Aggios: Aggregator-Based Voting using Proof of Partition

Marius Lombard-Platet and **Doron Zarchy***

University of Luxembourg

ACM AsiaCCS 2026

* Both authors contributed equally.

Roadmap

01

Hidden Partition Problem

Define the combinatorial object

Voters, candidates, hidden assignments

02

Extended Partition Argument (EPA)

Algebraic encoding

Polynomial relations

Zero-knowledge proof system

03

Aggios Voting Application

Off-chain aggregation

On-chain verification

Cost reduction

04

Security & Performance

Formal security properties

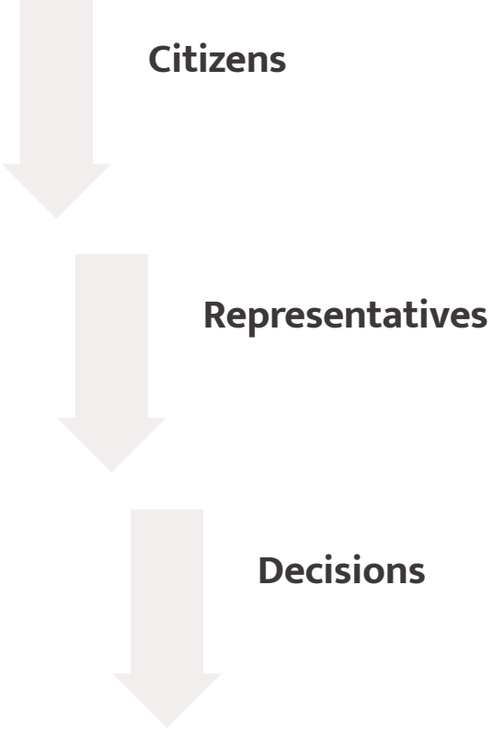
Proof complexity

Benchmarks

Why Voting Scalability Matters

Removing scalability barriers to frequent, publicly verifiable participation.

The Status Quo



Direct Participation

- More responsive
- More expressive
- Publicly verifiable

Does not scale to large populations!

Frequent Participation

+

Public Verifiability

↓

Need a Scalability Solution

Why Voting Scalability Matters

Removing scalability barriers to frequent, publicly verifiable participation.

The Status Quo

Citizens

Representatives

Decisions

Direct Participation

- More responsive
- More expressive
- Publicly verifiable

Does not scale to large populations!

Frequent Participation

+

Public Verifiability

↓

Need a Scalability Solution

- Concrete example nowadays: DAO with 100,000 members is not practical if communication costs grow with the number of voters.
- Goal: Remove scalability barriers to frequent, publicly verifiable voting. Costs should depend on the number of candidates — not the number of voters.

The Core Idea: Aggregate Off-Chain, Prove On-Chain

Instead of publishing every vote, voters delegate to aggregators who collect ballots privately — then publish only tally **M** and proof Π_{EPA}.

Voters Delegate

v1→Alice, v2→Bob, v3→Alice, v4→Alice, v5→Bob, v6→Carol, v7→Carol, v8→Alice, v9→Bob, v10→Alice

Aggregators Collect

Alice: 5 votes

Bob: 3 votes

Carol: 2 votes

Publish M & Π_{EPA}

Bulletin Board

Public observers verify the proof without ever learning individual voter assignments.

The Hidden Partition Problem

Introduce the combinatorial problem before discussing cryptography.

Voter Tokens: v1, v2, v3, v4, v5, v6, v7, v8, v9, v10

Alice

Size: 5

{v1, v3, v4, v8, v10}

Bob

Size: 3

{v2, v5, v9}

Carol

Size: 2

{v6, v7}

- Partition
  - Every voter belongs to exactly one subset
  - Subsets should cover the whole domain
- Hidden Partition:
  - Membership to subsets remains hidden
- Additional requirement:
  - Subset sizes are public

A New ZK Primitive: Hidden Partition Proofs

EPA is the primary contribution of this work — a new zero-knowledge primitive.

Existing Primitives

- Membership proofs
- Inclusion proofs
- Hidden subvector proofs

Missing Primitive

- Hidden partition proofs
  - Prove subsets are disjoint
  - Prove subsets cover the full domain
  - Prove public size constraints

From Partitions to Polynomials

Translate the combinatorial partition problem into an algebraic one.

The Partition

[v1, v2, v3, v4, v5, v6, v7, v8, v9, v10]

Alice

{v1, v3, v4, v8, v10}

Bob

{v2, v5, v9}

Carol

{v6, v7}

The Polynomial Encoding

Each object is represented as a polynomial over a finite field.

Object	Polynomial
Full vote vector (all n voters)	$C(X)$
Alice's subset I_A	$V_A(X)$
Bob's subset I_B	$V_B(X)$
Carol's subset I_C	$V_C(X)$

$C(X)$ encodes the committed vote vector. Each $V_j(X)$ encodes the voters supporting candidate j.

Encoding Hidden Subsets

A vanishing polynomial is the algebraic representation of a set of voters.

Set	Vanishing Polynomial
Full domain H $H = \{v1, v2, ..., v10\}$	$Z_H(X) = \prod_{i \in H}(X - \omega_i)$ roots = all voters
Hidden subset I_A (Alice) $I_A = \{v1, v3, v4, v8, v10\}$	$Z_{I_A}(X) = \prod_{i \in I_A}(X - \omega_i)$ roots = Alice's voters only

Just as $Z_{I_A}(X)$ represents Alice's hidden subset, $Z_H(X)$ represents the entire voting domain.

Why Do We Need the Consistency Constraint?

The partition alone is not enough — we must also connect it to the committed vote vector.

- Given polynomial C(X)
- $Z_{I_j}(X)$ — identifies the voters assigned to candidate j
- $V_j(X)$ — represents the corresponding candidate-j subvector
- w_j — is the public label associated with candidate j

We need aconsistency relation will enforce that $Z_{I_j}(X), V_j(X)$, and w_j are mutually consistent with $C(X)$ — for every candidate j.

EPA: Enforcing a Hidden Partition

These polynomial relations are exactly the partition statement.

Meaning	Constraint
subset I_j lies inside $H \Rightarrow$	$Z_{I_j} \mid Z_H$
subsets are disjoint and cover the entire domain $\Rightarrow$	$\prod_j Z_{I_j} = Z_H$
subset I_j has the prescribed public size $\Rightarrow$	$\deg(Z_{I_j}) = m_j$
V_j matches C on the hidden subset $I_j \Rightarrow$	Z_{I_j} divides $(C - w_j \cdot V_j)$

The Relation We Want to Prove

Formally define the statement that EPA proves.

Given

- Vector C (commits to the entire vote vector)
- Public candidate labels $w_1, ..., w_k$
- Hidden subsets $I_1, ..., I_k$

Prove:

1. Compute subset sizes $m_1, ..., m_k$
2. $|I_j| = m_j$ — each subset has the correct size
3. $I_i \cap I_j = \emptyset$ — subsets are disjoint
4. $\bigcup_j I_j = [n]$ — subsets cover all voters
5. Every voter in I_j is associated with label w_j in the committed vector.

The flow shows how the full vote vector is committed once, then interpreted through a hidden partition.

EPA Protocol

Connecting the mathematical relation to the actual proof protocol.

Prover

Public: $(C,\ w_1, \ldots, w_k)$

Witness: $(I_1, \ldots, I_k)$

1. Tally the vote and obtain $m_1, ..., m_k$
2. Construct $V_1, \ldots, V_k$ and $Z_{I_1}, \ldots, Z_{I_k}$
3. Commit to these polynomials (KZG): $\hat{V}, \hat{Z}$
4. Construct EPA proof π — auxiliary quotient polynomials (KZG proofs)
5. Send $m_1, ...m_k, \hat{V}, \hat{Z}, \pi$ to the verifier

EPA Protocol

Connecting the mathematical relation to the actual proof protocol.

Prover

Public: $(C,\ w_1, \ldots, w_k)$

Witness: $(I_1, \ldots, I_k)$

1. Tally the vote and obtain $m_1, ..., m_k$
2. Construct $V_1, \ldots, V_k$ and $Z_{I_1}, \ldots, Z_{I_k}$
3. Commit to these polynomials (KZG): $\hat{V}, \hat{Z}$
4. Construct EPA proof π — auxiliary quotient polynomials (KZG proofs)
5. Send $m_1, ...m_k, \hat{V}, \hat{Z}, \pi$ to the verifier

Verifier

Receives commitments and proof. Then checks:

- Partition relation: $\prod_j Z_{I_j} = Z_H$
- Size constraints: $\deg(Z_{I_j}) = m_j$ via $X^{d-m_j} Z_{I_j}$
- Consistency of C and subvector V_j:
  - Z_{I_j} divides $(C - w_j \cdot V_j)$
  - Divisibility checked via quotient polynomials

All identities verified using KZG pairing checks (batched to $k + 1$ pairings)

EPA Protocol

Connecting the mathematical relation to the actual proof protocol.

Prover

Public: $(C,\ w_1, \ldots, w_k)$

Witness: $(I_1, \ldots, I_k)$

1. Tally the vote and obtain $m_1, ..., m_k$
2. Construct $V_1, \ldots, V_k$ and $Z_{I_1}, \ldots, Z_{I_k}$
3. Commit to these polynomials (KZG): $\hat{V}, \hat{Z}$
4. Construct EPA proof π — auxiliary quotient polynomials (KZG proofs)
5. Send $m_1, ...m_k, \hat{V}, \hat{Z}, \pi$ to the verifier

Verifier

Receives commitments, then checks:

- Partition relation: $\prod_j Z_{I_j} = Z_H$
- Size constraints: $\deg(Z_{I_j}) = m_j$ via $X^{d-m_j} Z_{I_j}$
- Consistency of C and subvector V_j:
  - Z_{I_j} divides $(C - w_j \cdot V_j)$
  - Divisibility checked via quotient polynomials

All identities verified using KZG pairing checks (batched to $k + 1$ pairings)

EPA Security Guarantees

Completeness — Honest partitions are accepted.

Knowledge Soundness — An accepted proof implies knowledge of a valid hidden partition satisfying the constraints.

Zero Knowledge — The verifier learns nothing about the hidden partition beyond the public statement

Aggios Protocol

01

Setup

Generate SRS; encode candidate labels as public scalars $w_1, \ldots, w_k$

02

Registration

Voter i publishes voting key pk_i on-chain; aggregator commits to the registered voter-token vector C

03

Voting

Voter sends choice j privately to the aggregator off-chain

Voter receives a proof of inclusion from the aggregator

04

Posting

Aggregator publishes tally $M = (m_1, \ldots, m_k)$ and proof Π_{EPA}

05

Verification

Validators verify Π_{EPA} against C and M

Security Properties

- **Tally Integrity:** EPA knowledge soundness + KZG binding
- **No Double Voting:** Partition disjointness enforced by EPA soundness
- **Eligibility:** Voter tokens committed at registration
- **External Privacy:** Zero-knowledge of EPA + hidden index sets
- **Aggregator Privacy (basic Aggios):** None — aggregator learns each vote
- **Aggregator Privacy (Aggios-Split):** Deniability against up to t colluding aggregators

Aggios-Split: Hiding Votes from Aggregators

Basic Aggios

1 **Voter**

2 **Aggregator**

The aggregator learns the vote.

Aggios-Split

Vote information is distributed across multiple aggregators.

$A_1 \dots A_t$

No coalition of up to t aggregators can determine the vote.

Key Points

- Basic Aggios provides tally integrity but the aggregator learns the vote.
- Aggios-Split distributes vote information across multiple aggregators.
- Privacy holds against up to t colluding aggregators.
- Tradeoff: additional communication and coordination.

Aggios-Split strengthens privacy while preserving the EPA-based tally verification framework.

Performance

EPA Proof Complexity

- Benchmarks: up to 2^18 elements (~262,000 voters)
- Prover time: under 60s
- Verification: always under 300ms
- Proof size: 4k − 2 G_1 elements + k G_2 elements

Takeaways

- **EPA: A New ZK Primitive** — EPA provides an efficient proof system for hidden partitions with public constraints. Potential applications extend beyond voting.
- **Voting Aggregation as a Partition Problem** — Aggios reduces tally correctnes to proving a hidden partition of a committed vote vector.
- **Costs Scale with Candidates, Not Voters** — Verification and on-chain communication depend primarily on the number of candidates (k), rather than the number of voters (N).

Thank You

Questions?