Aggregator-Based Voting using proof of Partition

Marius Lombard-Platet[1] and Doron Zarchy[1⋆]

APSIA, SnT, University of Luxembourg, Luxembourg
first.last@uni.lu, doronz@gmail.com

Abstract. We present Aggios, a scalable and privacy preserving proxy voting system designed for frequent and large-scale elections such as Decentralized Autonomous Organizations (DAO), when storing votes on the bulletin board is expensive. To this end, Aggios introduces 'aggregators': entities to which voters delegate their votes, and who then post their batched proofs on the public ledger. Aggios achieves strong integrity guarantees: only authorized voters can vote, votes are counted correctly, voters are assured their vote is counted.

At the core of Aggios, lies a novel zero-knowledge argument, which we call the Extended Partition Argument (EPA), that allows a prover to demonstrate that a committed vector can be decomposed into multiple disjoint "subvectors" forming a partition, each subvector of public (or not) sizes. The argument is compatible with a universal SRS, does not require precomputation, and offers efficient proving and verification complexity. We prove security of the EPA in the algebraic group model. Our implementation of EPA shows suitability of the argument even for very large vectors.

Using the EPA as the central block to Aggios, we show that our voting scheme is at least 512 times more compact than naive casting of N votes, and can even be size-independent of the number of voters in the optimal case, thus offering a practical route to frequent and privacy-preserving voting at scale.

1 Motivation and Introduction

Maintaining communication efficiency is a critical challenge in large-scale electoral processes, particularly when voting is frequent and conducted on public bulletin boards. For instance, Decentralised Autonomous Organisations (DAOs) rely on frequent voting to determine the organisation's policy [27]. However, demands for transparency make off-chain voting DAOs significantly less popular than their on-chain counterparts, receiving 87% less funding on average [4]. However, publicly verifiable voting systems on bulletin boards may struggle with scalability, thus requiring significant voting costs and centralizing voting power [15]. [15] shows that DAOs have a low participation rate, especially among small token holders. The current solution, token delegation (e.g. [29]), only exacerbates the centralization issue [40].

⋆ Both authors contributed equally to this work.

We address this bottleneck by introducing a *new voting protocol* in which ballots are *aggregated* in groups before only the group counts (and a succinct proof of correctness) are published, decreasing on-chain costs by at least a factor of 500. Addressing DAOs' practical constraints also provides a building block for future liquid democracy and direct civic engagement that requires repeated, publicly verifiable voting with optional privacy [32].

Novel Partition Argument. Central to our approach is a powerful partition argument in zero-knowledge. In this scheme, a prover demonstrates that a committed vector, e.g., voting tokens, can be partitioned into multiple *disjoint* parts (subvectors[1]), whose sizes are publicly known. Formally, if we know of a committed vector $\vec{c}$, each subvector is intended to have m_j elements (with m_j being public and $\sum m_j = |\vec{c}|$), and the prover must show in zero-knowledge that these subvectors indeed contain exactly m_j distinct elements of $\vec{c}$, and that collectively the subvectors cover the entire vector, i.e. the subvectors form a partition of $\vec{c}$.

The partition argument is modular, in the sense that the sizes m_j can be individually made private if the prover wishes so. Moreover, the prover can also prove that it is not the subvectors $\vec{v_j}$ that form a partition of vector $\vec{c}$, but rather the $w_j \cdot \vec{v_j}$, where $\cdot$ represents the scalar multiplication of w_j with the subvector $\vec{v_j}$. This is useful for our voting scheme, where w_j will encode the vote for a specific candidate, and $\vec{v}_j$ the subvector of people voting for that candidate.

We describe our protocol and show its correctness, soundness, and zero-knowledge properties. We also implement and benchmark it, proving its efficiency over a wide range of parameters, while keeping the verifier time below 300 ms. Proving time remains mostly under a minute, even for 2^{18} elements.

While we only use our partition argument in relation to the voting scheme below, we also note that it can be used in a variety of scenarios, and for instance could be applied to proofs of mixing in specific blockchains like Monero, where each new block must prove that the sum of the inputs is correctly redistributed.

Aggregator-Based Voting (Aggios). Using our partition argument, we introduce a new voting system called *Aggios*, which achieves significantly improved scalability: its complexity of on-chain communication is independent of the total number of voters. Aggios operates as follows:

1. *Aggregator Model.* Each voter selects a trusted aggregator (from multiple options) to cast votes on their behalf. The aggregator *collects* many voter tokens.
2. *Subvector Partitioning per Candidate.* The aggregator constructs a single commitment covering all vote tokens, then presents a *partition* of this commitment—one subvector per candidate, each of size $\{m_j\}$ with m_j the number of votes for that candidate, according to public parameters.
3. *Public Verifiability.* Only an aggregate vote count (per candidate) plus a short zero-knowledge proof is published. Any validator can verify (a) the

[1] For $\vec{c} = (c_1, \ldots, c_n)$ a vector, $\vec{v} = (v_i)_{i \in S}$ is said to be a subvector of $\vec{c}$ over S iff $S \subset [n]$ and $\forall i \in S, v_i = c_i$. Note that the indexes of $\vec{v}$ are the ones in S.

aggregator included exactly the correct number of tokens for each candidate, (b) no tokens appear more than once, and (c) all tokens have been counted. This prevents fraud or double voting, yet conceals individual voter choices.

Considering a scenario in which both voters and aggregators can deviate from the protocol, our scheme is sound, as a dishonest aggregator cannot tamper with the number of tokens allocated to any candidate without invalidating the zero-knowledge proof. It also makes minimal use of 'on-ledger' data: in the best-case scenario, because each aggregator's proof is succinct, Aggios operates with *amortized sublinear communication* in the number of voters. In particular, for k candidates and L aggregators, the on-chain proof size is $O(kL)$, independent of the total number of voters served by each aggregator. We first describe Aggios, which is a proxy voting system in which the aggregators learn the votes. Finally, we present Aggios-Split, in which vote deniability is enforced even against colluding aggregators.

The rest of this paper is organized as follows: Section 2 introduces the preliminaries, before introducing security of our Extended Partition Argument in Section 3, proving its security in section 4, and benchmarking it in Section 5. Then, we introduce our voting schemes, Aggios and Aggios-Split, in Section 6, and analyse their security in Section 7. Section 8 concludes by reviewing related work.

2 Preliminaries

2.1 Security Parameters and Notations

Let $\lambda \in \mathbb{N}$ denote the security parameter. A function $neg : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all $c > 0$, there exists k_0 such that for all $k > k_0$, $neg(k) < \frac{1}{k^c}$.

A bilinear group is a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive cyclic groups of prime order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate bilinear map, and g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ respectively. The finite field with p elements is denoted by $\mathbb{F}_p$ and abbreviated as $\mathbb{F}$. For all $a \in \mathbb{F}$ and $\gamma \in \{1, 2, T\}$, the group element $a \cdot g_\gamma \in \mathbb{G}_\gamma$ is denoted by $[a]_\gamma$ (or sometimes by $\mathbf{a}$) and represents the commitment to the field element a in $\mathbb{G}_\gamma$. We denote the set $1, ..., n$ by $[n]$. We denote the set of polynomials over $\mathbb{F}$ of degree at most d as $\mathbb{F}_{\leq d}[X]$. The Structured Reference String (SRS) of a trusted setup commitment scheme is the set of powers $([x]_\gamma, \ldots, [x^d]_\gamma)_{\gamma \in \{1,2\}}$ generated at setup phase, with x a secret scalar. Note that the setup can be distributed [39]. The vanishing polynomial over set I is $Z_I(X) \leftarrow \prod_{i \in I}(X - i)$.

2.2 Algebraic Group Model

We analyze the soundness of our partition argument in the Algebraic Group Model (AGM) [16]. In the AGM, whenever someone outputs a group element $y \in \mathbb{G}_\gamma$ with $\gamma \in \{1, 2\}$, they also output a representation of y in the form of a linear

combination of SRS elements. For instance, given a commitment $[P(X)]_1 = \left[\sum p_i X^i\right]_1$ and the SRS $[x]_1, \ldots, \left[x^d\right]_1$, the algebraic adversary receives $[P(X)]_1$ as well as $\{(p_i, \left[x^i\right]_1)\}$, and can therefore reconstruct P.

2.3 The KZG Polynomial Commitment Scheme

Our constructions are based on the KZG polynomial commitment scheme [22]. KZG polynomial commitment consists of algorithms (KZG.Setup, KZG.Commit, KZG.Open, KZG.Verify) such that:

- $(\text{srs}, x) \leftarrow \mathsf{Setup}(\mathsf{pp})$: given the public parameters, perform the trusted setup, output a structured reference string $\text{srs} = \{\left[x^j\right]_{1,2}\}_{j=0}^{\mathsf{pp}.d}$ where x is a secret.
- $\mathrm{C} \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{srs}, P(X))$: outputs $\mathrm{C} = [P(x)]_1$ (since x is secret, the prover computes $[P(x)]_1$ from the srs).
- $(y, \pi) \leftarrow \mathsf{Open}\left(\mathsf{pp}, \text{srs}, P(X), i\right)$: outputs $(P(i), [Q(x)]_1)$, with $Q(X) = \frac{P(X)-P(i)}{X-i}$, proving to a verifier that $P(i) = y$
- $\{0,1\} \leftarrow \mathsf{Verify}\left(\mathsf{pp}, \text{srs}, [C]_, i, y, \pi\right)$: Verifier accepts the proof if $e\left(\mathrm{C} - [y]_1, [1]_2\right) = e\left(\pi, [X-i]_2\right)$

Multiple openings of a single polynomial at points $I = \{i_1, \ldots, i_n\}$ can be batched into a single proof element. Using the vanishing polynomial $Z_I \leftarrow \prod_{i \in I}(X-i)$, and $S(X) \leftarrow \sum_{i \in I} L_{I,i}(X)P(i)$ where $L_{I,i}(X)$ are the Lagrange interpolation polynomials over I:

- $((y_1, \ldots, y_n), \pi) \leftarrow \mathsf{Open}\left(\mathsf{pp}, \text{srs}, P(X), I\right)$: outputs the tuple $((P(i_1), \ldots, P(i_n)), [Q(x)]_1)$, with $Q(X) = \frac{P(X)-S(X)}{Z_I(X)}$
- $\{0,1\} \leftarrow \mathsf{Verify}\left(\mathsf{pp}, \text{srs}, \mathrm{C}, I, (y_1, \ldots, y_n), \pi\right)$: Verifier computes $Z_I(X)$ and $S(X)$, and accepts if $e\left(\mathrm{C} - [S]_1, [1]_2\right) = e\left(\pi, [Z_I]_2\right)$

Under the q-SDH assumption [22] (see Section A.1 for its definition) in the AGM, and using the Schwartz-Zippel lemma, a valid opening $((y_1, \ldots, y_n), \pi)$ of C at I is a proof that the equation $P(X) - S(X) = 0 \mod Z_I(X)$ holds on $\mathbb{F}$. In this paper, we will often refer to that equation rather than to the KZG opening.

This satisfies completeness and evaluation binding under the q-SDH assumption [18,22].

2.4 Degree check on a KZG commitment

Let us consider an SRS of maximum degree d. It is possible, as shown in [14,42], to prove that a commitment $[C]_1$ is a commitment to a polynomial $C(X)$ of degree at most $n \leq d$, by having the prover send to the verifier $n, [C]_1, \left[X^{d-n}C\right]_1$. The verifier accepts if $e([C]_1, \left[X^{d-n}\right]_2) = e(\left[X^{d-n}C\right]_1, [1]_2)$. [14] proves that if the verifier accepts, under the q-DLOG assumption (see Section A.1 for its definition), then $\deg(X^{d-n}C(X)) \leq n$, hence the result.

2.5 KZG as (Sub)Vector Commitment Scheme.

Vectors of size m and polynomials of degree $m-1$, both defined over a field $\mathbb{F}$, are isomorphic to each other, with the canonical one-to-one mapping of $\vec{c} = (c_1, ..., c_m) \in \mathbb{F}^m$ to $C(X) = \sum_{i=1}^{m} c_i L_i(X)$ where $\{L_i(X)\}_{i=1}^{m}$ are the Lagrange interpolation polynomials over $\mathbb{H}$, a multiplicative subgroup of $\mathbb{F}$ of size m, generated by ω. In particular, this mapping is demonstrated by the fact that $C(\omega^i) = c_i$. This allows us to relate vectors and position binding polynomial commitment schemes.

As observed by [37,34], batched KZG proofs can be seen as a subvector proof. For this, we say that $\vec{v} = (v_i)_{i \in S}$ is a subvector of $\vec{c} = (c_1, \ldots, c_n)$ over the domain $S \subset [n]$ if and only if $\forall i \in S, v_i = c_i$. Similarly to vectors, subvectors can be mapped to polynomials. In this paper[2], we will consider the mapping that associates the subvector $\vec{v}$ to $V(X) = \sum_{i \in S} v_i L_i(X)$, thus of degree $n-1$.

Let us consider a polynomial C defined over $\mathbb{H}$. One can prove, following [37,34], that C_I maps to a subvector of C over I simply by proving the following relation on their associated polynomials.

$$C(X) - C_I(X) = 0 \mod Z_I$$

With Z_I the vanishing polynomial over I. Of course, this opening is neither hiding the values c_i of C, nor the set of indexes I where the vector is opened.

If we want to hide the set I, it simply suffices to not disclose it to the verifier. However, doing so, the prover must now send a commitment to Z_I (since the verifier cannot anymore compute it on their own), and prove that it is indeed a vanishing polynomial (or a constant). For this, as presented in [30], the prover can show that Z_I divides $Z_{\mathbb{H}}$, the (public) vanishing polynomial over $[n]$. Thus the prover has to prove the following relations:

$$\begin{aligned} Z_{\mathbb{H}}(X) &= 0 \mod Z_I(X) \\ C(X) - C_I(X) &= 0 \mod Z_I(X) \end{aligned}$$

Finally, hiding the values c_i of C_I is tackled by blinding the commitments, which we discuss later on.

3 Extended Partition argument

In many cryptographic protocols, it is useful to prove that a committed vector can be decomposed into several disjoint parts. For instance, to show that each part corresponds to a well-formed vote, a batch input, or a token allocation. The subvector argument [25] allows a prover to show that a committed vector contains a particular subvector at certain positions, without revealing those positions.

[2] It is more common to find the mapping $v \mapsto V(X) = \sum_{i \in S} v_i L_{S,i}(X)$ where $L_{S,i}(X)$ is the Lagrange basis over S, thus obtaining a polynomial of degree $|S|-1$. However in our protocol, the Lagrange basis of $\mathbb{H}$ offers better performances.

We build on this concept and introduce the *extended partition argument (EPA)* - a new type of zero-knowledge argument that allows a prover to demonstrate that a committed vector C can be expressed as the union of multiple (possibly hidden) subvectors, each scaled by a known public scalar, and that these subvectors cover the entire domain without overlap.

Consider a bilinear tuple with associated prime field $\mathbb{F}$, and $\mathbb{H}$ be a multiplicative subgroup of $\mathbb{F}$, represented as a group of roots of unity of size $n \in \mathbb{N}$ with generator ω.

Let $\vec{c} = (c_1, ..., c_n)$ be a vector and $C(X)$ be the polynomial of degree $n-1$ that evaluates to c_i at point ω^i for all $i \in [n]$.

We also consider k subvectors $\{\vec{v_j}\}_{j \leq k}$ of domains I_j, and k values $w_j \in \mathbb{F}$, such that $w_j \cdot \vec{v_j}$ is a subvector of $\vec{c}$ over I_j for all $j \in [k]$, with $\cup_{j\in[k]} I_j = [n]$ and $\sum_{j\in[k]} |I_j| = n$, i.e. that the $w_j \cdot \vec{v_j}$ form a partition of $\vec{c}$, in the sense that the subvectors $w_j \cdot \vec{v}_j$ are subvectors of $\vec{c}$ over I_j, with $\{I_j\}_{j\in[k]}$ being a partition of $[n]$.

A trusted party selects at random a secret x and computes $SRS_i(x) = \{[x]_i, \left[x^2\right]_i, ..., \left[x^d\right]_i\}$ for $i \in \{1, 2\}$ (that is, commitments of powers less than or equal to d of x in $\mathbb{G}_1$ and $\mathbb{G}_2$). Let $Z_{\mathbb{H}}(X) = \prod_{i\in[n]}(X - \omega^i) = X^n - 1$ and $Z_{I_j}(X) = \prod_{i\in I_j}(X - \omega^i)$ be the vanishing polynomials over $\mathbb{H}$ and I_j, respectively.

The extended partition argument is a zero-knowledge argument for the relation $\mathcal{R}_{\text{EPA}}$ defined as:

$$\left[\begin{array}{l|l} \{\left[x^{i-1}\right]_1, \left[x^{i-1}\right]_2\}_{i\in[d]} ; & [C]_1 = \mathbf{c}; \sum_{j\in[k]} m_j = n \\ \mathbf{c}, \{w_j, m_j\}_{j\in[k]}, \mathbb{H}, \omega ; & \forall j \in [k], |I_j| = m_j \\ C(X), \{I_j\}_{j\in[k]} & \cup_{j\in[k]} I_j = [n] \\ & \forall j \in [k], \exists V_j \in \mathbb{F}_{\leq n}[X], \\ & \quad \forall i \in I_j, w_j V_j(\omega^i) = C(\omega^i) \end{array}\right]$$

As a side observation, setting all w_j to 1 is a relation for a partition with prescribed (i.e., public) sizes. Similarly, setting all w_j to 1, dropping any clause about m_j, and adding the requirement that $\forall i \neq j, I_i \cap I_j = \emptyset$ (which is given at no cost by our protocol) gives a relation for undisclosed size partition, where the subvector sizes are undisclosed (and potentially null). Both variants can be of individual interest.

3.1 Intuition for the EPA

We first give an explanation of why the EPA protocol, described below, will give us an argument for the relation $\mathcal{R}_{\text{EPA}}$.

Leveraging subvector proofs for partition argument Let us consider a vector $\vec{c}$ of n points and its associated polynomial C, defined over $\mathbb{H}$ a cyclic group of order n and generator ω. We set $k > 1, j \in [k]$, and some polynomials C_{I_j}, Z_{I_j} of respective degree $n-1$ and n that fulfill the following two equations,

i.e., that prove that each C_{I_j} maps to a subvector of C over an undisclosed domain I_j, for any $j \leq k$ (see section 2.5).

$$\begin{aligned} Z_{\mathbb{H}}(X) &= 0 \mod Z_{I_j}(X) \\ C(X) - C_{I_j}(X) &= 0 \mod Z_{I_j}(X) \end{aligned}$$

Moreover, if the sets I_j are disjoint and form a partition of $[n]$, then the vanishing polynomials Z_{I_j} (which can also be seen as accumulators for I_j) have no common roots, and $\prod_j Z_{I_j} = Z_{\mathbb{H}}$.

Note that so far, we prove that the prover knows of a partition, but the size of each subset is not known by the verifier, since the Z_{I_j} are not public.

Adding a degree check To ensure that the subvectors sizes match a public prescription, we add a degree check, as described in section 2.4, on each Z_{I_j}, proving that their degree is at most m_j, with $\sum m_j = n$. Another advantage of the degree check is that it will ensure validity of the protocol when using a universal SRS, i.e. one where the maximum degree d of constructible commitments is higher than n.

Proving a product with pairing checks Proving the validity of the product $\prod_{j\in[k]} Z_{I_j} = Z_{\mathbb{H}}$ is feasible, but requires $k-1$ pairing checks and $k-2$ extra commitments sent to the verifier. For this, the prover commits to $[Z_{I_1}]_1$, and for any $j > 1$, to $\left[Z_{I_j}\right]_2$. Furthermore, the prover commits to partial products $\left[Z_{I_{1,2}}\right]_1$, $\left[Z_{I_{1,2,3}}\right]_1, \ldots, \left[Z_{I_{1,2,\ldots,k-1}}\right]_1$, where $Z_{I_{1,\ldots,i}}(X) \leftarrow \prod_{j=1}^{i} Z_{I_j}(X)$, and sends them to the verifier.

Finally, the verifier computes $[Z_{\mathbb{H}}]_1 = [X^n - 1]_1$ and checks that:

- $\forall 1 < j \leq k, e(\left[Z_{I_{1,\ldots,j-1}}\right]_1, \left[Z_{I_j}\right]_2) = e(\left[Z_{I_{1,2,\ldots,j}}\right]_1, [1]_2)$
- $e(\left[Z_{I_{1,2,\ldots,k-1}}\right]_1, [Z_k]_2) = e([Z_{\mathbb{H}}]_1, [1]_2)$

Transforming a partition argument into an extended partition argument In our voting protocol below, we do not want to exhibit a partition, but rather a series of subvectors over subsets $I_j \subset [n]$, such that if a certain scalar multiplication is applied to each subvector, they then form a partition. In other words, we want to prove that there exists some $V_j(X)$ such that, for some scalars w_j known to the verifier, the sets represented by the polynomials $C_{I_j}(X) \leftarrow w_j V_j(X)$ form a partition of $C(X)$.

As such, to prove an EPA, the prover must show that

$$\begin{cases} Z_{\mathbb{H}}(X) = 0 \mod Z_{I_j}(X) \\ C(X) - w_j V_j(X) = 0 \mod Z_{I_j}(X) \\ \prod_j Z_{I_j} = Z_{\mathbb{H}} \\ \deg(Z_j) \leq m_j, s.t. \sum_{j\in[k]} m_j = n \end{cases}$$

3.2 Blinding

Finally, to achieve zero-knowledge, we blind Z_{I_j} by $Z'_{I_j} \leftarrow r_j Z_{I_j}$ with $r_j \in \mathbb{F}^*$, and blind V_j by $V'_j \leftarrow V_j + s_j Z'_{I_j}$, with $s_j \in \mathbb{F}$. C can optionally be blinded too by adding a multiple of $Z_{\mathbb{H}}$. Furthermore, since the product of the blinded Z'_{I_j} must be equal to $Z_{\mathbb{H}}$, we require that $\prod_{j\in[k]} r_j = 1$.

3.3 Protocol

The following protocol is then a zero-knowledge argument for $\mathcal{R}_{\text{EPA}}$.

Extended Partition Argument

Public inputs:

- Prime order additive groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, with generators $[1]_1, [1]_2$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and associated scalar field $\mathbb{F}$
- Structured reference string (SRS) $[x]_1, \ldots, \left[x^d\right]_1, [x]_2, \ldots, \left[x^d\right]_2$

Common inputs:

- Multiplicative subgroup $\mathbb{H} \subset \mathbb{F}$ of order $n \leq d$ and generator ω
- KZG commitment $[C]_1$ to $C(X)$, of degree $n-1$
- $(w_j)_{j\in[k]}$ a list of scalars from $\mathbb{F}^*$
- $(m_j)_{j\in[k]}$ the size of the subsets, with $\sum_{j\in[k]} m_j = n$

Witness inputs:

- Sets of indices $I_1, ..., I_k \subset [n]$ such that $\bigcup_{j\in[k]} I_j = [n]$ and $|I_j| = m_j$ (note that since $\sum_{j\in[k]} m_j = n$ the sets I_j do not intersect each other)
- $\{c_i\}_{i\in[n]}$ such that $C(X) = \sum_{i\in[n]} L_i(X)c_i$, with $(L_i(X))$ the Lagrange basis over $\mathbb{H}$

Round 1 Prover: As seen in 'Proving a product with a pairing check', Z_{I_1} is not committed in the same group as the other Z_{I_j}. We thus introduce grp such that $\mathsf{grp}(1) = 1$, and $\mathsf{grp}(j) = 2$ if $j > 1$. Similarly, $\overline{\mathsf{grp}} = 3 - \mathsf{grp}$.

For each $j \in [k]$:

- Compute $Z_{I_j} \leftarrow \prod_{i\in I_j}(X - \omega^i)$, $V_j(X) \leftarrow w_j^{-1} \sum_{i\in I_j} L_i(X)c_i$
- Sample blinding factors $r_j \leftarrow \mathbb{F}^*, s_j \leftarrow \mathbb{F}$. If $j = k$, let $r_k \leftarrow \prod_{i\in[k-1]} r_i^{-1}$
- Compute blinded $Z'_{I_j}(X) \leftarrow r_j Z_{I_j}(X)$ and $V'_j(X) \leftarrow V_j(X) + s_j Z'_{I_j}(X)$
- Compute $Q_{1,j}(X) = \frac{X^n-1}{Z_{I_j}(X)} = Z_{\overline{I_j}}(X)$ and $Q_{2,j}(X) = \frac{C(X)-w_j V_j(X)}{Z_{I_j}(X)}$

- Compute blinded witnesses $P_{1,j}(X) \leftarrow r_j^{-1} Q_{1,j}(X)$ and $P_{2,j}(X) \leftarrow r_j^{-1} Q_{2,j}(X) - w_j s_j$
- If $1 < j < k$, compute blinded partial accumulator $Z'_{1,\ldots,j}(X) \leftarrow \prod_{i=1}^{j} Z'_{I_j}(X)$
- Publish $\mathbf{x}_j \leftarrow \left[X^{d-m_j} Z'_{I_j}\right]_1, \mathbf{z}_j \leftarrow \left[Z'_{I_j}\right]_{\mathsf{grp}(j)}, \mathbf{v}_j \leftarrow \left[V'_j\right]_1$. If $1 < j < k$ also publish $\mathbf{y}_j \leftarrow \left[Z'_{1,\ldots,j}\right]_1$

Round 1 Verifier: Send random challenges $\chi_1, \ldots, \chi_k$
Round 2 Prover: For each $j \in [k]$:

- Publish $\mathbf{p}_j \leftarrow [P_{1,j} + \chi_j P_{2,j}]_{\overline{\mathsf{grp}}(j)}$

Round 2 Verifier (unoptimized):
Check that:

- $e(\mathbf{z}_1, \left[X^{d-m_1}\right]_2) \stackrel{?}{=} e(\mathbf{x}_1, [1]_2)$
- $e(\mathbf{z}_1, \mathbf{p}_1) \stackrel{?}{=} e([X^n - 1]_1 + \chi_1([C]_1 - w_1 \mathbf{v}_1), [1]_2)$

Set values $\mathbf{y}_1 \leftarrow \mathbf{z}_1$ and $\mathbf{y}_k \leftarrow [X^n - 1]_1$.
For each $j \in [2, k]$, check that:

- $e(\left[X^{d-m_j}\right]_1, \mathbf{z}_j) \stackrel{?}{=} e(\mathbf{x}_j, [1]_2)$
- $e(\mathbf{y}_{j-1}, \mathbf{z}_j) \stackrel{?}{=} e(\mathbf{y}_j, [1]_2)$
- $e(\mathbf{p}_j, \mathbf{z}_j) \stackrel{?}{=} e([X^n - 1]_1 + \chi_j([C]_1 - w_j \mathbf{v}_j), [1]_2)$

Accept if all the checks pass; reject otherwise.

Note that setting all w_j to 1 yields a proof of partition with prescribed sizes, which we call EPA-linear. If one furthermore desires to hide the size of one or several of the subsets Z_{I_j} to the verifier (e.g. to obtain a protocol for undisclosed sizes partition argument), they will have to remove all the associated $\mathbf{x}_j$ from the transcript, and similarly the verifier will remove these checks (EPA-linear-nosize). The original EPA version will be used in Section 6, while EPA-linear, will be useful in Section 6.3 on top of the EPA.

3.4 Optimizations

Classical batching gives the verifier checks, for ϕ a random scalar:

- $e(\mathbf{z}_1, \mathbf{p}_1 + \phi_1 \left[X^{d-m_1}\right]_2) \stackrel{?}{=} e([X^n - 1]_1 + \chi_1([C]_1 - w_1 \mathbf{v}_1) + \phi_1 \mathbf{x}_1, [1]_2)$
- $e(\mathbf{p}_j + \phi_j \left[X^{d-m_j}\right]_1 + \phi_j^2 \mathbf{y}_{j-1}, \mathbf{z}_j) \stackrel{?}{=} e([X^n - 1]_1 + \chi_j([C]_1 - w_j \mathbf{v}_j) + \phi_j \mathbf{x}_j + \phi_j^2 \mathbf{y}_j, [1]_2)$ for any $j \in [2, k]$

Furthermore, a series of checks of the form $e(a_i, b_i) \stackrel{?}{=} e(c_i, [1]_2)$ can be batched by picking a random $\psi \in \mathbb{F}^*$ and checking that $e(\sum \psi^{i-1} c_i, [1]_2) = \sum \psi^{i-1} e(a_i, b_i)$. This decreases the number of pairing computations from the unoptimised $6k-2$ to simply $k+1$.

Finally, Miller loop optimizations [31] can be used to compute the sum of n pairings faster than computing n pairings.

We chose to omit these optimizations from the description of the algorithm (and subsequent proofs) to improve readability. However, our implementation relies on the optimized batched verification.

Some other optimizations, such as computing the partial products $Z_{1,\dots,j}$ in a binary tree rather than an unbalanced tree, offer little advantage in practice.

4 Security of EPA

In this section, we prove that EPA is a zero knowledge argument of knowledge the completeness, knowledge soundness, and zero-knowledge properties of the EPA. For improving the readability, we only prove these properties using the unbatched version of the protocol, with $3k-1$ verifier checks. An adaptation of these proofs to the batched version is immediate.

4.1 Completeness

We first show that the EPA protocol is complete in the sense of Definition 7 (Section A), i.e., that an honest prover will convince an honest verifier with probability 1.

Theorem 1. *The protocol given in Section 3.3 is complete.*

Proof. First, all commitments are constructible since all polynomials have a degree $\leq d$. Second, $\forall j > 1$:

- $e(\left[X^{d-m_j}\right]_1, \mathbf{z}_j) = e(\left[X^{d-m_j}\right]_1, \left[Z'_{I_j}\right]_2) = e(\left[X^{d-m_j} Z'_{I_j}\right]_1, [1]_2) = e(\mathbf{x}_j, [1]_2)$
- $e(\mathbf{y}_{j-1}, \mathbf{z}_j) = e(\left[Z'_{1,\dots,j-1}\right]_1, \left[Z'_j\right]_2) = e(\left[Z'_{1,\dots,j-1} Z'_j\right]_1, [1]_2) = e(\left[Z'_{1,\dots,j}\right]_1, [1]_2) = e(\mathbf{y}_j, [1]_2)$
- $\begin{aligned} e(\mathbf{p}_j, \mathbf{z}_j) &= e(\left[r_j^{-1} Q_{1,j} + \chi_j (r_j^{-1} Q_{2,j} - w_j s_j)\right]_1, \left[r_j Z_{I_j}\right]_2) \\ &= e(\left[Z_{\mathbb{H}} + \chi_j (C - w_j V_j - w_j s_j r_j Z_{I_j})\right]_1, [1]_2) \\ &= e(\left[X^n - 1 + \chi_j (C - w_j (V_j + s_j Z'_{I_j}))\right]_1, [1]_2) \\ &= e([X^n - 1]_1 + \chi_j ([C]_1 - w_j \mathbf{v}_j), [1]_2) \end{aligned}$

Similarly for the case $j = 1$, hence the verifier always accepts.

4.2 Knowledge Soundness

To prove knowledge soundness in the algebraic group model, we setup a game Game_{KS} with an adversary $\mathcal{A}$ and an extractor $\mathcal{E}$. The goal of $\mathcal{A}$, given an SRS, is to produce a valid argument for an instance where they do not know the witness. $\mathcal{A}$ wins the game if and only if their argument is accepted by a verifier, and there is no efficient algebraic $\mathcal{E}$ that from the transcript produces C, I_j that are a witnesses to the relation $\mathcal{R}_{\text{EPA}}$. The protocol is knowledge sound if $\mathcal{A}$ wins the game Game_{KS} with at most a negligible probability. A more precise definition is given in Definition 8 of Section A.

We first recall [17, Lemma 2.2], reformulated below.

Lemma 1 (Real pairing check, [17]). *Assume the q-DLOG for $(\mathbb{G}_1, \mathbb{G}_2)$. Given an algebraic adversary $\mathcal{A}$, the probability of any real pairing check $e([A]_1, [B]_2) \stackrel{?}{=} e([C]_1, [D]_2)$ passing is larger by at most an additive negligible factor than the probability the corresponding ideal check $A(X)B(X) \stackrel{?}{=} C(X)D(X)$ holds.*

Theorem 2. *Assuming the q-SDH for $(\mathbb{G}_1, \mathbb{G}_2)$, given an algebraic adversary $\mathcal{A}$, the protocol given in Section 3.3 satisfies knowledge soundness for the EPA relation.*

Proof. Let us suppose that the adversary outputs for all $j \leq k$, $\mathbf{x}_j, \mathbf{v}_j, \mathbf{y}_j \in \mathbb{G}_1, \mathbf{z}_j \in \mathbb{G}_{\mathsf{grp}(j)}, \mathbf{p}_j \in \mathbb{G}_{\overline{\mathsf{grp}}(j)}$, along with their algebraic representations.

The real pairing checks[3] $e(\mathbf{p}_j, \mathbf{z}_j) = e([Z_H]_1 - \chi_j([C]_1 - w_j\mathbf{v}_j), [1]_2)$ correspond, according to Lemma 1, with overwhelming probability to the ideal pairing checks $(P_{1,j} - \chi_j P_{2,j})Z'_{I_j} = Z_{\mathbb{H}} - \chi_j(C - w_j V'_{I_j})$. Because the χ_j are sampled after the prover commits to Z'_{I_j} and V'_j, with probability $1 - \frac{1}{|\mathbb{F}|}$ and assuming the q-SDH, we must have $Z'_{I_j} \mid Z_{\mathbb{H}}$ and $Z'_{I_j} \mid C - w_j V'_{I_j}$. Let I_j be the set of roots of Z'_{I_j}, and $V_j \leftarrow V'_j \mod Z'_{I_j}$. Because $Z'_{I_j} \mid Z_{\mathbb{H}}$, we conclude that $I_j \subset \mathbb{H}$, and because $Z'_{I_j} \mid C - w_j V'_{I_j}$ (and thus $Z'_{I_j} \mid C - w_j V_{I_j}$), we conclude that $C(y) = w_j V_j(y)$ for any $y \in I_j$. Note that currently, the extractor does not yet know whether I_j is empty or not.

According to Lemma 1 and since q-SDH implies q-DLOG, the real pairing checks $e(\left[X^{d-m_j}\right]_1, \mathbf{z}_j) = e(\mathbf{x}_j, [1]_2)$ prove with overwhelming probability that $\deg(Z'_{I_j}) \leq m_j$, i.e. that $|I_j| \leq m_j$. Similarly, $e(\mathbf{y}_{j-1}, \mathbf{z}_j) = e(\mathbf{y}_j, 1)$ show by recursion with overwhelming probability $(1 - \frac{1}{|\mathbb{F}|})^k$, that assuming the $\prod_{j\in[k]} Z'_{I_j} = Z_{\mathbb{H}}$, which itself indicates that $n = |\cup_{j\in[k]} I_j| \leq \sum_{j\in[k]} |I_j| = \sum m_j$. Moreover, because we know that $\sum_j m_j = n$, we conclude that the I_j do not intersect each other and that $|I_j| = m_j$.

Therefore, an extractor receiving (notably) $\mathbf{c}, \mathbf{z}_j$ will simply extract the committed polynomials C and Z'_{I_j} using the algebraic representation of their commitments. The extractor will also extract I_j, either by factoring Z'_{I_j} or by comparing V_j with C over $\mathbb{H}$. According to the previous paragraphs, the tuple $(C, \{I_j\}_{j\in[k]})$

[3] The same reasoning can be trivially applied to the case where $j = 1$, therefore we omit the distinction for simplicity.

satisfies the relation $\mathcal{R}_{\text{EPA}}$, hence the adversary loses the game with overwhelming probability.

4.3 Zero Knowledge

Finally, we also prove that the protocol is zero-knowledge, where zero-knowledge is defined as Definition 9 in Section A.

Theorem 3. *The protocol given in Section 3.3 satisfies zero knowledge for the EPA relation.*

Proof. We describe the Simulate algorithm that, given an instance $\mathbf{c}, w_j, m_j$ and the SRS trapdoor x, produces a (valid) transcript that is indistinguishable from a honest prover's transcript. For simplicity, we will treat the case $j = 1$ similarly to the other cases, even though in reality the groups for some commitments are switched.

Let the simulator create a random partition $(I_j)_{j\in[k]}$ of I, with $|I_j| = m_j$ and let $Z_{I_j}(X) \leftarrow \prod_{i\in I_j}(X - \omega^i)$. For any $j \leq k$, the simulator samples α_j, β_j at random from $\mathbb{F}$, except for $\alpha_k \leftarrow \prod_{j\leq k} \alpha_j^{-1}$. The simulator then outputs $\mathbf{z}_j \leftarrow \alpha_j \left[Z_{I_j}\right]_2, \mathbf{x}_j \leftarrow \left[X^{d-m_j}\alpha_j Z_{I_j}\right]_1, \mathbf{y}_j \leftarrow \left[\prod_{i\leq j} \alpha_i Z_{I_i}\right]_1, \mathbf{v}_j \leftarrow (\mathbf{c} - \beta_j)w_j^{-1}$, and receives challenges χ_j. Finally, the simulator outputs $\mathbf{p}_j \leftarrow \alpha_j^{-1}\left[Z_{\overline{I}_j}\right]_1 + \chi_j\beta_j\alpha_j^{-1}\left[(Z_I(x))^{-1}\right]_1$ with $\overline{I}_j \leftarrow I\backslash I_j$.

The simulator's outputs verify the pairing checks:

$$e(\left[X^{d-m_j}\right]_1, \mathbf{z}_j) = e(\mathbf{x}_j, [1]_2) \tag{1}$$

$$e(\mathbf{y}_{j-1}, \mathbf{z}_j) = e(\mathbf{y}_j, [1]_2) \tag{2}$$

$$e(\mathbf{p}_j, \mathbf{z}_j) = e([Z_H]_1 + \chi_j([C]_1 - w_j\mathbf{v}_j), [1]_2) \tag{3}$$

Equations (1) and (2) are clearly true, and Equation (3) is proven by $e(\mathbf{p}_j, \mathbf{z}_j) = e(\left[\alpha_j^{-1}Z_{\overline{I_j}}(X)\right]_1 + \chi_j\left[(Z_I(x))^{-1}\beta_j\alpha_j^{-1}\right]_1, \alpha_j\left[Z_{I_j}(X)\right]_2) = e([Z_{\mathbb{H}}(x) + \chi_j\beta_j]_1, [1]_2) = e([Z_{\mathbb{H}}]_1 + \chi_j([C]_1 - ([C]_1 - \beta_j)), [1]_2) = e([Z_{\mathbb{H}}]_1 + \chi_j([C]_1 - w_j\mathbf{v}_j), [1]_2)$.

Finally, the distribution of output elements uniquely matches a valid distribution because for $j < k, \mathbf{z}_j$ is uniformly blinded by r_j for the prover and α_j for the simulator; $\mathbf{z}_k$ uniquely satisfies the relation $\prod \mathbf{z}_j = Z_{\mathbb{H}}(x)$; $\mathbf{x}_j$ uniquely satisfies the pairing check from eq. (1); $\mathbf{y}_j$ uniquely satisfies the pairing check from eq. (2); $\mathbf{w}_j$ is uniformly blinded by s_j for the prover and β_j for the simulator; and $\mathbf{p}_j$ uniquely satisfies the pairing check from eq. (3).

5 Evaluation of EPA

Benchmark setup Our implementation of EPA is available at `https://github.com/mariuslp/aggios_epa_benchmark`, using the `arkworks` library [3]. Tests

were conducted on a laptop with Intel i5-10310U CPU, 16 GB RAM, with parallelization (4 cores, 8 threads in total). Prover and verifier times were measured over values n (elements in the whole set) ranging from 2^8 to 2^{18}, and k (number of partitions) ranging in $\{2, 5, 10, 50, 100\}$, and averaged over 50 runs. Our implementation supports only power-of-2 partition size, but padding with a predefined scalar or zero is possible. The complete benchmark took around 300 CPU hours, 43 hours of real time. RAM usage remained under 5.2 GB. Proving is mostly faster than 60 seconds, verification is always faster than 300 ms.

We give some results in Figure 1. More detailed results are available in Section B.

$O(n \log n)$ polynomial division The `ark-poly` library only implements a quadratic polynomial division algorithm, thus rendering the prover impractically slow for large vectors. We implemented a $O(n \log n)$ division algorithm, following [38, Section 9.1], and our pull request has been merged with the `ark-poly` library. The same optimization was also described in [43].

$O(k)$ verifier time Similarly, `ark-poly-commit` does not handle fast commitments to sparse polynomials, thus making commitments to e.g. $X^n - 1$ needlessly slow in $O(n)$. We have submitted a pull request, approved for merge, for committing to sparse polynomials in $O(1)$. This made the verifier up to 95% faster.

EPA complexity Our proving complexity, assuming that each of the k partitions has a size of n/k, is $O(kn \log^2 n)$: generating $V_j, Q_j, Z_{1,\dots,j}$ takes individually $O(n \log n)$ and generating Z_{I_j} takes individually $O(n/k \log^2 n)$ [38, Section 10]. Proof size is $4k - 2$ elements in $\mathbb{G}_1$, k elements in $\mathbb{G}_2$, and verifier work is $O(k)$ group operations, $k + 1$ pairings.

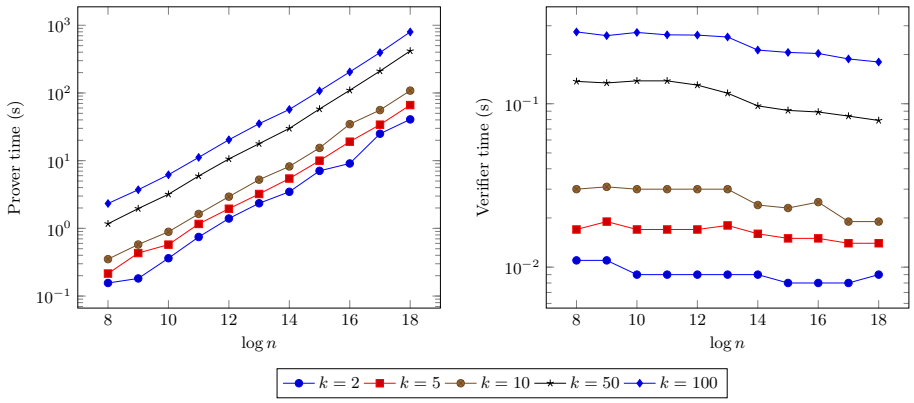

Fig. 1. Prover and verifier's time, in seconds, for various set sizes n and number of partitions k, averaged over 50 runs.

Related implementations To the best of our knowledge, there is no existing partition proof using other classical primitives such as RSA accumulators or Merkle tree, thus making comparison with other schemes difficult. While a generic implementation on Merkle trees using Groth16 is feasible, in practice according to [42] a single proof of knowledge of 1 element took between 2 seconds ($n = 2^6$) to 10 seconds ($n = 2^{22}$), thus a generalization of the scheme to k subsets of thousands of elements would quickly become impractical.

6 Aggios: the Aggregator-based Voting Protocol

Leveraging the EPA, we now present Aggios, aggregator-based voting protocol. We first describe a simple version in which the voters give the vote in plaintext to the aggregator, which corresponds to proxy voting. Proxy voting is regularly used in corporate environments, where small representatives delegate their voting power to larger entities [28], but also, more recently, by numerous DAOs [20]. Then, we introduce a variant of Aggios, in which the individual votes are split between aggregators.

6.1 High-level description of Aggios

Protocol description The protocol involves three entities:

- **Voter:** Voter selects an aggregator to cast votes on her behalf.
- **Aggregator:** Collects votes from its registered voters, publishes the tally and an EPA-based zero-knowledge proof of the tally.
- **Validator:** Verifies the proofs, ensuring integrity (no forged or omitted votes) and validating the aggregator's final tally. Anyone can be a validator.

Trust and privacy. Aggios is *proxy voting*: the voter sends her choice privately to the selected aggregator, which therefore learns the vote, while the public ledger learns only tallies and a succinct proof. Aggios-Split strengthens privacy by distributing each ballot across multiple aggregators, so that no coalition of up to t aggregators learns the vote with certainty.

The protocol proceeds in four phases. Unless specified otherwise, all communications are public and on the bulletin board:

1. *Setup:* A trusted election authority generates an SRS and each voter publishes their public key.
2. *Registration:* Voters generate a pair of voting keys and publish their voting public key. The voters also choose their aggregators and send them a registration request. The aggregator publishes commitments to the registered voters. Validators check that no voter is registered with two distinct aggregators.
3. *Voting:* Each voter privately sends her choice to the aggregator. After votes are collected, the aggregator publishes the tally and an EPA proof. Each voter privately receives proof that their vote has been counted.

4. *Dispute mechanism (public accountability)*: Voters sign their committed vote, and the aggregator Merkle-commits per-voter evidence (per-voter ACE elements in Aggios-Split). Upon a dispute, revealing the corresponding leaf and Merkle path lets anyone adjudicate the claim without revealing the vote (see Appendix E).

The setup is done once. Registration occurs only when users enter the system (i.e., select their aggregator for the first time) or switch to other aggregators, which we assume does not happen frequently. Therefore, the associated communication cost on the public bulletin board can be amortized over several elections.

Security properties and system setting As we will develop further down in Section 7, Aggios offers vote integrity (votes cannot be modified without detection), eligibility (only registered voters can vote), no double voting, and vote privacy (external observers cannot guess the voter's choice), even again active malicious attackers. In Aggios-Split, the aggregators themselves are prevented from knowing someone's vote with certainty below a certain threshold of corrupted actors.

6.2 Detailed Protocol

In Algorithm 1, we detail our protocol. We use a collision-resistant hash function $H : \{0,1\}^* \mapsto \mathbb{G}_1$ for BLS signatures [8], and a ZK proof *Eqlog* such as the Chaum-Pedersen [9, Section 20.3.6], proving that $a = [\alpha]g_1$ and $b = [\alpha]g_2$, have equal log with respect to bases g_1 and g_2. See Section A.5 for a formal definition of the equal log. All messages from voters/aggregators are signed.

Algorithm 1 Aggios

Setup

- Election Authority posts SRS $\{[x^{t-1}]_1, [x^{t-1}]_2\}_{t=1}^{d}$ where d is an upper bound on the number of voters.
- Election Authority publishes distinct non-zero scalars $\{w_1, \ldots, w_k\} \subset \mathbb{F}$, one per candidate, which serve as the public labels in the partition argument.
- Each voter i publishes long-term key $pk_i = [sk_i]_1$.

Registration

Voter i:

- Chooses aggregator Ag
- Generates *public-secret voting* key pair (pkv_i, skv_i), with $pkv_i = [skv_i]_2$ and publishes (i, pkv_i).
- Prepares BLS signature $\sigma_i(Ag) \leftarrow [sk_i] \cdot H(Ag)$ showing delegation to aggregator Ag.
- Sends $(\sigma_i(Ag), skv_i, pk_i)$ to Ag over a private channel.

Aggregator Ag:

- Collects $(\sigma_i(Ag), skv_i, pk_i)$ from each voter i who wishes to register. Let I be the set of such voters who sent a valid registration message to Ag.
- For each $i \in I$:
  1. Compute $\tau_i \leftarrow [skv_i L_{i,I}(X)]_2$, with $(L_{i,I}(X))_{i\in I}$ Lagrange polynomials on I.
  2. Construct $\pi_i^{Lagrange}$ (using $EqLog$) showing that τ_i and pkv_i share the same exponent with respect to bases $[L_{i,I}]_2$ and g_2, thus proving τ_i's well-formedness.
- Compute $\sigma_I(Ag) \leftarrow \sum_{i\in I} \sigma_i(Ag)$ (batched BLS signatures).
- Publish $(\{(i, \tau_i, \pi_i^{Lagrange})\}_{i\in I}, \sigma_I(Ag))$ on the bulletin board.

Validators (Registration Check):

- Verify $e(\sigma_I(Ag), g_2) \stackrel{?}{=} e\left(H(Ag), \sum_{i\in I} pk_i\right)$.
- For each $i \in I$, check $EqLog.\text{Verify}\left(g_2, pkv_i, [L_{i,I}(X)]_2, \tau_i, \pi_i^{Lagrange}\right) \stackrel{?}{=} 1$.
- If any check fails, the aggregator is rejected. Otherwise, validators accept I as the set of registered voters.

Voting

Voter i:

- Chooses candidate among election defined integers $(w_j)_{j\leq k}$, and privately sends it to Ag.

Aggregator Ag:

- Wait until a cutoff time T_1 to receive votes from the registered voters.
- Let $I^{(t)} \subseteq I$ be the subset of voters who submitted a ballot (recall that I is the set of registered voters.). Compute $[C]_1 \leftarrow \sum_{i\in I^{(t)}} [skv_i L_{i,I}(X)]_1$ (effectively nullifying the commitment of the voters who did not cast a ballot)
- For each candidate $j \in \{1, \ldots, k\}$, gather $I_j \subseteq I^{(t)}$, the subset of voters that chose candidate j. Let $\mathcal{M} = (m_1, \ldots, m_k, m_{k+1})$ with $m_j = |I_j|$ for $j \leq k$ and $m_{k+1} = |I \setminus I^{(t)}|$.
- For each $j \leq k$ set $\mathbf{v}_j \leftarrow \frac{1}{w_j} \sum_{i\in I_j} [skv_i L_{i,I}(X)]_1$ and $\mathbf{v}_{k+1} \leftarrow [0]_1$, $w_{k+1} = 0$.
- Compute the EPA proof Π from the aggregated votes and publish $(\Pi, I^{(t)}, \mathcal{M})$ on the bulletin board.
- For each $i \in I_j^{(t)}$, compute $\left(skv_i, \pi_i^{KZG}(j)\right) = \text{KZG.Open}\left(\mathbf{v}_j, i\right)$ and privately send $\pi_i^{KZG}(j)$ to voter i, allowing voter i to check that skv_i was indeed included in the subset $\mathbf{v}_j$ corresponding to their chosen candidate j.

Verification

- *Validators*: Obtain $(\Pi, I^{(t)}, \mathcal{M})$ from the bulletin board and reconstruct $[C]_1$ from $\left[\sum_{i\in I^{(t)}} \tau_i\right]_1$ and . Then, verify the proof Π along with $(I^{(t)}, \mathcal{M})$

against $[C]_1$, ensuring validity of the aggregation. If verification passes, then accept the final tally $\mathcal{M} = (m_1, \ldots, m_k)$.
- *Each voter i*: Reconstruct $[C]_1$ from $\left[\sum_{i \in I^{(t)}} \tau_i\right]_1$. Then the voter checks proof of inclusion in the correct subset. Send a BLS signed ACK to Ag if and only if we have the equality KZG.Verify$([C]_1, i, skv_i, \pi_i^{KZG}) = 1$.
- *Aggregators*: Compute $\sigma = \sum_i \sigma_i$ the aggregation of the signed ACKs sent by the voters and publish σ on the bulletin board.

Dispute Mechanism (Liveness Safeguard)

If aggregator ignores/omits a voter i:
- Voter i does not send an acknowledgment BLS signature to Ag, thus opening a dispute.

Outcome:
- Flag any aggregator that cannot prove correct execution in case of dispute, and allow registered voters to cast their votes again with another aggregator.

Note that currently, the dispute mechanism cannot allow an external observer to conclude whether the voter or the aggregator misbehaved, without revealing the voter's vote. Using the ACE protocol for Aggios-Split (respectively described in Section C.2 and Section 6.3), we can obtain a better resolution conflict, in which the aggregator is able to prove they behaved honestly, without revealing the voter's vote. This is developed in Section E.

6.3 Aggios-Split: ballots via vote dilution

We describe an enhancement of Aggios, named Aggios-Split, that hides each voter's choice from a set of t corrupted accumulators.

Description of Aggios-Split Instead of voting for one candidate among k using a single aggregator, each voter votes $kt + 1$ times using $kt + 1$ different aggregators. Of these, kt votes are uniformly distributed among all k candidates (i.e., t decoy votes per candidate), thus effectively canceling each other out. The final vote encodes the voter's choice. Aggregators publish their tallies the same way as in Aggios, and the overall tally (including the decoy votes) is obtained. Assuming N voters, the real final tally can be recovered by subtracting Nt votes from each candidate (since each voter provably sends t decoy votes to each candidate). See an example of vote casting and tallying in Appendix D.1. By the pigeonhole principle, an attacker needs at least $t+1$ colluding aggregators to determine with certainty the choice of a specific voter. Therefore, Aggios-Split provides deniability against any coalition of up to t colluding aggregators.

Required proofs for correctness First and obviously, the voter needs to prove to aggregators that they indeed cast a vote using kt uniformly distributed

decoy votes in addition to their own vote, we describe how to do so in section 6.3, with protocol VWF.

However, while the VWF protocol protects against colluding aggregators or cheating voters, it does not protect against a voter colluding with an aggregator to change their vote. Say, for instance, that a voter commits to the votes $[1, 2, 3, 2]$, i.e. votes for candidate 2 with $k = 3, t = 1$. The voter also agrees with, say, aggregator of index 0, to change the delegated vote from a vote for candidate 1 to a vote for candidate 2. Since only the voter can detect the alteration of their vote, that fraud would be undetected by other honest users.

Hence, we must introduce another check, that is run between the aggregators and works as long as at least one of the aggregators is willing to detect the cheating (enforcing this rule can be done by for instance forcing each voter to register to at least one aggregator run by each candidate). During that check, each aggregator proves to the other aggregators that the vote they counted for each voter matches the committed vote they received. While the idea is simple, the exact execution requires careful planning, which we will explain in Section 6.3, with protocol ACE.

Proving voter well formedness (VWF) The vote of voter i is a commitment to a polynomial $S_i(X)$ that contains $kt + 1$ votes: t decoy votes to each of the k candidates, as well the real vote. To show well-formedness, we will show that there exists a sub-vector $T_i(X)$ of $S_i(X)$, representing kt votes, such that T_i can be partitioned into k (sub-)subvectors, each subsubvector consisting of t votes for every single candidate[4]. The partition argument is an instantiation of EPA-linear (see the end of Section 3.3).

We thus define, in Section C.1, the subprotocol of voter well-formedness (VWF), where $\mathsf{VWF.Prove}(S_i(X), k, t)$ returns various commitments and proofs, that will allow $\mathsf{VWF.Verify}$ to correctly assess the validity of the voter claim that S_i is well-formed.

The voter will also produce individual opening proofs of S_i at each point j, that will privately prove to aggregator j the value of $S_i(\omega^j)$, i.e. the vote delegated to them.

Proving aggregator correct execution (ACE) As mentioned earlier, a colluding voter and aggregator could agree to change a vote. Therefore, aggregators must check between themselves the validity of each other's votes. For this, each aggregator $l \leq kt+1$ sends to the other aggregators involved with voter i a proof that the vote they cast for voter i is indeed the one committed in $S_i(X)$ at index occupied by l.

Given the definition of the subvectors $(V_j^{(l)})_{j \leq k}$ in the EPA proof of aggregator l (see Section 3.3, where they are named V_j) and the fact that Aggios partitions the vector $C^{(l)}$ (C in Section 3.3) of voting keys skv_i registered with

[4] We could also prove that S only contains valid vote values, however this is not needed since aggregators will dismiss the voter in case of an invalid vote.

aggregator l (see Algorithm 1), we observe that if i voted for candidate j, then $V_j^{(l)}(\omega^i) \times w_j = skv_i$ and $V_m^{(l)}(\omega^i) \times w_j = 0$ for any $m \neq j$. Thus, assuming the $V_j^{(l)}$ are well-formed, $\sum_{j\in[k]} V_j^{(l)}(\omega^i) \times \mathsf{vote_of}^{(l)}(\omega^i) = C^{(l)}(\omega^i)$, with $\mathsf{vote_of}^{(l)}(\omega^i)$ being the vote of voter i delegated to aggregator l. Let L_i be the public list of aggregators that i is using. For each aggregator a_l, we define M_l as the polynomial that for ω^i associates the index of a_l in L_i, assuming voter i registered with aggregator a_l: $L_i[M_l(\omega^i)] = a_l$ (see an example in Section D.2). We additionally have $\mathsf{vote_of}^{(l)}(\omega^i) = S_i(M_l(\omega^i))$, thus we have, if everything is well-formed, that $\sum_{j\in[k]} V_j^{(l)}(\omega^i) \times S_i(M_l(\omega^i)) = C^{(l)}(\omega^i)$. Would the aggregator be able to prove this relation, then other aggregators would be convinced of the honest vote casting.

However, the aggregator does not know the polynomial S_i, just its opening at one specific point. To address this issue, the voter will have to send extra information to the aggregator: a commitment to $S_i(M_l(X))$, a proof of its well-formedness, and a proof $\rho_1^{(i)}$ that $S_i(M_l(\omega^i)) = \mathsf{vote_of}^{(l)}(\omega^i)$. The aggregator, receiving all such proofs from all voters, constructs the polynomial $R^{(l)}(X)$ such that, for all voter i registered to the aggregator, $R^{(l)}(\omega^i) = \mathsf{vote_of}^{(l)}(\omega^i)$. The aggregator also computes the opening proof $\rho_2^{(i)}$ of the opening of R at ω^i. Then, $\rho_1^{(i)} - \rho_2^{(i)}$ is a proof that $S_i(M_l(X)) = R^{(l)}(X) \mod X - \omega^i$, i.e. that $S_i \circ M_l$ is equal to $R^{(l)}$ at ω^i, i.e. the vote in R is equal to the vote delegated by the voter.

Finally, the aggregator computes and discloses to the other aggregators the proof of the relation $C^{(l)}(\omega^i) - \sum_{j\in[k]} V_j^{(l)}(\omega^i) \times R^{(l)}(\omega^i) = 0 \mod X - \omega^i$, proving that the vote counted for voter i is equal to the vote sent to the aggregator in S_i, via the equality $S_i(M_l(\omega^i)) = R^{(l)}(\omega^i)$, for each voter i. This can be batched.

We give a full description of the protocol ACE in Section C.2. We also prove its soundness and zero-knowledge.

Modifications on the Aggios protocol We will now describe the registration and voting phase in Aggios-Split in Algorithm 2. The rest of the protocol will be similar to Aggios. We consider a set of candidates C of size k, N voters need to be registered with $Kt + 1$ aggregators ($K \geq k$). Verification and contest phases are unchanged.

Algorithm 2 Aggios-Split

Registration phase

Voter i

- Register to at least $kt + 1$ aggregators in the same way as Aggios (Algorithm 1). The order of these aggregators is public, say $L_i = [a_1, a_7, a_8, \dots]$.

Everyone

- Gather the registration list L_i of every voter.
- For each aggregator a_l, compute $M_l(X)$ such that $M_l(\omega^i) = j$ if $L_i[j] = a_l$.

Voting phase

Voter i

- Pick a candidate $c \in C$, create the voting vector $s^{(i)} \leftarrow \text{Shuffle_vec}([sk_i^{c_1}, \ldots, sk_i^{c_k}] \times t \,||\, [sk_i^c])$.
- Compute and send $\mathsf{VWF.Prove}(S_i(X), k, t)$ (Section C.1) to all aggregators i is registered with for that election
- Run the voter part of $\mathsf{ACE.Prove}$, send the proofs to the corresponding aggregators

Aggregator l

- Receives proofs from voter i and a delegated vote, represented by a KZG opening sent privately and its proof $\pi^{(i)}_{open,l}$
- Raise an error if the index of the opening of $\pi^{(i)}_{open,l}$ does not match $M_l(\omega^i)$, the index of l for voter i.
- With all other aggregators registered by i, check that everybody received the same from proofs from $\mathsf{VWF.Prove}$, and that $\mathsf{VWF.Verify}$ accepts.
- If any of the checks fail, dismiss voter i and signal it to other aggregators.
- Compute $\mathsf{ACE.Prove}$ (Section C.2) and send its proof to all other aggregators.
- For each ACE proof received from another aggregator, run $\mathsf{ACE.Verify}$ and fail if some checks do not succeed.
- If all the checks succeed, publish the tally, and privately send a proof of inclusion to the voter.

6.4 Efficiency

In this section, we study the efficiency of the basic Aggios voting compared to the naive solution where all voters publish their votes on the public ledger.

We assume that only a few voters switch aggregators, so amortized registration costs remain low. The following observations justify this assumption:

1. Honest users tend to stick with their chosen aggregator unless there is a compelling reason to switch (e.g., fees or service issues).
2. Dishonest users who repeatedly switch aggregators to disrupt the system will be easy to detect and can be penalized.
3. Any aggregator acting maliciously, by dropping or forging deltas, will be flagged through public monitoring or by voters lodging signed dispute claims.

Similarly, assuming participation remains similar at each election, to further reduce on-chain data, in each election round an aggregator needs only to publish the *difference* of participants $\Delta I^{(t)} = I^{(t)} \setminus I^{(t-1)}$ between two elections, rather than $I^{(t)}$.

Furthermore, we show how quickly vote aggregation becomes efficient compared to the naïve solution of storing all the votes.

During setup, each user sends a public voting key pkv_i of $\mathbb{G}_1$ (using the BLS12-381 curves, uncompressed $\mathbb{G}_1$ and $\mathbb{G}_2$ points are respectively 96 and 192 bytes). Registration costs $96 + 628|I|$ uncompressed bytes (1 BLS signature in $\mathbb{G}_1$, $|I|$ IDs of, e.g., 32 bits, $|I|$ commitments in $\mathbb{G}_2$, $|I|$ Chaum Pedersen proofs of 432 bytes [9]).

The voting and verification bandwidth consists of an EPA proof, the list $\Delta I^{(t)}$ of updated voters, and one BLS signature. $\Delta I^{(t)}$ is represented either as bitset of n bits in the worst case, a list of $\log n$ bits integers in the better case where $|\Delta I^{(t)}| \leq n/\log n$, and 0 bits if no change in actual voters occurs. Assuming participation does not change between elections, Aggios thus has constant size.

An EPA proof contains $4k-2$ elements of $\mathbb{G}_1$ and k elements of $\mathbb{G}_2$. Hence, the size of EPA proof + BLS signature + size of $\Delta I^{(t)}$ (the tally is omitted since it is also present in normal voting) is of size $576k - 96 + n/8$ bytes at worst. In comparison, n ElGamal votes (as a basic comparison baseline) on a 256 bits curve require $64n$ bytes, thus in the worst case our scheme is more efficient than the naïve solution when aggregating more than $\frac{576k-96}{64-1/8} \approx 9k-2$ votes, showing its practicality. Asymptotically and in the worst case, our scheme is $64n/(n/8) = 512$ times more compact than naïve voting (an additional factor of 2 can be obtained by using compressed points). In the best case, where $\Delta I^{(t)}$ can be represented in sublinear or constant space, the ratio goes to infinity.

Finally, since Aggios-split publishes the same amount of data on the public ledger, the same conclusions apply for Aggios-Split. The main difference is that in Aggios-Split, the voter needs to register to numerous aggregators, therefore increasing the registration cost. However this is amortized over the repeated votes.

7 Security of Aggios

In this section, we formalize the key security properties of our aggregator-based voting protocol using game-based definitions.

7.1 Security Model

We consider a malicious adversary that may corrupt any subset of voters and aggregators and cause corrupted parties to deviate arbitrarily and collude (voters with voters, voters with aggregators, and aggregators with each other). For Aggios, this means that the adversary may corrupt the (single) posting aggregator and any number of voters. For Aggios-Split with L aggregators, the adversary may corrupt up to t aggregators (and any number of voters). We assume at least one aggregator remains honest (i.e., does not deviate in the same way as the adversary), as required by the cross-checking/accountability layer. All on-ledger data are public, and correctness is publicly auditable via the verification algorithms (EPA, and in Aggios-Split also ACE).

Theorem 4. *Under the hardness assumptions of KZG and EPA, Aggios (Protocol 3.3) provides the following:*

1. ***Integrity:*** *Votes cannot be modified without detection.*
2. ***Eligibility:*** *Only registered voters can vote.*
3. ***No Double Voting:*** *A voter's key cannot be counted more than once.*

4. ***Privacy:*** *External entities cannot link an honest voter to its chosen candidate.*

Disputes/accountability. Using the ACE transcript to prove correct aggregator execution, said transcript can be condensed into a Merkle tree where leaves are per-voter proof, therefore allowing for a public resolution of voter/aggregator conflict, without revealing the voter's vote. See Appendix E for detailed description.

7.2 Integrity of the Tally

We define the *Integrity* game $\mathcal{G}_I$ between a challenger and an adversary $\mathcal{A}$ (representing a corrupted aggregator, potentially colluding with up to $n-1$ corrupted voters):

1. **Setup.** The challenger generates system parameters params (including the SRS for KZG, the group generator ω for the partition indices, etc.). The challenger gives params to $\mathcal{A}$.
2. **Registration/Voting Phase.** $\mathcal{A}$ may corrupt any subset of voters except at least one honest voter. Each (honest) voter i provides a valid registration (public key) and eventually sends a ballot to $\mathcal{A}$. The adversary $\mathcal{A}$ sees all corrupted voters' ballots and can attempt to omit or alter honest votes.
3. **Tally Submission.** The adversary $\mathcal{A}$ outputs:
   - A polynomial commitment $[P]_1$ (purporting to represent all voter keys/ballots),
   - A partition argument Π using EPA claiming that exactly some partition $I = \{I_1, ..., I_k\}$ yields tallies $(T_1, ..., T_k)$.
4. **Verification.** The challenger (playing the role of validators) checks Π using the EPA verification algorithm.
5. **Winning Condition.** $\mathcal{A}$ *wins* if:
   - The protocol's EPA verification *accepts*,
   - All voters accept the KZG proof of inclusion to the correct sub-polynomial V_j,
   - The final posted tallies do *not* match the actual set of ballots submitted by the honest voters (i.e., one or more honest votes are replaced/omitted/altered) beyond what could have been cast by corrupted voters.

Definition 1 (Integrity Security). *We say the protocol satisfies* integrity *if, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ participating in $\mathcal{G}_I$,* $\Pr[\mathcal{A}$ *wins* $\mathcal{G}_I] \leq neg(\lambda)$, *where* $neg(\lambda)$ *is a negligible function in the security parameter* λ.

Theorem 5. *Assuming q-SDH, Aggios satisfies* Integrity.

Proof. Assume that there exists a polynomial-time adversary $\mathcal{A}$ that wins $\mathcal{G}_I$ with non-negligible probability.

We construct an adversary $\mathcal{B}$ attempt to break the soundness of EPA using $\mathcal{A}$ or the binding property of KZG polynomial commitment. $\mathcal{B}$ simulates the environment for $\mathcal{A}$.

- *Initialization.* $\mathcal{B}$ is provided a KZG challenge and EPA Integrity challenges.
- *Simulating Honest Voters.* For each honest voter i, $\mathcal{B}$ generates n voting keys that are consistent with the KZG commitment (which $\mathcal{B}$ wishes to break) values and the keys as the protocol requires. For any query of $\mathcal{A}$ to open these keys, $\mathcal{B}$ answers consistently, as if it were the real protocol environment.
- *Corrupted Voters.* For corrupted voters, $\mathcal{A}$ directly controls them, so $\mathcal{B}$ just relays their actions.
- $\mathcal{A}$ outputs a commitment $[C(X)]_1$, an EPA proof Π for subsets $\{I_j\}$ and tallies $\{T_j\}$, and KZG inclusion proofs π_i^{KZG} for each honest voter i claiming index i is in sub-polynomial V_j for some candidate j.
- $\mathcal{B}$ extracts the proof Π and proofs π_i^{KZG} from $\mathcal{A}$ and publishes $\mathbf{c}$ and the proofs.

$\mathcal{A}$ winning $\mathcal{G}_I$ implies that at least one honest vote is omitted from the commitments $v_1, ..., v_k$ or they don't satisfy the extended partition arguments while all checks (EPA + KZGs) pass. By assumption, one of the events (out of $n+1$ events) must happen with non-negligible probability.

- $\mathcal{A}$ claims "i is in V_j" but the local transcript shows i's actual vote is a different value or not included at all, namely $\mathcal{A}$ succeeds in forging a KZG proof of inclusion to V_j.
- The aggregator does not use a valid witness for EPA (e.g., skipping some honest index i), thereby forging a passing EPA proof.

$\mathcal{B}$ examines Π (the EPA proof) and $\{\pi_i^{KZG}\}_{i\in[n]}$ for each voter i. $\mathcal{B}$ extracts all the proofs created by $\mathcal{A}$. If the first type of event happens, $\mathcal{B}$ is able to forge at least one KZG proof of inclusion (out of n). However, by KZG binding, providing two consistent but distinct openings is impossible with more than negligible probability.

If the second type of event (forging KZG proof), $\mathcal{B}$ is able to demonstrates a "partition forgery" that contradicts the EPA soundness (Theorem 2).

In either scenario, $\mathcal{B}$ uses $\mathcal{A}$'s success to produce a valid break of the underlying assumption. This shows that if $\mathcal{A}$ wins with probability larger than $\epsilon(\lambda)$ (note that there must be one event can exists with probability larger than $\frac{\epsilon(\lambda)}{n}$). Therefore, $\mathcal{B}$ is a able to breaks KZG binding or breaking the EPA soundness with probability at least $\frac{\epsilon(\lambda)}{n} - \mathsf{neg}(\lambda)$, a contradiction to binding property of KZG (given $q - DLOG$ is hard) or to the fact that EPA satisfies proof of knowledge (Theorem 2).

Hence, $\mathcal{A}$'s advantage must be negligible, implying that no adversary can violate integrity with more than negligible probability.

7.3 Eligibility

We define the eligibility game $\mathcal{G}_E$ between a challenger and an adversary $\mathcal{A}$ as follows:

1. **Setup.** The challenger generates the system parameters $(\mathsf{params}, \mathsf{SRS})$ for the KZG commitment, an index generator ω (for evaluating polynomials at points ω^i), and any public information needed by the EPA partition argument, and provide it to a malicious aggregator $\mathcal{A}$.
2. **Registration.** A set of n voters, some of which may be corrupted, registers with the aggregator. Each *honest* voter i provides a public voting key that the aggregator is supposed to incorporate into a polynomial $C(X)$. Let I_{reg} be the set of *all* registered (honest + corrupted) voter indices.
3. **Aggregator's Polynomial and Commitments.** Eventually, $\mathcal{A}$ (acting as the aggregator) constructs a polynomial $C(X)$ that supposedly includes every registered voter's key and a *KZG commitment* $\mathbf{c}$ to $C(X)$.
4. **Voting / Aggregation.** $\mathcal{A}$ collects votes for some subset in I_{reg} (representing who actually voted after registration). Then it produces final "ballot commitments" $\mathbf{v}_1, \ldots, \mathbf{v}_k$, a tally of the votes, along with EPA partition argument Π claiming that these $\{\mathbf{v}_j\}_{j\in[k]}$ collectively partition $C(X)$ in some valid manner.
5. **Winning Condition.** We say $\mathcal{A}$ *wins* $\mathcal{G}_E$ if:
   (a) The partition argument Π is accepted by the EPA verifier (i.e., it passes all checks),
   (b) The aggregator's final commitments $\mathbf{v}_1, \ldots, \mathbf{v}_k$ include at least one *unregistered* index $i' \notin I_{\text{reg}}$ (thus "inserting" a phantom voter into the final tallies without being detected)

Definition 2 (Eligibility Security). *A protocol satisfies* Eligibility *if, for every polynomial-time adversary $\mathcal{A}$ in the game $\mathcal{G}_E$ defined above,* $\Pr[\mathcal{A} \text{ wins } \mathcal{G}_E] \leq \mathsf{neg}(\lambda)$*, where* $\mathsf{neg}(\lambda)$ *is a negligible function in the security parameter* λ*.*

Theorem 6 (Eligibility). *Assuming q-SDH, Aggios satisfies* Eligibility*.*

Proof. We adapt the proofs from above but focus on the unique aspect of "phantom voters".

Assume that $\mathcal{A}$ wins $\mathcal{G}_E$ with non-neglible probability. Let $\mathcal{B}$ be an adversary that simulates the environment for $\mathcal{A}$ and attempts to break the EPA proof of knowledge. $\mathcal{B}$ forwards params and scalars $\{w_j\}_{j\in[k]}$ to $\mathcal{A}$ and simulates all honest voter registrations exactly as in the real protocol. Let I_{reg} be the set of actual (honest + corrupted) indices of the voters who truly register.

At the end, $\mathcal{A}$ presents the commitment $\mathbf{c}$ to $C(X)$, a tally result, the ballot commitments $\mathbf{v}_1, \ldots, \mathbf{v}_k$ such that there is an index $i' \notin I_{\text{reg}}$ that is included in, say, $\mathbf{v}_j$, and EPA proof Π proving (allegedly) that $\{\mathbf{v}_j\}$ partition $C(X)$. Then $\mathcal{B}$ extract the proof Π from $\mathcal{A}$ and publish ($\mathbf{c}$ and Π (which includes $\{\mathbf{v}_j\}_{j\in[k]}$).

Since, by assumption, $\mathcal{A}$ wins the game with non-neglible probability, $\mathcal{B}$ is able to provide a passing argument (specifically $\mathbf{v}_j$ while $t_j V_j(X) \neq C(X) \bmod Z_{I_j}(X)$) with probability of at least $\epsilon(\lambda) - \mathsf{neg}(\lambda)$, although it does not form a valid extended partition argument of $C(X)$, breaking EPA knowledge soundness - leading to contradiction of Theorem 2). Hence $\mathcal{A}$'s chance of winning $\mathcal{G}_E$ by inserting a phantom index is negligible.

7.4 No Double Voting (One Voter, One Vote)

The No Double Vote Security Game, denoted $\mathcal{G}_{NDV}$, is identical to the game $\mathcal{G}_E$ except for the winning condition. We say $\mathcal{A}$ *wins* $\mathcal{G}_{NDV}$ if:

1. The EPA proof Π *accepts*,
2. All honest voters see a valid KZG proof that *some* sub-polynomial V_j includes their index i,
3. There exists at least one honest index i that is *included in more than one subset* (i.e., double-counting) or included multiple times for a single subset, leading to a double vote for that index (at least one honest index i is *double-counted* – that is, the same index i appears in more than one I_j or is effectively assigned to multiple sub-polynomials V_{j_1}, V_{j_2}, or the aggregator tries to claim multiple distinct openings for the same index in $C(\cdot)$).

Definition 3 (No Double Voting Security).
A protocol satisfies No Double Voting *if, for every polynomial-time adversary* $\mathcal{A}$ *participating in* $\mathcal{G}_{NDV}$, $\Pr[\mathcal{A} \text{ wins } \mathcal{G}_{NDV}] \leq \mathsf{neg}(\lambda)$, *where* $\mathsf{neg}(\lambda)$ *is a negligible function in the security parameter* λ.

Theorem 7. *Assuming q-SDH, Aggios satisfies No Double Voting.*

Proof. Suppose, for contradiction, that there exists a PPT adversary $\mathcal{A}$ that wins $\mathcal{G}_{NDV}$ with non-negligible probability $\epsilon(\lambda)$.

We construct an adversary $\mathcal{B}$ to show that $\mathcal{A}$'s success breaks the knowledge soundness of EPA is non-negligible. As in the previous games, $\mathcal{B}$ simulates the environment of $\mathcal{A}$ and acts as the challenger of the game $\mathcal{G}_{NDV}$. $\mathcal{B}$ provides the params (e..g, SRS) and forwards it to $\mathcal{A}$ exactly as in the real protocol.

Whenever an honest voter i would register or vote, $\mathcal{B}$ generates the voter keys, and provides these to $\mathcal{A}$ as the aggregator. For corrupted voters, $\mathcal{A}$ controls them directly. Eventually, $\mathcal{A}$ outputs:

- The polynomial commitment $[C]_1$,
- The EPA proof Π (with commitments $\mathbf{v}_1, ..., \mathbf{v}_k$), partitioning the set of indices into subsets $\{I_j\}_{j=1}^k$, along with the tally.
- A claim that each honest voter is in exactly one subset, yet $\mathcal{A}$ has *double-counted* at least one honest i in multiple subsets (or in the same subset).

$\mathcal{B}$ inspects the aggregator's subsets $I_1, \ldots, I_k$. Assuming $\mathcal{A}$ wins $\mathcal{G}_{NDV}$, implies that there is some voter with index i that appears in both I_j and $I_{j'}$ (namely, the public key skv_iG is "included" in $V_j(X)$ or $V_{j'}(X)$), or if one "double voted" for the same $V_j(X)$, then there is one vote missing in one of the other polynomials.

Namely $\mathcal{A}$ generated proof Π is a passing proof that does not satisfy the EPA argument. Then, $\mathcal{B}$ extract this proof from $\mathcal{A}$.

Therefore, $\mathcal{B}$ manage to produce an accepting proof for a chosen instance that does not satisfies the EPA argument with probability at least $\epsilon(\lambda)-$(negligible term) - leading to contradiction of the EPA proof of knowledge (Theorem 2). Hence, $\mathcal{A}$ cannot double-count an honest index while passing EPA verification, except with negligible probability.

7.5 Privacy

We define a game $\mathcal{G}_P$ between a challenger $\mathcal{C}$ and a distinguisher (or adversary) $\mathcal{D}$:

1. **Setup.** The system parameters $(\mathsf{params}, \mathsf{SRS})$ for KZG and the public data for the EPA argument are generated and given to $\mathcal{D}$ (as they would be public).
2. **Voting.** Honest voters privately send their votes (or ephemeral secrets) to $\mathcal{A}$ over an encrypted channel. No corruption of $\mathcal{A}$ is assumed regarding privacy.
3. **Aggregator Output.** The aggregator publishes a KZG commitment $\mathbf{c}$ to $C(X)$, the candidate commitments $\mathbf{v}_1, \ldots, \mathbf{v}_k$, and associated scalars, the tallies $m_1, \ldots, m_n$, and an EPA zero-knowledge argument Π that $\{\mathbf{v}_j\}$ partitions $C(X)$.
4. **Distinguisher's Task.** $\mathcal{D}$ sees $(\mathbf{c}, \mathbf{v}_1, \ldots, \mathbf{v}_k, \Pi)$ and tries to guess which candidate j the honest voter i voted for.
5. **Winning Condition.** If $\mathcal{D}$ can identify the index-candidate pair (i, j) with probability (in distinguishing advantage) more than $\frac{1}{k} + \mathsf{neg}(\lambda)$, we say $\mathcal{D}$ *wins* $\mathcal{G}_P$.

Definition 4 (Privacy). *A protocol is* private against third parties *if every PPT $\mathcal{D}$ has only negligible advantage in winning $\mathcal{G}_P$.*

Theorem 8. *Assuming q-DLOG, Aggios satisfies Privacy.*

Proof. Recall that we assume that the aggregator $\mathcal{A}$ is trusted not to reveal any index-to-candidate pairing.

Suppose, for contradiction, that there is a distinguisher $\mathcal{D}$ winning $\mathcal{G}_P$ with probability $\epsilon(\lambda)$ (non-negligible). We build a reduction $\mathcal{B}$ that uses $\mathcal{D}$ to break the zero-knowledge property of the EPA.

$\mathcal{B}$ simulates all honest voters in the standard manner, ensuring each honest voter i sends her vote to $\mathcal{A}$ privately. $\mathcal{A}$ forms zero knowledge commitments $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and proves EPA with Π. Because $\mathcal{B}$ does not reveal how each index is mapped to each $\mathbf{v}_j$, $\mathcal{D}$ sees only the public data $(\mathbf{c}, \mathbf{v}_1, \ldots, \mathbf{v}_k, \Pi)$. If $\mathcal{D}$ can link a specific honest voter i to candidate j with non negligible probability, $\mathcal{B}$ obtains a non-negligible advantage that indicates the zero-knowledge property of Π is compromised, namely that it learned subset membership from Π or the commitments beyond random guessing.

Hence, if $\mathcal{D}$ has non-negligible success, $\mathcal{B}$ breaks zero-knowledge of the EPA argument — leading to contradiction of Theorem 3. Therefore, no external third party can break voter-to-candidate privacy, establishing Theorem 8.

7.6 Conclusion of Game-Based Security

By bounding the adversary's winning probability in each of these games using the EPA knowledge soundness property (Theorem 2), KZG binding, and standard cryptographic assumptions (like q-DLOG in the AGM), we conclude that the

aggregator-based voting protocol achieves *integrity*, *eligibility*, *no double voting*, and *privacy* with high confidence. Consequently, except with negligible probability, an adversarial aggregator cannot alter, insert, or omit any honest voter's ballot without detection, nor can it double-count a single voter's vote.

8 Related Work

Set membership proofs gained popularity with ring signatures [10,19], and their applications to blockchain. Proofs of sub-vectors (which can be seen as batched vector membership proofs) on KZG commitments have been introduced in [37,36], while the notion of subvector proof itself is older [13,22]. Subvector proofs can be based on assumptions other than pairing-based, for example, RSA accumulators [2], PCP [25], hidden order groups [11], or Merkle trees + zk-SNARK [21]. Merkle tree constructions are not homomorphic and rely on Groth16 or other generic zero-knowledge frameworks to reach succinctness. Thus, they are usually slower by two or three orders of magnitude [42] than their counterparts. RSA-based accumulators can be zero-knowledge [41,12,7], but are usually slower by an order of magnitude than pairing-based constructions [33,42,35], with some exceptions [41]. However, creating an (extended) partition argument on RSA accumulators remains an open problem. As such, our partition argument is the only one, to the best of our knowledge, that can efficiently prove a partition over a committed set. Notus, a recent zk RSA accumulator, offers table lookup cost of $O(n \log n)$ precomputation and $O(1)$ lookup [41]. Implementing a partition argument on Notus to compare efficiency with our solution is out of the scope of this paper (notably, proving subset size in RSA accumulators is an open problem). Still, a naïve implementation would rely on quadratic (in k) non-membership proofs to prove non-intersection of subsets, each of size $O(n + \lambda)$, around 6KB for usual settings [5], rendering the scheme impractical. We also observe that our scheme requires no precomputations, except for the distributed setup of KZG SRS, which is typically simpler than setting up RSA groups. Moreover, the KZG SRS can be updated—an essential feature in case of trapdoor leak.

Proxy voting has been examined in several studies [23,24,44,26]. Implementations designed for medium-sized organizations have also been developed [1]. For example, Kulyk et al. [23,24] explore vote delegation scenarios in an e-voting system, focusing on aspects such as coercion resistance and usability in relatively small groups. Zwattendorfer et al. [44] propose secure proxy voting but do not address large-scale efficiency concerns—mainly when frequent votes occur. Li et al. [26] propose a proxy e-voting method with anonymity and unforgeability guarantees, emphasizing cryptographic soundness, but do not introduce any specialized aggregation step for batched vote handling. Meanwhile, Adler [1] demonstrates a working system for 'liquid democracy' in medium-sized organizations, but does not scale to the level of thousands or millions of voters. However, in all these works, the total communication cost and verification time remain linear in the number of voters per election. That is, none of them addresses

scalability in terms of proof size or verification time for frequent or large-scale elections.

Acknowledgements

Doron Zarchy received support from the Luxembourg National Research Fund (FNR) under the CORE project (PABLO grant reference 16326754).

Marius Lombard-Platet received support from the Luxembourg National Research Fund (FNR) under the CORE project (C21/IS/16221219/ImPAKT).

References

1. Adler, A. Liquid democracy-norm, code and developers of democracy beyond representation. In *Conference for E-Democracy and Open Governement* (2013), p. 459.
2. Agrawal, S., and Raghuraman, S. Kvac: Key-value commitments for blockchains and beyond. In *Advances in Cryptology - ASIACRYPT* (2020), Springer.
3. arkworks contributors. `arkworks` zksnark ecosystem, 2022.
4. Bellavitis, C., and Momtaz, P. P. Voting governance and value creation in decentralized autonomous organizations (daos). *Journal of Business Venturing Insights 23* (2025), e00537.
5. Benarroch, D., Campanelli, M., Fiore, D., Gurkan, K., and Kolonelos, D. Zero-knowledge proofs for set membership: efficient, succinct, modular. *Designs, Codes and Cryptography 91*, 11 (Nov 2023).
6. Boneh, D., and Boyen, X. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology 21*, 2 (Apr 2008), 149–177.
7. Boneh, D., Bünz, B., and Fisch, B. Batching techniques for accumulators with applications to iops and stateless blockchains. In *Advances in Cryptology – CRYPTO 2019* (Cham, 2019), A. Boldyreva and D. Micciancio, Eds., Springer International Publishing.
8. Boneh, D., Lynn, B., and Shacham, H. Short signatures from the weil pairing. In *Advances in Cryptology — ASIACRYPT 2001* (Berlin, Heidelberg, 2001), C. Boyd, Ed., Springer Berlin Heidelberg.
9. Boneh, D., and Shoup, V. A graduate course in applied cryptography, v0.6.
10. Bootle, J., et al. Short accountable ring signatures based on ddh. In *European Symposium on Research in Computer Security* (2015), Springer.
11. Campanelli, M., Fiore, D., Greco, N., Kolonelos, D., and Nizzardo, L. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In *Advances in Cryptology - ASIACRYPT* (2020), S. Moriai and H. Wang, Eds., Springer.
12. Campanelli, M., Fiore, D., Han, S., Kim, J., Kolonelos, D., and Oh, H. Succinct zero-knowledge batch proofs for set accumulators. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2022), CCS '22, Association for Computing Machinery.
13. Catalano, D., and Fiore, D. Vector commitments and their applications. In *Public-Key Cryptography - PKC* (2013), K. Kurosawa and G. Hanaoka, Eds., Springer.

14. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., and Ward, N. P. Marlin: Preprocessing zksnarks with universal and updatable SRS. In *Advances in Cryptology - EUROCRYPT 2020* (2020), Springer.
15. Feichtinger, R., Fritsch, R., Vonlanthen, Y., and Wattenhofer, R. The hidden shortcomings of (d)aos – an empirical study of on-chain governance, 2023.
16. Fuchsbauer, G., Kiltz, E., and Loss, J. The algebraic group model and its applications. In *CRYPTO* (2018), Springer.
17. Gabizon, A., Williamson, Z. J., and Ciobotaru, O. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive* (2019).
18. Groth, J. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT 2016* (Berlin, Heidelberg, 2016), M. Fischlin and J.-S. Coron, Eds., Springer Berlin Heidelberg, pp. 305–326.
19. Groth, J., and Kohlweiss, M. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), Springer.
20. Han, J., Lee, J., and Li, T. A review of dao governance: Recent literature and emerging trends.
21. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N., et al. Zcash protocol specification. *GitHub: San Francisco, CA, USA 4*, 220 (2016), 32.
22. Kate, A., Zaverucha, G. M., and Goldberg, I. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010* (Berlin, Heidelberg, 2010), M. Abe, Ed., Springer Berlin Heidelberg, pp. 177–194.
23. Kulyk, O., Marky, K., Neumann, S., and Volkamer, M. Introducing proxy voting to helios. In *2016 11th International Conference on Availability, Reliability and Security (ARES)* (2016), IEEE.
24. Kulyk, O., Neumann, S., Marky, K., Budurushi, J., and Volkamer, M. Coercion-resistant proxy voting. *computers & Security 71* (2017).
25. Lai, R. W. F., and Malavolta, G. Subvector commitments with application to succinct arguments. In *Advances in Cryptology - CRYPTO* (2019), A. Boldyreva and D. Micciancio, Eds., Springer.
26. Li, W., and Xiong, H. Efficient proxy re-encryption scheme for e-voting system. *KSII Transactions on Internet & Information Systems 15*, 5 (2021).
27. Lumineau, F., Wang, W., and Schilke, O. Blockchain governance – a new way of organizing collaborations? *Organization Science* (03 2020).
28. McCAHERY, J. A., SAUTNER, Z., and STARKS, L. T. Behind the scenes: The corporate governance preferences of institutional investors. *The Journal of Finance 71*, 6 (2016), 2905–2932.
29. Nazirkhanova, K., Gunjur, V., Jesus, X. P. C.-D., and Boneh, D. Kite: How to delegate voting power privately, 2025.
30. Posen, J., and Kattis, A. A. Caulk+: Table-independent lookup arguments. *Cryptology ePrint Archive* (2022).
31. Scott, M. Pairing implementation revisited. *IACR Cryptol. ePrint Arch.* (2019).
32. Sharma, T., Potter, Y., Pongmala, K., Wang, H., Miller, A., Song, D., and Wang, Y. Future of algorithmic organization: Large-scale analysis of decentralized autonomous organizations (daos), 2024.
33. Srinivasan, S., Karantaidou, I., Baldimtsi, F., and Papamanthou, C. Batching, aggregation, and zero-knowledge proofs in bilinear accumulators. In

Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (New York, NY, USA, 2022), CCS '22, Association for Computing Machinery.
34. Tomescu, A., Abraham, I., Buterin, V., Drake, J., Feist, D., and Khovratovich, D. Aggregatable subvector commitments for stateless cryptocurrencies. In *12th International Conference, SCN 2020* (Berlin, Heidelberg, 2020), Springer-Verlag.
35. Tomescu, A., Bhupatiraju, V., Papadopoulos, D., Papamanthou, C., Triandopoulos, N., and Devadas, S. Transparency logs via append-only authenticated dictionaries. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2019), CCS '19, Association for Computing Machinery.
36. Tomescu, A., Chen, R., Zheng, Y., Abraham, I., Pinkas, B., Golan-Gueta, G., and Devadas, S. Towards scalable threshold cryptosystems. In *Symposium on Security and Privacy, SP* (2020), IEEE.
37. Tomescu Nicolescu, I. A. *How to keep a secret and share a public key (using polynomial commitments).* PhD thesis, Massachusetts Institute of Technology, 2020.
38. von zur Gathen, J., and Gerhard, J. *Modern Computer Algebra*, 3rd ed. Cambridge University Press, 2013.
39. Wang, F., Cohney, S., and Bonneau, J. Sok: Trusted setups for powers-of-tau strings. In *International Conference on Financial Cryptography and Data Security* (2025), Springer, pp. 197–218.
40. Weidener, L., Laredo, F., Kumar, K., and Compton, K. Delegated voting in decentralized autonomous organizations: a scoping review. *Frontiers in Blockchain Volume 8 - 2025* (2025).
41. Xin, J., Haghighi, A., Tian, X., and Papadopoulos, D. Notus: dynamic proofs of liabilities from zero-knowledge rsa accumulators. In *Proceedings of the 33rd USENIX Conference on Security Symposium* (2024), SEC '24, USENIX Association.
42. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., and Simkin, M. Caulk: Lookup arguments in sublinear time. In *SIGSAC* (2022).
43. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., and Ràfols, C. Baloo: Nearly optimal lookup arguments. *Cryptology ePrint Archive* (2022).
44. Zwattendorfer, B., Hillebold, C., and Teufl, P. Secure and privacy-preserving proxy voting system. In *2013 IEEE 10th International Conference on e-Business Engineering* (2013), IEEE.

A Definitions

A.1 q-SDH and q-DLOG assumptions

The q-SDH assumption was first introduced in [6] (in a single group but has been extended to the full bilinear setting in e.g. [17]), and asserts that for in a group $\mathbb{G}$ and any SRS $(g, g^x, \ldots, g^{x^q})$, for any secret x, the adversary cannot compute $(c, g^{\frac{1}{x-c}})$ for any $c \neq x$. More formally,

Definition 5 (q-SDH). *Let $\lambda \in \mathbb{N}$, $\mathbb{G} \xleftarrow{\$} \mathsf{Setup}(\lambda)$ be a group of prime order p generated by g, and $x \in \mathbb{F}_p$ be a random number. Given as input the tuple*

$(g, g^x, \ldots, g^{x^q})$, *for every PPT adversary* $\mathcal{A}_{q-SDH}$, *the probability* $Pr[\mathcal{A}_{q-SDH}(g, g^x, \ldots, g^{x^q}) = (c, g^{\frac{1}{x-c}})]$ *is negligible in* λ *for any* $c \in \mathbb{F}_p \backslash \{x\}$.

Similarly, the q-DLOG assumption was introduced in [16], reformulated in [17] and states that from an SRS, the adversary cannot retrieve the secret x that generated said SRS. More formally,

Definition 6 (bilinear q-DLOG). *Let* $\mathbb{G}_1, \mathbb{G}_2$ *be pairing friendly groups of security parameter* λ, *and prime order* p. *Let* $x \in \mathbb{F}_p$ *be a random number. Given the SRS* $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x, \ldots, g_2^{x^q})$, *for every PPT adversary* $\mathcal{A}_{q-DLOG}$ *trying to guess the discrete log, the probability* $Pr[\mathcal{A}_{q-DLOG}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x, \ldots, g_2^{x^q}) = x]$ *is negligible in* λ.

It is immediate that breaking q-DLOG implies breaking q-SDH.

A.2 Completeness

A protocol satisfies completeness, if a prover giving a honest proof for a statement will almost always convince a verifier. Formally,

Definition 7. *A protocol between a prover* Prove *and a verifier* Verify *for a relation* R *and public parameters* pp *is complete if* $\forall (s, w) \in R$,

$$P\left[\mathsf{Verify}(srs, s, \pi) = 1 \,\middle|\, \begin{matrix}(srs, td) \leftarrow \mathsf{Setup}(pp) \\ \pi \leftarrow \mathsf{Prove}(pp, srs, (s, w))\end{matrix}\right] = 1$$

A.3 Knowledge-soundness

A protocol satisfies knowledge soundness, if the probability that a cheating prover can create a valid proof for a statement for which they know no witness, is negligible. We model the knowledge of the witness by the existence of an extractor that would be able to extract the witness from the proof. Formally,

Definition 8. *A protocol between a prover* Prove *and verifier* Verify *for a relation* R *and public parameters* pp *has Knowledge Soundness in the Algebraic Group Model if there exists an efficient* $\mathcal{E}$ *such that the probability of any algebraic adversary* $\mathcal{A}$ *winning the following game is negligible in* λ*:*

$$\left[\begin{matrix}(s, w) \notin R \\ \mathsf{Verify}(srs, s, \pi) = 1\end{matrix} \,\middle|\, \begin{matrix}(srs, x) \leftarrow \mathsf{Setup}(pp) \\ (s, \pi) \leftarrow \mathcal{A}(pp, srs) \\ w \leftarrow \mathcal{E}(srs, s, \pi)\end{matrix}\right]$$

A.4 Zero-knowledge

A non-interactive protocol is zero-knowledge if one cannot distinguish between the distribution of transcripts from between a honest prover and a verifier, and from a simulation that does not know the witness. More formally,

Definition 9. *A non-interactive protocol between a prover* Prove *and verifier* Verify *for a relation R and public parameters pp is (computational) zero-knowledge if there exists a PPT* Simulate *such that, for any PPT $\mathcal{A}$, the following distributions:*

$$\left\{ s, \pi \;\middle|\; \begin{array}{l} (srs, x) \leftarrow \mathsf{Setup}(pp) \\ s \leftarrow \mathcal{A}(srs) \\ \pi \leftarrow \mathsf{Prove}(pp, srs, (s, \mathcal{O}^R(s)) \end{array} \right\}$$

and

$$\left\{ s, sim \;\middle|\; \begin{array}{l} (srs, x) \leftarrow \mathsf{Setup}(pp) \\ s \leftarrow \mathcal{A}(srs) \\ sim \leftarrow \mathsf{Simulate}(pp, srs, x, s) \end{array} \right\}$$

are (computationally) indistinguishable, where $\mathcal{O}^R(s)$ is an oracle that for any s from R, outputs w such that $(s, w) \in R$.

A.5 Equal log statement

Definition 10. *Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of the same order, with respective generators g_1, g_2. Elements $a \in \mathbb{G}_1, b \in \mathbb{G}_2$ are said to have equal log with respect to g_1 and g_2 if and only if there exists $\alpha \in \mathbb{N}$ such that $a = [\alpha]g_1$ and $b = [\alpha]g_2$.*

We also define the proving algorithm $\mathsf{EqLog.Prove}(g_1, a, g_2, b, \alpha)$ that outputs a proof π and $\mathsf{EqLog.Verify}(g_1, a, g_2, b, \pi)$, the verification algorithm such that we have correct verification, i.e.
$\mathsf{EqLog.Verify}(g_1, a, g_2, b, \mathsf{EqLog.Prove}(g_1, a, g_2, b, \alpha)) = 1$ if and only if a, b have equal log α.

The Chaum-Pedersen protocol [9, Section 20.3.6] is an example on how to implement the equal log proof system.

B Benchmark detailed results

In Table 1, we give the proportion of the time spent in each part of the prover's algorithm, for $k = 10$. One can observe that the commitments are the main bottlenecks. The figures are very similar for other values of k. In Tables 2 and 3, we give the computation time for the prover and verifier for the EPA protocol, in seconds.

C Subprotocols for Aggios-Split

In this section, we explain, describe and prove the soundness and zero-knowledge properties of the two subprotocols that we use in Aggios-Split.

	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}
$P_{1,j}$	3%	3%	4%	4%	5%	6%	7%	9%	10%	11%	12%
$P_{2,j}$	3%	3%	4%	5%	6%	7%	8%	10%	11%	12%	13%
Commit R1	53%	52%	53%	51%	49%	49%	48%	44%	42%	41%	41%
Commit R2	37%	39%	36%	36%	37%	34%	31%	32%	32%	29%	26%
Other	4%	4%	4%	4%	3%	4%	5%	4%	5%	6%	7%

Table 1. Bottlenecks in the EPA protocol, showing the proportion (rounded to the closest unit) of time passed in each part of the algorithm, for different values of n and $k = 10$ subsets, averaged over 50 runs.

	$\mathbf{2^8}$	$\mathbf{2^9}$	$\mathbf{2^{10}}$	$\mathbf{2^{11}}$	$\mathbf{2^{12}}$	$\mathbf{2^{13}}$	$\mathbf{2^{14}}$	$\mathbf{2^{15}}$	$\mathbf{2^{16}}$	$\mathbf{2^{17}}$	$\mathbf{2^{18}}$
2	0.156	0.182	0.362	0.746	1.398	2.347	3.467	7.074	9.085	25.003	40.682
3	0.182	0.257	0.444	0.928	1.573	2.674	4.948	8.628	14.355	27.352	52.446
5	0.215	0.432	0.575	1.162	1.951	3.219	5.443	9.995	19.069	34.008	66.201
10	0.351	0.577	0.890	1.632	2.940	5.234	8.210	15.426	34.605	55.861	108.293
50	1.169	1.959	3.190	5.936	10.602	17.804	29.927	57.990	109.577	210.796	417.576
100	2.329	3.719	6.186	11.171	20.343	35.164	57.060	107.249	204.450	394.478	797.238

Table 2. Prover computation time for the EPA protocol, in seconds, as a function of n and k. Results have been averaged over 50 runs.

	$\mathbf{2^8}$	$\mathbf{2^9}$	$\mathbf{2^{10}}$	$\mathbf{2^{11}}$	$\mathbf{2^{12}}$	$\mathbf{2^{13}}$	$\mathbf{2^{14}}$	$\mathbf{2^{15}}$	$\mathbf{2^{16}}$	$\mathbf{2^{17}}$	$\mathbf{2^{18}}$
2	0.011	0.011	0.009	0.009	0.009	0.009	0.009	0.008	0.008	0.008	0.009
3	0.012	0.012	0.012	0.011	0.012	0.011	0.011	0.009	0.010	0.010	0.010
5	0.017	0.019	0.017	0.017	0.017	0.018	0.016	0.015	0.015	0.014	0.014
10	0.030	0.031	0.030	0.030	0.030	0.030	0.024	0.023	0.025	0.019	0.019
50	0.137	0.134	0.138	0.138	0.130	0.116	0.097	0.091	0.089	0.084	0.079
100	0.275	0.261	0.273	0.264	0.263	0.256	0.213	0.206	0.203	0.188	0.180

Table 3. Verifier computation time for the EPA protocol, in seconds, as a function of n and k. Results have been averaged over 50 runs.

C.1 Voter well-formedness (VWF)

Protocol As stated in Section 6.3, the voter needs to prove that their vote $S(X)$ is a commitment to $kt+1$ votes, with at least t votes for each of the k candidates.

This is summarised by the relation $\mathcal{R}_{\text{VWF}}$ described as

$$\left[\begin{array}{c|c} \{[x^{i-1}]_1, [x^{i-1}]_2\}_{i\in[d]}\ ; & [S]_1 = \mathbf{s}, [T]_1 = \mathbf{t} \\ \omega, k, t, (w_j)_{j\le k}, \mathbf{s}, \mathbf{t}; & \exists I \subset [kt+1], |I| = kt \\ S(X), T(X) & \forall i \in I, T(\omega^i) = S(\omega^i) \\ & \exists I_1, \ldots, I_k \subset I \\ & |I_j| = t, \cup_j I_j = I \\ & \forall j \in [k], \forall i \in I_j, T(\omega^i) = w_j \end{array}\right]$$

The VWF is a simple combination of an EPA proof, showing that $T_i(X)$ consists of t votes for each candidate, and a subvector proof, showing that $T_i(X)$ is indeed a subvector of $S_i(X)$, over some unknown domain. We then prove that the domain is of size at least kt by adding a degree binding proof. Thus, the protocol is as follows:

Voter well-formedness (VWF)

Prove$(S_i(X), k, t)$:

- Create and commit to a vector $T_i(X)$ such that T_i is a subvector of S_i over an undisclosed domain D_i of size kt, and such that T_i contains equally as many votes for each of the k candidates.
- Randomly sample the blinding factors $\beta_1^{(i)}$, $\beta_2^{(i)}, \beta_3^{(i)}$, and compute blinded polynomials $S_i'(X) \leftarrow S_i(X) + \beta_1^{(i)} Z_{\mathbb{H}}(X)$, $T_i'(X) \leftarrow T_i(X) + \beta_2^{(i)} Z_{\mathbb{H}}(X)$ and $Z'_{D_i}(X) \leftarrow \beta_6^{(i)} Z_{D_i}(X)$
- Compute $Q_1(X) \leftarrow (S_i'(X) - T_i'(X))/Z'_{D_i}(X)$ and $\pi_1^{(i)} \leftarrow [Q_1]_1$
- Prove that Z'_{D_i} has correct degree by degree-binding $\pi_1^{(i)}$, with the computation of $\pi_2^{(i)} \leftarrow [X^n Q_1(X)]_1$
- Prove that $\mathbf{zd}_i \leftarrow [Z'_{D_i}]_2$ is a vanishing polynomial by computing $\pi_3^{(i)} \leftarrow [Q_2]_1$, with $Q_2(X) \leftarrow Z_{\mathbb{H}}(X)/Z'_{D_i}(X)$
- Compute $\pi_4^{(i)}$, the EPA-linear proof that T_i' partitions into t votes for each candidate.
- Compute $\pi_5^{(i)} \leftarrow [X^{n-kt+1} T'(X)]_1$
- Compute $\mathbf{s}_i \leftarrow [S_i']_1, \mathbf{t}_i \leftarrow [T_i']_1$
- Return $\mathbf{s}_i, \mathbf{t}_i, \mathbf{zd}_i, \pi_1^{(i)}, \pi_2^{(i)}, \pi_3^{(i)}, \pi_4^{(i)}, \pi_5^{(i)}$

Verify$(\mathbf{s}_i, \mathbf{t}_i, \mathbf{zd}_i, \pi_1^{(i)}, \pi_2^{(i)}, \pi_3^{(i)}, \pi_4^{(i)}, \pi_5^{(i)})$

- Check $e(\mathbf{s}_i - \mathbf{t}_i, [1]_2) = e(\pi_1^{(i)}, \mathbf{zd}_i)$
- Check $e([Z_{\mathbb{H}}]_1, [1]_2) = e(\pi_3^{(i)}, \mathbf{zd}_i)$
- Check $e(\pi_2^{(i)}, [1]_2) = e(\pi_1^{(i)}, [X^n]_2)$
- Check $\mathsf{EPA\text{-}linear.Verify}(\pi_3^{(i)}, \mathbf{t}_i, k)$ the validity of the EPA-linear proof of partition over $\mathbf{t}_i$
- Check $e(\pi_5^{(i)}, [1]_2) = e(\mathbf{t}_i, \left[X^{n-kd+1}\right]_2)$

Accept if and only if all the checks pass. Note that the lower bound of T_i degree is indirectly checked by the EPA proof.

Note: To make the ACE protocol (see Section C.2) zero-knowledge, we will require extra blinding parameters on S', as detailed in Theorem 12.

Soundness

Theorem 9. *The protocol VWF is sound in the AGM, assuming the q-SDH for* $(\mathbb{G}_1, \mathbb{G}_2)$.

Proof. The proof is quite straightforward, as it is a combination of a subvector proof, an EPA proof and a degree bound. Let us suppose that the adversary outputs $\mathbf{s}_i, \mathbf{t}_i, \mathbf{zd}_i, \pi_1^{(i)}, \pi_2^{(i)}, \pi_3^{(i)}, \pi_4^{(i)}, \pi_5^{(i)}$. The proofs in $\pi_4^{(i)}$ prove that, assuming q-DLOG (implied by q-SDH), with overwhelming probability and because of Theorem 2, that $T_i(X)$ is a vector consisting of k times the value $sk_i^{c_j}$, for each $j \in [k]$. The real pairing check $e([Z_{\mathbb{H}}]_1, [1]_2) = e(\pi_3^{(i)}, \mathbf{zd}_i)$ corresponds with overwhelming probability to the ideal pairing check $Z_{\mathbb{H}}(X) = 0 \mod Z_{D_i}(X)$, which indicates that Z_D is indeed a vanishing polynomial, or a scalar.

The EPA proof $\pi_4^{(i)}$ proves that T_i has at least kt values, i.e. $deg(T_i) \geq kd-1$. The check $e(\pi_5^{(i)}, [1]_2) = e(\mathbf{t}_i, \left[X^{n-kd+1}\right]_2)$ indicates that with overwhelming probability, $deg(T_i(X)) \leq kd - 1$. We therefore conclude that $deg(T_i) = kd$.

The real pairing check $e(\mathbf{s}_i - \mathbf{t}_i, [1]_2) = e(\pi_1^{(i)}, \mathbf{zd}_i)$ corresponds with overwhelming probability to the ideal pairing check $S_i(X) - T_i(X) = 0 \mod Z_{D_i}(X)$. Knowing that Z_D is a vanishing polynomial (or a scarlar), we conclude that T_i is a subvector of S_i over Z_D, and because of the degree check on π_1, we conclude with overwhelming probability that Z_D has degree $deg(S_i) - deg(Q_1) = kt + 1$, hence the kt values of T_i on which the EPA partition argument was run form indeed a subvector of S_i.

As such, given a valid transcript output, the extractor extracts $S_i(X), T_i(X)$ from $\mathbf{s}_i, \mathbf{t}_i$, which, given the previous paragraphs, is a witness to $\mathcal{R}_{\text{VWF}}$, therefore the adversary loses the game.

It should be noted that, from this protocol, we do not know if the commitment to S_i consists of only kt elements or rather $kt + 1$. However, assuming the last vote is invalid, this will be revealed during the ACE .

Zero-knowledge

Theorem 10. *The protocol VWF is zero-knowledge.*

Proof. Since there is no commitment in the common inputs of the protocol, we observe that a simulator could simply run a honest execution with randomly generated polynomial $S_i(X)$ and a random subvector $T_i(X)$. The proof π_3 is similarly generated by a honest execution of EPA-linear over T_i. We then observe that the output elements uniquely match a random valid distribution because:

- $\mathbf{s}_i$ is uniformly blinded by $\beta_1^{(i)}$
- $\mathbf{t}_i$ is uniformly blinded by $\beta_2^{(i)}$
- $\mathbf{z}d_i$ is uniformly blinded by $\beta_3^{(i)}$
- $\pi_1^{(i)}$ uniquely satisfies $e(\mathbf{s}_i - \mathbf{t}_i, [1]_2) = e(\pi_1^{(i)} \mathbf{zd}_i)$
- $\pi_2^{(i)}$ uniquely satisfies $e(\pi_2^{(i)}, [1]_2) = e(\pi_1^{(i)}, [X^n]_2)$
- $\pi_3^{(i)}$ uniquely satisfies $e([Z_{\mathbb{H}}]_1, [1]_2) = e(\pi_3^{(i)}, \mathbf{zd}_i)$
- $\pi_5^{(i)}$ uniquely satisfies $e(\pi_5^{(i)}, [1]_2) = e(\mathbf{t}_i, \left[X^{n-kd+1}\right]_2)$
- The components of $\pi_4^{(i)}$ also match a uniquely valid random distribution, as proven in Theorem 3

C.2 Proof of aggregator correct execution

In this section, we describe how to prove that an aggregator has correctly executed their votes, and then prove that our protocol is sound and zero-knowledge.

Protocol As seen in Section 6.3, the aggregator l essentially needs to prove two things:

- For each $j \in [k]$, the polynomial $V_j^{(l)}$ is equal to 0 outside of where it is a subvector of $C^{(l)}$: for $Z_{I_j}^{(l)}$ the vanishing polynomial associated to $V_j^{(l)}$ in the EPA proof, ${V'}_j^{(l)}(X) = 0 \mod Z_{\overline{I_j}}$ with $Z_{\overline{I_j}} \leftarrow Z_{\mathbb{H}}/Z'_{I_j}$.
- $C^{(l)}(X) - \sum_{j\in[k]} S_i(M_l(X))V_j^{(l)}(X) = 0 \mod (X - \omega^i)$ for each voter i registered with the aggregator l;

For the second point, as noted previously, the aggregator does not have access to $S_i(X)$. However, they know the opening of S_i at point $M_l(\omega^i)$, with M_l a publicly computable polynomial. They will thus derive another polynomial $R^{(l)}$, prove its equality with S_i at point $M_l(\omega^i)$, and substitute S for R in the relation.

For this protocol to work, the voter must send additional data to the aggregator, on top of its vote and a WVF proof. Namely, a commitment to $S_i(M_l(X))$, a proof of soundness of said commitment, and an opening proof at $S_i(M_l(\omega^i))$ that the aggregator will reuse to prove that its polynomial $R^{(l)}$ is well-formed at ω^i, without revealing information about $R^{(l)}(\omega^i)$.

More formally, the voter i provides to the aggregator l a ZKP for the relation $\mathcal{R}_{\text{ACE1}}$ defined as follows:

$$\left[\begin{array}{c|c} \{[x^{i-1}]_1, [x^{i-1}]_2\}_{i\in[d]}; & \mathbf{s} = [S]_1 \\ \mathbf{s}, M(X), \omega, w_i, i; & S(M(\omega^i)) = w_i \\ S(X) & \end{array}\right]$$

Following that proof, the aggregator l generates a ZKP to the other aggregators of the relation $\mathcal{R}_{\text{ACE2}}$ defined as follows:

$$\left[\begin{array}{c|l} & [S_i]_1 = \mathbf{s}_i, [S_i \circ M]_2 = \mathbf{sm}_i, [C]_1 = \mathbf{c} \\ \{[x^{i-1}]_1, [x^{i-1}]_2\}_{i\in[d]}; & \exists R \in \mathbb{F}[X], \forall i \in G, R(\omega^i) = S_i(M(\omega^i)) \\ G, (\mathbf{s}_i, \mathbf{sm}_i)_{i\in G}, M(X), \mathbf{c}; & \exists \text{ EPA partitioning C over G and:} \\ (S_i(M(\omega^i)))_{i\in G} & \quad \forall i \in G \backslash I_j, V_j(\omega^i) = 0 \\ & \quad \forall i \in G, C^{(l)}(\omega^i) = R(\omega^i) \sum_j V_j(\omega^i) \end{array}\right]$$

where G is the set of voters registered to the aggregator, and V_j are the subvectors of C over I_j in the EPA proof.

We thus use the following protocol.

Aggregator l Correct Execution (ACE)

Voter i, voting phase: For each aggregator l it registered to:

- Compute polynomial $M_l(X)$, or retrieve it from a trusted source
- Compute $\mathbf{sm}_i^{(l)} \leftarrow [S_i'(M_l(X))]_2$, compute a challenge $\alpha_{i,l}$ via the Fiat-Shamir heuristic
- Compute $y_1^{(i,l)}, \rho_1^{(i,l)} \leftarrow \mathsf{KZG.Open}(S_i'(X), M_l(\alpha_{i,l}))$
- Compute $_, \rho_2^{(i,l)} \leftarrow \mathsf{KZG.Open}(S_i'(M_l(X)), \alpha_{i,l})$
- Compute $y_3^{(i,l)}, \rho_3^{(i,l)} \leftarrow \mathsf{KZG.Open}(S_i'(M_l(X)), \omega^i)$. $y_3^{(i,l)}$ is the vote of voter i delegated to l and must be equal to one of the predefined candidate values, say $w_i^{(l)}$.
- Send $\mathbf{sm}_i^{(l)}, \rho_1^{(i,l)}, \rho_2^{(i,l)}, y_1^{(i,l)}$ to a channel shared by all aggregators
- Send $\rho_3^{(i,l)}, w_i^{(l)}$ to aggregator l

Aggregator l, voting phase:

- For each registered voter, check that $\mathsf{VWF.Verify}= 1$, and that $w_i^{(l)}$ is a valid vote.
- Compute $M_l(X)$ and $\alpha_{i,l}$ for each voter i, and check that $\mathsf{KZG.Verify}(\mathbf{s}_i, M_l(\alpha_{i,l}), y_1^{(i,l)}, \rho_1^{(i,l)})$, $\mathsf{KZG.Verify}(\mathbf{sm}_i^{(l)}, \alpha_{i,l}, y_1^{(i,l)}, \rho_2^{(i,l)})$, $\mathsf{KZG.Verify}(\mathbf{sm}_i^{(l)}, \omega^i, w_i^{(l)}, \rho_3^{(i,l)})$ for each registered voter i. Note that $\mathbf{sm}_i^{(l)} \in \mathbb{G}_2$ but the verification algorithm can be trivially adapted.
- Abort and signal voter to other aggregators if the checks fails.

- Construct $R^{(l)}(X)$ such that $R^{(l)}(\omega^i) = y_3^{(i,l)}$. Sample a random blinding factors $\epsilon^{(l)}$, and compute blinded $R_2^{(l)}(X) \leftarrow R^{(l)}(X) + \epsilon^{(l)} Z_{\mathbb{H}}(X)$
- Compute $_, \sigma_4^{(i,l)} \leftarrow \mathsf{KZG.Open}(R_2^{(l)}, \omega^i)$, as well as $\rho_4^{(l,i)} \leftarrow \sigma_4^{(i,l)} - \rho_3^{(i,l)}$
- Compute $\rho_5^{(i,l)}$, the proof of the relation $C^{(l)}(X) - R_2^{(l)}(X) \sum_j V_j^{(l)}(X) = 0 \mod (X - \omega^i)$ for each i. These proofs can be batched in a single one.
- Compute $\mathbf{r}^{(l)} \leftarrow \left[R_2^{(l)}\right]_2$, and $\mathbf{rv}^{(l)} \leftarrow \left[R_2^{(l)} \sum_j V_j^{(l)}\right]_1$
- Retrieve all the votes delegated to the aggregator by honest voters. Compute $\rho_6^{(l)} \leftarrow \mathsf{EPA.Prove}$, the proof of the votes delegated to aggregator l, and extract the polynomials $V_j'(X)$ and sets I_j.
- Compute $Q_j(X) \leftarrow V_j'(X)/Z'_{I_j}(X)$ and $\rho_7^{(l,j)} \leftarrow [Q_j]_1$
- Disclose to other aggregators $\mathbf{r}^{(l)}, \mathbf{rv}^{(l)}, \rho_4^{(i,l)}, \rho_5^{(i,l)}, \rho_7^{(l,j)}$. Publicly disclose $\rho_6^{(l)}$.

Aggregator, verification phase: Receive ACE and EPA proofs from all other aggregators. For each received proofs from aggregator l:

- Check $\mathsf{EPA.Verify}(\rho_6^{(l)}) = 1$. Extract $\mathbf{c}^{(l)}, v_j^{(l)}, \mathbf{z}_j^{(l)}$ from $\rho_6^{(l)}$.
- Check $e(\mathbf{v}_j^{(l)}, \mathbf{z}_j^{(l)}) = e(\rho_7^{(l,j)}, [X^n - 1]_2)$ for all $j \in [k]$
- Check $e([1]_1, \mathbf{r}^{(l)} - \mathbf{sm}_i^{(l)}) = e(\rho_4^{(i,l)}, \left[X - \omega^i\right]_2)$ for all i registered with l
- Check $e(\mathbf{rv}^{(l)}, [1]_2) = e(\sum_j \mathbf{v_j^{(l)}}, \mathbf{r}^{(l)})$
- Check $e(\mathbf{c}^{(l)} - \mathbf{rv^{(l)}}, [1]_2) = e(\rho_5^{(i,l)}, \left[X - \omega^i\right]_2)$ for each voter i registered with aggregator l
- Abort and signal aggregator on the public channel if the checks fail. Otherwise, accept.

Soundness

Theorem 11. *The protocol ACE is knowledge sound in the AGM, assuming the q-SDH hypothesis on* $(G_1, \mathbb{G}_2)$.

Proof. The proof is separated in two parts, to prove first the soundness of what the voter sends to the aggregator, then of what the aggregator sends to other aggregators.

Knowledge soundness of $\mathcal{R}_{\text{ACE1}}$

We first prove that if the checks that the aggregator runs on the voter input pass, then there is knowledge soundness of the relation $\mathcal{R}_{\text{ACE1}}$. Assume the aggregator receives $\mathbf{s}, \mathbf{sm}, y_1, w, \rho_1, \rho_2$ from voter i. Let $S(X)$ be the polynomial associated with $\mathbf{s}$ and $SM(X)$ the polynomial associated with $\mathbf{sm}$. The

verification checks $\mathsf{KZG.Verify}(\mathbf{s}, M_l(\alpha), y_1, \rho_1)$ and $\mathsf{KZG.Verify}(\mathbf{sm}, \alpha, y_1, \rho_2)$ correspond, with overwhelming probability under the q-DLOG assumption (implied by q-SDH), to the ideal check $S(M_l(\alpha)) = y_1$ and $SM(\alpha) = y_1$, thus, using the Schwartz-Zippel lemma, proving that $SM(X) = S(M_l(X))$ with overwhelming probability. Finally, the verification $\mathsf{KZG.Verify}(\mathbf{sm}_i, \omega^i, y_3^{(i,l)}, \rho_3^{(i,l)})$ proves with overwhelming probability under the q-SDH assumption that $S(M_l(\omega^i)) = w$. As such, an extractor would extract S from $\mathbf{s}$ and thus with overwhelming probability find a witness to the relation R_{ACE1}, therefore the adversary loses the game.

Knowledge soundness of $\mathcal{R}_{\text{ACE2}}$

We now prove the second part of the knowledge soundness, namely that if an aggregator can convince other aggregators, then they know a witness to $\mathcal{R}_{\text{ACE2}}$. For simplicity, we assume here that all voters are registered with all aggregators (i.e. $G = [n]$), but the proof can be adapted otherwise.

Let us consider a valid transcript $\mathbf{r}^{(l)}, \mathbf{rv}^{(l)}, \rho_4^{(i,l)}, \rho_5^{(i,l)}, \rho_6^{(l)}, \rho_7^{(l,j)}$, with voter input $\mathbf{sm}_i, \rho_1^{(i,l)}, \rho_2^{(i,l)}, y_1^{(i,l)}$. Let $SM(X), RV(X), R(X), C(X), V_j(X)$ be the polynomials derived from $\mathbf{sm}_i, \mathbf{rv}^{(l)}, \mathbf{r}^{(l)}, \mathbf{c}^{(l)}, \mathbf{v}_j^{(l)}$, with $\mathbf{c}^{(l)}, \mathbf{v}_j^{(l)}$ coming from the EPA proof in $\rho_6^{(l)}$.

According to Theorem 2, $\rho_6^{(l)}$ proves that the submitting aggregator knows a witness to the EPA statement on its registered voters, with overwhelming probability. The real pairing check $e(\mathbf{rv}^{(l)}, [1]_2) = e(\sum_j \mathbf{v_j^{(l)}}, \mathbf{r}^{(l)})$ corresponds with overwhelming probability to the ideal pairing check $RV(X) = R(X) \sum_j V_j(X)$. Since the pairing check $e(\mathbf{c}^{(l)} - \mathbf{rv}^{(l)}, [1]_2) = e(\rho_5^{(i,l)}, [X - \omega^i]_2)$ corresponds with overwhelming probability to the ideal pairing check $C(X) - RV(X) = 0 \mod (X - \omega^i)$, we conclude that with overwhelming probability, $C(\omega^i) = R(\omega^i) \sum_j V_j(\omega^i)$. Finally, the ideal pairing check $e([1]_1, \mathbf{r}^{(l)} - \mathbf{sm}_i^{(l)}) = e(\rho_4^{(i,l)}, [X - \omega^i]_2)$ corresponds with overwhelming probability to the ideal check $R(X) - SM(X) = 0 \mod (X - \omega^i)$, which implies that $R(\omega^i) = SM(\omega^i)$. The aggregator having received a proof that $\mathbf{sm}_i^{(l)}$ is a valid commitment to $S_i'(M_l(X))$ in the previous phase, we get that with overwhelming probability, $R(\omega^i) = S_i'(M_l(\omega^i))$. Finally, the real pairing check $e(\mathbf{v}_j^{(l)}, \mathbf{z}_j^{(l)}) = e(\rho_7^{(l,j)}, [X^n - 1]_2)$ corresponds with overwhelming probability to the ideal pairing check $V_j^{(l)}(X) Z_{I_j}^{(l)}(X) = 0 \mod Z_{\mathbb{H}}$, i.e. $V_j^{(l)}(X) = 0 \mod Z_{I_j}^{(l)}$. Thus, the values of $S_i \circ M_l$ over G are a witness to the relation $\mathcal{R}_{ACE2}$. As such, an extractor given a valid transcript will extract these values and the adversary loses the game with overwhelming probability.

Zero-knowledge

Theorem 12. *If, in the protocol VWF, the polynomial S_i is blinded by $2t$ extra random blinding factors, then the protocol ACE is zero-knowledge.*

Proof. We now consider that the polynomial $S_i(X)$ is blinded into $S'_i(X) = S(X) + (\beta_1^{(i)} + \sum_{l \leq t} \left[\gamma_{l,1}^{(i)} X^{2l+1} + \gamma_{l,2}^{(i)} X^{2l+2}\right]) Z_{\mathbb{H}}(X)$, with $(\gamma_{l,m}^{(i)})_{l \leq k, m \in [2]}$ randomly generated scalars.

We now describe a simulator that, given an instance of ACE, and the SRS trapdoor x, generates a valid transcript that is indistinguishable from a honest prover's transcript.

Proof for ACE1

The simulator receives an instance $\mathbf{s}_i, w_i^{(l)}$, samples and publishes $\mathbf{sm}_i^{(l)} \leftarrow r_1^{(i,l)}$ from $\mathbb{F}$ and receives a challenge $\alpha_{i,l}$. It then samples $y_1^{(i,l)} \leftarrow r_2^{(i,l)}$ and computes $\rho_1^{(i,l)} \leftarrow (\mathbf{s}_i - \left[y_1^{(i,l)}\right]_1)(x - M_l(\alpha_{i,l}))^{-1}, \rho_2^{(i,l)} \leftarrow (\mathbf{sm}_i^{(l)} - \left[y_1^{(i,l)}\right]_1)(x - \alpha_{i,l})^{-1}, \rho_3^{(i,l)} \leftarrow (\mathbf{sm}_i^{(l)} - \left[w_i^{(l)}\right]_1)(x - \omega^i)^{-1}$, and publishes the result.

It appears that the pairing checks by the verifier all pass, and the output uniquely matches a random valid distribution since

- $\mathbf{sm}_i^{(l)}$ is blinded by $r_1^{(i,l)}$ for the simulator and $\gamma_{l,1}^{(i)}$ for the prover
- $y_1^{(i,l)}$ is blinded by $r_2^{(i,l)}$ for the simulator and $\gamma_{l,2}^{(i)}$ for the prover
- $\rho_1^{(i,l)}, \rho_2^{(i,l)}, \rho_3^{(i,l)}$ uniquely match the pairing checks run by the verifier.

Proof for ACE2

The simulator receives an instance $\mathbf{s}_i, \mathbf{sm}_i^{(l)}$. It then samples $\mathbf{r}^{(l)} \leftarrow \left[r_3^{(l)}\right]_1$, simulates $\rho_6^{(l)}$ a correct execution of EPA (see Theorem 3), from which it extracts $\mathbf{c}^{(l)}, \mathbf{v}_j^{(l)}$ and outputs $\mathbf{rv}^{(i)} \leftarrow r_3^{(l)}(\sum_j \mathbf{v}_j^{(l)}), \rho_4^{(i,l)} \leftarrow (x - \omega^i)^{-1}(\mathbf{r}^{(l)} - \mathbf{sm}_i^{(l)}), \rho_5^{(i,l)} \leftarrow (x - \omega^i)^{-1}(\mathbf{c}^{(l)} - \mathbf{rv}^{(i)}), \rho_7^{(j,l)} \leftarrow Z_j^{(l)}(x)(x^n - 1)^{-1} \left[V_j^{(l)}\right]_1$. One can verify that the pairing checks pass, furthermore:

- $\mathbf{r}^{(l)}$ is blinded by $r_3^{(l)}$ for the simulator and $\epsilon^{(i)}$ for the prover
- $\mathbf{rv}^{(l)}$ uniquely satisfies its pairing check
- $\rho_4^{(i,l)}, \rho_5^{(i,l)}$ uniquely satisfy their KZG opening
- $\rho_6^{(l)}$ is uniformly distributed, as per Theorem 3
- $\rho_7^{(j,l)}$ uniquely satisfies its pairing check

D Example: Aggios-Split

D.1 Casting the vote

In the following example, we demonstrate Aggios-Split vote encoding and aggregation. Let $k = 2$ be the number of candidates, $n = 5$ the number of aggregators, $t = 2$ the corruption threshold, and assume that there are $N = 6$ voters. Each real vote is encoded as a vector of $kt + 1 = 5$ votes over $\{1, 2\}$, such that each candidate receives t decoy votes (in this case, 2 votes) while the actual chosen candidate receives $t + 1$ votes (in this case, 3 votes). The votes, either decoy or legitimate, are randomly shuffled in the vector that the voter commits to.

With our current values, unshuffled votes can be either $[1, 2, 1, 2, 1]$ (vote for 1) or $[1, 2, 1, 2, 2]$ (vote for 2). The voter will then shuffle the vector corresponding to their choice, obtain e.g. $[2, 1, 1, 1, 2]$, and commit to that vector.

The committed vector is shared to all aggregators, and each aggregator receives an opening of the vote delegated to them.
Say that the votes each aggregator receives are:

	Agg 1	Agg 2	Agg 3	Agg 4	Agg 5	Vote for
Voter 1	1	2	1	2	1	1
Voter 2	2	2	1	1	1	1
Voter 3	2	1	2	1	2	2
Voter 4	2	2	2	1	1	2
Voter 5	1	2	2	2	1	2
Voter 6	1	1	2	2	2	2

Each aggregator receives the proof for the vote delegated to them and computes a tally based on the values it received, using an EPA proof:

Aggregator	**Votes for 1**	**Votes for 2**
Agg 1	3	3
Agg 2	2	4
Agg 3	2	4
Agg 4	3	3
Agg 5	4	2

Summing the votes for each candidate, the total of votes (decoy + legitimate) for candidate 1 is 14, and the total votes for candidate 2 is 16.

To compute the final tally, subtract from the initial tally the Nt decoy votes that each candidate received. The final tally is then

$$\left(\sum_{i=1}^{n} \text{tally}(Agg_i)\right) - N \cdot t = (14, 16) - 12 = (2, 4)$$

Namely, candidate 1 received 2 votes and candidate 2 received 4 votes.

D.2 Constructing the polynomial M

Keeping the assumption from Section D.1 that $t = 2, k = 2$ and hence each voter registers with $kt + 1 = 5$ aggregators, assume that voter v_1 registers, in this order, with aggregators a_1, a_5, a_3, a_2, a_4, therefore creating the registration list $L_1 \leftarrow [a_1, a_5, a_3, a_2, a_4]$. The voter then casts a vote with the polynomial $S_1(X)$, representing the vector $W_1 \leftarrow [w_1, w_2, w_3, w_4, w_5]$, i.e. $S_1(0) = w_1, S_1(1) = w_2$ and so on, and publishes a commitment to S_1 to aggregators $a_1, \ldots, a_5$. Furthermore, vote $W_1[j]$ is sent in clear (along with an opening proof of S_1) to aggregator $L_1[j]$, for j in $\{0, \ldots, 4\}$: a_1 recieves w_1, a_5 receives w_2 etc.

Let us now consider aggregator e.g. a_2 and how to build their related polynomial $M_2(X)$. $M_2(\omega^1)$ is defined as the index of a_2 in L_1, assuming that v_1 registered with a_2. In this case, we have $M_2(\omega^1) = 3$ and $S_1(M_2(\omega^i)) = w_4$ since a_2 is at index 3 in L_1.

Similarly, assuming voter e.g. v_7 has the registration list $L_7 \leftarrow [a_2, a_3, a_8, a_{10}, a_1]$, then $M_2(\omega^7) = 0$ since the index of a_2 in L_7 is 0.

Since the registration phase is public, anyone can reconstitute the list of aggregators to whom a voter has subscribed, therefore the polynomial M_2 is public. The same can be applied for all other aggregators: $M_l(\omega^i) = \text{index_of}(a_l, L_i)$, assuming that voter v_i registered with aggregator a_l.

E Better conflict resolution in Aggios

As seen in Algorithm 1, the dispute mechanism in Aggios does not allow to differentiate between a lying voter, wrongly claiming aggregator misbehaviour, and a lying aggregator. In this appendix, we offer a solution relying on the observation that the ACE protocol (Section C.2) is a zero-knowledge proof that the aggregator behaved honestly. Furthermore, one can see Aggios as a specific instance of Aggios-Split with $t = 0$. As such, plugging the ACE protocol on top of Aggios is straightforward.

However, one issue remains, namely the bandwidth of the ACE proof. Since $t = 0$, we only have one aggregator, but according to Section C.2 the transcript still consists of $2N + k + 2$ elements (plus the EPA proof), for N voters and k candidates. As such, publishing the whole ACE transcript on the ledger would ruin the compactness efforts of Aggios. Instead, the aggregator creates a Merkle tree, in which each leaf consists of:

- the commitment to $S_i(X)$, the blinded polynomial of the votes of voter i (which in this scenario only contains a single vote)
- a signature by voter i of the above commitment, for non-repudiation
- the ACE proof elements pertaining to i, namely $\rho_4^{(i)}, \rho_5^{(i)}$

Finally, the aggregator publishes the root of the Merkle tree, along with the other elements of the ACE transcript (namely $\mathbf{r}, \mathbf{rv}, \rho_7^{(j)}$).

Then, in the case where voter i claims incorrect execution of the aggregator, the latter reveals the path from the root of the Merkle tree to the leaf containing the proof elements relative to voter i, allowing everyone to check that the aggregator did act honestly. To avoid pointless disputes which would saturate the ledger, a slashing protocol can be put in place, punishing incorrect dispute claims.