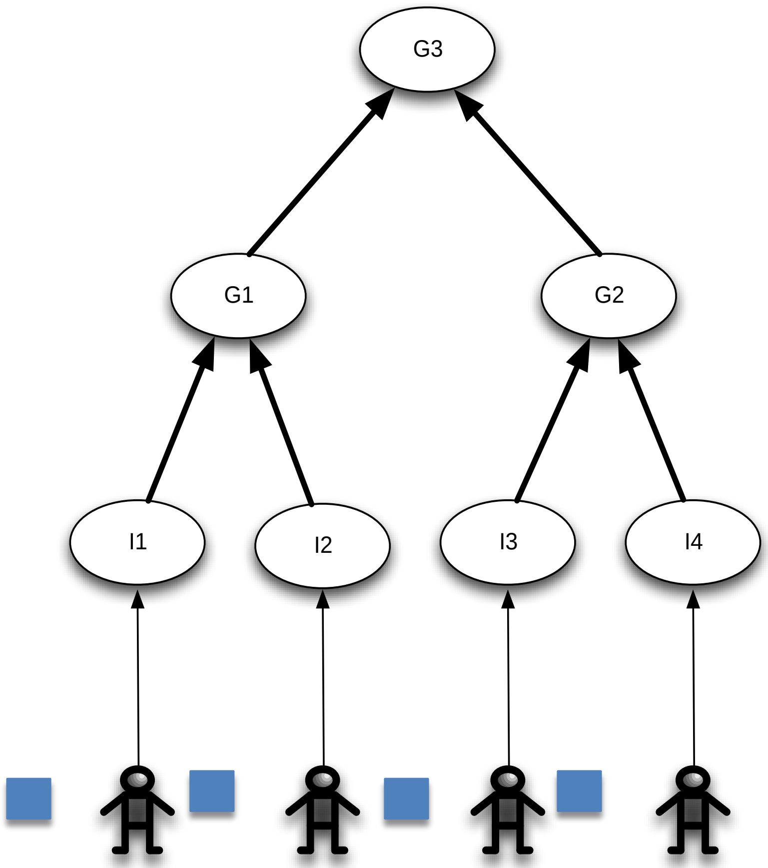
Count ready inputs

Threshold Counting

- Problem
  - n good parties with a bit initially set to 0.
  - At least τ parties eventually set their bits to 1.
  - Goal: parties learn when at least τ bits are 1.
- Solution: τ-Counter
  - Bits sent/received per party: $O(\log n)$
  - Computation per party: $O(\log n)$
  - Total latency: $O(\log n)$

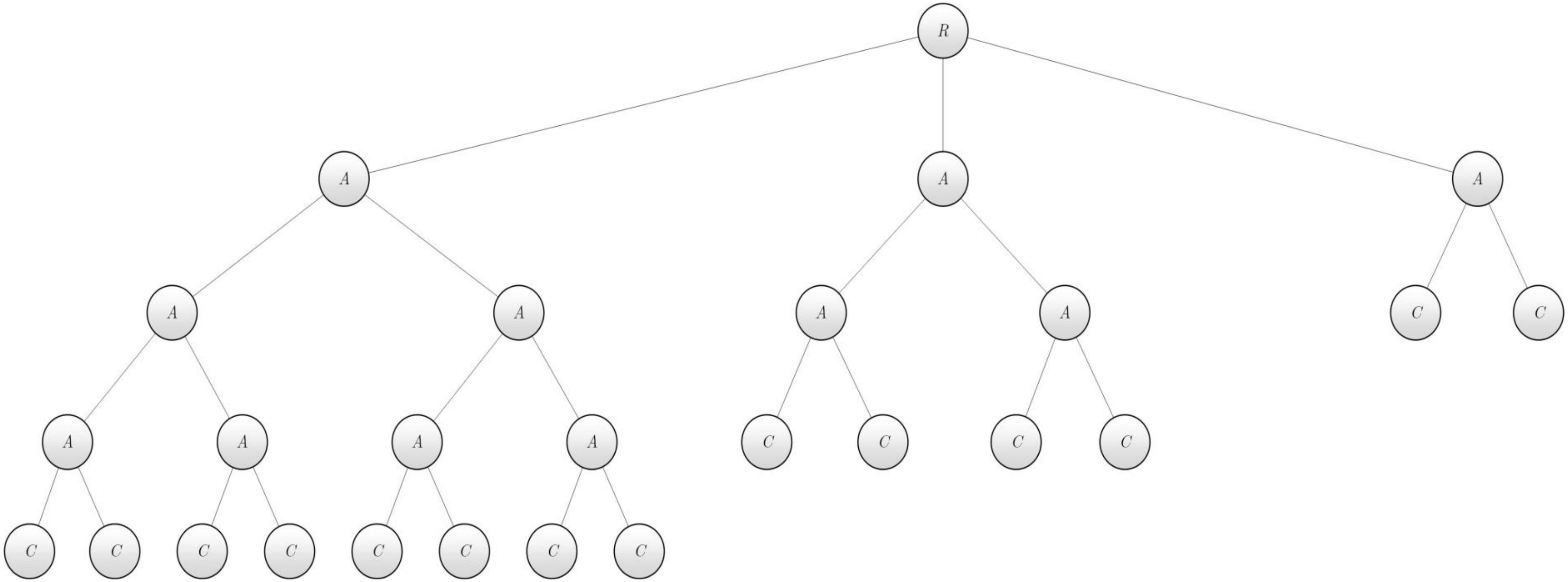
Approach

- Naïve approach: binary tree
  - A message is sent up for each new input
  - Problem: Load balancing
- τ-Counter
  - Forwards to random nodes
  - Collects inputs and aggregates them before forwarding

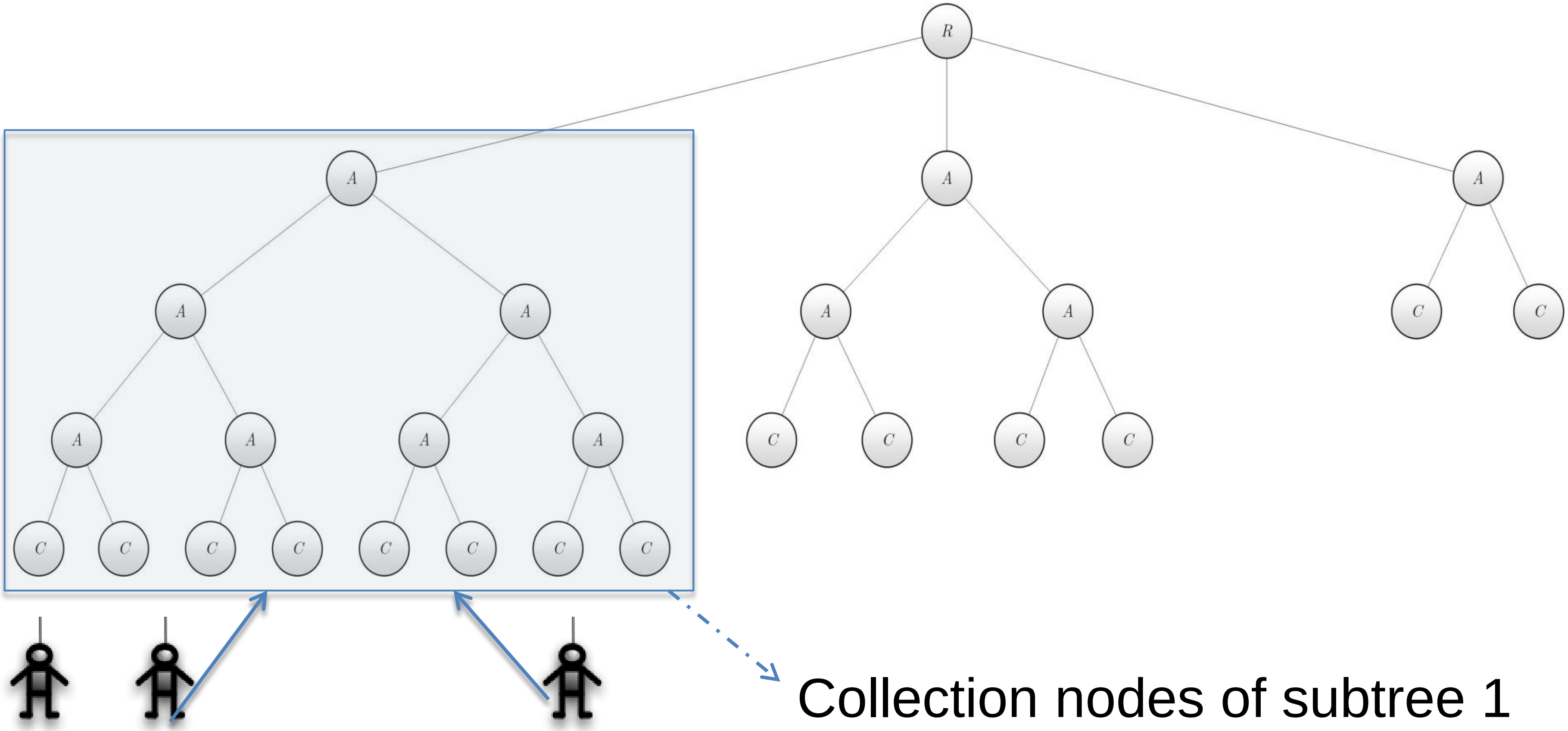


τ-Counter Data Structure

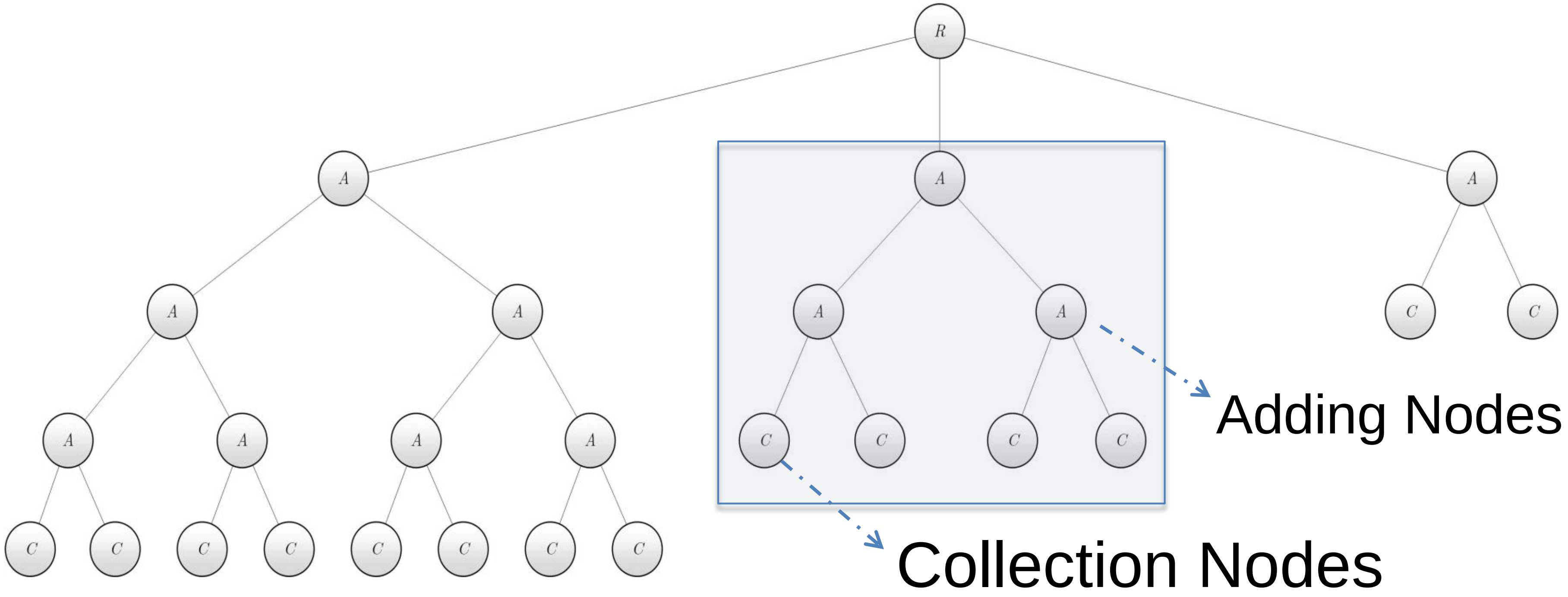
- Root node has $O(\log n)$ children.



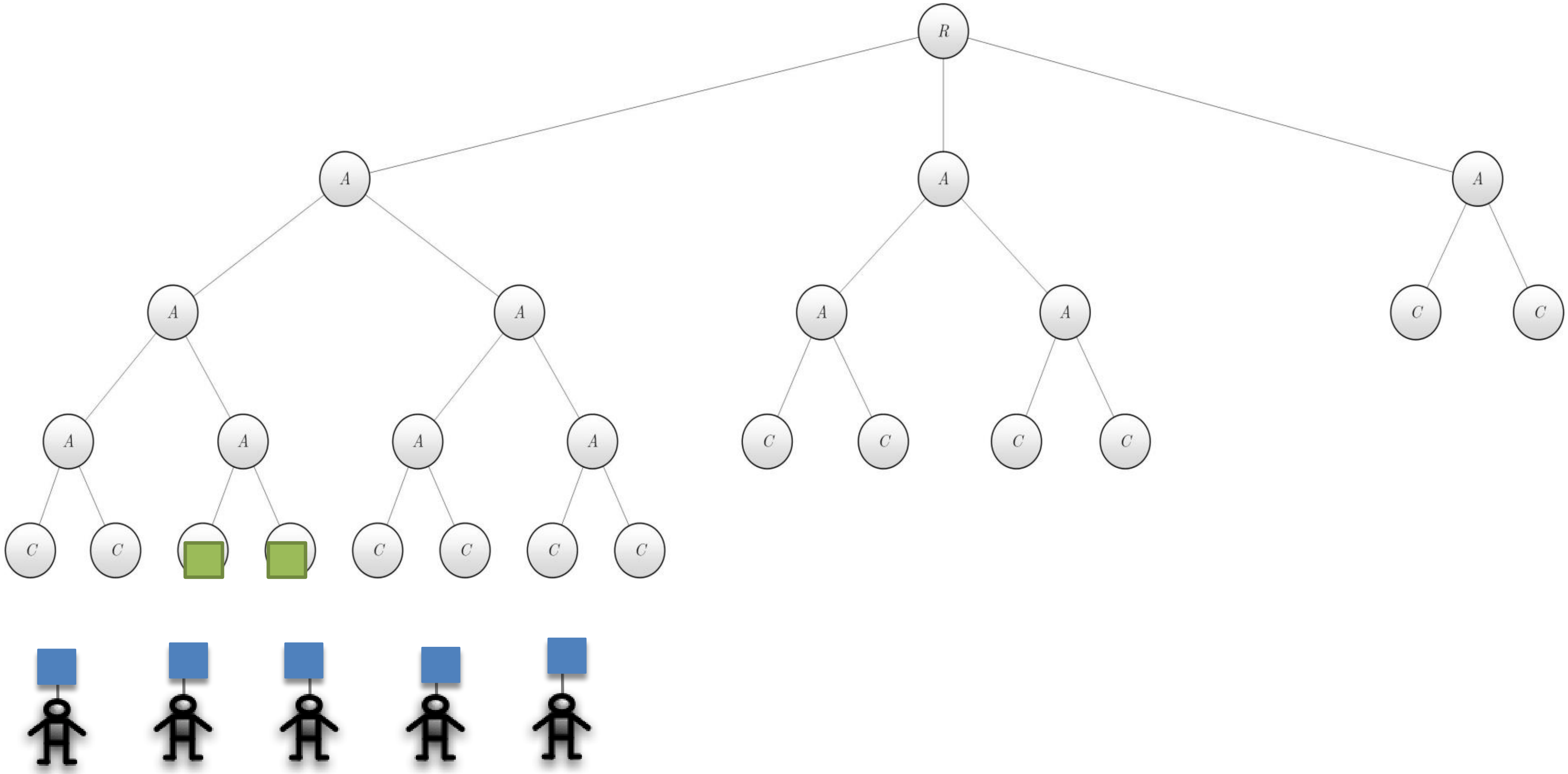
τ-Counter Up Stage



τ-Counter Up Stage



τ-Counter



Next Step

Our MPC Construction

Previous Work

- [BGW88], [CR93], [CDD99], [HM01], [BH06], [AIK10], …
- Assume the circuit has m gates
  - Each party sends $O(mn)$ messages
  - Each party performs $O(mn)$ computation

Our Contribution

- Improve computation and communication costs
- In average, each party
  - Sends $\tilde{O}\left(\frac{m}{n}\right)$ bits
  - Performs $\tilde{O}\left(\frac{m}{n}\right)$ computation
- We solve MPC *w.h.p.* meaning
  - $1 - O(1/n^c)$ for any fixed c

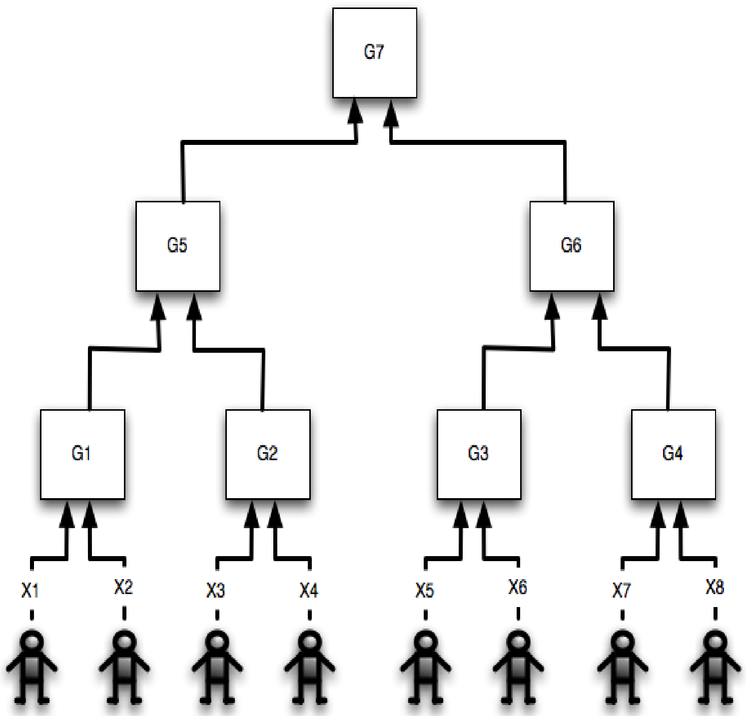
Algorithm Overview

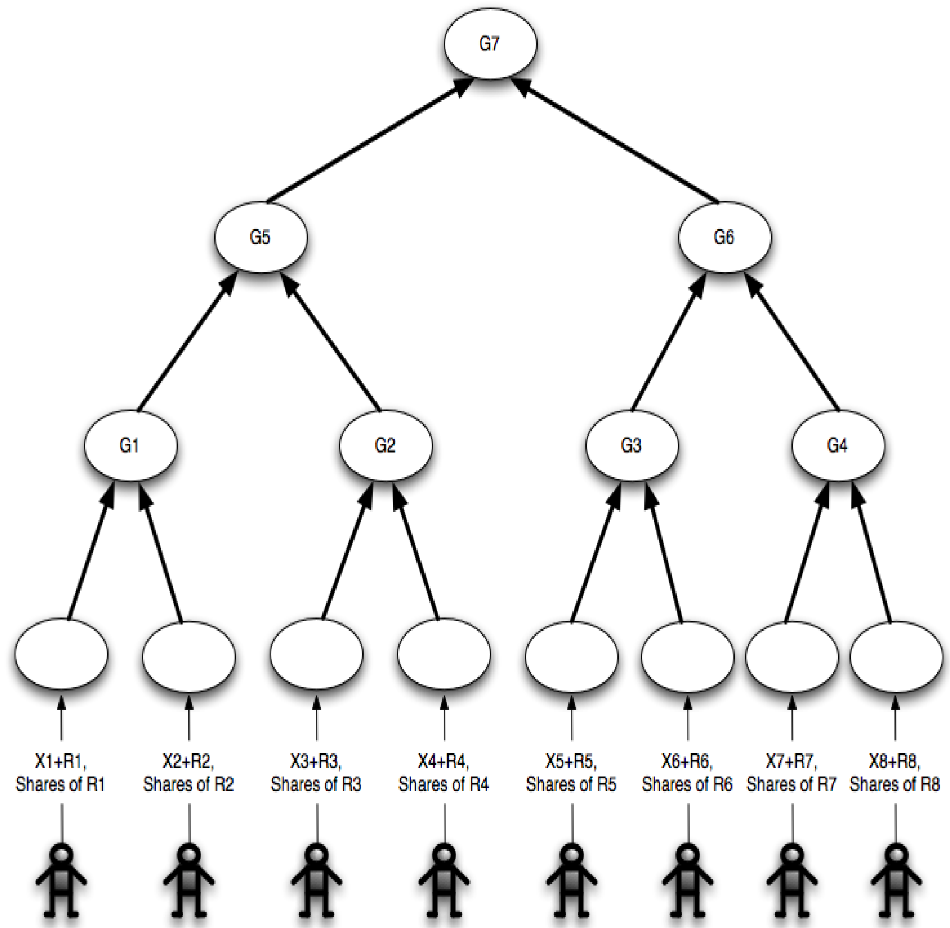
- Use quorums
  - Has $\theta(\log n)$ parties, $< 1/8$ fraction are bad
  - Used for input counting
  - Each gate is computed by a quorum
- Preserve privacy
  - Mask gate inputs and output with random values
  - Random number are known collectively via verifiable secret sharing
- Asynchronous issues
  - τ-Counter counts the number of ready inputs

Tools

- Quorum building
  - Parties agree on n quorums *w.h.p* [BGH13]
- Preserve privacy and computation
  - Verifiable secret sharing of [CR93]
  - MPC in quorums [CR93]

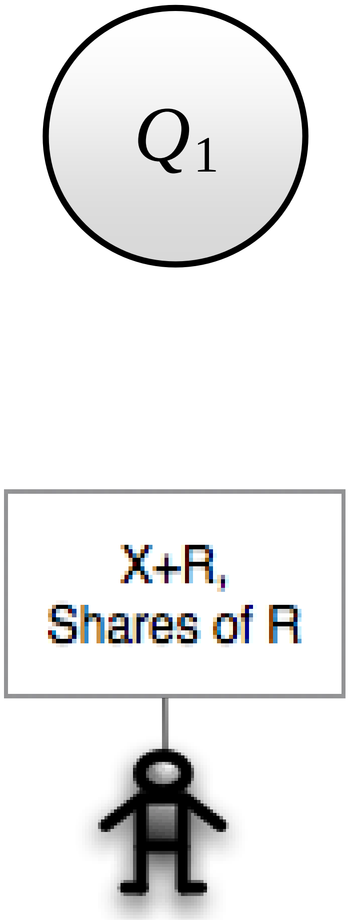
Circuit


G7
G5
G6
G1
G2
G3
G4
X1
X2
X3
X4
X5
X6
X7
X8

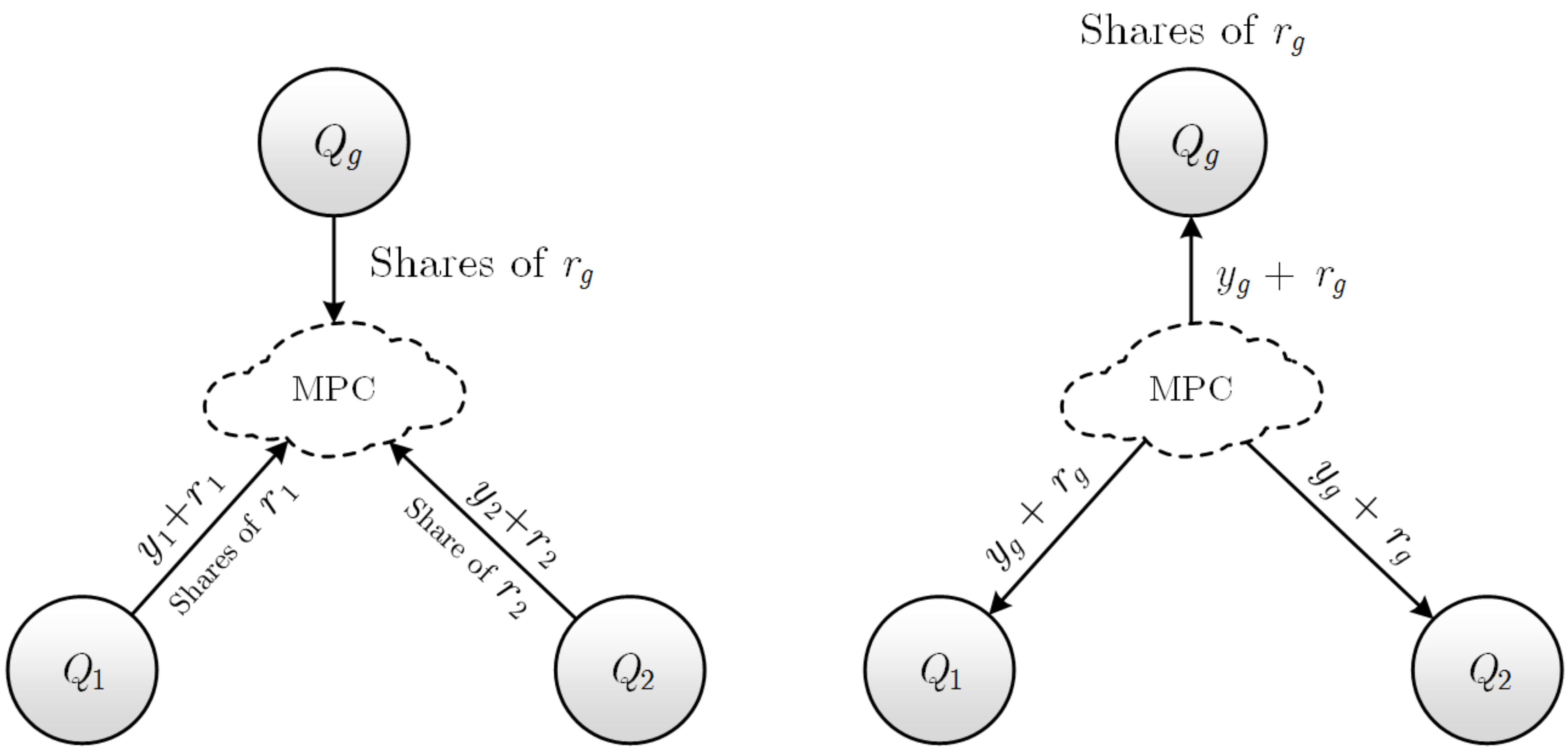

G7
G5
G6
G1
G2
G3
G4
X1+R1, Shares of R1
X2+R2, Shares of R2
X3+R3, Shares of R3
X4+R4, Shares of R4
X5+R5, Shares of R5
X6+R6, Shares of R6
X7+R7, Shares of R7
X8+R8, Shares of R8

The Algorithm: Input commitment

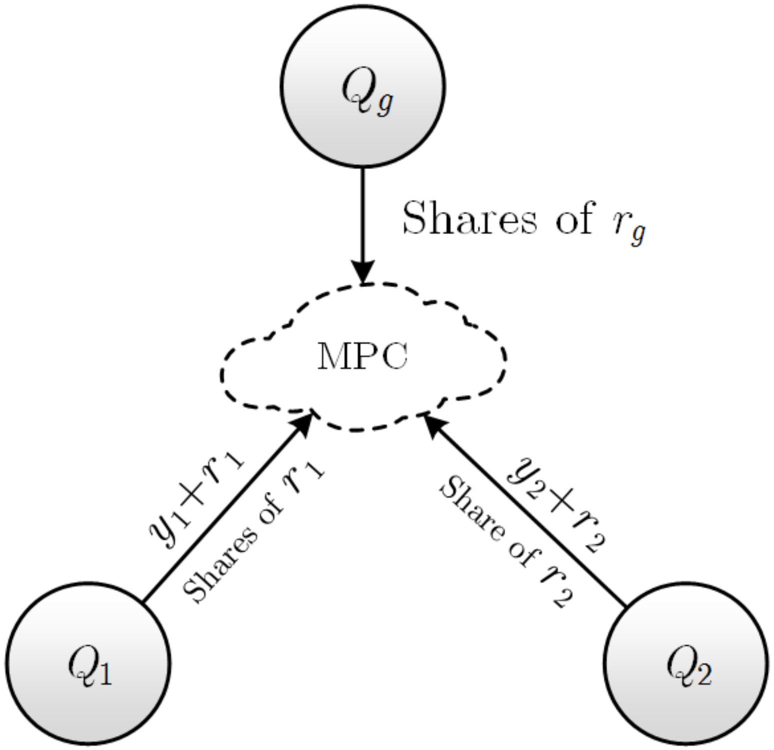
- Quorums form a τ-Counter
- Each party VSS a random mask to an input quorum
- Each party sends its masked input to an input quorum



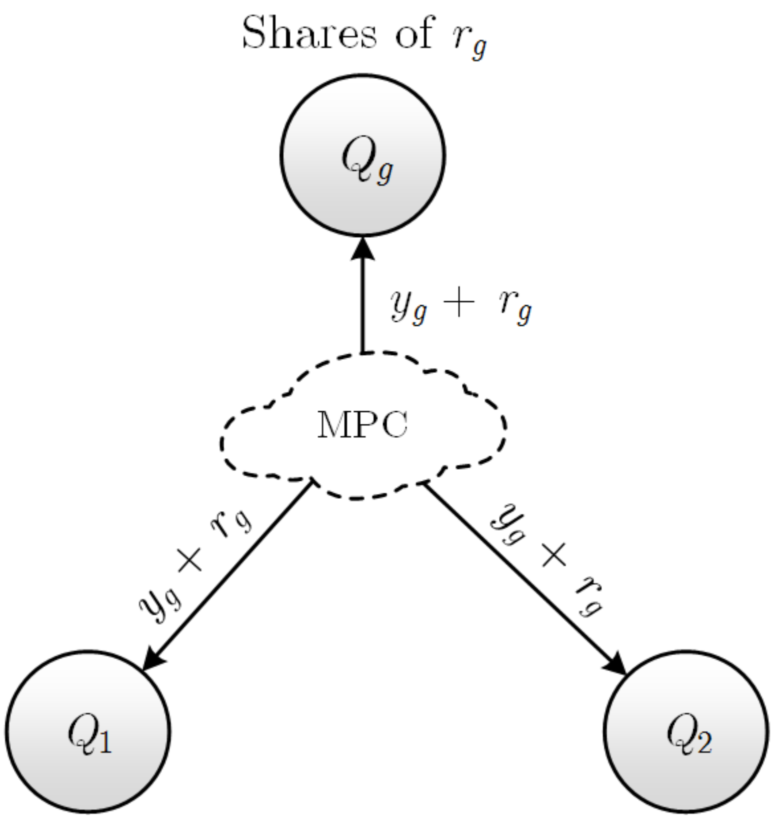
The Algorithm: Computation



Conclusion



Q_g
Shares of r_g
MPC
$y_1 + r_1$
Shares of r_1
$y_2 + r_2$
Share of r_2
Q_1
Q_2



Shares of r_g
Q_g
$y_g + r_g$
MPC
$y_g + r_g$
$y_g + r_g$
Q_1
Q_2

More Open Problem

- Designing scalable interactive computation
- Scalable interactive Coding for Multiparty Protocols
- Server based MPC based on quorums

Thank you!