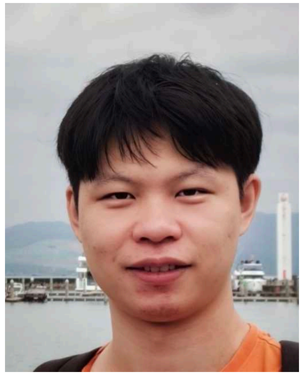
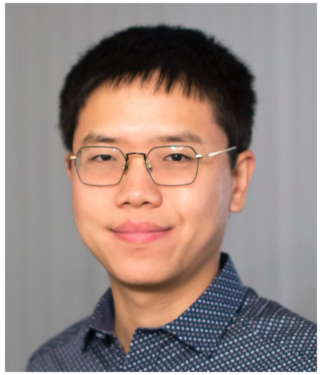
Asynchronous Consensus without Trusted Setup or Public Key Cryptography








Sourav Das Sisi Duan Shengqi Liu Atsuki Momose Ling Ren Victor Shoup

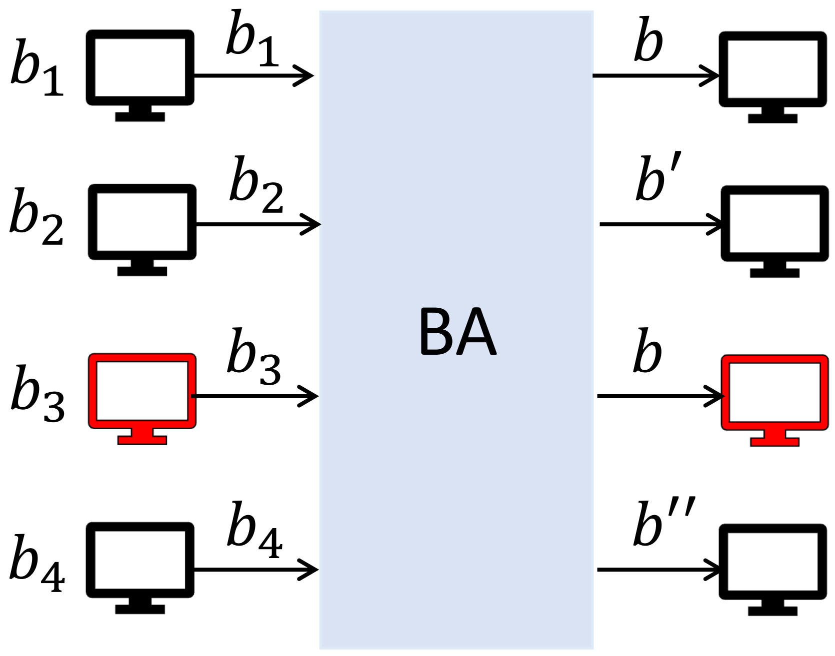
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

souravd2@Illinois.edu

To appear at ACM CCS 2024

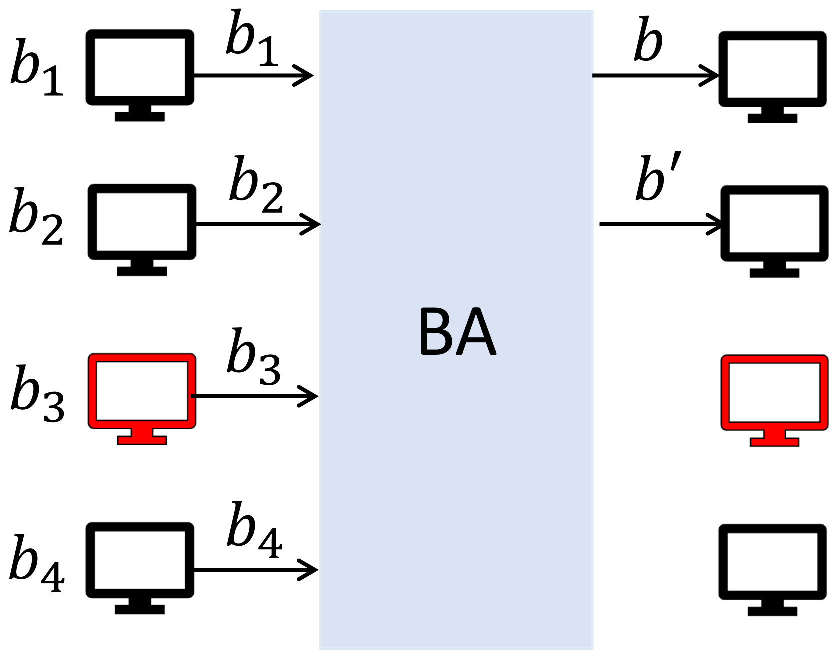
Binary Agreement/Consensus [Lamport 1982]

n party protocol tolerating t failures to agree on a single bit



Binary Agreement/Consensus [Lamport 1982]

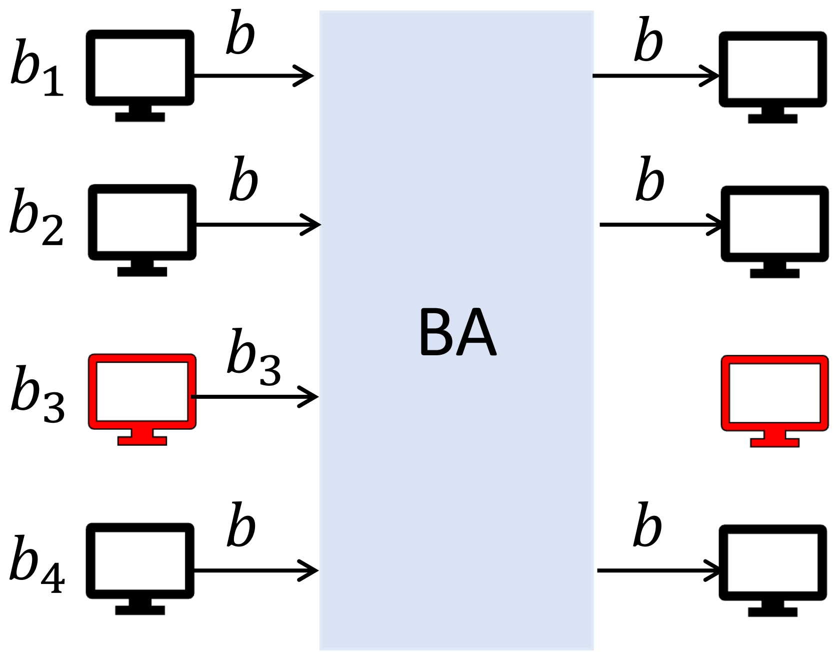
n party protocol tolerating t failures to agree on a single bit



- Agreement
  - $b' = b$

Binary Agreement/Consensus [Lamport 1982]

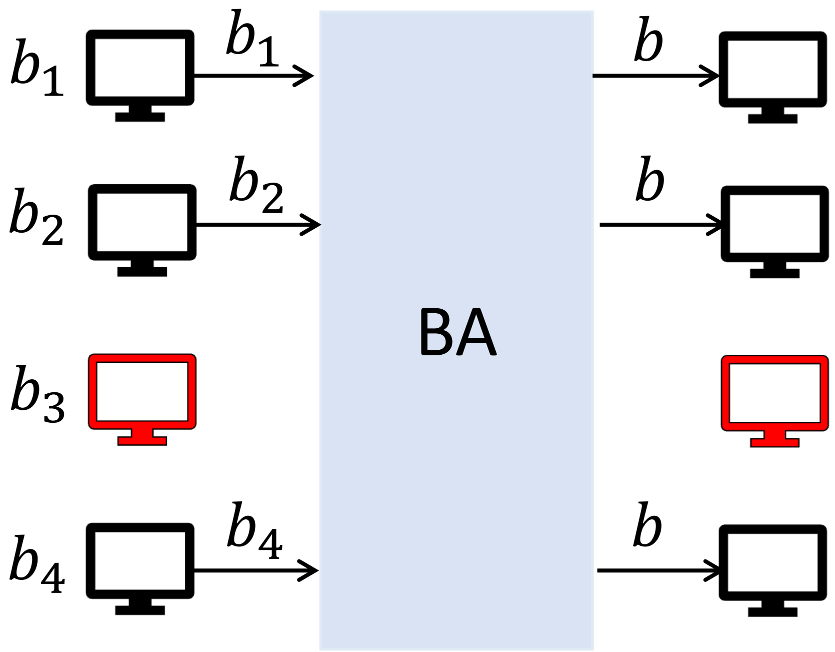
n party protocol tolerating t failures to agree on a single bit



- Agreement
  - $b' = b$
- Validity
  - All honest node input $b \Rightarrow$ BA outputs b

Binary Agreement/Consensus [Lamport 1982]

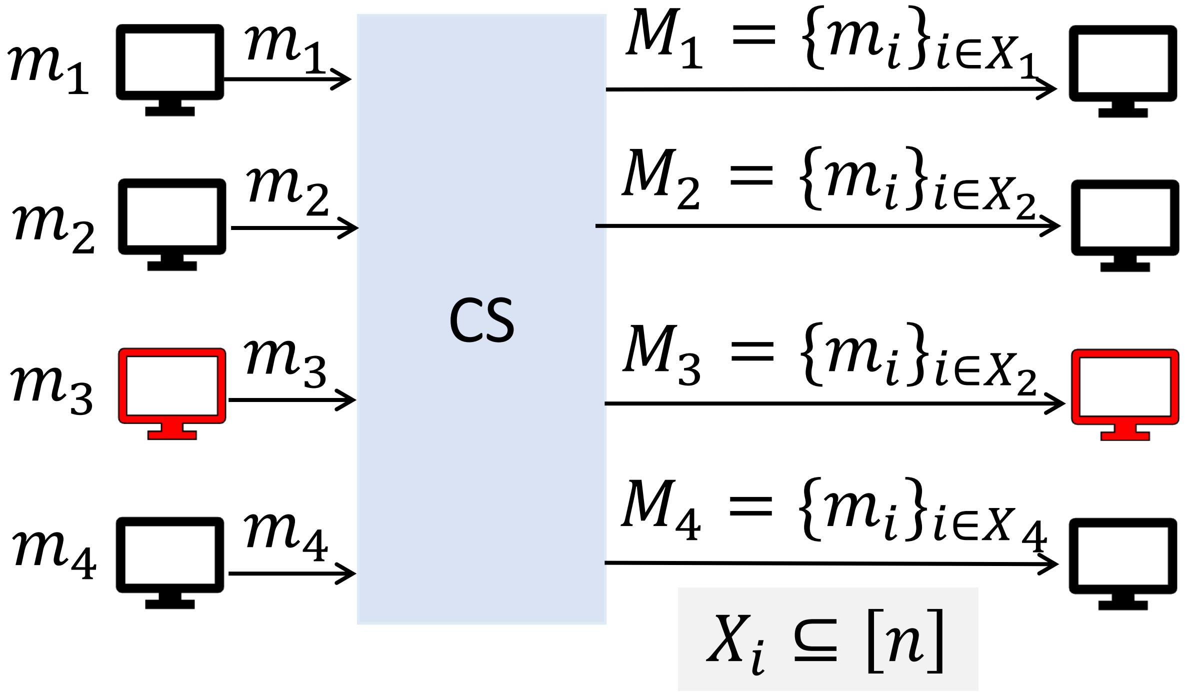
n party protocol tolerating t failures to agree on a single bit



- Agreement
  - $b' = b$
- Validity
  - All honest party input $b \Rightarrow$ ABA outputs b
- Termination
  - The protocol eventually terminates

Common Subset or Interactive Consistency

n party protocol tolerating t failures to agree on a subset of inputs



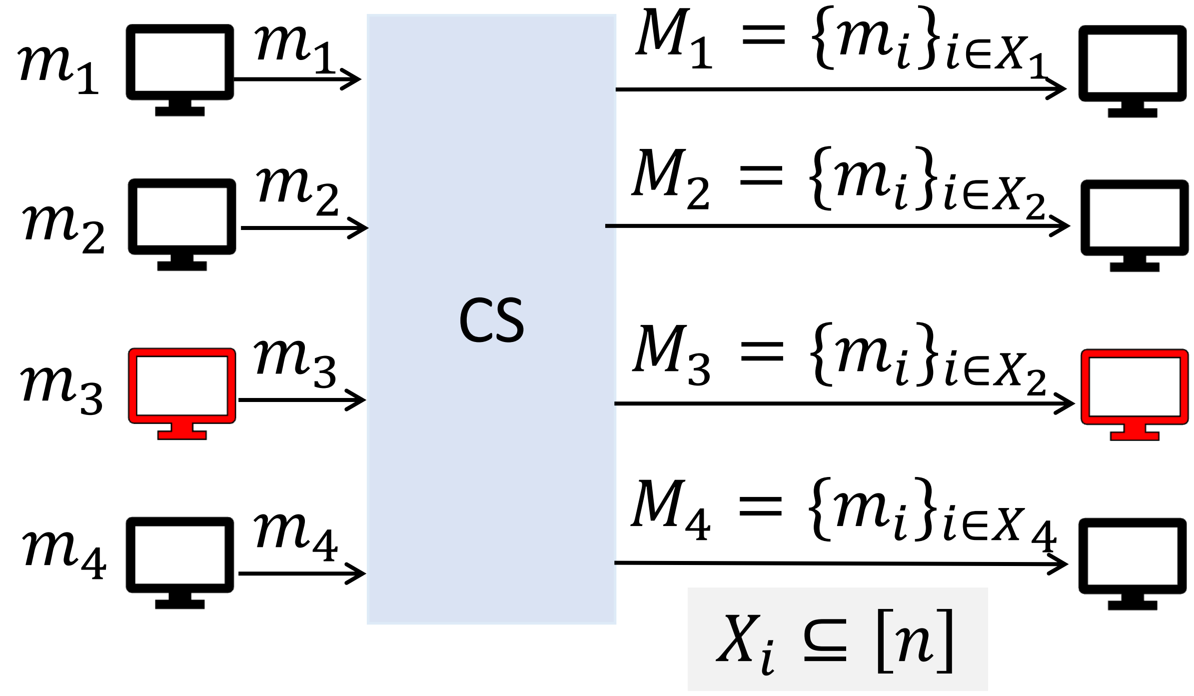
Agreement:
$M_i = M_j = M$ for honest parties (i, j)

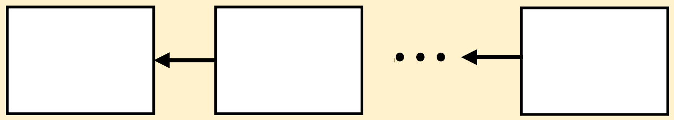
Validity:
M contains at least $n - 2t$ honest input

Termination:
The protocol terminates

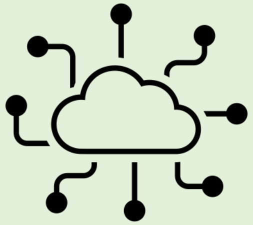
Applications of Common Subset

n party protocol tolerating t failures to agree on a subset of inputs

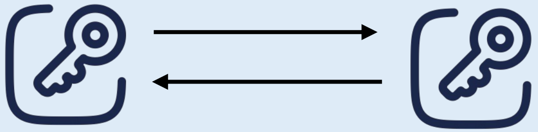




Blockchains



Secure multi-party computation

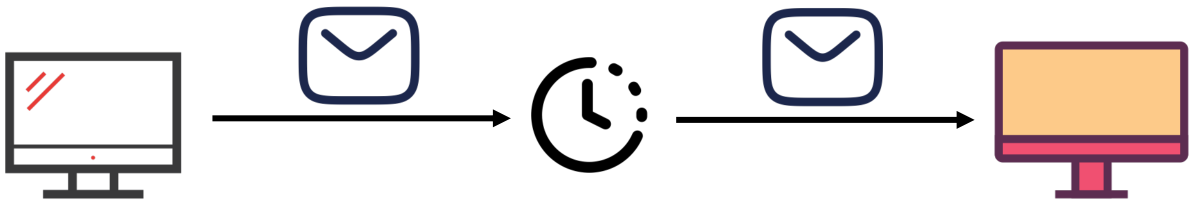


Distributed Key Generation

Asynchronous Networks

Definition (Asynchronous Networks):

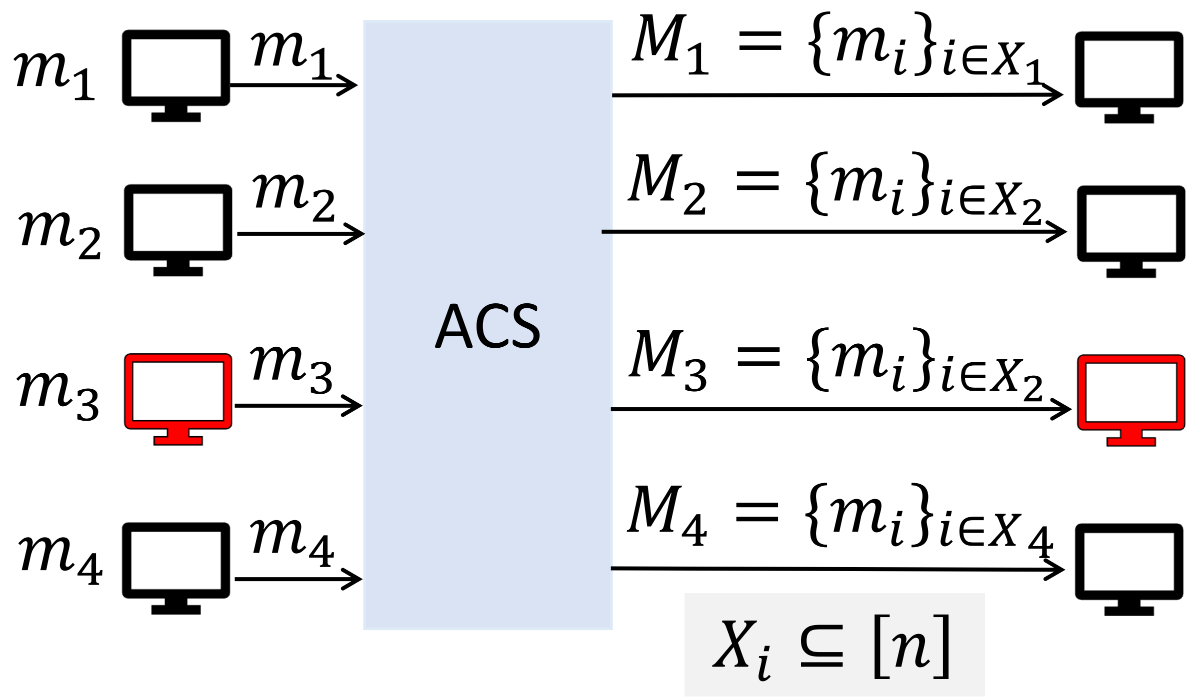
1. Message delays between honest parties are unbounded
2. Messages must eventually arrive



Great to model communication over the internet!

Asynchronous Common Subset (ACS)

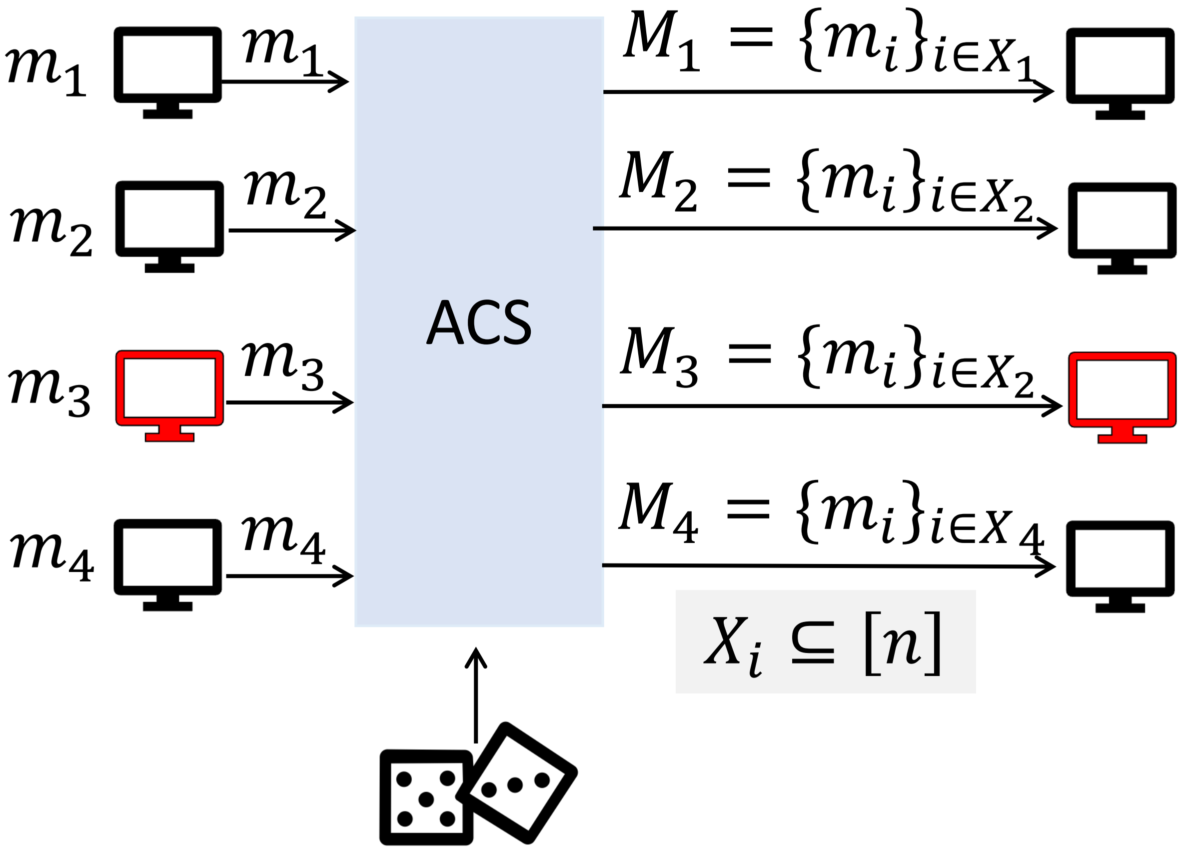
Common subset protocol over asynchronous network



Deterministic consensus is impossible in asynchronous network [FLP86]

ACS with external common coin

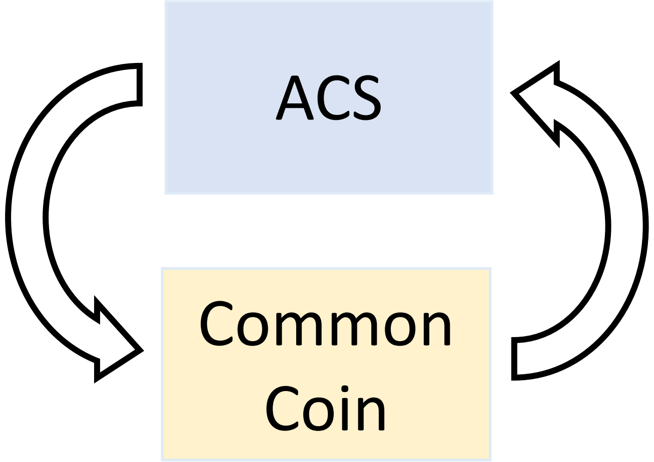
ACS protocols needs to be randomized!



Common Coin

How to build an asynchronous common coin protocol?

Rely on ACS ☹



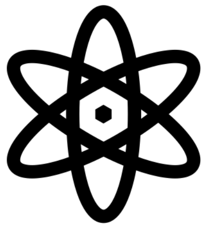
Circular dependency ☹

Asynchronous Consensus: Prior Works

Protocols	Communication Cost	Expected #Round	Assumptions
AJM+'21, GLL+'22	$O(\kappa n^3)$	$O(1)$	SXDH, RO
DXKR'22	$O(\kappa n^3)$	$O(\log n)$	DDH, RO

Relies on public key cryptography assumption

Insecure against quantum computers [Shor 99]



Asynchronous Consensus: Prior Works

Protocols	Communication Cost	Expected #Round	Assumptions
AJM+'21, GLL+'22	$O(\kappa n^3)$	$O(1)$	SXDH, RO
DXKR'22	$O(\kappa n^3)$	$O(\log n)$	DDH, RO
AGPS'23 ($n \geq 4t + 1$)	$O(n^5)$	$O(1)$	None

What about information theoretic protocols?

Asynchronous Consensus: Prior Works

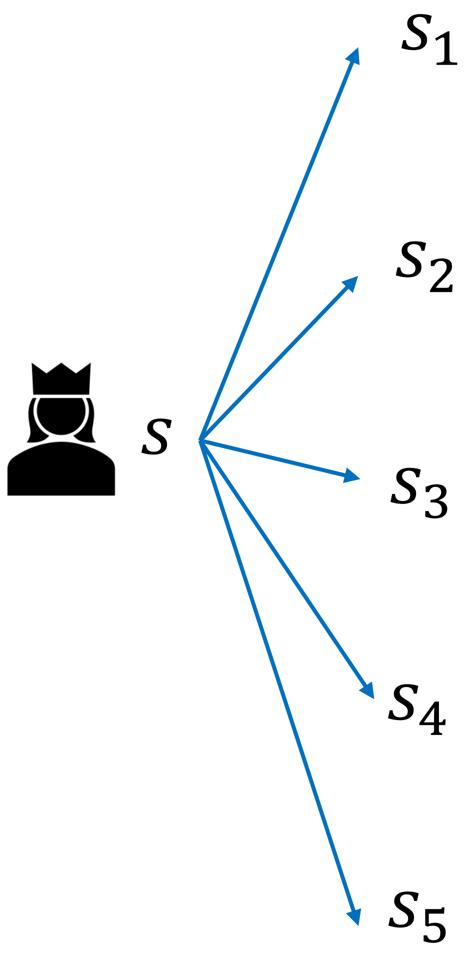
Protocols	Communication Cost	Expected #Round	Assumptions
AJM+'21, GLL+'22	$O(\kappa n^3)$	$O(1)$	SXDH, RO
DXKR'22	$O(\kappa n^3)$	$O(\log n)$	DDH, RO
AGPS'23 ($n \geq 4t + 1$)	$O(n^5)$	$O(1)$	None
This work	$O(\kappa n^3)$	$O(1)$	RO^*

Can we design a post-quantum secure ACS protocol without setup? ✔

Is it concretely efficient? ✔

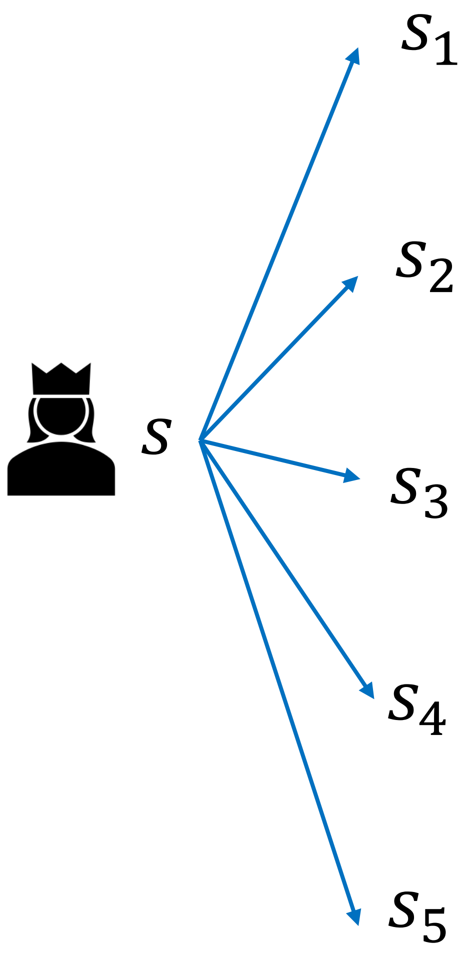
Background

(n, t) Threshold Secret Sharing



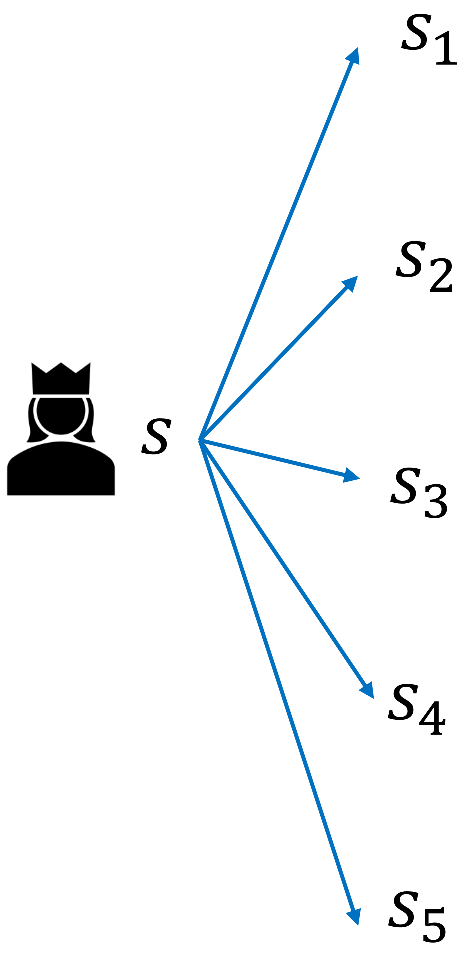
- A mechanism to share a secret s into n shares

(n, t) Threshold Secret Sharing

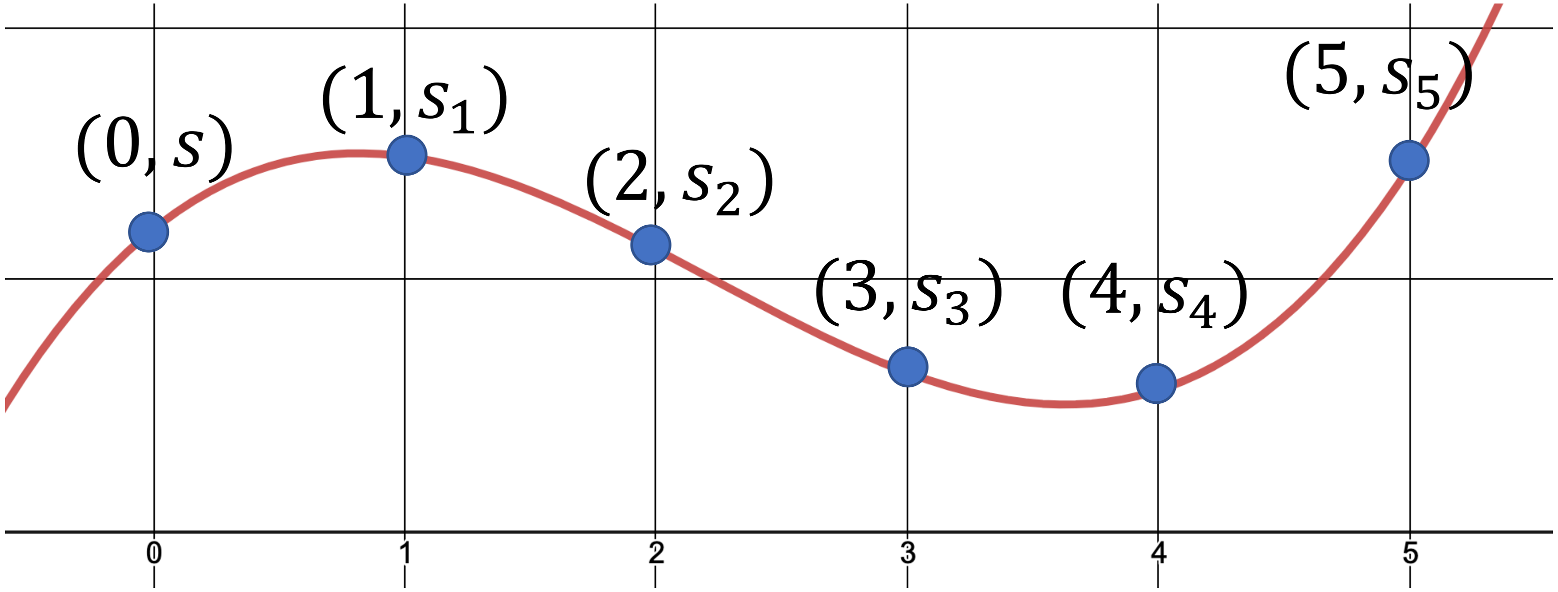


- A mechanism to share a secret s into n shares
- Any subset of $t + 1$ shares reveal the secret
- Any subset of t or less shares reveal nothing about s

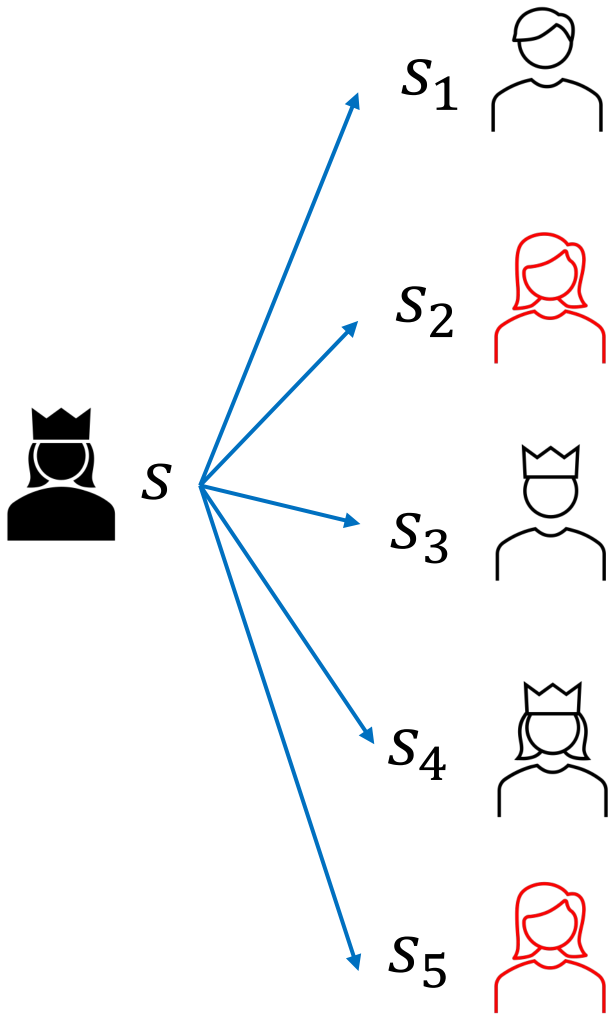
(n, t) Threshold Secret Sharing



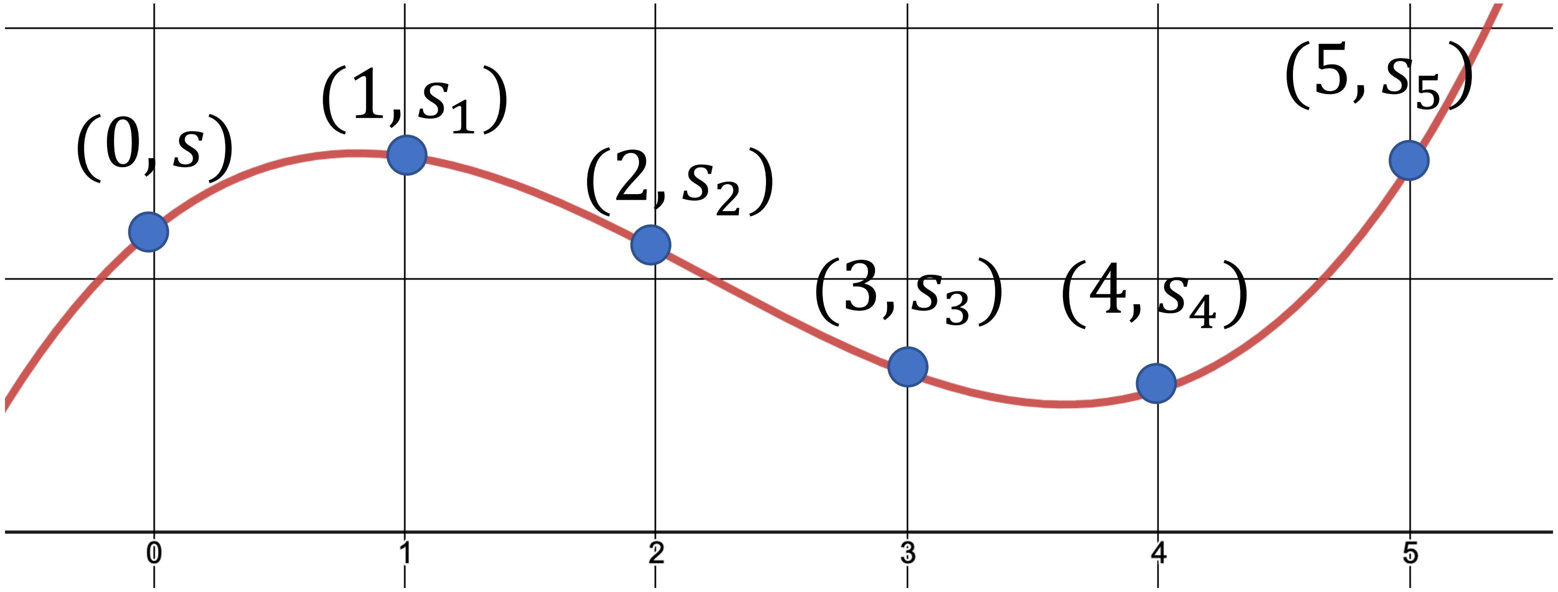
- A mechanism to share a secret s into n shares
- Any subset of $t + 1$ shares reveal the secret
- Any subset of t or less shares reveal nothing about s
- An example of $(5,3)$ threshold secret sharing scheme



(n, t) Verifiable Secret Sharing

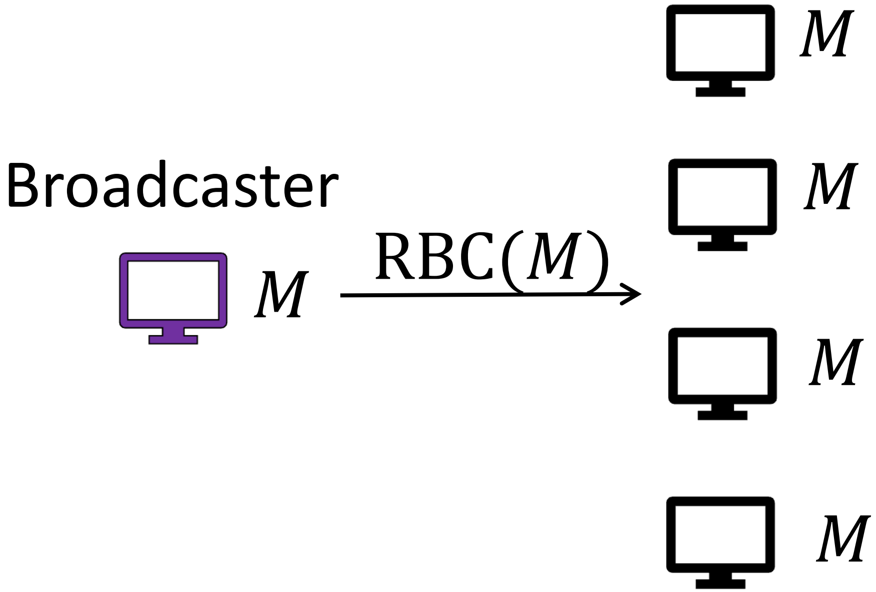


- A protocol to share a secret s into n parties
- Any subset of $t + 1$ parties can recover the secret
- Any subset of t or less parties learn nothing about s
- An example of $(5,3)$ threshold secret sharing scheme

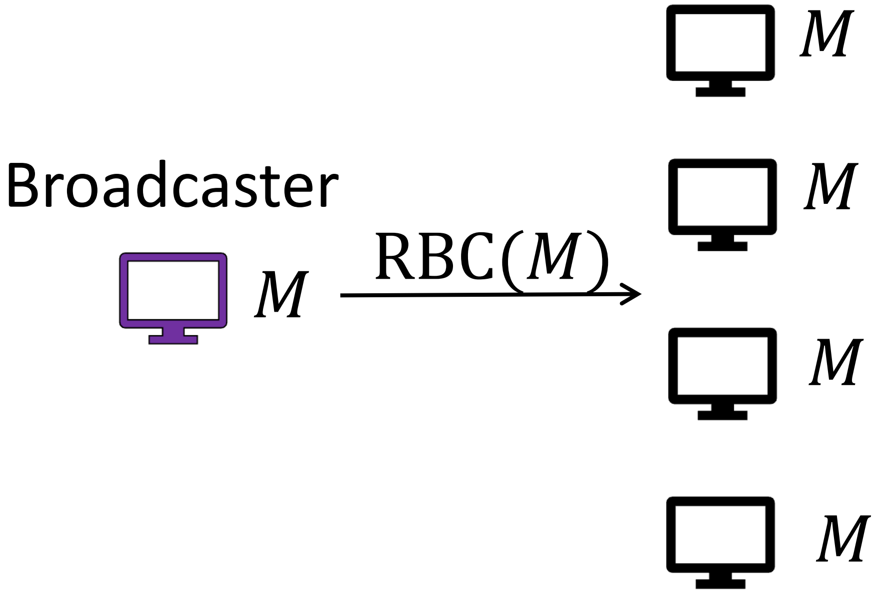


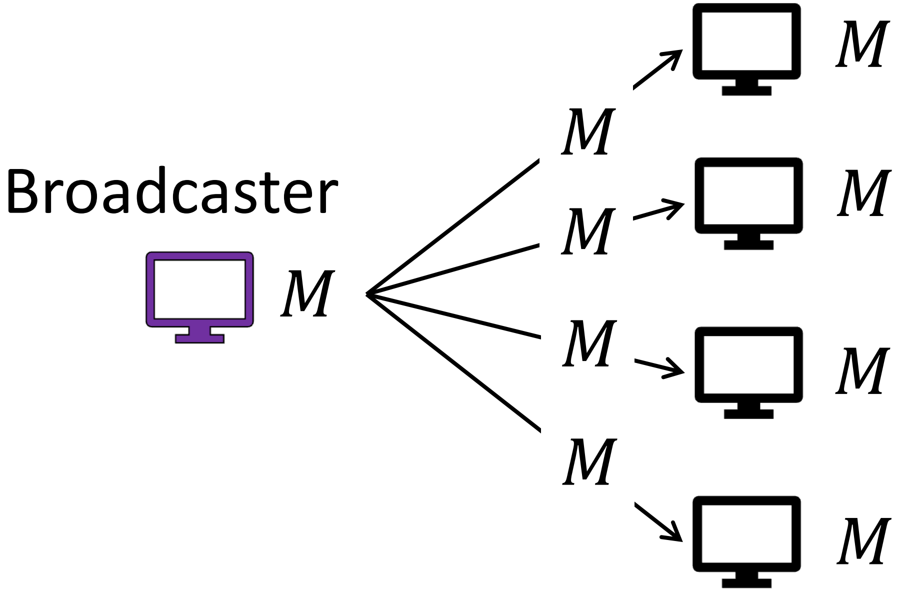
Parties can validate that they received valid shares.

Reliable Broadcast (RBC) [Bracha 84]

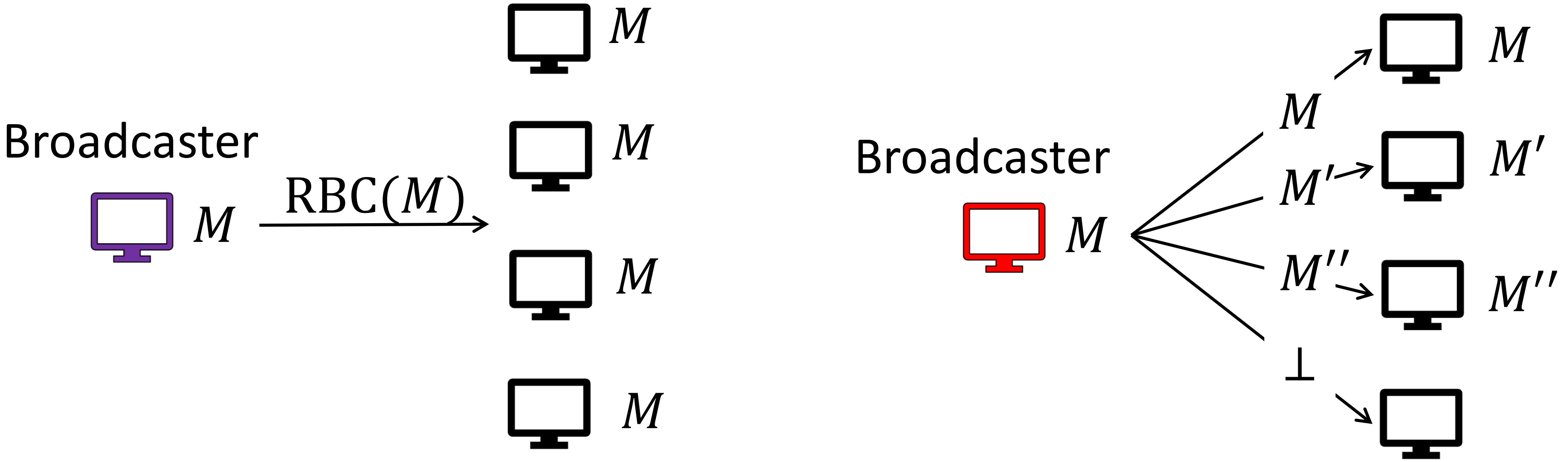


Reliable Broadcast (RBC) [Bracha 84]





Reliable Broadcast (RBC) [Bracha 84]

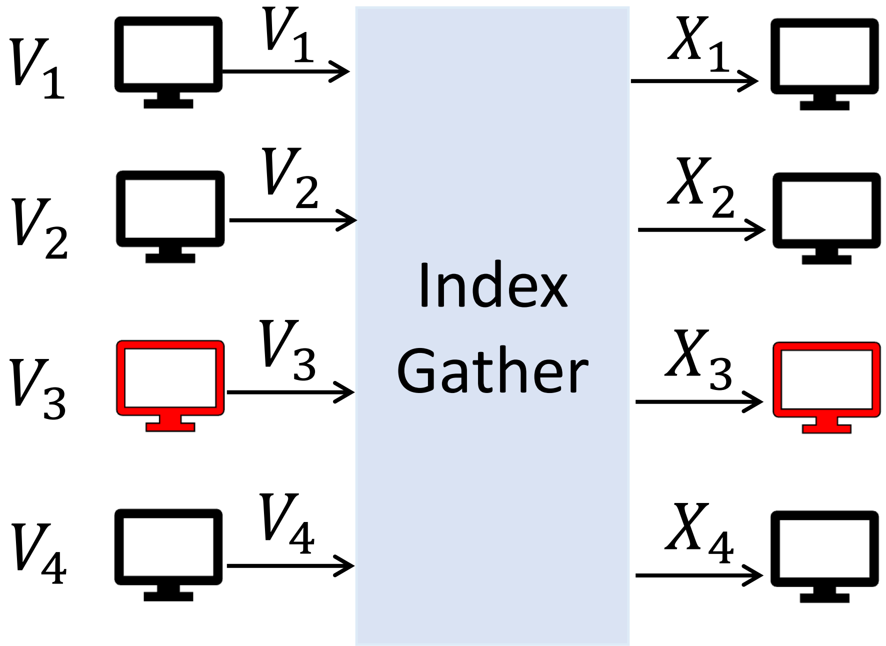


We need asynchronous RBC for long messages [CKLS02, DXR21].

Index Gather

Definition: Asynchronous consensus with weak agreement property

Party validation: Parties validate each other based on the actions taken by others



Termination: Same as ACS*

Validity: $X_i \subseteq [n]$ contains only validated parties

Binding Core:
- When the first honest party outputs
- There exists a core set X of $|X| \geq n - t$
- Everyone outputs a superset of X i.e, $X_j \supseteq X$

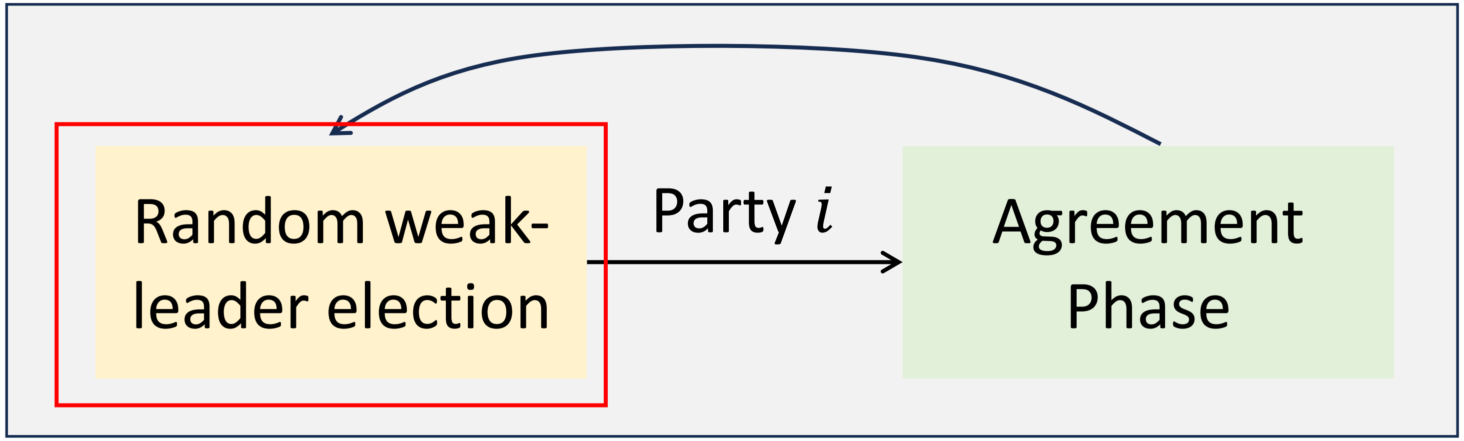
$V_i \subseteq [n]$ is the set of parties party i has locally validated

Earliest gather protocol: [Canetti-Rabin'93]

Prior approach to construct setup free ACS

ACS Construction from Weak Leader Election

Protocol proceed in iterations. Each iteration has two phases:



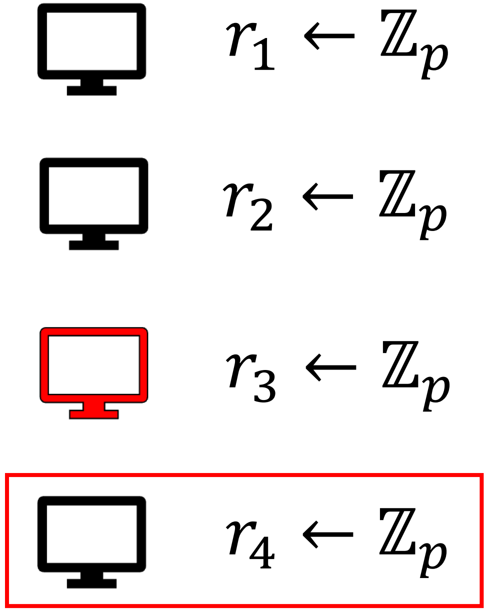
Weak = parties disagree on who the leader is with constant probability

When parties agree on an honest leader, the protocol terminates successfully

Weak-leader election protocol

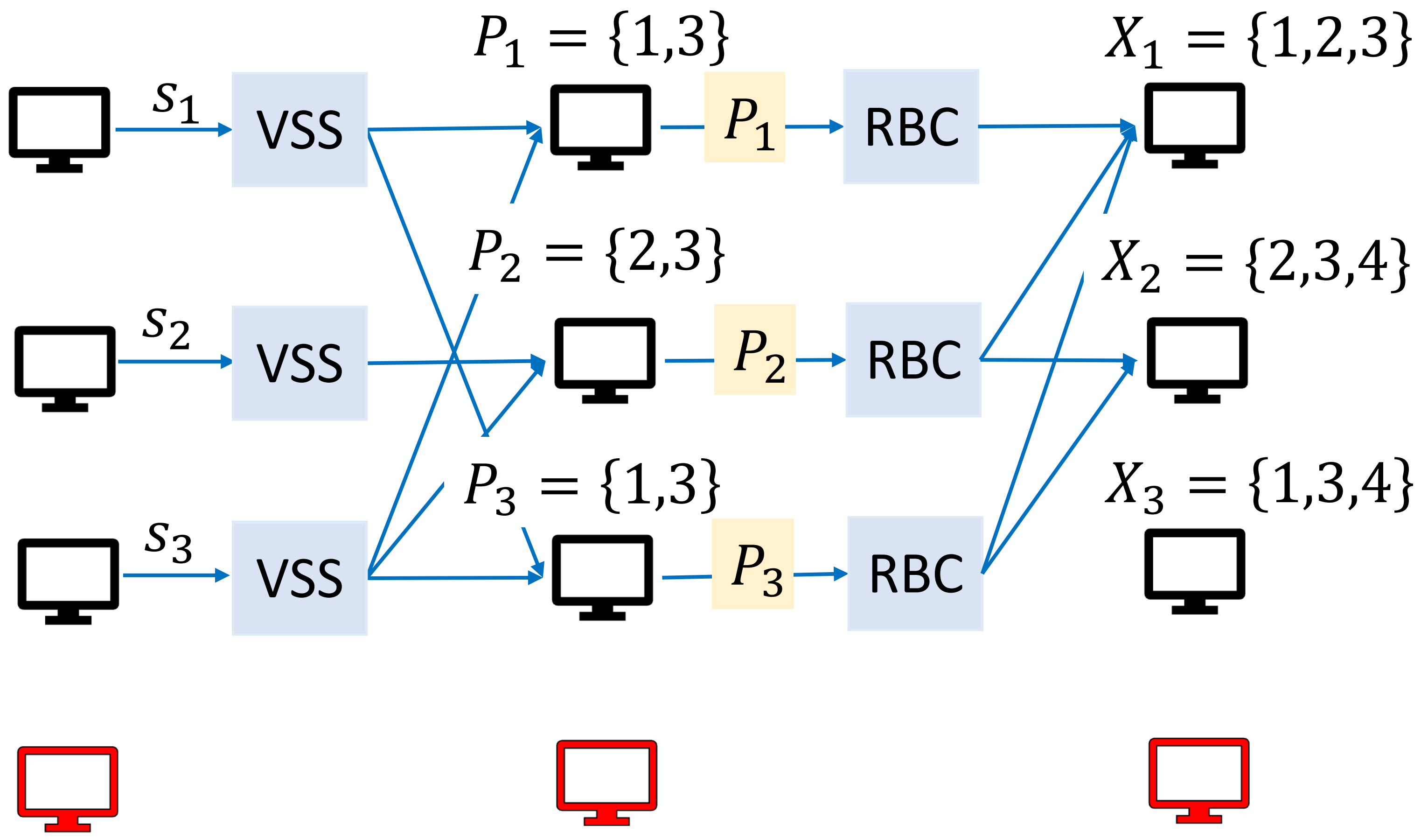
Protocol:
1. Assign each party a random rank
2. Party with highest rank is the leader

Easy with an external common coin!

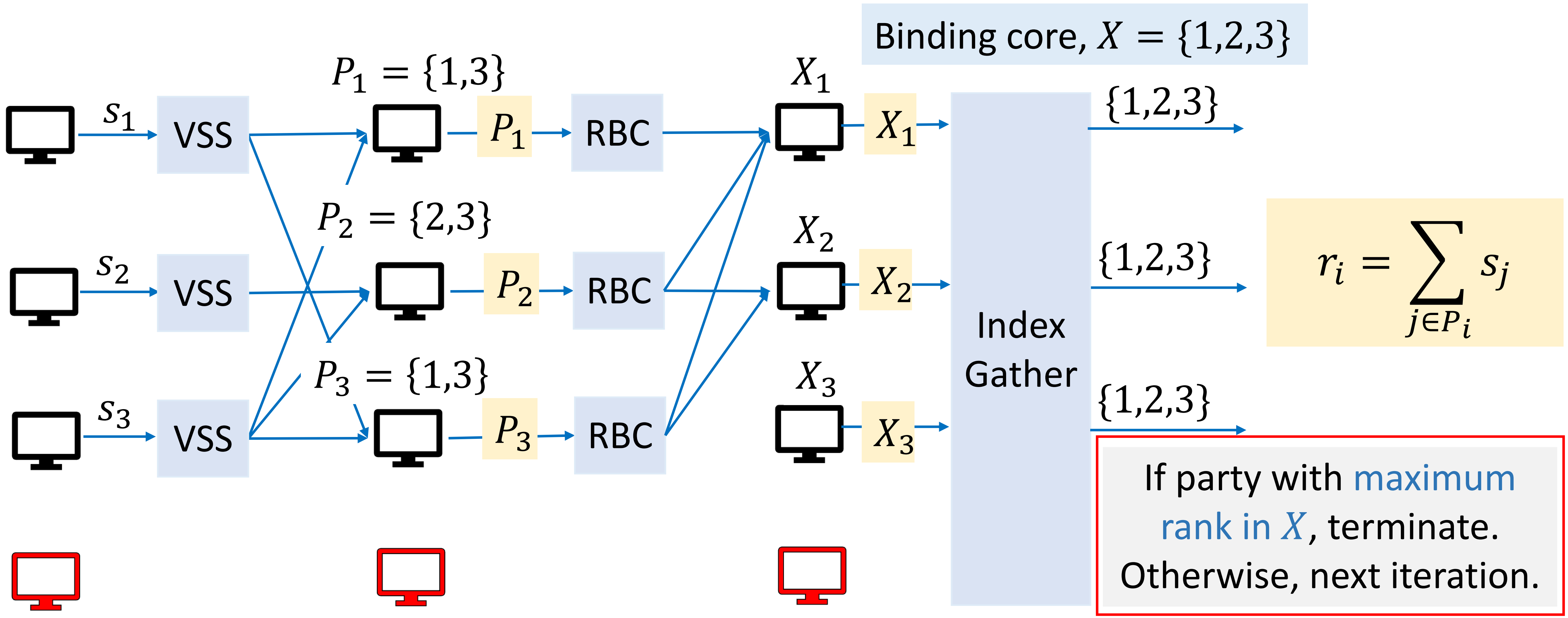


We focus on protocols that do not rely on external common coin

Weak-leader election protocol without external coin



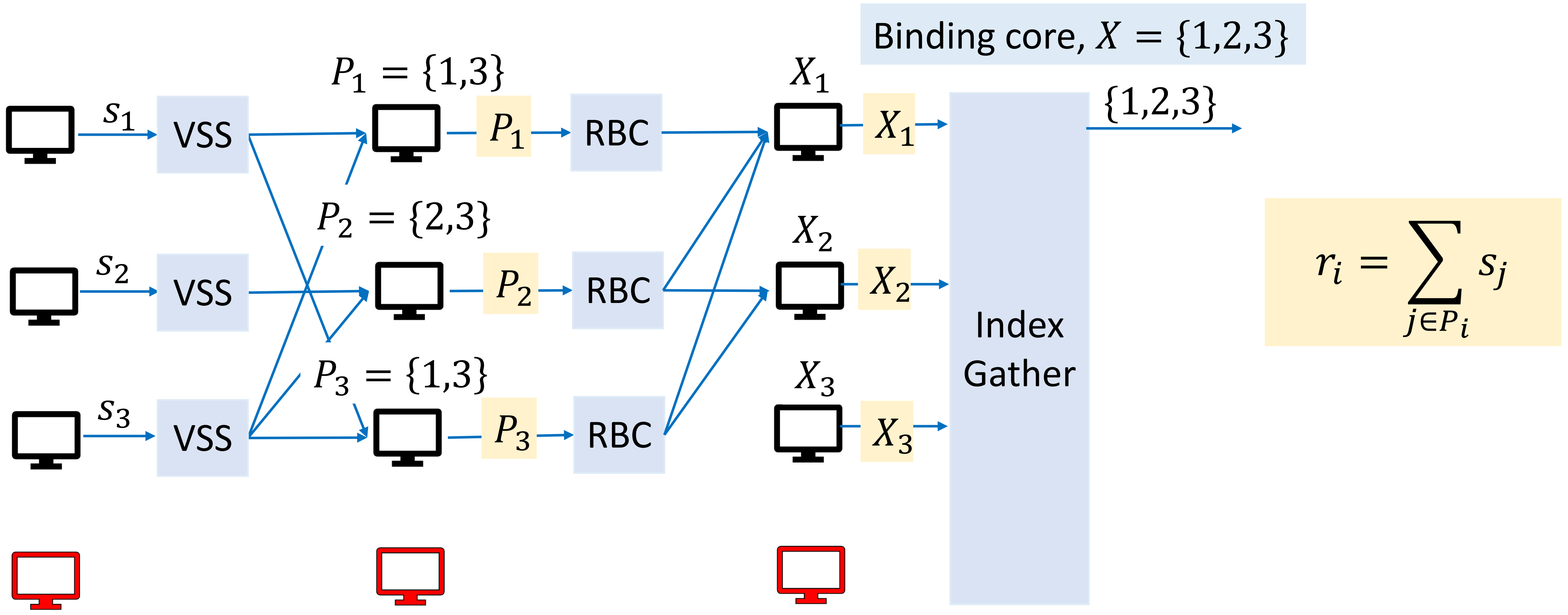
Weak-leader election protocol without external coin



This approach is insecure ☹

Attack: Weak-leader election protocol

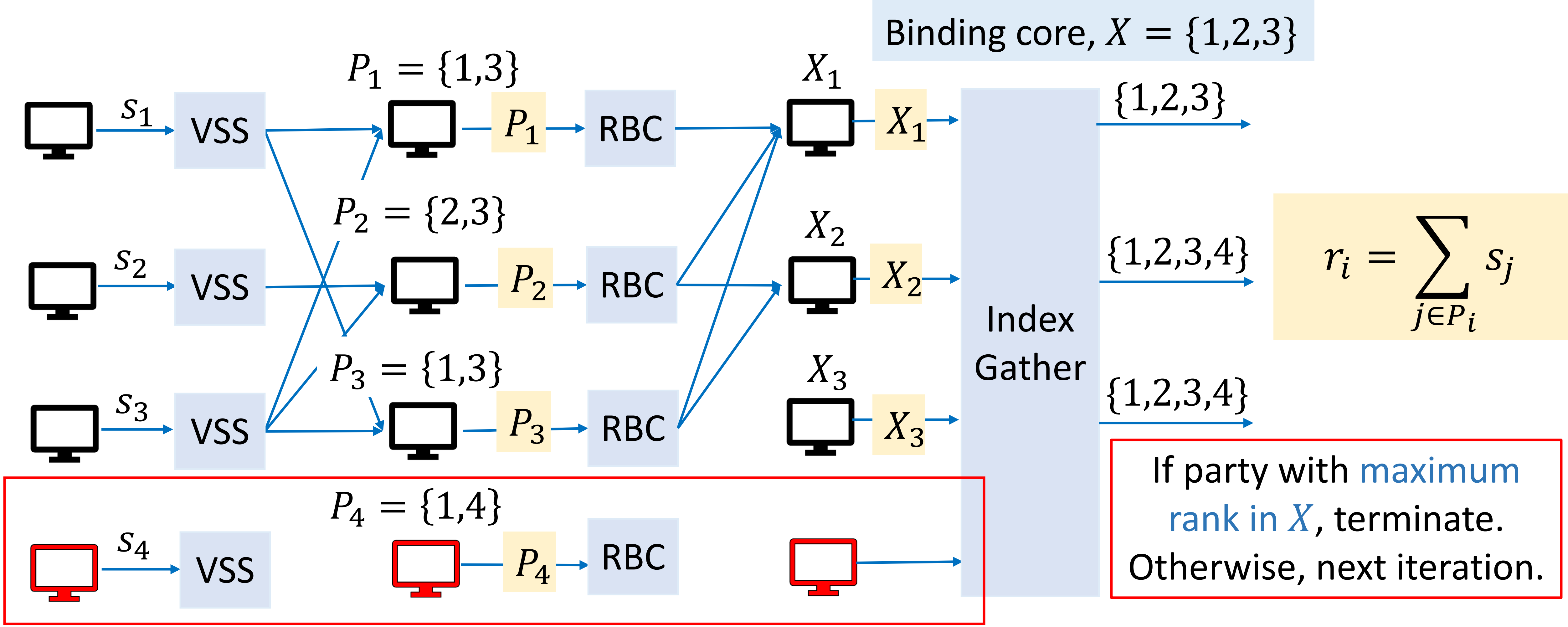
Attack: Weak-leader election protocol



Adversary learns s_1, s_2, s_3 after first gather output

Pick s_4 such that $r_4 = s_1 + s_4$ is the highest rank

Attack: Weak-leader election protocol

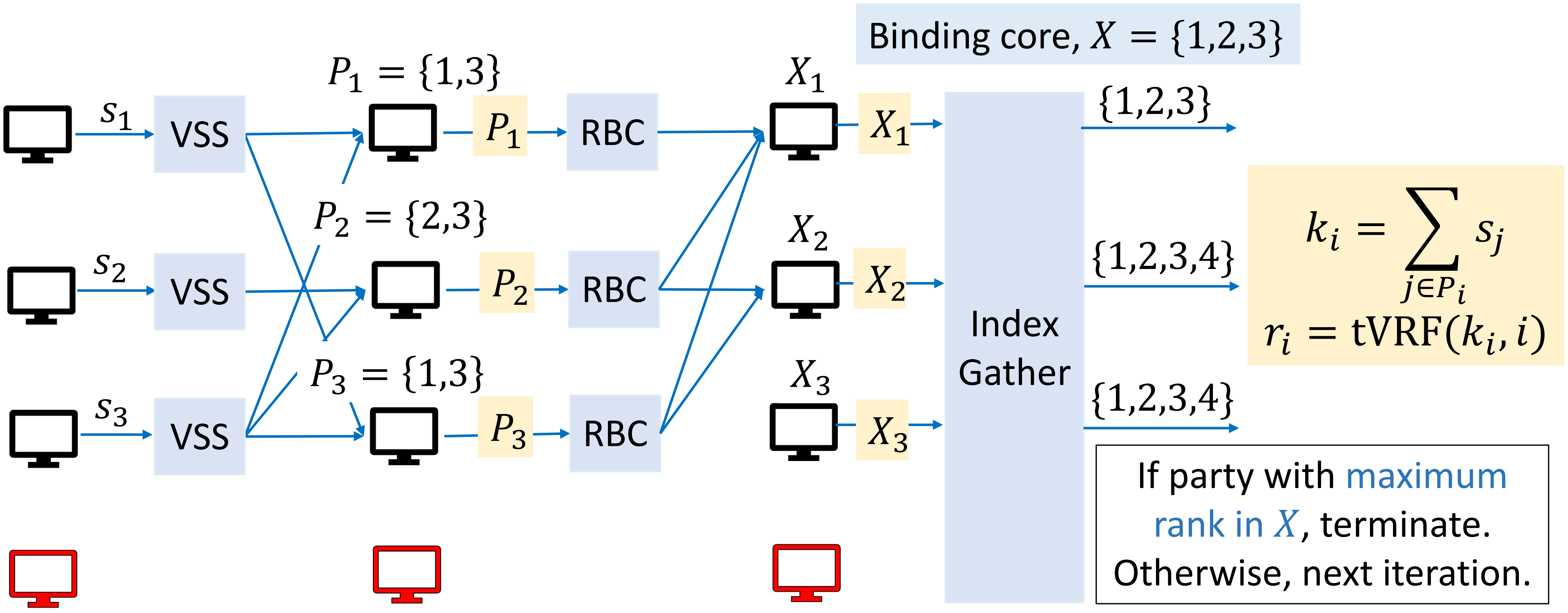


Adversary learns s_1, s_2, s_3 after first gather output

Pick s_4 such that $r_4 = s_1 + s_4$ is the highest rank

Fix 1: Use threshold Verifiable Random Function

Fix 1: Use threshold Verifiable Random Function

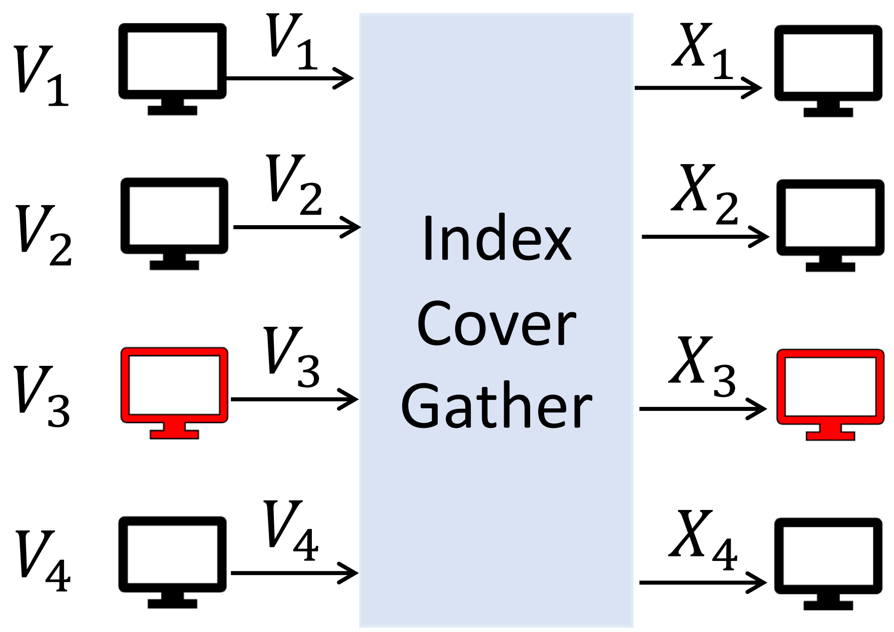


We do not know any threshold VRF without public key cryptography

Our Approach

Index Cover Gather

Index Gather with binding cover property



$V_i \subseteq [n]$ is the set of parties party i has locally validated

Binding Core:

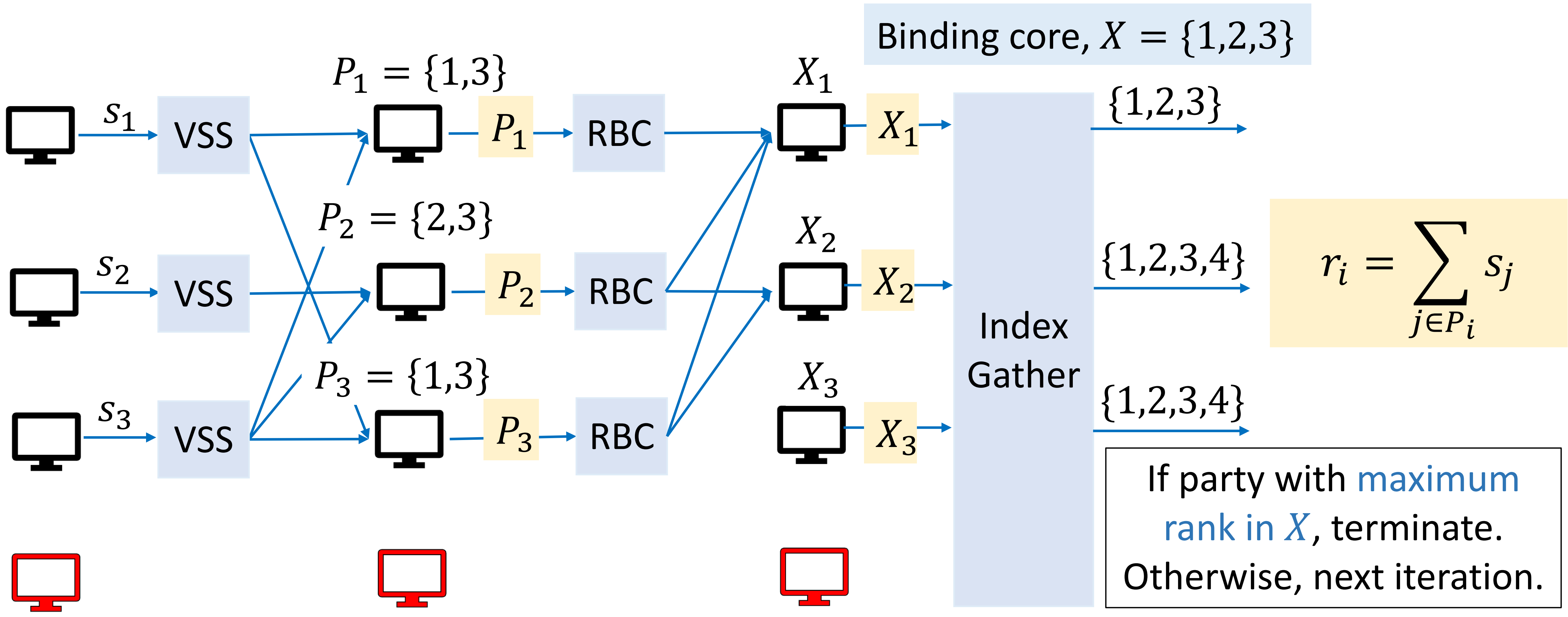
- When the first honest party outputs
- There exists a core set X of $|X| \geq n - t$
- Everyone outputs a superset of $X_j \supseteq X$

Binding Cover:

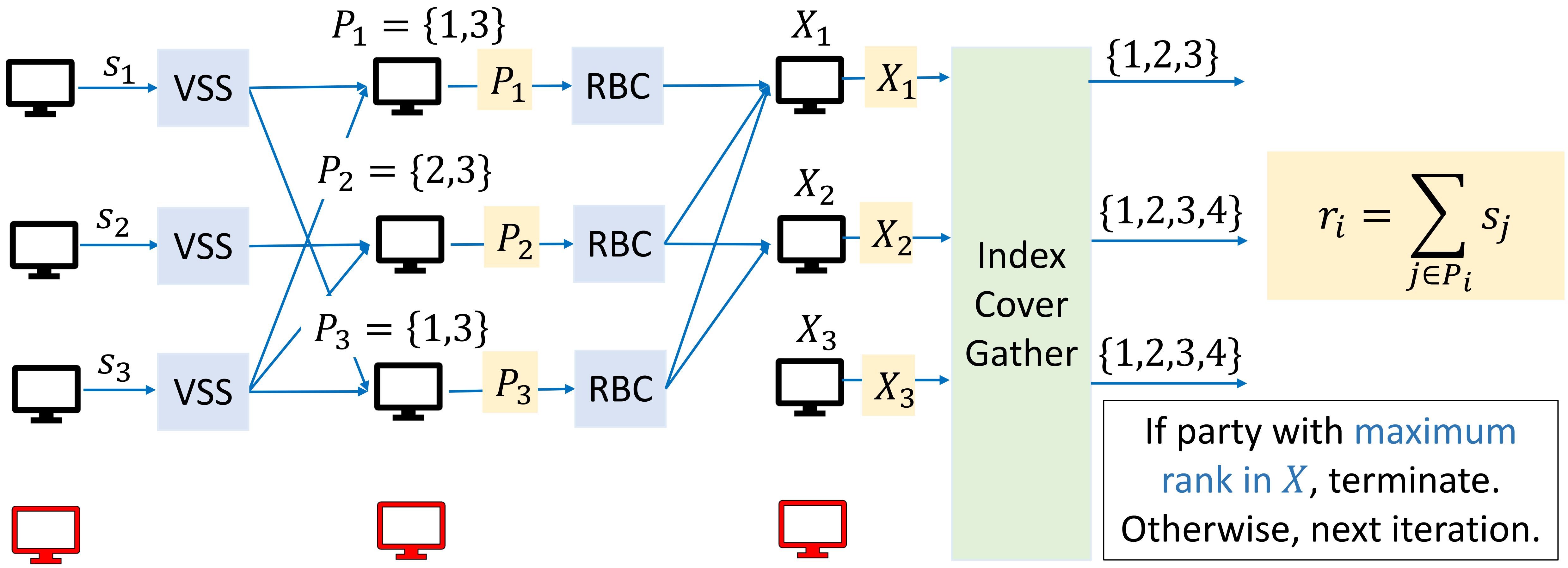
- When the first honest party outputs
- There exists a cover set Y of valid parties
- Everyone outputs a subset of $X_j \subseteq Y$

Index cover gather protocol with $O(n^3)$ communication cost

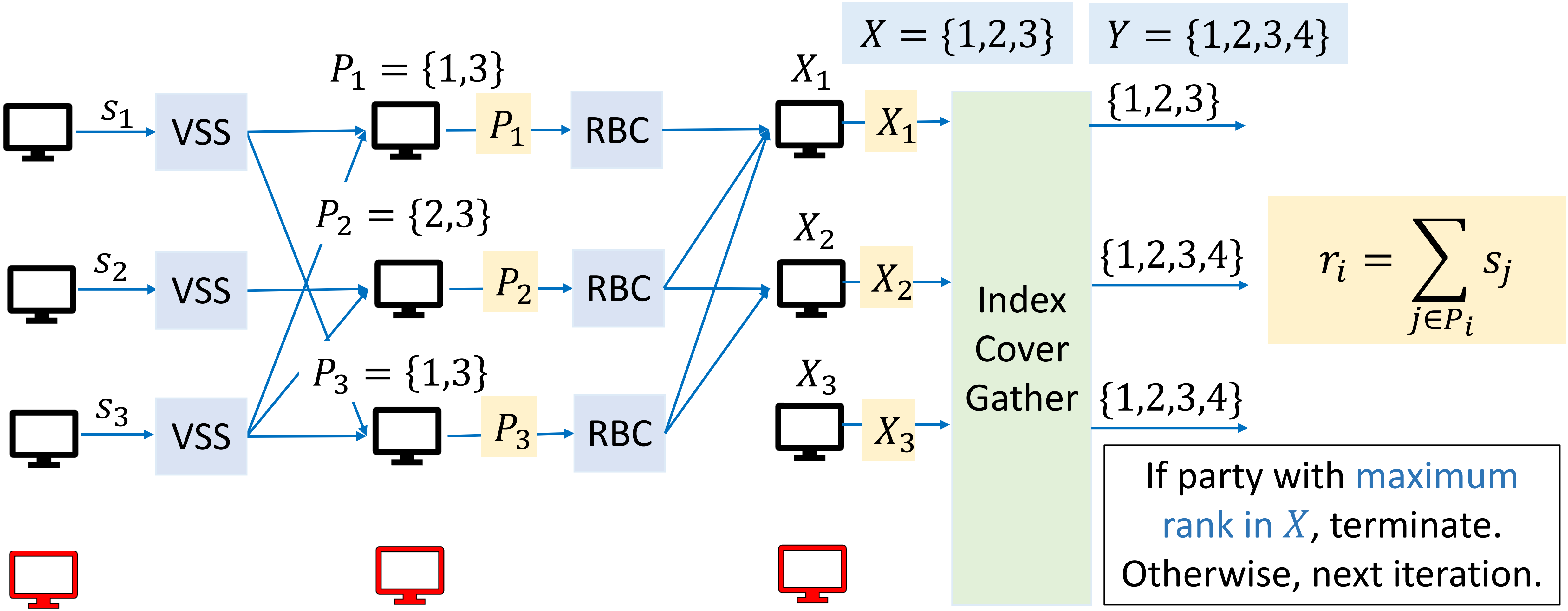
Insecure weak-leader election



Our secure weak-leader election

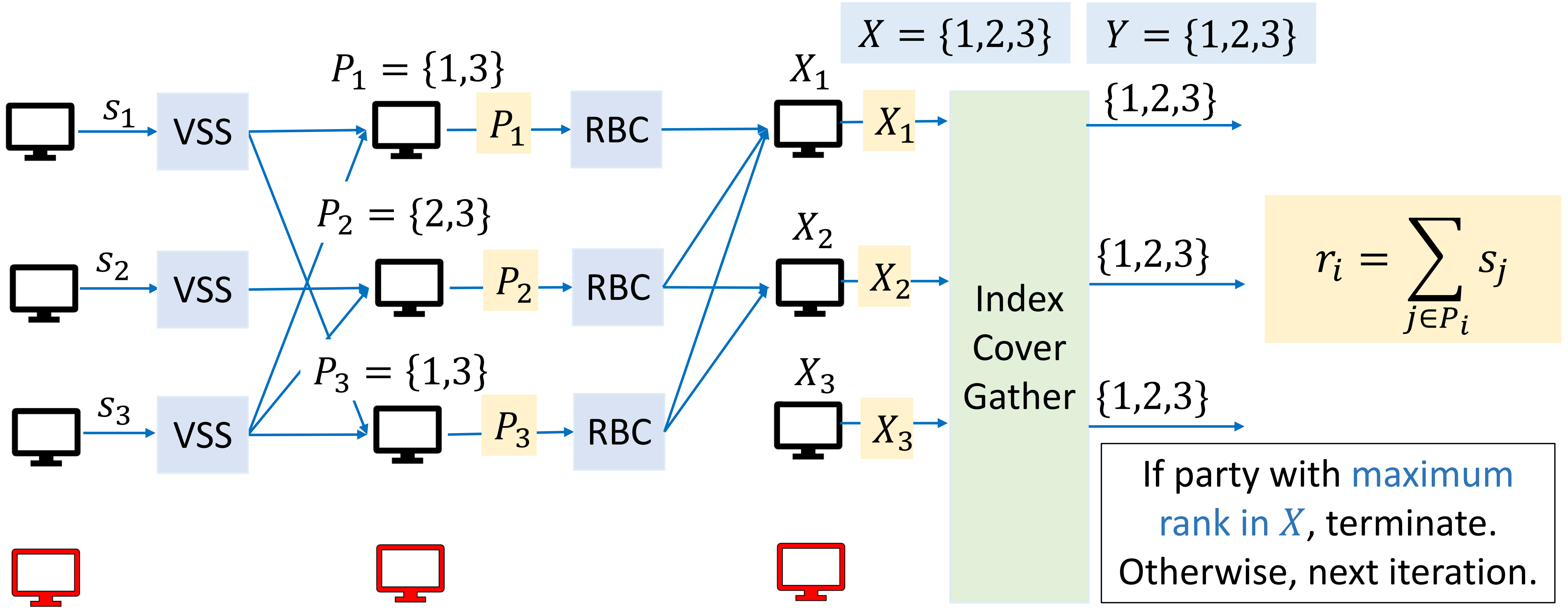


Our secure weak-leader election



Lets try to attack our weak-leader election

Lets try to attack our weak-leader election

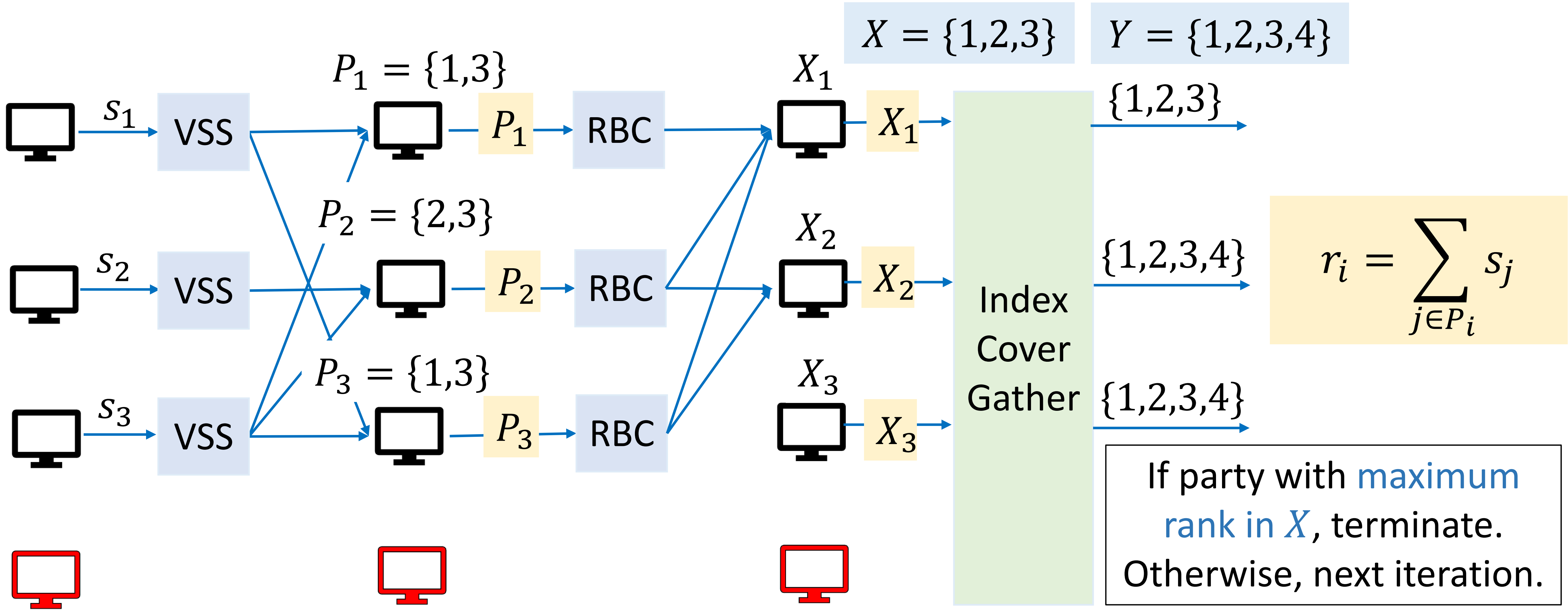


Adversary learns s_1, s_2, s_3 after first gather output

Pick s_4 such that $r_4 = s_1 + s_4$ is the highest rank

Attack is inconsequential!

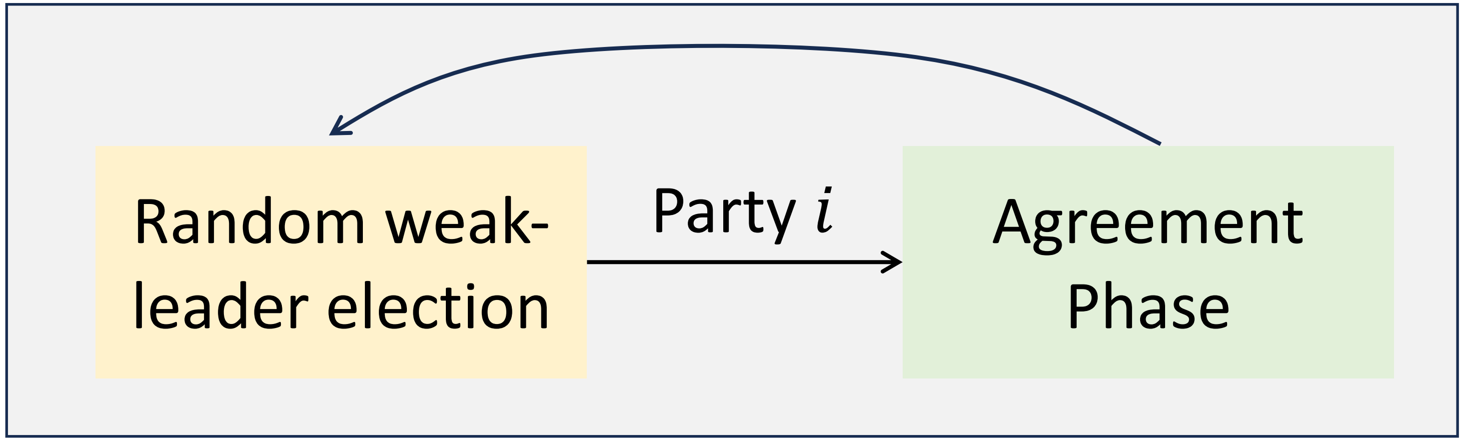
Lets try to attack our weak-leader election



Adversary need to fix s_4 before the index cover gather outputs at any honest party

ACS Construction from Weak Leader Election

Protocol proceed in iterations. Each iteration has two phases:



Contributions:

1. Efficient weak-leader election protocol using only hash functions
2. Concrete optimizations in the agreement phase
3. Simple hash-based weak asynchronous VSS protocol

Implementation and Evaluation

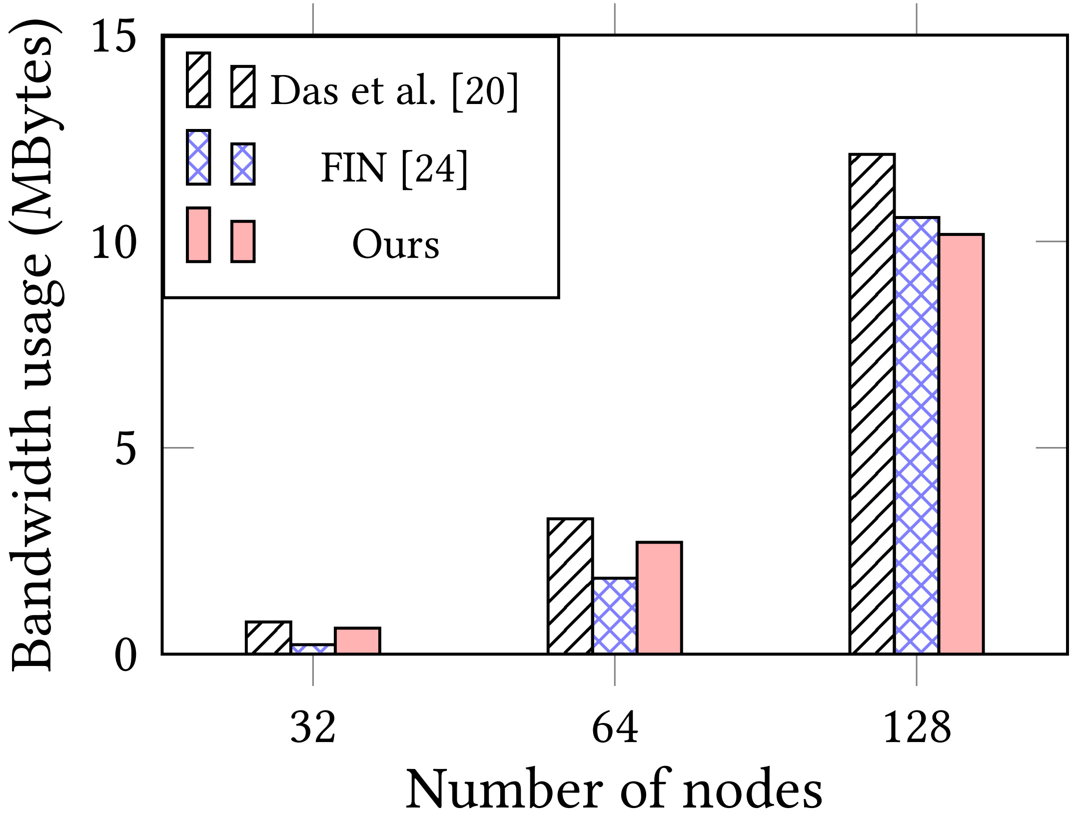
Implementation and Evaluation

- Atop the [DXKR’22] asynchronous DKG code base
- python for networking
- bls12381 field operations for finite field arithmetic
- Up to 128 geo-distributed nodes in AWS across 8 regions
- Baselines:
  1. ACS protocol from [DXKR’22]
  2. ACS from FIN (assumes common-coin)

Lots of scope for optimizing the implementation!

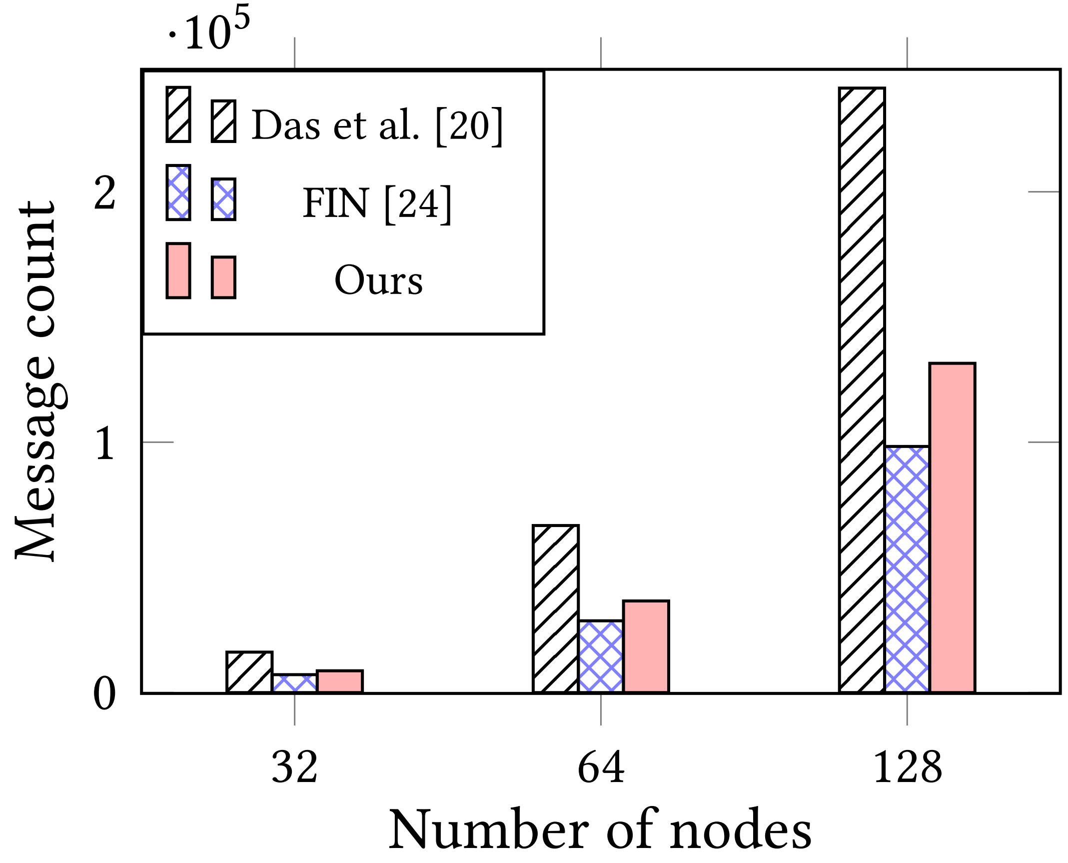
Evaluation: Bandwidth Usage

Definition: Amount of data a party sends during the index ACS protocol



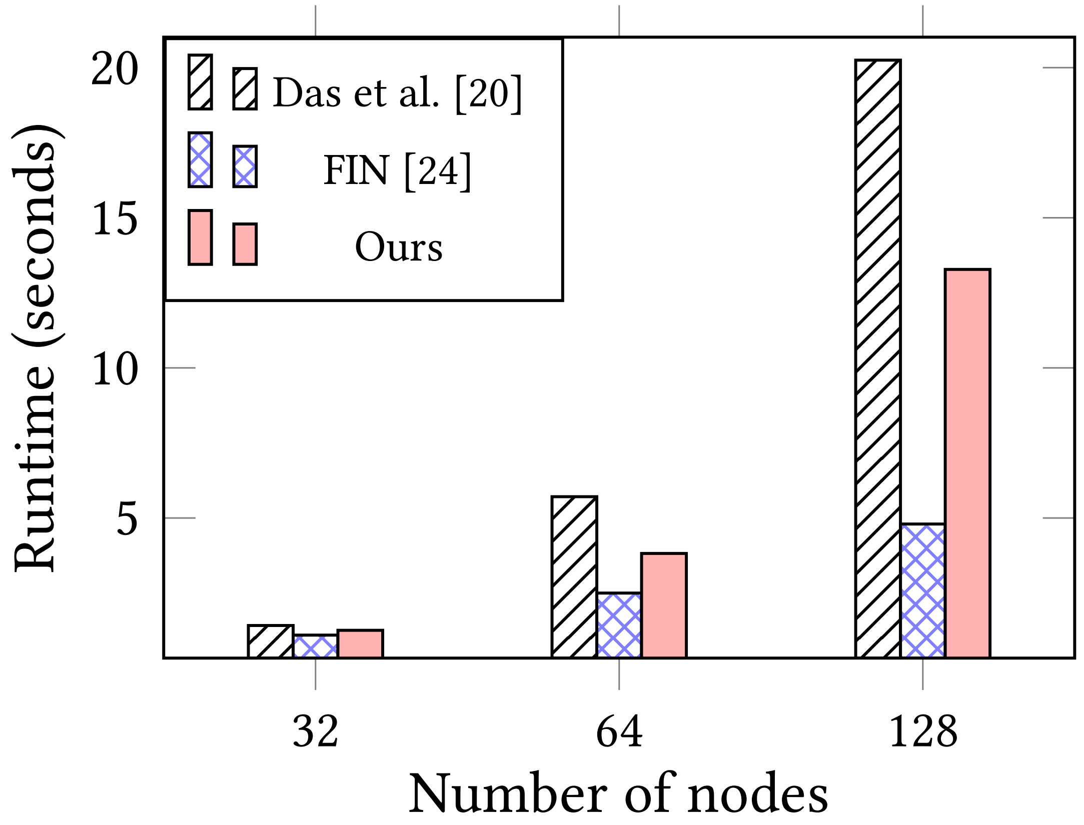
Evaluation: Message Count

Definition: Number of messages a party sends during the index ACS protocol



Evaluation: Runtime

Definition: End-to-end latency of the index ACS protocol



Summary:

- Asynchronous Consensus Protocol assuming only Hash-function
- Optimal fault-tolerance $n \geq 3t + 1$
- $O(\kappa n^3)$ communication cost;
- $O(1)$ round complexity
- Protocol with concrete efficiency

Open Problems

- Improve efficiency or prove lower bounds
- Better implementation (will be generally helpful for the community)

Let’s agree on a common subset, in a flash

Asynchronously, with nothing more than a hash

Resilience has to be perfect

Communication, just cubic

In theory and practice, we hope to make a splash

- *Victor Shoup*

Paper link

Code

Thank You! (souravd2@Illinois.edu)!