Communication Locality in Secure Multi-Party Computation

How to Run Sublinear Algorithms in a Distributed Setting

Elette Boyle
MIT

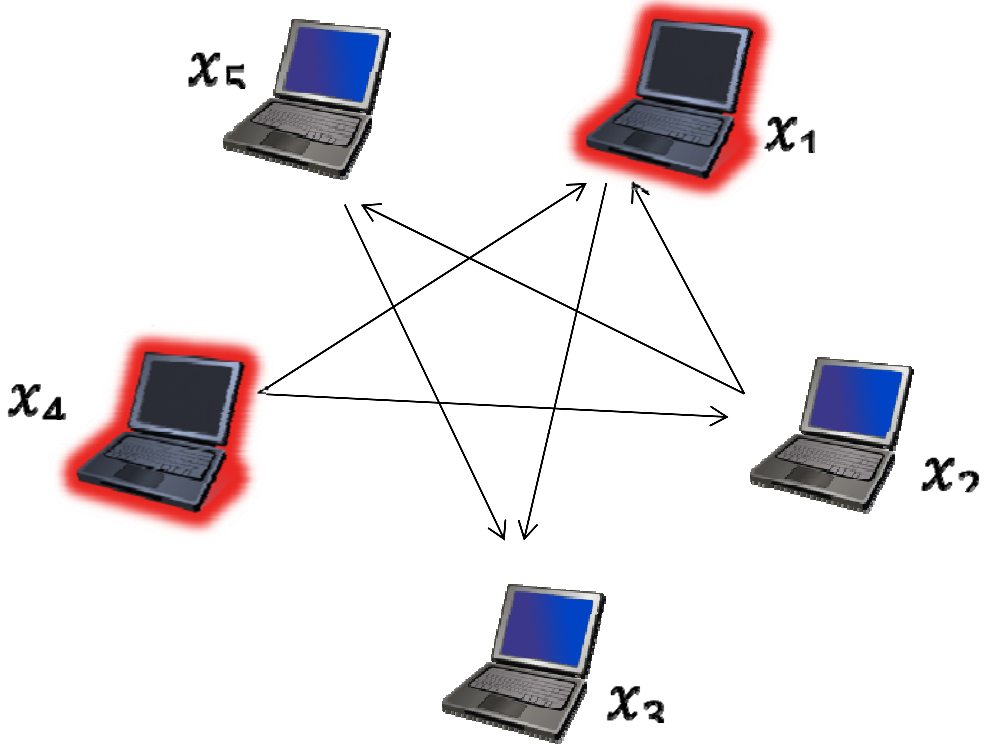
Shafi Goldwasser
MIT & Weizmann

Stefano Tessaro
MIT

Secure Multi-Party Computation (MPC)

[Goldreich-Micali-Wigderson87]

Jointly compute function f on secret inputs



Learn *only* $f(x_1, \ldots, x_n)$

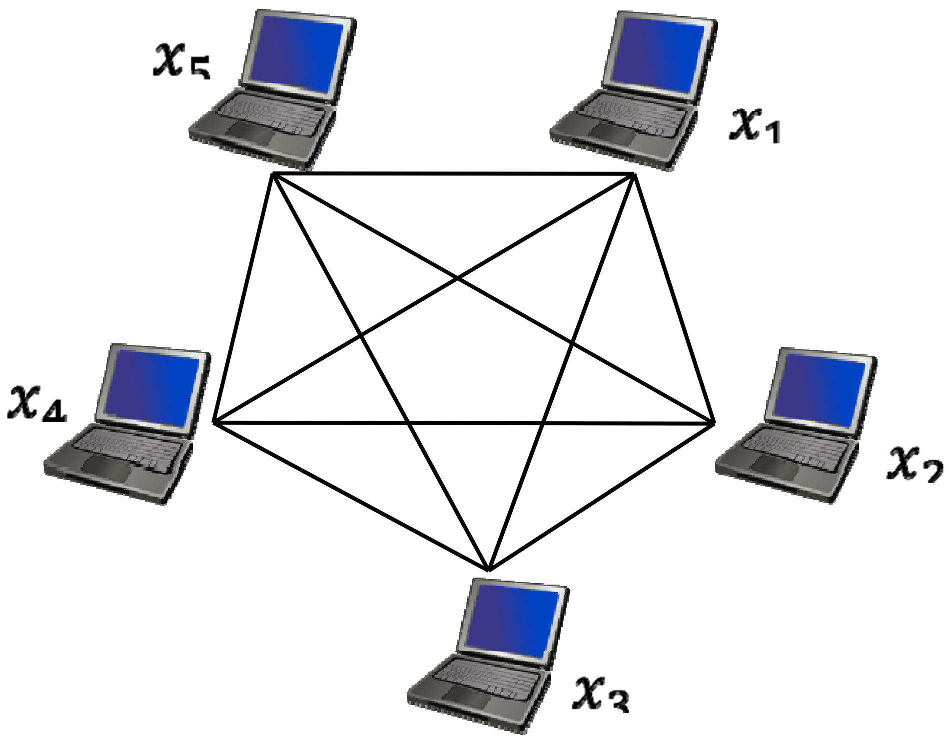
Selection of Prior MPC Work

- **Original Constructions** [GMW87, BGW88, CCD88,…]
  - Communication complexity: $\Omega(n^2 \cdot |f|)$
- **Scalable MPC** [Damgard-Ishai06, Damgard-Nielsen07, …]
  - Communication complexity: $O(poly(n) + |f|)$
- **FHE-Based** [Asharov-Jain-LopezAlt+12]
  - Communication complexity: $\Omega(n^2)$ independent of f

Everyone talks to everyone

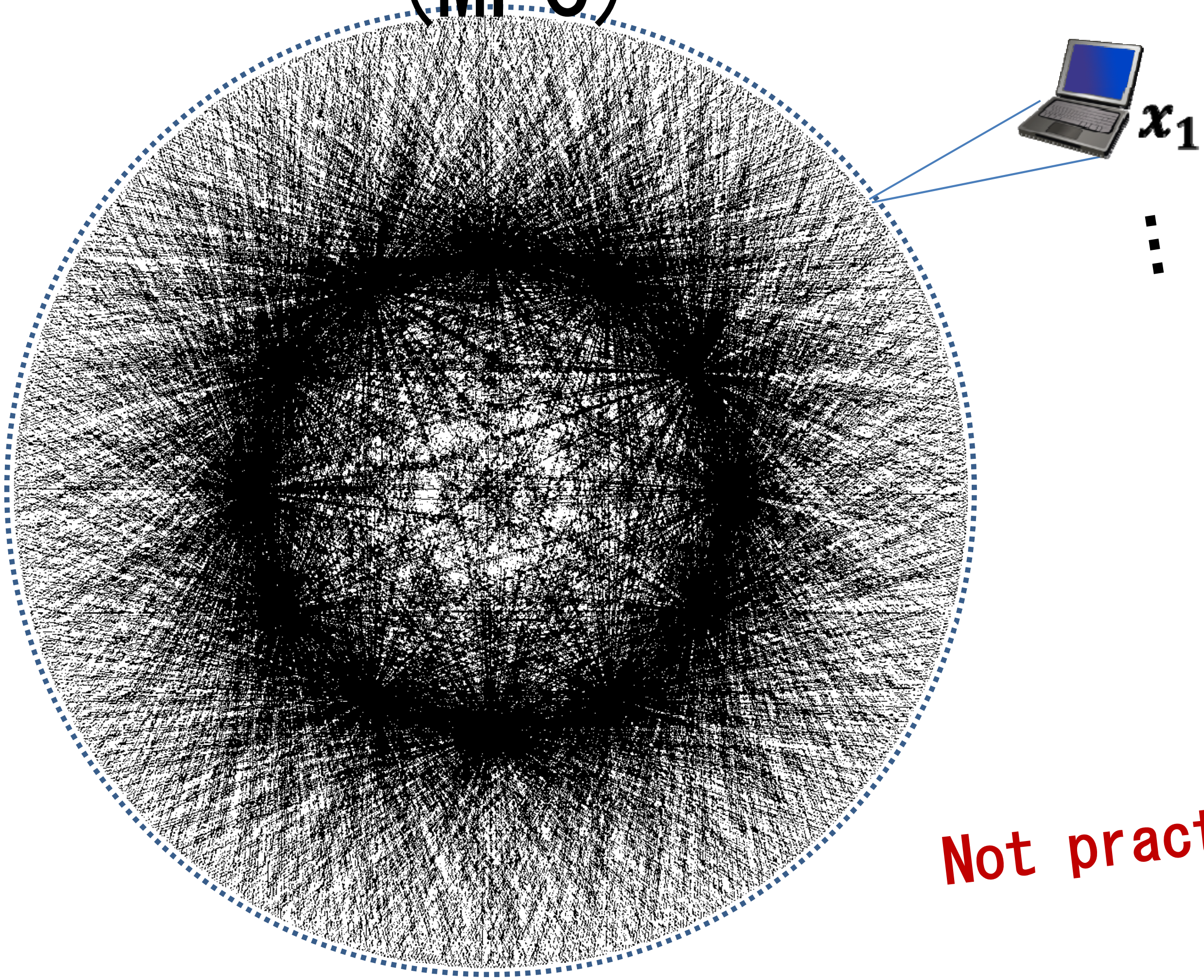
Secure Multi-Party Computation (MPC)

parties:
$n = 5$



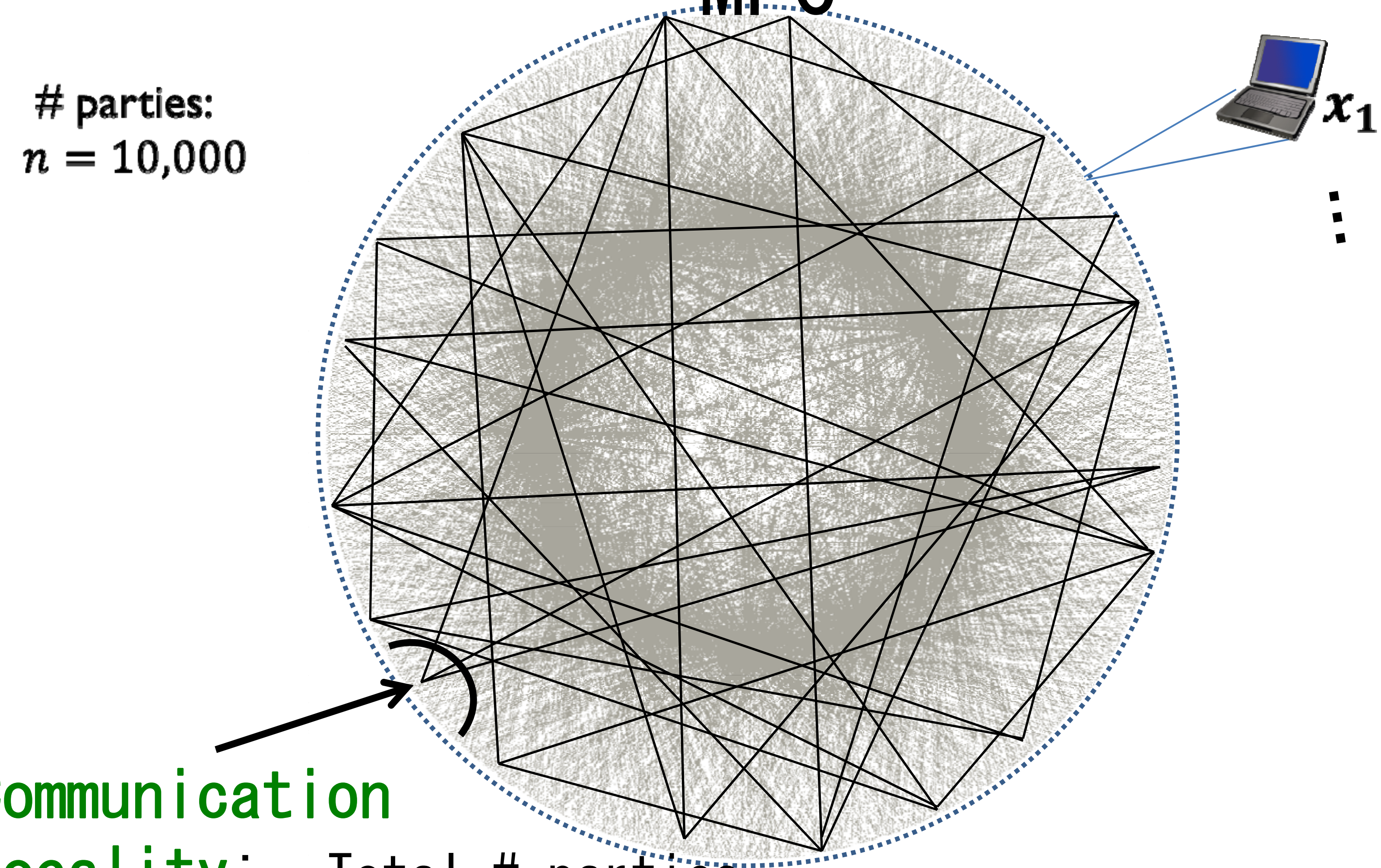
Secure Multi-Party Computation (MPC)

\# parties:
$n = 10{,}000$



Not practical!

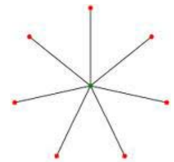
Today: Communication Locality in MPC



Communication Locality: Total # parties each party communicates with throughout protocol lifetime

Prior Work

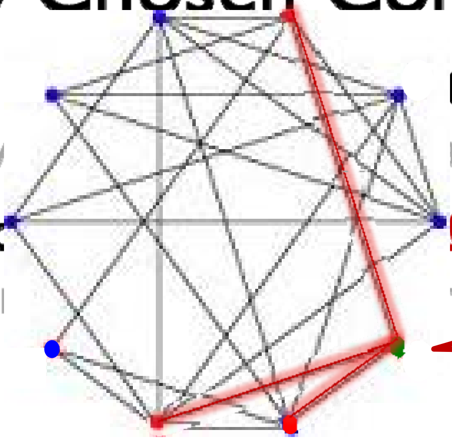
- MPC when Underlying Network is Incomplete
  - Almost-everywhere MPC for certain polylog(n)-deg networks
    [Chandran-G…y-Ostrovsky10, Chandran-Garay-Ostrovksy12]
  - Star gr… …vi-Lindell-Pinkas11]
    weake… …functionality



$n - o(n)$ parties receive correct output
[Dwork-Peleg-Pippenger-Upfal88]

"Everywhere" MPC not possible

- Dynamically Chosen Communication Network
  - Almost-every… …tion, BA: CC = $\Omega(\text{polylog}(n))$
    [King-Saia-Sanw… …10]
  - Leader Electi… $\Omega(\sqrt{n})$
    [King-Saia09, Ki… …gan-Sa… …]



Isolated honest party

This Work:

Full MPC,

polylog(n) communication locality,

using cryptography

General MPC

Theorem 1: There exists **general** n-party MPC s.t.

a) Tolerates $(1/3 - \epsilon)n$ static Byzantine faults

b) **Communication locality: polylog(n)**

c) Comm per party: $O\big(n \cdot \ell \cdot \mathrm{polylog}(n)\big)$ bits, (ℓ=input size)

d) Setup: signature keys + Common Random String (CRS)

Uses: digital signatures, FHE, simulation-sound NIZKs

* Can achieve (a) & (b) assuming *only* signatures + PKE

Special Focus: Sublinear Algorithms

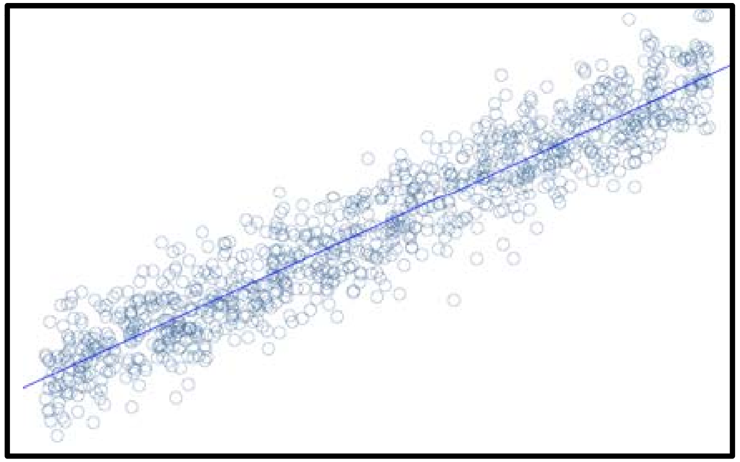
Input query complexity of f is $q(n) \in o(n)$

Execution of f with randomness r:

$$x_1 \; x_2 \; \boxed{x_3} \; x_4 \; x_5 \; x_6 \; \boxed{x_7} \; x_8 \; x_9 \; x_{10} \; x_{11} \; x_{12} \; \boxed{x_{13}} \; \boxed{x_{14}} \; x_{15} \; \cdots \; \boxed{x_n}$$

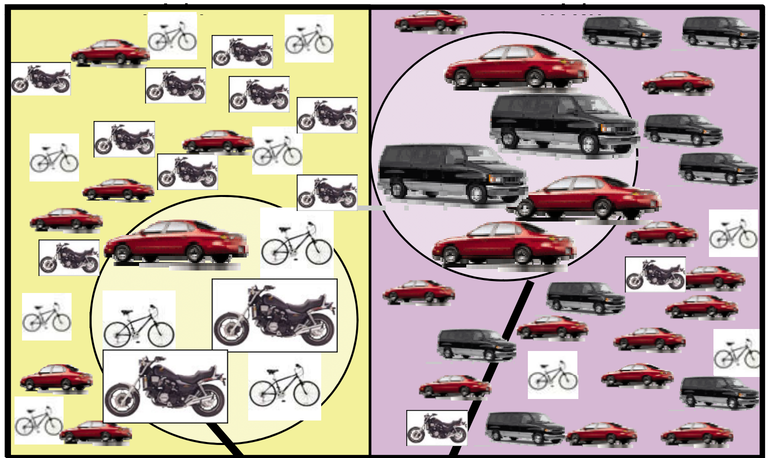
$\rightarrow f(x_1, \dots, x_n; r)$

- Example applications:



Distribution testing

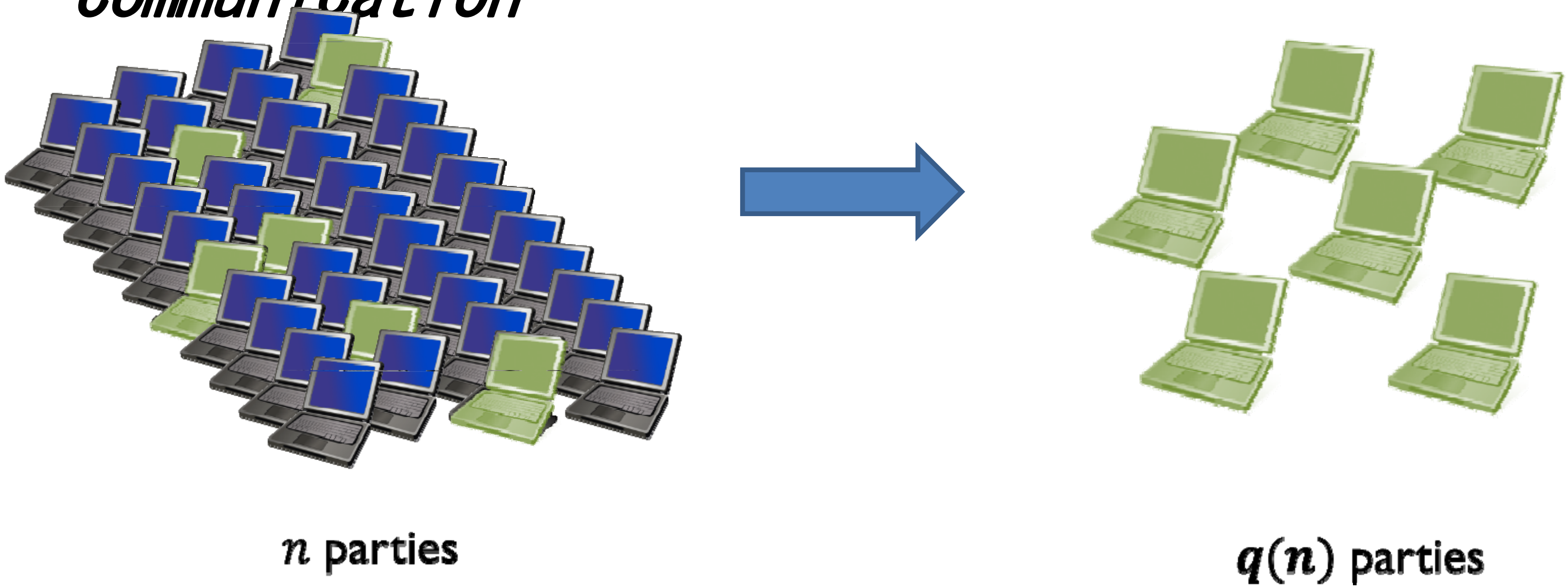
Transactions of 20-30 yr Transactions of 30-40 yr



Testing for trends

Securely Evaluating Sublinear Algorithms

In principle: requires much ***less communication***



Main Challenge: Must hide ***which inputs*** are used!

Related Work: Sublinear *Two-Party* Setting

- Communication-Preserving MPC [Naor-Nissim01]
  - Sublinear communication
  - Super-polynomial computation
- MPC on RAM programs
  [Ostrovsky-Shoup97, Damgard-Meldgard-Nielsen11, Gordon-Katz-Kolesnikov+12, Lu-Ostrovsky13]
- Sublinear MPC for specific functions
  [Feigenbaum-Ishai-Malkin+01, Indyk-Woodruff06, …]

MPC for Sublinear Algorithms

<u>Theorem 2</u>: MPC for *sublinear algorithms* f with query complexity q s.t.

a) Tolerates $(1/3 - \epsilon)n$ static Byzantine faults

b) Communication locality: $q \cdot \mathrm{polylog}(n)$

c) Comm per party: $O\left(\boxed{(n + \ell)} \cdot \mathrm{polylog}(n)\right)$ bits

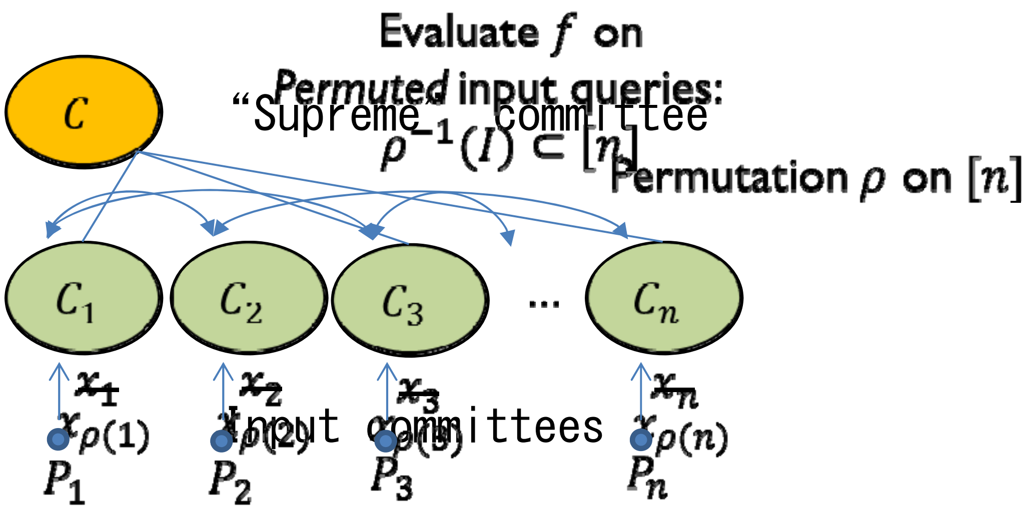
d) Setup: signature keys + Common Ra CRS)

Compared to
$(n \cdot \ell)$

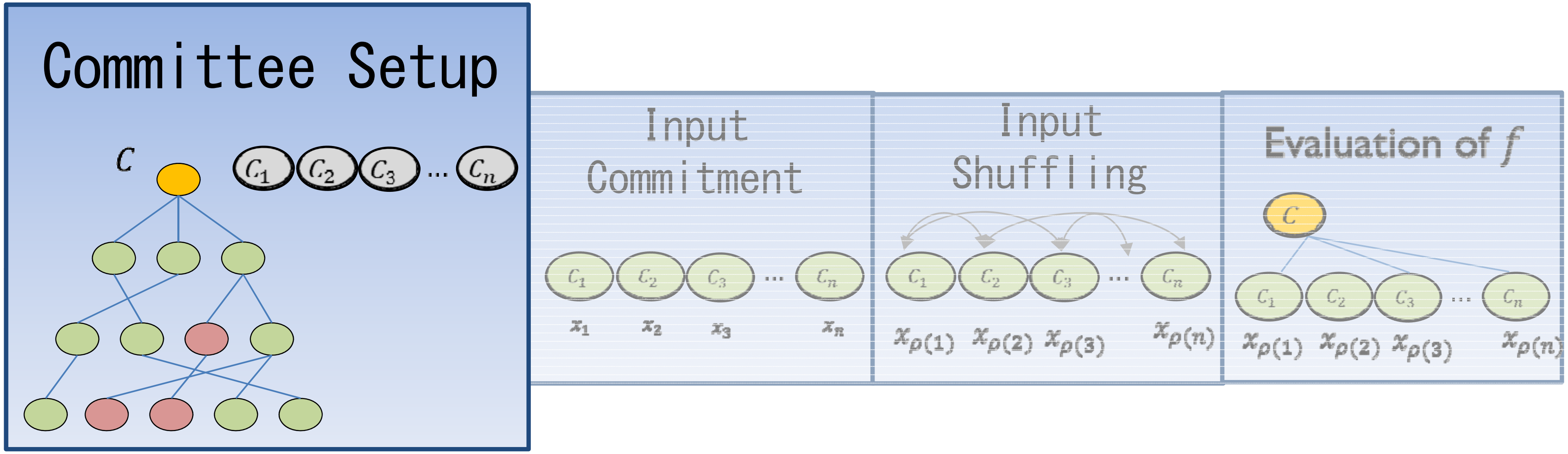
Uses: digital signatures, FHE, simulation-sound NIZKs

Protocol for Sublinear Algorithms: Overview of Nonadaptive Case

1. Committee Setup
2. Input Commitment
3. Oblivious Input Shuffling
4. Evaluation of f

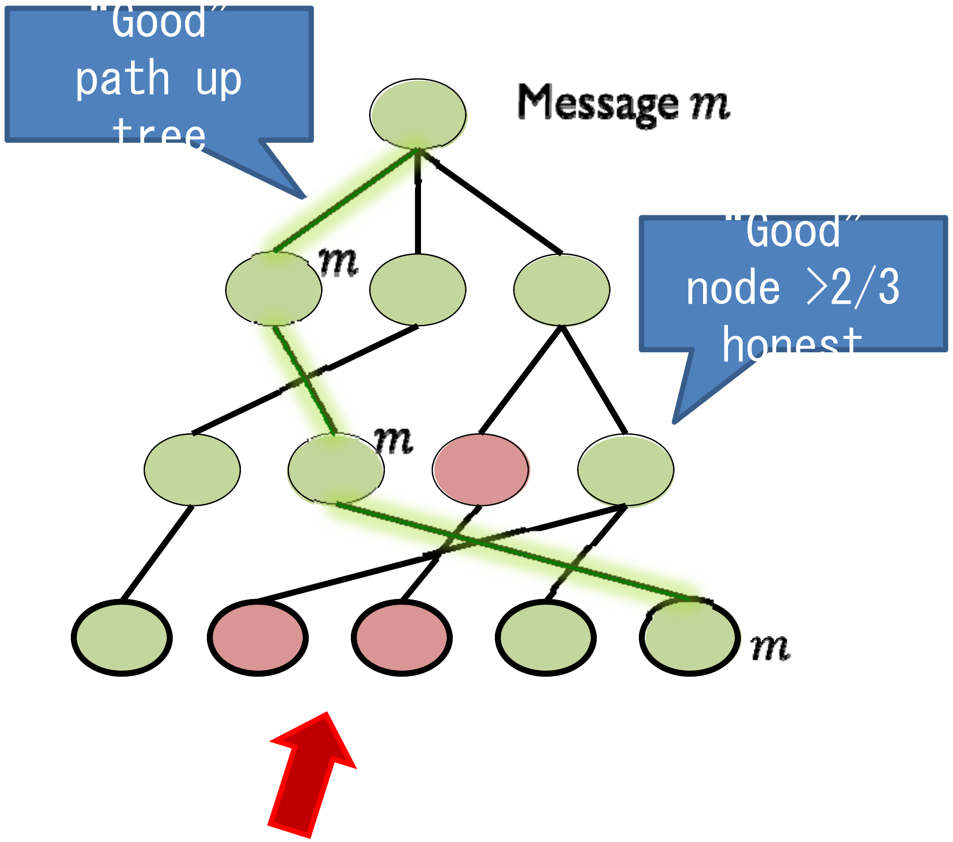


PHASE 1: COMMITTEE SETUP



Starting Point: *Almost-Everywhere* Committee Election

[King-Saia-Sanwalani-Vee06]



Properties:

- Depth, degree polylog(n)
- polylog(n) parties per node
- $(1 - o(1))$ fraction "good" nodes per layer

$\Rightarrow$ **$(1 - o(1))$ fraction of leaves have "good" path up tree**

Problem: $o(1)$ fraction of leaf nodes may receive bad information!

Toward *Full* Agreement

(Starting point)

This is where signatures come in

Almost-everywhere → *"Certified"* almost-everywhere → *Full*

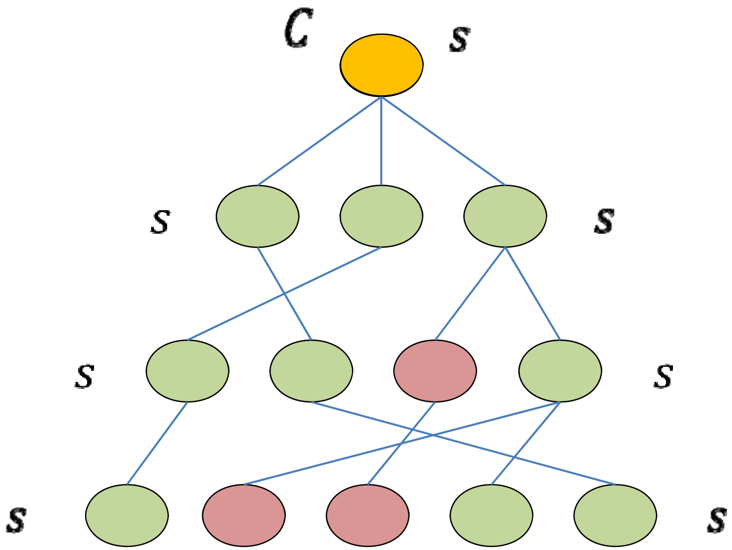
$n - o(n)$ parties agree

$n - o(n)$ parties agree
+
hold "**certificate**" of **correctness**

All parties agree

Supreme & Input Committees

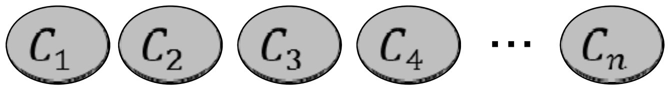
- Supreme Committee



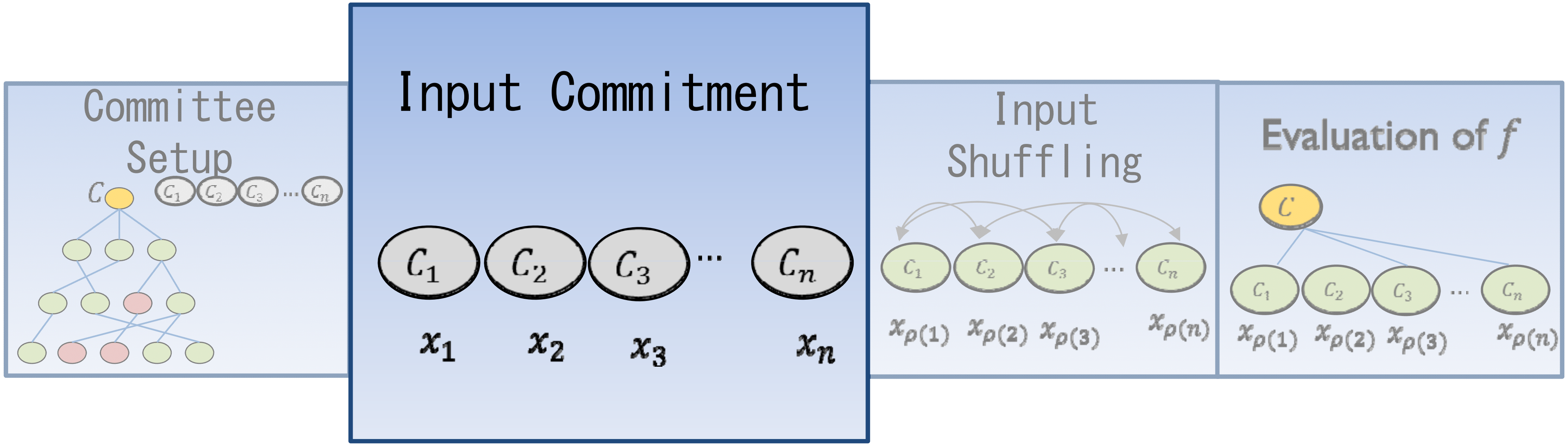
- Input Committees

Defined using PRF

$C_i \coloneqq F_s(i)$



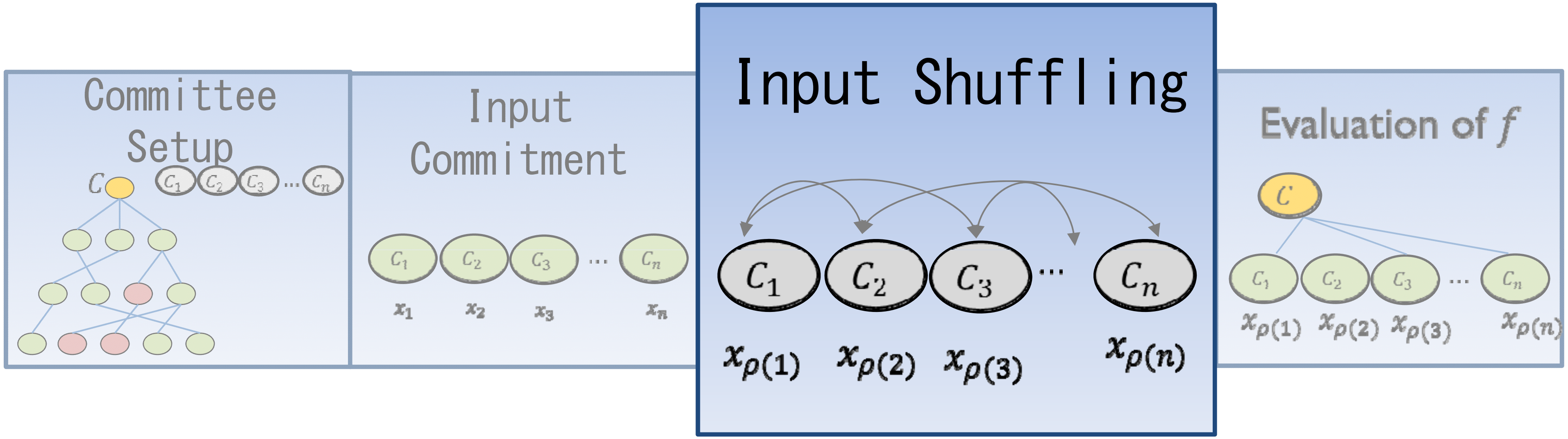
Phase 2: Input Commitment



Each party P_i Verifiably Secret Shares (VSS) his input to C_i

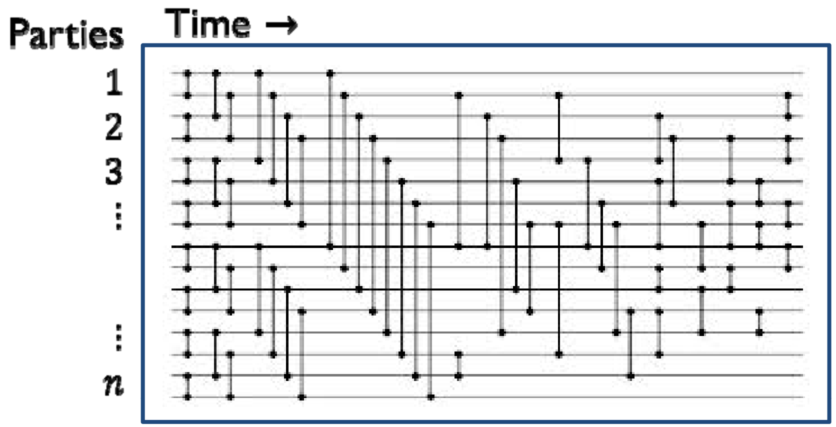
[Chor-Goldwasser-Micali-Awerbuch85]

Phase 3: Input Shuffling



Switching Networks

- Pairwise swaps among C_i's



Switching network:

0/1 choice per swap gate yields a permutation on $[n]$

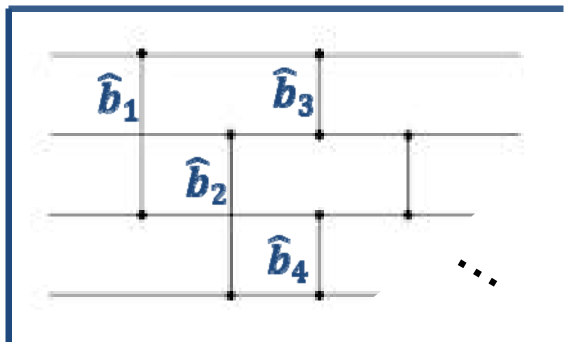
- <u>Lemma</u>: $\exists$ polylog(n)-depth switching network SN s.t.

Random choice of swap bits in SN	$\approx_s$	Random permutation on $[n]$

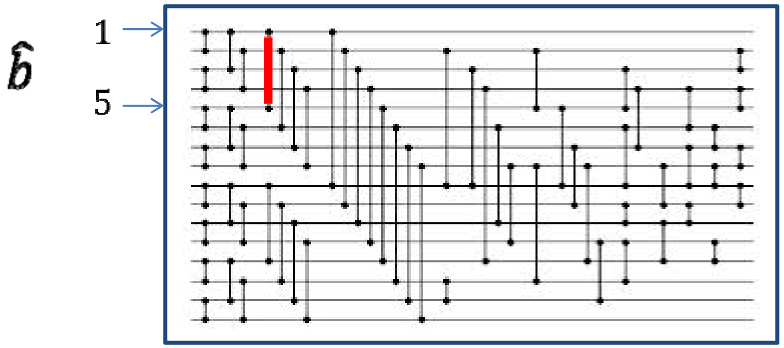
Oblivious Shuffling

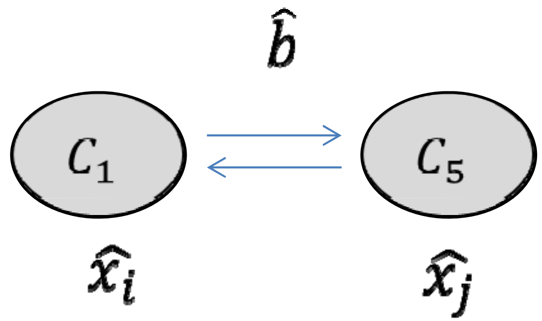
- **Generating Shuffle:**

  C obliviously generates FHE-*encrypted* swap bits $\hat{b}_j$



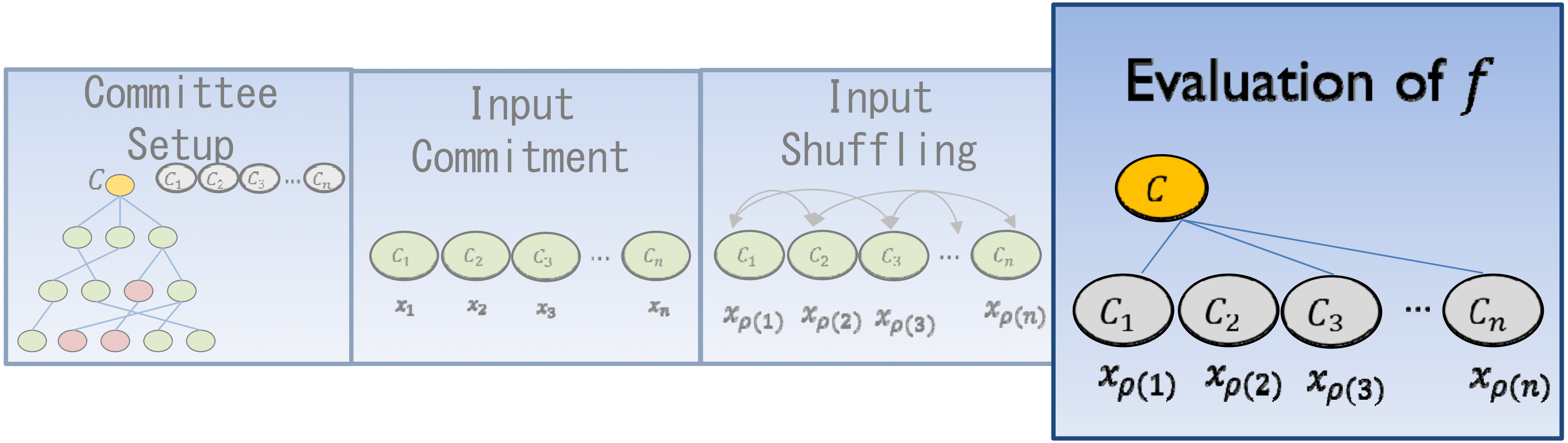
- **Implementing Shuffle:**





Homomorphically Evaluate: *Swap-Or-Not*(b, x_i, x_j)

PHASE 4: EVALUATION OF f



Summary of Contributions

Communication Locality:
A New Metric in MPC

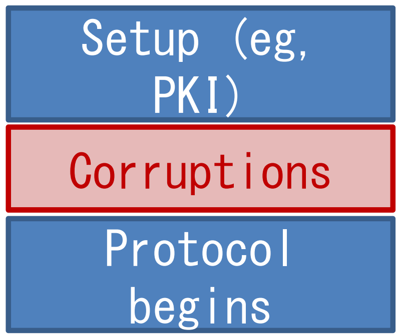
Achieve communication locality $\mathbf{polylog}(n)$
using cryptography

Sublinear Algorithms in MPC Context

Our Model

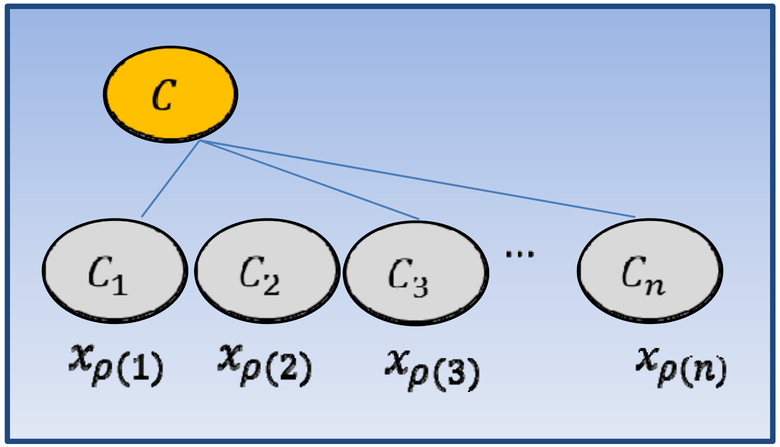
- Network Model
  - Synchronous network
  - Public-Key Infrastructure (PKI)
  - *Ability* to communicate with anyone

- Adversarial Model
  - Byzantine faults of t parties
  - Faults scheduled:
    - post PKI
    - pre protocol execution



C Leads Evaluation of f

- Committee C samples *permuted* input indices: $\rho^{-1}(I)$ & queries each input committee $C_{\rho^{-1}(i)}$
- Each queried committee $C_{\rho^{-1}(i)}$ sends held input value
- C evaluates f on these inputs, corresp to I

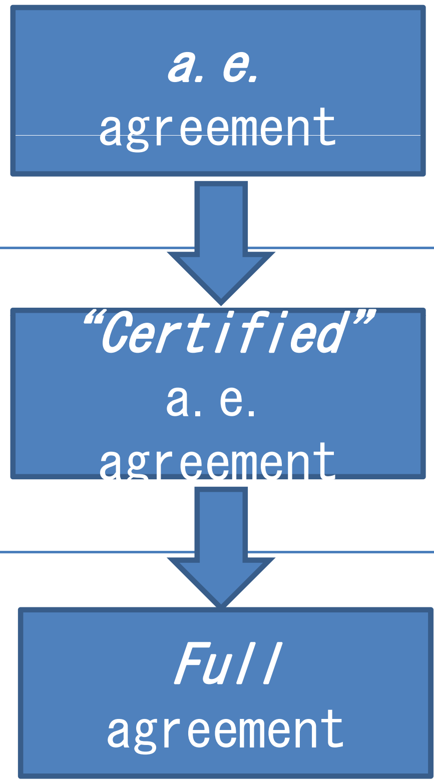


Phase 1 Overview: Committee Setup

- First goal: Elect "good" committee C

<u>Starting point:</u>

Almost-everywhere (a.e.) election :
$1-o(1)$ fraction of parties agree on good C



$1-o(1)$ fraction of parties agree on good C
and hold "certificate" of correctness

All parties agree on good C

Phase 1 Overview: Committee Setup

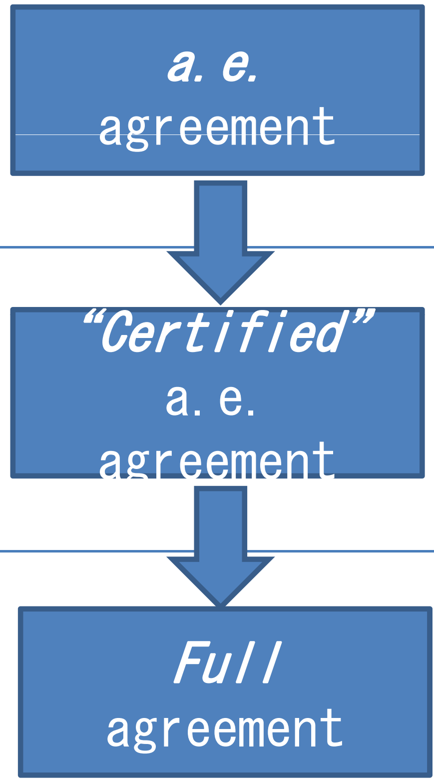
- Second Goal: Allow C to ***broadcast***

<u>Starting point:</u>

$1-o(1)$ agreement on some value

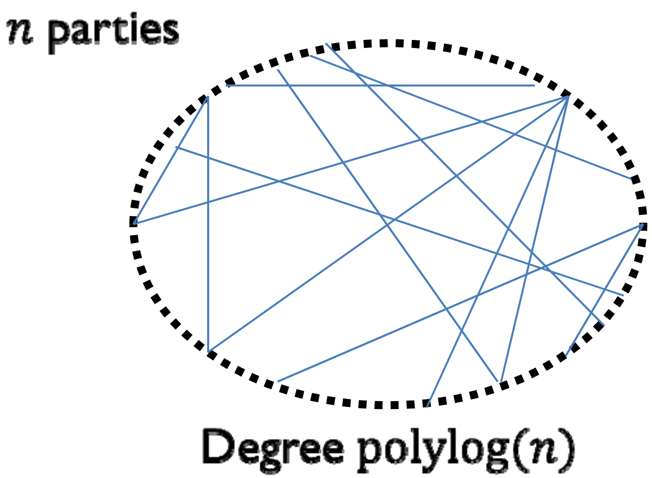
$1-o(1)$ fraction of parties agree on value
and hold "certificate" of correctness

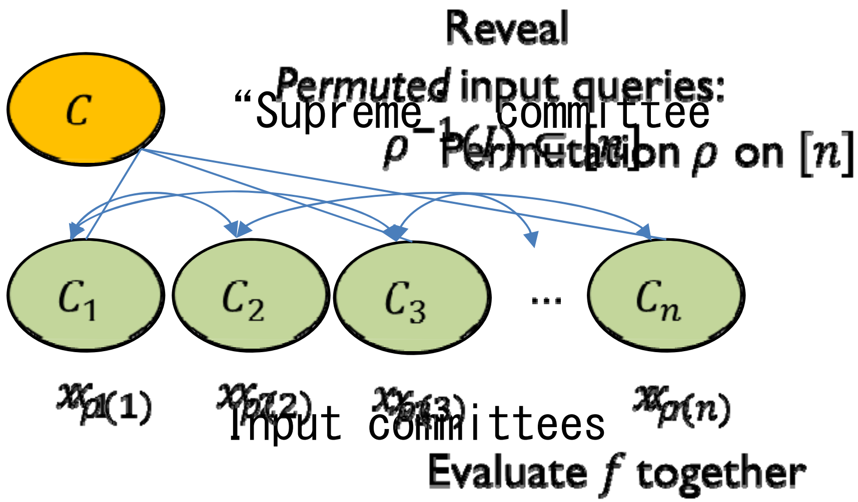
All parties agree on value



Protocol for Sublinear Algorithms: Overview

1. Communication Graph + Committee Setup
2. Input Commitment
3. Input Shuffling
4. **Evaluation of f**





Combining Signatures into Certificate

- Option 1: Append as list

$$\sigma_1, \sigma_2 \rightarrow \sigma' = (\sigma_1, \sigma_2)$$

"Certificate" for m $\equiv (\sigma_1, \sigma_2, \ldots, \sigma_k)$

with $\geq n/2$ valid signatures

- Option 2: Use *Multisignatures* [***]

Multisigs:

Can combine sigs on same msg into short object

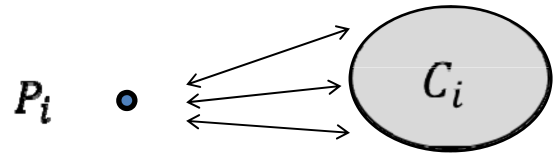
(**Gen**, **Sign**, **Verify**)	-	standard signature algorithms
$\boldsymbol{Combine}(\{vk_i, \sigma_i\}, m)$	-	outputs multisignature
$\boldsymbol{MultiVer}(\{vk_i\}, m, \sigma')$	-	verifies σ' wrt $\{vk_i\}$

$$vk_1, \sigma_1,\; vk_2, \sigma_2 \rightarrow \sigma' \leftarrow Combine(\sigma_1, \sigma_2)$$

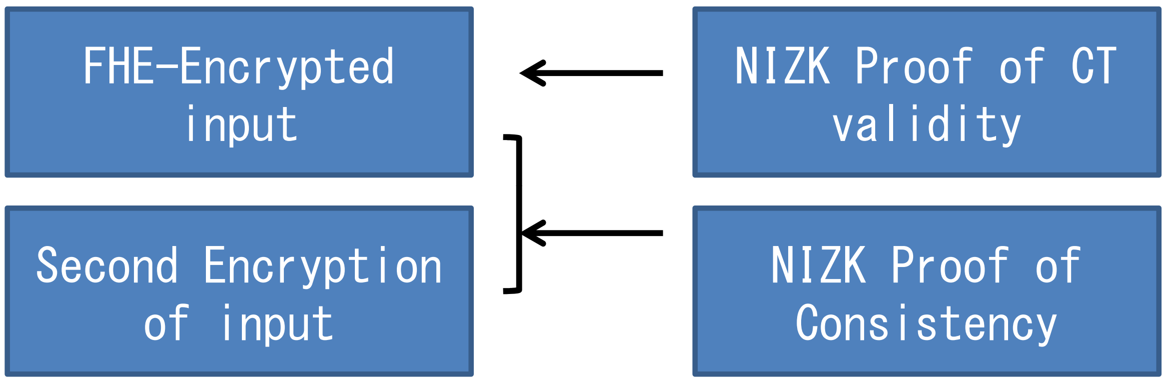
"Certificate" for m $\equiv \sigma', 1_S$

Valid multisig wrt $\{vk_i\}_{i \in S}$, $|S| \geq n/2$

Step 2: Input Commitment

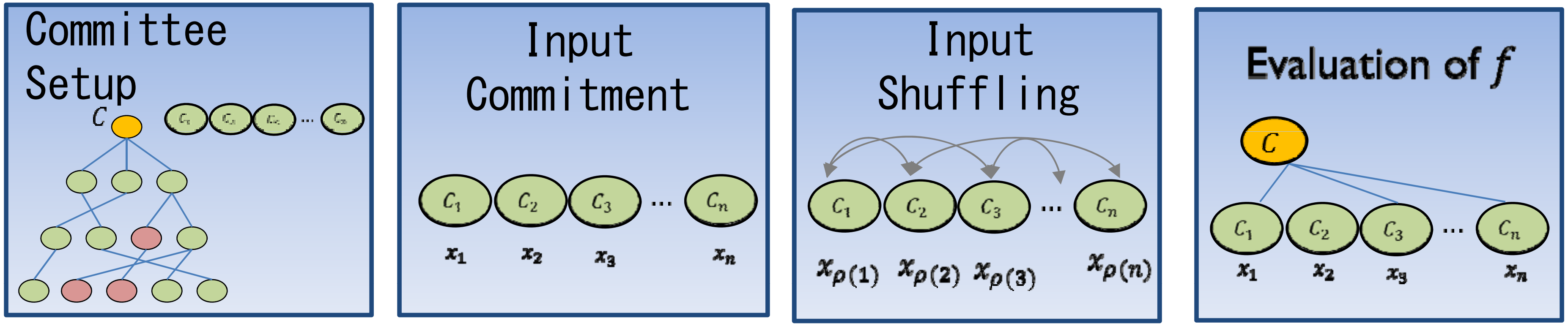


- Use Verifiable Secret Sharing (VSS) [CGMA85]
  - Values to share:



  - Public keys generated by C & "broadcast"
- C_i verifies proofs, keeps FHE ciphertext

PHASE 1:
COMMITTEE SETUP



Analyzing Communication

Protocol Step	Comm Locality	Comm cxy	# Rounds
A.e. leader election			
Certifying a.e.		$O(n \cdot pl(n))$	
To full agreement		$O(n \cdot pl(n))$	
Input commitment		$\ell \cdot pl(n)$	
Gen shuffle perm		$O(n \cdot pl(n))$	
Implementing shuffle		$\ell \cdot pl(n)$	
Choosing inputs	$q \cdot pl(n)$	$\ell \cdot pl(n)$	$O(q) + pl(n)$
Evalua[illegible]	$q \cdot pl(n)$	$\ell \cdot pl(n)$	

n bits to describe certifying set $\subset [n]$

Permutation on $[n]$: $n \cdot \log(n)$ bits

For *adaptive* algorithms

C talks to q input committees

$\boxed{}$ $= pl(n) = O(polylog(n))$

This Talk:

Protocol for ***sublinear algorithms*** (Thm 2)

+ Complexity Analysis

Extension to ***general*** functions (Thm 1)

Sanjam Garg Abhishek Jain Amit Sahai Stefano Tessaro

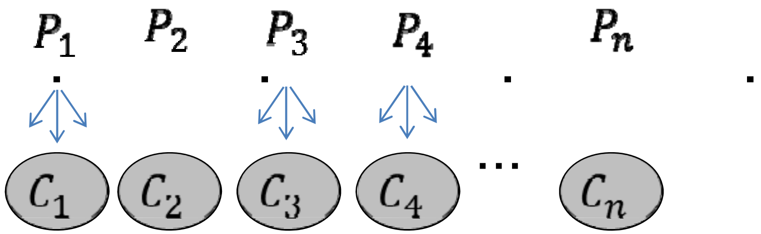
Shafi Goldwasser Yael Tauman Gil Segev Daniel Wichs

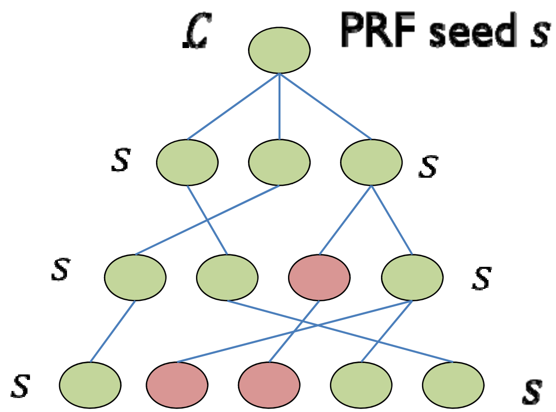
Achieving *Full* Agreement

Almost-everywhere → *"Certified"* almost-everywhere → *Full*

What about isolated honest parties??

Solution: Enlightened parties P_i "spread the word" to *new* polylog(n)-size groups C_i





Properties of C_i's:

- $\forall i, \exists$ "enlightened" j s.t. $P_i \in C_j$
- Each C_i is "good" ($\geq 2/3$ honest)

To be used later!

Can achieve with Pseudorandom Function Family: $C_i = F_s(i)$