Enforcing Winner-Only Disclosure: Verifiable Tally Hiding for Weighted DAO Governance

Jiayu Li[1] and Gongli Li[1⋆]

Henan Normal University, Xinxiang, Henan, China
2408324056@stu.htu.edu.cn, ligl522@163.com

Abstract. Token-weighted governance can fingerprint individual voters: when weights are unequal and externally known, the exact weighted tally may reveal who voted which way. Publishing only the winning outcome hides the tally from the public, but a system that reconstructs the tally in order to decide still hands it to whoever runs the computation. Publishing only the winner is not the same as computing only the winner. We present a verifiable protocol for weighted DAO votes in which no protocol role ever receives the weighted tally in plaintext. Each ballot proves in zero knowledge that its encrypted contribution is exactly zero or the voter's registered weight. Trustees add accepted contributions and compare the sum with a threshold entirely under encryption using the Cortier–Gaudry–Yang (CGY) conditional-gate toolbox, and decrypt only the final decision. A public verifier checks that the computation consumed exactly the ciphertexts accepted by the governance contract. We prove accepted-result correctness even when all trustees are malicious; restricted-I/O SUC security of the tally against static active corruption of any sub-threshold trustee coalition, which is any four of five in our reference instance; and winner-only disclosure relative to the real sealed board. With 1,024 voters, the encrypted tally takes 157 s and independent replay 115 s, while ballot proving dominates the 28.5-minute end-to-end pipeline. A fresh Arbitrum Sepolia execution exercises the complete public-chain path.

Keywords: Weighted voting · Tally hiding · DAO governance · Threshold cryptography · Secure computation · Public verifiability

1 Introduction

Token-weighted governance makes influence proportional to registered stake. When weights are unequal and externally known, however, the exact weighted tally can itself identify votes. With five participants of weights $1, 2, 4, 8, 16$, a published tally of 26 has the unique decomposition $2 + 8 + 16$, revealing all five choices. The effect is not confined to toy examples: an exact-integer replay of the released B-Privacy corpus [2] recovers choices in 3,112 of 3,844 proposals across 31 DAOs; on affected proposals it identifies 41.53% of ballots and 85.08%

⋆ Corresponding author.

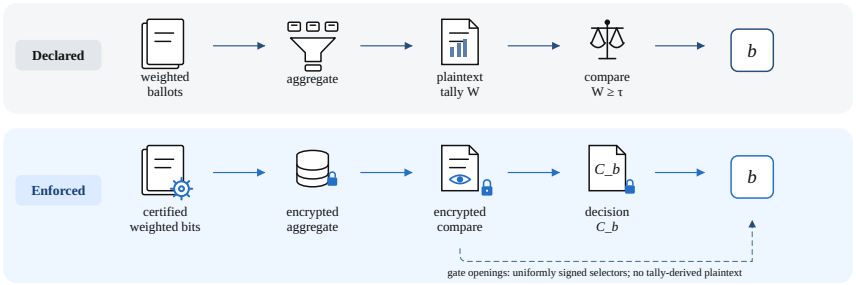


Fig. 1. Declared versus enforced winner-only disclosure. Both release b; only the upper path reconstructs W in plaintext.

of voting weight on average (Appendix J.5). These measurements motivate our disclosure target rather than constitute a security property.

A natural response is to publish only the outcome b rather than the tally W. We call this *declared* winner-only disclosure. It limits public output but still permits the computing party to reconstruct W before publishing b. **Publishing only the winner is not the same as computing only the winner.** Our target is *enforced* winner-only disclosure: aggregation and comparison keep W encrypted, and only the final predicate is opened (Fig. 1).

For voter i with registered integer weight $0 \le w_i < 2^{L_{\text{vote}}}$ and choice $v_i \in \{0, 1\}$, the target decision is

$$W = \sum_{i=1}^{n} v_i w_i, \qquad b = [W \ge \tau], \tag{1}$$

for public threshold τ fixed before voting. No prescribed protocol role receives W, its margin $W - \tau$, a carry, a partial sum, an aggregate bit, or an individual contribution $x_i = v_i w_i$.

The difficulty is twofold. First, weighted contributions are multi-bit integers, so addition and comparison create carries and intermediate bits that are themselves functions of W. Second, a governance result is meaningful only if the encrypted computation consumes *exactly* the ciphertexts accepted by the contract: a re-encrypted copy, reordered board, or filtered subset is a different vote. We therefore combine a registry-bound ballot relation with exact chain/backend binding and CGY's encrypted arithmetic [4].

Contributions. (1) We enforce winner-only disclosure inside the computation: registered weighted contributions are added and compared without opening the tally or its intermediates. (2) A single ballot relation certifies that every accepted contribution is zero or the registered weight, while a public acceptance predicate binds the tally to the chain-accepted ciphertexts coordinate by coordinate. (3) We prove three layered claims: every publicly accepted result is correct even with a fully malicious committee (Theorem 1); the tally inherits CGY's restricted-I/O

Table 1. Comparison with representative tally-hiding systems.

	Ordinos	CGY	Kryvos/PQ	WEAPT	This work
Voter-specific weights	×	×	×	✓	✓
Public verifiability	✓	✓	✓	✓	✓
Public tally hidden	✓	✓	✓	✓	✓
No plaintext-tally recipient	✓	✓	×[b]	×[a]	✓
Flexible result functions	✓	✓	✓	✓	×[c]

[a]WEAPT publishes only the result but a counter reconstructs the tally. [b]Kryvos/PQKryvos allow tallying authorities to learn the aggregate. [c]The implemented rule is one encrypted threshold predicate; Section 9 discusses extensions.

SUC security against any static active sub-threshold trustee coalition (Theorem 2); and, relative to the real sealed board, the tally suffix adds no tally-derived plaintext beyond b (Corollary 1). We implement the complete path and verify every run by independent public replay; at 1,024 voters the encrypted backend takes 156.7 s and the complete pipeline 28.50 minutes, dominated by ballot proving.

2 Background and Related Work

B-Privacy [2] quantifies how public weighted tallies reveal individual choices; we use it as the raw-tally disclosure baseline. Exactness here concerns registered integers, so fractional balances require a pre-poll integerization rule. Voting-Bloc Entropy [19] studies governance concentration, a separate question from tally disclosure.

Threshold ElGamal elections and Helios establish public verifiability [5,1]. Receipt-freeness and coercion resistance are addressed by JCJ and zkVoting [14,22], and by Hyperion and recent threshold/private-board systems [23,24,25]. Our construction is not receipt-free: a voter may still produce transferable evidence of her own vote. Ordinos [15] and CGY [4] evaluate result functions over encrypted ballots; CGY provide the conditional gate, bitwise arithmetic, comparison, and restricted-I/O SUC composition that we instantiate. Kryvos and PQKryvos [12,13] formalize public tally hiding but permit tallying authorities to learn the aggregate. WEAPT [18] supports weighted voting but designates a counter that reconstructs the full tally. Cicada [20] and QV-net [26] remove tallying authorities under different disclosure goals but ultimately make aggregate tallies recoverable. KZH-Fold [21] addresses accountable large-scale accumulation; we claim no sublinear scaling.

Thus CGY supply the secure restricted-I/O computation engine; our work supplies registry-bound weighted admission, exact chain binding, a winner-only weighted-governance disclosure policy, and an end-to-end realization.

3 System and Threat Model

The registry fixes each eligible credential's integer weight and publishes a Poseidon Merkle root. Voters hold an identity secret, registered weight, and choice. A committee of a trustees shares the decryption key. The governance contract verifies ballots and seals the accepted set; a publisher posts the outcome claim; and anyone can replay the public transcript.

Public data include poll parameters, accepted ballots and proofs, trustee public keys, the complete tally transcript, the decision bit, and the on-chain claim. Secret tally-derived data include individual choices and contributions, W, its margin, and all carries and partial sums. The registry's credential-to-weight map is also private, although external token-balance information can independently reveal weights.

The key is Shamir-shared so that any q_{dec} trustees can decrypt and fewer cannot. Our theorems allow any committee size a and $q_{\mathrm{dec}} \leq a$; the reference instance fixes $(a, t, q_{\mathrm{dec}}) = (5, 4, 5)$. Thus all five must contribute, while any coalition of four remains below the decryption threshold. This maximizes privacy within the static model at the cost of availability: any missing trustee can prevent an opening. CGY's gate protocol already requires every trustee to contribute to each gate [4].

The three claims use nested adversaries. For correctness, the adversary controls all trustees, the publisher, any voters, and the network, subject only to the public setup check; it may abort. For tally-computation security, the adversary statically and actively corrupts any $f < q_{\mathrm{dec}}$ trustees and controls the publisher and scheduling within the authenticated common-view channel. For disclosure, it additionally controls malicious voters during board formation. Invalid or conflicting messages abort; missing messages wait. No abort opens a tally intermediate.

Trustees communicate through an authenticated common-view synchronous resource as in CGY [4]; the reference system realizes it with a canonical board service (Appendix J.3). The service is trusted for availability, authentication, atomic sealing, and non-equivocation, but not for secrecy or correctness. The verifier additionally trusts its chain view relative to a trusted deployment configuration, and the registry is assumed well formed.

4 Protocol Overview

Consider a voter Alice registered with weight $8 = 00001000_2$. A yes ballot encrypts those eight bits; a no ballot encrypts eight zeros. A single Groth16 proof states that the ciphertext vector encodes either zero or the weight registered to a credential she holds, and binds the nullifier to that credential and poll. The contract checks the proof, rejects repeated nullifiers, and records the exact ciphertext coordinates. Once sealed, the trustees add the recorded contributions and compare the encrypted aggregate with τ; they decrypt only the resulting decision ciphertext C_b.

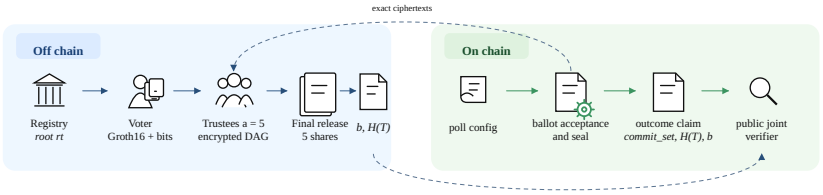


Fig. 2. Protocol dataflow. The chain fixes the accepted ciphertext set; the tally consumes exactly those ciphertexts. Every gate uses all five trustees, and every opening uses five verified shares.

The protocol has five phases: *setup* fixes the registry root, poll parameters, committee key, and trustee verification keys; *admission* verifies one weighted ballot proof and appends its ciphertext vector to the accepted-set commitment; *tally* evaluates the weighted-threshold circuit as CGY conditional gates; *release* rerandomizes and five-share decrypts C_b; and *verification* checks the complete chain/backend acceptance predicate. Figure 2 summarizes the split.

A conditional gate computes an encryption of xy from ciphertexts of x and a bit y. The trustees first encode the selector as $2y - 1 \in \{\pm 1\}$ and each multiply both wires by a secret sign. Only the product of those signs with $2y - 1$ is opened. With one honest trustee the opening is uniform and independent of y. Every gate output is recomputed from public values, so no trustee supplies an unchecked output. No gate opens a weighted sum, carry, contribution, margin, or aggregate bit; the only tally-derived value decrypted by the protocol is the final decision.

CGY supply the gate, proofs, rerandomization, verified decryption, and SUC composition. Our construction adds registry-bound weighted admission, exact binding to the chain-accepted ciphertexts, compilation of the weighted threshold into bit-typed selectors, and the joint verifier.

Algorithm 1. End-to-end winner-only weighted vote.

1 SETUP. Fix registry root rt, poll identifier p, threshold τ, committee public key h, and trustee verification keys.
2 BALLOT. Voter i forms $x_i = v_i w_i$, encrypts its L_{vote} bits, and proves the existing Groth16 weighted-ballot relation.
3 ADMISSION. The contract verifies the proof and nullifier uniqueness, then records the exact ciphertext coordinates.
4 SEAL. After admission closes, seal the canonical accepted board BB.
5 TALLY. Compile $\Gamma_{n,L_{\text{vote}},\tau}$ and evaluate it over exactly BB using CGY conditional gates.
6 TERMINAL. Derive $Q = D - C_*$ and $C_b = E_1 - Q$; authorize only C_b for final release.
7 RELEASE. Rerandomize C_b and decrypt it using five verified shares.
8 CLAIM. Record $(\mathit{commit}_{\text{set}}, H(\mathcal{T}), b)$.
9 VERIFY. Accept iff $\mathsf{Chain} \wedge \mathsf{Snapshot} \wedge \mathsf{Backend} \wedge \mathsf{Binding} \wedge \mathsf{Hash} \wedge \mathsf{Bit} \wedge \mathsf{Claim}$.

5 Weighted Ballot Admission and Exact Binding

All encryption uses exponential ElGamal in the prime-order BabyJubjub subgroup $\mathbb{G} = \langle g \rangle$ with $g = \mathsf{Base8}$ [17]:

$$\mathsf{Enc}_h(m; r) = ([r]g, [m]g + [r]h), \qquad E_m = \mathsf{Enc}_h(m; 0).$$

Voter i writes $x_i = v_i w_i = \sum_{k<L_{\text{vote}}} x_{i,k} 2^k$ and encrypts each bit

$$CT_{i,k} = (R_{i,k}, S_{i,k}) = ([r_{i,k}]g, [r_{i,k}]h + [x_{i,k}]g). \tag{2}$$

The single Groth16 relation proves membership of (s_i, w_i) under the registry root, $0 \le w_i < 2^{L_{\text{vote}}}$, $v_i \in \{0, 1\}$, $x_{i,k} = v_i w_{i,k}$ for every bit, correct encryption randomness, and $\nu_i = \text{Poseidon}_2(s_i, p)$. Hence every accepted contribution is either zero or exactly the registered weight. The instantiated relation has $L_{\text{vote}} = 8$, a depth-10 registry tree, 37 public signals, and one relation/reference string. L_{vote} is an encoding width rather than a governance weight cap; deployments choose it to cover the registered weight range. The implementation uses one relation and one reference string; the released reference string is test-only.

The contract binds the root, poll identifier, and committee key to the poll; rejects repeated nullifiers and identity R components; and records all ciphertext coordinates exactly. Sealing fixes a canonical accepted board BB. A public verifier reconstructs the snapshot from chain events, recomputes its domain-separated accumulator, and authenticates it against a trusted deployment configuration.

Exact backend binding. The tally consumes the recorded ciphertexts themselves, not re-encryptions or encryptions of the same plaintexts. At $n = 8$ the verifier compares all 64 ciphertexts (256 canonical field coordinates) position by position. Reordering, duplication, omission, addition, bit reversal, re-encryption, and rerandomization therefore fail before tally processing. This is the cross-layer step that turns a private encrypted circuit into a governance computation over the vote the contract actually accepted.

The publisher records $(commit_{\text{set}}, H(\mathcal{T}), b)$. The contract checks the sealed-set commitment; the public verifier checks the transcript and result. It accepts only if

$$\mathsf{Accept} = \mathsf{Chain} \wedge \mathsf{Snapshot} \wedge \mathsf{Backend} \wedge \mathsf{Binding} \wedge \mathsf{Hash} \wedge \mathsf{Bit} \wedge \mathsf{Claim}. \tag{3}$$

Here Chain and Snapshot authenticate the accepted set; Backend verifies every gate and final release; Binding checks coordinate equality; and Hash, Bit, and Claim bind the verified bytes, decoded bit, and on-chain claim to the same sealed set. Appendix F gives the exact predicate. An on-chain claim event alone establishes neither transcript validity nor correctness.

6 Winner-Only Tally

Keys and conditional gate. The committee uses a degree-t Shamir polynomial $f(X) = \sum_{\ell=0}^{t} a_\ell X^\ell$ with $sk = f(0)$, $h = [sk]g$, shares $s_j = f(j)$, and public

verification keys $h_j = [s_j]g$. The reference instance fixes $a = 5$, $t = 4$, and $q_{\mathrm{dec}} = 5$. Parties check that the public values are consistent with one degree-t sharing.

For a gate on ciphertexts X of x and Y of bit y, let $Y_0 = [2]Y - E_1$. Trustee j samples $\sigma_j \in \{\pm 1\}$ and fresh randomness and publishes

$$X_j = [\sigma_j]X_{j-1} + \mathsf{Enc}_h(0; r_{X,j}), \quad Y_j = [\sigma_j]Y_{j-1} + \mathsf{Enc}_h(0; r_{Y,j}), \quad e_j = [r_{X,j}]\tilde{h}, \tag{4}$$

with CGY's same-branch disjunctive proof. The auxiliary point is

$$\tilde{h} = H_{\mathbb{G}}(\mathsf{DST}_{\mathrm{aux}} \| \mathsf{sid} \| \mathrm{enc}(g) \| \mathrm{enc}(h)).$$

After all verified masking and rerandomization contributions, the trustees five-share decrypt only the signed selector $y' \in \{\pm 1\}$ and everyone computes

$$Z = [2^{-1}](X + [y']X'). \tag{5}$$

It encrypts xy. Appendix B gives the fixed-generator specialization and Appendix D maps the construction to CGY's algorithms.

Weighted threshold. A balanced tree of CGY ripple adders computes the exact encrypted sum; a CGY subtractor compares it with the public τ. For $n = 2^k$, $L_{\mathrm{agg}} = L_{\mathrm{vote}} + k$ prevents overflow. The circuit uses

$$C_{\mathrm{total}} = (2L_{\mathrm{vote}} + 1)n - 2$$

conditional gates. At $(n, L_{\mathrm{vote}}, L_{\mathrm{agg}}) = (8, 8, 11)$ it uses 134 gates (113 for aggregation and 21 for comparison), depth 72, and maximum ready width 4. Appendix E gives the full arithmetic, typing proof, count, and schedule.

The final borrow Q encrypts $1 - b$. If C_* is the physical output of the last conditional gate and $D = E_{\tau_{L-1}} + R_{L-2} - A_{L-1}$ is fixed before that invocation, the terminal adapter is

$$Q = D - C_*, \qquad C_b = E_1 - Q. \tag{6}$$

Only C_b enters the final rerandomize-then-decrypt release; attempts to release C_* are rejected. All five verified shares are required, and the decoded group element must be $\mathcal{O}$ or g.

7 Security

The claims separate three questions: whether an accepted result can be wrong, what a sub-threshold malicious trustee coalition learns from the encrypted computation, and whether the tally suffix discloses plaintext beyond the decision relative to the real sealed board.

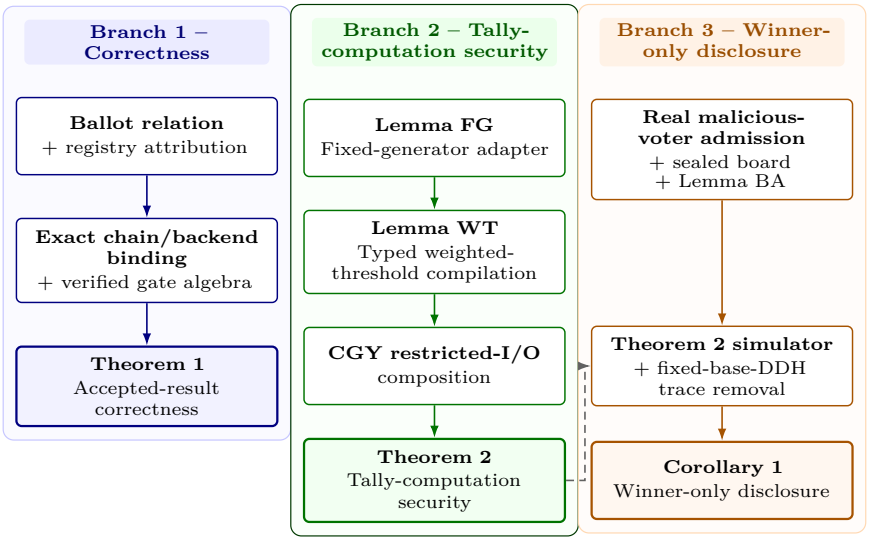


Fig. 3. Security dependency map. Theorem 1 requires neither DDH nor an honest trustee. Theorem 2 is the restricted-I/O tally-computation result. Corollary 1 retains the real admission prefix and is relative to the sealed board. The figure does not assert whole-voting SUC security.

Assumptions. (A1) fixed-base DDH holds in $\mathbb{G} = \langle \mathsf{Base8} \rangle$. (A2) $H_{\mathbb{G}}$ is a programmable group-valued random oracle and Fiat–Shamir uses a programmable 256-bit hash oracle instantiated with SHA-256, interpreted as a big-endian integer reduced modulo q. (A3) Groth16 has ordinary auxiliary-input computational knowledge soundness for the unchanged ballot relation under a reference string generated independently of committee setup. (A4) Poseidon provides leaf, node, and credential binding and target-preimage resistance for unused leaves. (A5) the board accumulator and transcript hashes are collision resistant. (A6) the authenticated common-view broadcast resource is available and verifiers have an authentic chain view relative to the trusted deployment configuration. The released artifact uses a test-only Groth16 reference string.

Two adapter lemmas bridge our deployment to CGY. Lemma FG (Appendix B) specializes CGY's gate proof to the fixed BabyJubjub generator by programming the auxiliary group oracle rather than the generator. Lemma BA (Appendix C) derives witnesses for every ballot on a finite sealed board from ordinary auxiliary-input knowledge soundness, without giving witnesses to any runtime party or simulator. Lemma WT (Appendix E) proves that all compiled selectors encrypt bits and establishes the terminal identity; Lemma EB (Appendix F) formalizes exact chain/backend binding.

Theorem 1 (Accepted-result correctness) Setting. *Fix qualifying parameters* $(n = 2^k, L_{\mathrm{vote}}, L_{\mathrm{agg}}, \tau)$ *(Definition A.5) and an external registry* $\mathcal{R} = ((c_j, w_j))_j$ *with distinct active credentials and* $0 \le w_j < 2^{L_{\mathrm{vote}}}$. *Assume (A2)–*

(A6); in particular, SHA-256 is modeled as a random oracle and challenges are derived from it as in (A2). Let the Groth16 reference string be correctly generated. Let $\mathcal{A}_{\mathrm{corr}}$ *be a PPT adversary that controls all a trustees (including their setup, subject only to the public setup check), the publisher, any subset of voters, and the network, and that makes at most* q_H *oracle queries.*

Guarantee. *Suppose the joint verifier returns* $\mathsf{Accept} = 1$ *with recorded bit* $\hat{b}$ *for a sealed board of* n *accepted ballots. Then there exist an injective attribution* ι *from the accepted set* Acc *to active registry credentials and Boolean votes* (v_i) *such that each accepted ciphertext vector encrypts the bits of* $v_i w_{\iota(i)}$ *and*

$$\hat{b} = \Big[\sum_{i \in \mathrm{Acc}} v_i \, w_{\iota(i)} \geq \tau \Big],$$

except with probability at most

$$\varepsilon_{\mathrm{corr}} \leq n \, \varepsilon_{\mathrm{KS}} + \varepsilon_{\mathrm{reg}} + (q_H + N_{\mathrm{FS}}) \, p_{\mathrm{FS}} + \varepsilon_{\mathrm{hash}}.$$

Terms.

- $\varepsilon_{\mathrm{reg}} = \varepsilon_{\mathrm{leaf}} + \varepsilon_{\mathrm{node}} + \varepsilon_{\mathrm{cred}} + \varepsilon_{\mathrm{unused}}$ *collects the registry-binding advantages.*
- $N_{\mathrm{FS}} = (3a + q_{\mathrm{dec}})C_{\mathrm{total}} + a + q_{\mathrm{dec}}$ *is the number of Fiat–Shamir proofs the verifier checks (2,690 at the reference configuration).*
- $p_{\mathrm{FS}} = 43/2^{256} \approx 2^{-250.6}$ *is the maximum probability of any single challenge value under the implemented challenge map. The SHA-256 output is read as a big-endian integer and reduced modulo* q *without rejection sampling, and* $2^{256} = 42q + r$ *with* $0 < r < q$.
- $\varepsilon_{\mathrm{hash}}$ *bounds collisions in the board accumulator and transcript hash.*

Verifier rejection, transcript refusal, and stalls are aborts and carry no obligation.

The theorem needs neither DDH nor an honest trustee. Its proof follows the ballot from registry attribution through exact board binding, gate-by-gate algebraic correctness, the terminal identity, verified release, and claim binding. Outside the explicitly union-bounded bad events, every accepted gate proof fixes one common sign branch, every rerandomization preserves plaintext, verified shares reconstruct the correct decryption term, and Eq. (5) therefore yields the intended encrypted product. Full details are in Appendix G.

Theorem 2 (Tally-computation security) *Let a trustees hold a consistent degree-t Shamir sharing of the committee key, with* $q_{\mathrm{dec}} = t + 1 \leq a$*, and fix qualifying parameters. Let* Π_{WT} *be the conditional-gate protocol of Lemma FG composed over* $\Gamma_{n,L_{\mathrm{vote}},\tau}$ *and followed by rerandomize-then-decrypt of* C_b*. Assume fixed-base DDH in* $\langle \mathsf{Base8} \rangle$*, and model* $H_{\mathbb{G}}$ *and the Fiat–Shamir hash as programmable random oracles (A2). Consider any PPT tally adversary: it statically and actively corrupts* $f < q_{\mathrm{dec}}$ *trustees and controls the publisher and message scheduling within the authenticated common-view resource. Then, in the* $(\mathcal{F}_{\mathrm{RO}}, \mathcal{F}_{\mathrm{B}})$*-hybrid model and for every restricted environment,* Π_{WT} *SUC-securely computes* $\mathcal{F}_{\mathrm{WT}}^{\mathrm{trace}}$ *against that adversary. For the reference instance* $(a, t, q_{\mathrm{dec}}) = (5, 4, 5)$*, the guarantee covers every coalition of at most four trustees.*

The ideal functionality receives the common public inputs and trustee shares, aborts on inconsistency, decrypts the input bits, evaluates the weighted-threshold circuit, and returns fresh encryptions of every gate output together with b. Restricted environments give all trustees the same public inputs, bit encryptions as selectors, and a consistent sharing. Lemma FG supplies secure gates; Lemma WT supplies the restriction for every invocation; CGY's restricted composition theorem replaces all real gates; and the terminal adapter lets the release simulator know the randomness of C_b without learning any tally intermediate. Appendix H gives the complete reduction.

Simulator interface. The disclosure simulator receives exactly (1) pp and BB, with the public prefix transcript; (2) the continuation state of the corrupted parties, including their shares; (3) its own random-oracle simulation state, programmed from first queries onward; and (4) the bit b. It receives no honest ballot witness, contribution, carry, partial sum, aggregate W, honest share, committee secret, or abort pattern.

Corollary 1 (Winner-only disclosure)
Fix qualifying parameters and (a, t, q_{dec}) with $f < q_{\mathrm{dec}}$. Assume the hypotheses of Theorem 2 together with (A3)–(A5). Let $\mathcal{A}$ be any PPT disclosure adversary: the tally adversary of Theorem 2, additionally controlling malicious voters during the real admission prefix. Then there is a PPT simulator $\mathcal{S}$, with the interface above, such that the joint outputs of the real and ideal-suffix experiments are computationally indistinguishable with advantage at most

$$\varepsilon_{\mathrm{SUC}} + n\,\varepsilon_{\mathrm{KS}} + \mathrm{Adv}^{\text{fb-ddh}}_{\mathbb{G},g} + 1/q,$$

where $\varepsilon_{\mathrm{SUC}}$ is the negligible advantage of Theorem 2. Conditioned on the real sealed board, the tally suffix therefore reveals no additional tally-derived plaintext beyond b. Moreover, $b = [\sum_i v_i w_{\iota(i)} \geq \tau]$ for the attribution of Theorem 1, except with probability $\varepsilon_{\mathrm{reg}} + n\,\varepsilon_{\mathrm{KS}}$.

The proof first replaces the real suffix using Theorem 2, then removes the encrypted trace by a fixed-base DDH hybrid. Lemma BA extracts only inside the reduction, allowing it to compute the true gate plaintexts without knowing the committee key; a random DDH tuple makes the simulated trace uniform and independent of those plaintexts while the terminal adapter preserves C_b. The final simulator itself needs only the sealed board, corrupted state, oracle state, and b. Appendix I gives the hybrids and advantage accounting.

Corollary 1 is intentionally a suffix claim. Confidentiality of honest ballots already present on the board rests on DDH, Groth16 zero knowledge, and threshold-key secrecy. A combined whole-voting privacy theorem would additionally require ballot non-malleability. Giving the simulator b even in executions that abort after learning it is an explicit leakage upper bound.

Table 2. Local scale. Times are wall-clock seconds; total includes proving, admission, backend, release, and replay.

n	gates	proving	backend	replay	total	result/oracle
8	134	10.04	4.05	0.92	22.12	0/0
64	1,086	146.51	15.98	11.68	184.41	0/0
256	4,350	368.45	47.13	32.91	462.13	1/1
1,024	17,406	1,414.93	156.72	114.54	1,709.95	1/1

8 Evaluation

The implementation consists of a Circom/Groth16 ballot circuit, a Solidity poll contract, JavaScript and native Rust implementations of the same tally protocol, and a separate public replay verifier. The ballot circuit has 53,156 nonlinear and 4,984 linear constraints with 37 public signals. The reference backend fixes $(a, t, q_{\mathrm{dec}}) = (5, 4, 5)$ and uses the fixed-generator/hash-to-subgroup adapter of Appendix B. Three test suites pass: 46 JavaScript tests, four threshold/final-release tests, and four native tests. Appendix J gives the code-to-algorithm map, mutation matrix, board-service semantics, benchmark methodology, and public-chain identifiers.

Table 2 gives the essential local results; Appendix J.6 preserves the full timing and transcript-size tables. Each run uses fresh proof-valid ballots, exact accepted input coordinates, the native backend, five-share release, independent public replay in a separate process, and an isolated cleartext oracle consulted only after replay.

At 1,024 voters, ballot proving consumes 1,414.9 s of the 1,709.9 s pipeline (82.7%, or 1.38 s amortized per ballot); the encrypted backend takes 156.7 s and public replay 114.5 s. Because voters prove independently, ballot proving need not lie on the tally critical path. Gate count grows 130× from $n = 8$ to $n = 1{,}024$, while backend time grows 38.7× because the native evaluator runs the ready frontier in parallel. The complete 1,024-voter transcript is 3.55 GB in canonical JSONL and 0.96 GB in the lossless compact codec. The co-located benchmark excludes network latency; communication rounds scale with schedule depth rather than total gate count.

Figure 4 visualizes the composition of the end-to-end runtime in Table 2. It is intended as a visual summary only; these measured configurations do not claim an asymptotic law.

Figure 5 isolates the effect of native ready-frontier scheduling under the final Q5 profile. The serial backend remains near 22 gates/s, while the parallel scheduler reaches about 151 gates/s at frontier widths 16–32. This is a scheduler microbenchmark rather than an end-to-end scaling claim.

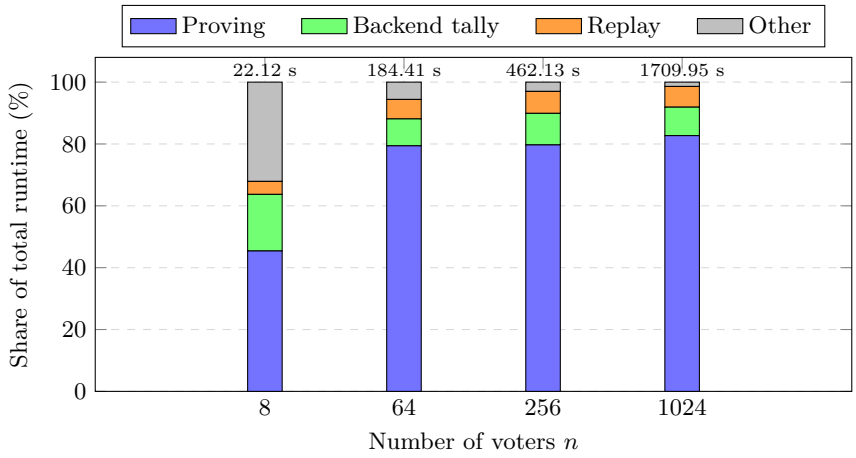


Fig. 4. Runtime composition of the end-to-end pipeline for the four local-scale reference runs. Each bar is normalized to 100% of the total wall-clock time, and the label above each bar shows the corresponding total runtime from Table 2. The figure is a visual summary of the measured runs rather than an asymptotic scaling claim.

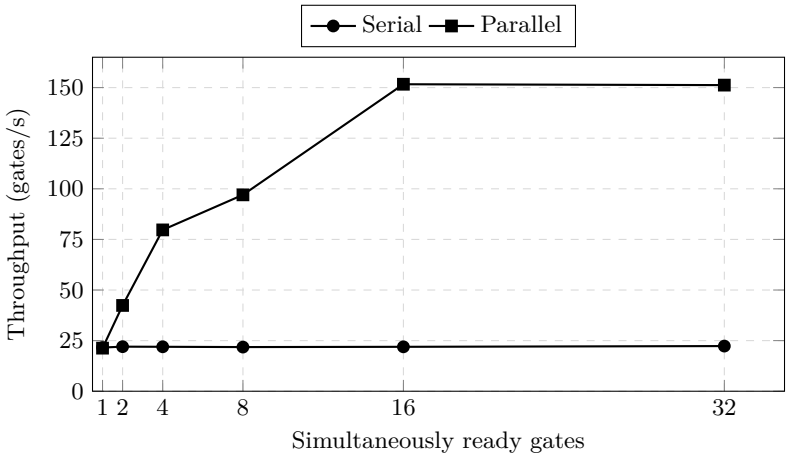


Fig. 5. Ready-frontier microbenchmark under the final Q5 profile. Each point reports median throughput for batches of mutually independent simultaneously ready gates, using the same public verifier for both execution modes. The parallel scheduler benefits substantially from wider ready frontiers, while the serial implementation remains nearly flat. This is a scheduler microbenchmark rather than an end-to-end scaling claim.

A fresh Arbitrum Sepolia execution exercises the public-chain path at the deployed contract's fixed $N = 8$: chain ID 421614; poll ID 910004; $\tau = 500$; 8 accepted and 0 rejected ballots; 134 gates; five-of-five release; 10.360267 s

ballot proving; 41.261974 s deployment plus admission; 3.797074 s backend; 4.226528 s release; 63.223557 s producer total including the claim; 62.143068 s independent public replay; median/p95 ballot gas 558,892/584,778; and 76,652 gas for the outcome claim. Binding, replay, and claim verification all succeed, and the released result matches the isolated oracle (1/1). Appendix J.4 records the public identifiers.

9 Discussion and Limitations

The released bit b is itself tally-derived and, together with public weights, can support inferences such as those studied by B-Privacy. Accepted-ballot count and transaction metadata are public. The tally, margin, carries, partial sums, and individual contributions remain encrypted within the stated model.

The tally theorem covers static active corruption of any sub-threshold trustee coalition: at most four of five in the reference instance. Adaptive corruption is outside the model. Five-of-five decryption makes any trustee able to stall. Correctness survives stalls, and aborts open no intermediate, but fairness is not provided: parties may learn b and then withhold completion. Corollary 1 deliberately gives the simulator b even on such aborting executions.

The registry's credential-to-weight map is a trust boundary, and external token balances may reveal a voter's weight independently. The winner-only tally guarantee does not require weights to be secret; externally visible weights only make disclosure of the exact tally more informative. Receipt-freeness, coercion resistance, ballot independence, and a combined whole-voting privacy theorem are complementary properties outside this tally-layer result. The implemented predicate is $[W_{\text{yes}} \geq \tau]$ for public precommitted τ; majority-of-cast-weight, quorum-plus-majority, and multi-option plurality would require additional encrypted aggregates and comparisons. Theorems and the circuit generator cover $n = 2^k$; other board sizes require padding or an unbalanced tree. The deployed contract fixes $N = 8$, while larger public polls need batched admission.

Key provisioning and the ballot CRS are test-only in the artifact, and the reference board is a concrete realization of the assumed common-view communication resource. A production deployment would replace them with a reviewed DKG [9,31], a multi-party reference-string ceremony, and a Byzantine broadcast or consensus layer exposing the same interface.

10 Conclusion

For weighted governance, the privacy question is not only what is published but what must be learned to compute it. A system that publishes only the winner can still expose the exact weighted tally to whoever computes the result, and public weights can turn that tally into a fingerprint of individual choices.

Our protocol never reconstructs the tally. Registered weights bind to accepted ballots inside zero-knowledge proofs; a public verifier ties the computation to exactly the ciphertexts accepted by the contract; CGY's encrypted computation

performs aggregation and threshold comparison; and a verified rerandomize-then-decrypt release opens only the decision bit. Every accepted result is correct even with a fully malicious committee; the tally inherits CGY's restricted-I/O SUC security below the decryption threshold; and, relative to the real sealed board, the tally adds no tally-derived plaintext beyond b. The implementation runs 1,024-voter tallies in minutes and completes end to end on a public chain. Production deployment would add a reviewed DKG, a production reference-string ceremony, and a Byzantine broadcast layer.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: 17th USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
2. Breckenridge, S., Vilardell, D., Fàbrega, A., Zhao, A., McCorry, P., Juels, A.: B-Privacy: Defining and enforcing privacy in weighted voting. In: 35th USENIX Security Symposium (USENIX Security 26). USENIX Association (2026). https://www.usenix.org/conference/usenixsecurity26/presentation/breckenridge
3. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO '92. LNCS, vol. 740, pp. 89–105. Springer (1993). https://doi.org/10.1007/3-540-48071-4_7
4. Cortier, V., Gaudry, P., Yang, Q.: A toolbox for verifiable tally-hiding e-voting systems. In: ESORICS 2022. LNCS, vol. 13555, pp. 631–652. Springer (2022). https://doi.org/10.1007/978-3-031-17146-8_31. Full version: Cryptology ePrint Archive, Report 2021/491 (Aug. 2022)
5. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: EUROCRYPT '97. LNCS, vol. 1233, pp. 103–118. Springer (1997). https://doi.org/10.1007/3-540-69053-0_9
6. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074
7. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th FOCS, pp. 427–437. IEEE (1987). https://doi.org/10.1109/SFCS.1987.4
8. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1987). https://doi.org/10.1007/3-540-47721-7_12
9. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptology **20**(1), 51–83 (2007). https://doi.org/10.1007/s00145-006-0347-3
10. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: 30th USENIX Security Symposium, pp. 519–535. USENIX Association (2021)
11. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_11
12. Huber, N., Küsters, R., Krips, T., Liedtke, J., Müller, J., Rausch, D., Reisert, P., Vogt, A.: Kryvos: Publicly tally-hiding verifiable e-voting. In: CCS '22, pp. 1443–1457. ACM (2022). https://doi.org/10.1145/3548606.3560701
13. Huber, N., Küsters, R., Reisert, P.: PQKryvos: Post-quantum secure e-voting with flexible ballot formats and public tally-hiding. PoPETs **2026**(4), 1149–1171 (2026). https://doi.org/10.56553/popets-2026-0164
14. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: WPES '05, pp. 61–70. ACM (2005)
15. Küsters, R., Liedtke, J., Müller, J., Rausch, D., Vogt, A.: Ordinos: A verifiable tally-hiding e-voting system. In: IEEE EuroS&P 2020, pp. 216–235. IEEE (2020). https://doi.org/10.1109/EuroSP48549.2020.00022
16. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176
17. WhiteHat, B., Bellés, M., Baylina, J.: ERC-2494: Baby Jubjub elliptic curve. Ethereum Improvement Proposals, ERC-2494 (2020). https://eips.ethereum.org/EIPS/eip-2494

18. Zeng, C., Zhang, Y., Song, Y., He, X., Liu, P., He, Y., Li, H.: Verifiable weighted electronic voting against tally leakage for popular voting methods. In: 34th ICCCN, pp. 1–9. IEEE (2025). https://doi.org/10.1109/ICCCN65249.2025.11134014
19. Fàbrega, A., Zhao, A., Yu, J., Austgen, J., Allen, S., Babel, K., Kelkar, M., Juels, A.: Voting-Bloc Entropy: A new metric for DAO decentralization. In: 34th USENIX Security Symposium, pp. 1299–1318. USENIX Association (2025)
20. Glaeser, N., Seres, I.A., Zhu, M., Bonneau, J.: Cicada: A framework for private non-interactive on-chain auctions and voting. Cryptology ePrint Archive, Paper 2023/1473 (2023)
21. Kadianakis, G., Zapico, A., Hafezi, H., Bünz, B.: KZH-Fold: Accountable voting from sublinear accumulation. In: CCS '25, pp. 409–422. ACM (2025). https://doi.org/10.1145/3719027.3744796
22. Park, S., Choi, J., Kim, J., Oh, H.: zkVoting: Zero-knowledge proof based coercion-resistant and E2E verifiable e-voting system. Cryptology ePrint Archive, Paper 2024/1003 (2024)
23. Damodaran, A., Rastikian, S., Rønne, P.B., Ryan, P.Y.A.: Hyperion: Transparent end-to-end verifiable voting with coercion mitigation. In: ESORICS 2025. LNCS, vol. 16054, pp. 124–143. Springer (2026). https://doi.org/10.1007/978-3-032-07891-9_7
24. Doan, T.V.T., Pereira, O., Peters, T.: Threshold receipt-free voting with server-side vote validation. In: E-Vote-ID 2025. LNCS, vol. 16028, pp. 55–72. Springer (2026). https://doi.org/10.1007/978-3-032-05036-6_4
25. Farzaliyev, V., Willemson, J.: End-to-end verifiable internet voting with partially private bulletin boards. In: E-Vote-ID 2025. LNCS, vol. 16028, pp. 73–89. Springer (2026). https://doi.org/10.1007/978-3-032-05036-6_5
26. Zhou, Z., Zhang, Z., Hao, F., Zheng, B., Masyhur, Z.: QV-net: Decentralized self-tallying quadratic voting with maximal ballot secrecy. In: CCS '25, pp. 453–467. ACM (2025). https://doi.org/10.1145/3719027.3744810
27. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE (2001)
28. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: CRYPTO 2015. LNCS, vol. 9216. Springer (2015)
29. Nielsen, J.B.: On Protocol Security in the Cryptographic Model. Ph.D. thesis, University of Aarhus (2003)
30. Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: ASIACRYPT 2004. LNCS, vol. 3329. Springer (2004)
31. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: EUROCRYPT '91. LNCS, vol. 547. Springer (1991)
32. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat–Shamir heuristic and applications to Helios. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer (2012)
33. Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R.S., Wood, C.A.: Hashing to elliptic curves. RFC 9380, IRTF (2023)

A Notation, Board, and Admission Definitions

Notation.

– λ is the security parameter.

- $\mathbb{G} = \langle g \rangle$, with $g = \mathsf{Base8}$, is the prime-order BabyJubjub subgroup of order q and identity $\mathcal{O}$, written additively.
  - Scalars live in $\mathbb{Z}_q$.
  - Point coordinates live in the base field of prime order $p_{\mathbb{F}}$.
- Encryption:
  - $\mathsf{Enc}_h(m; r) = ([r]g, [m]g + [r]h)$, and $E_m = \mathsf{Enc}_h(m; 0)$;
  - for small m, $\mathsf{Dec}_{sk}(R, S) = \log_g(S - [sk]R)$.
- $[P]$ is the Iverson bracket.
- $\ell_{S,j} = \prod_{\ell \in S \setminus \{j\}} \ell/(\ell - j) \bmod q$ are the Lagrange coefficients at zero.
- $[2^{-1}]$ denotes multiplication by the inverse of 2 modulo q.
- $p_{\mathrm{FS}} = 43/2^{256}$ is the maximum probability of any single Fiat–Shamir challenge value under the implemented challenge map.

Definition A.1 (Ballot relation). A statement and witness are

$$s = (rt, p, h, \nu, (CT_k)_{k<L_{\mathrm{vote}}}), \qquad \omega = (sec, w, v, (sib_\ell, d_\ell)_{\ell<\mathsf{depth}}, (r_k)_{k<L_{\mathrm{vote}}}).$$

We have $R_{\mathrm{ballot}}(s, \omega) = 1$ iff all of the following hold:

1. $w = \sum_k w_k 2^k$ with every $w_k \in \{0, 1\}$;
2. the leaf $\mathrm{Poseidon}_2(\mathrm{Poseidon}_1(sec), w)$ authenticates to rt along siblings sib_ℓ with Boolean directions d_ℓ;
3. $v \in \{0, 1\}$;
4. $x_k = v\, w_k$ for every k;
5. $CT_k = ([r_k]g, [r_k]h + [x_k]g)$, with r_k read modulo q;
6. $\nu = \mathrm{Poseidon}_2(sec, p)$.

Definition A.2 (Encrypted-bit language). $\mathcal{L}_{\mathrm{bits}}(h) = \{([r]g, [r]h + [x]g) : r \in \mathbb{Z}_q, x \in \{0, 1\}\}$. Accepted ballot ciphertexts additionally satisfy $R \neq \mathcal{O}$. The trivial encryptions E_0 and E_1 belong to $\mathcal{L}_{\mathrm{bits}}(h)$.

Definition A.3 (Accepted and sealed board). An accepted board $\mathsf{BB} = ((\nu_i, \mathbf{CT}_i, \pi_i))_{i=1}^n$ lists accepted ballots in canonical position order, together with its accumulator value $commit_{\mathrm{set}}$. The board is sealed once the contract's sealing transaction is final. All later processing uses BB unchanged.

Definition A.4 (Setup experiment and admissible real prefix). The setup experiment proceeds as follows:

1. The static corrupted set C_T is fixed.
2. The ideal setup samples a uniform degree-t Shamir sharing $(h, (h_j), (s_j))$.
3. The Groth16 reference string is generated honestly and independently.
4. Registry and poll parameters are fixed.

An *admissible real-prefix process* is any PPT process that receives pp and the corrupted shares and then:

- runs honest voters' ballot algorithms, with the real prover on real witnesses;

– runs arbitrary malicious-voter algorithms, interleaving submissions adaptively after observing public chain state;
– stops when the board is sealed, outputting $(\mathsf{pp}, \mathsf{BB}, \mathsf{st}_{\mathcal{A}})$.

The process has access to neither a Groth16 simulation oracle nor an honest decryption oracle. In the correctness experiment (Theorem 1), the committee setup is instead part of the adversary's computation, subject to the public setup check.

Definition A.5 (Qualifying parameters). A tuple $(n, L_{\text{vote}}, L_{\text{agg}}, \tau)$ is *qualifying* if:

– $n = 2^k$ with $k \geq 1$, and $L_{\text{vote}} \geq 1$;
– $L_{\text{agg}} = L_{\text{vote}} + k$;
– $0 \leq \tau < 2^{L_{\text{agg}}}$, with τ public and fixed before voting;
– n and L_{vote} are polynomial in λ.

The circuit $\Gamma_{n,L_{\text{vote}},\tau}$ is defined in Appendix E. Gate identifiers and session identifiers (protocol, setup hash, execution, gate, phase) are serialized injectively with domain separation.

Table 5. Reference configuration.

Parameter	Value
$(n, L_{\text{vote}}, L_{\text{agg}})$	$(8, 8, 11)$
Trustees / degree / decryption threshold	5 / 4 / 5
Gates (aggregation + comparison)	134 (113 + 21)
Depth / maximum ready width	72 / 4
Ballot public signals / implemented relations	37 / 1
Bound ciphertexts / coordinates	64 / 256
Fiat–Shamir proofs per complete transcript N_{FS}	2,690
Challenge map / maximum point mass p_{FS}	SHA-256, big-endian, reduced mod q / $43/2^{256}$
Ballot circuit (nonlinear / linear constraints)	53,156 / 4,984
Outcome-claim transaction (Arbitrum Sepolia)	76,652 gas

Table 6. Designed leakage beyond the public transcript.

Party	v_i	w_i	W	b	Scope
Public observer	no*	no*	no	yes	Cor. 1; ballot-layer confidentiality

Party	v_i	w_i	W	b	Scope
Any static coalition of at most four trustees	no*	no*	no	yes	Cor. 1, $f < q_{\mathrm{dec}}$
All five trustees	partial†	partial†	yes	yes	beyond the privacy claims
Registry	no	yes	no	yes	trust boundary
Publisher / chain	no*	no*	no	yes	no decryption capability

* The tally layer need not reveal the credential-to-weight binding; token balances or other external information may still reveal a voter's weight. Confidentiality of honest ballots on the board rests on DDH, Groth16 zero knowledge, and threshold-key secrecy. † Five trustees recover $x_i = v_i w_i$: if $x_i > 0$, then $v_i = 1$ and $w_i = x_i$; if $x_i = 0$, neither value is determined.

B Lemma FG: Fixed-Generator Specialization

Lemma B.1 (Lemma FG).
Groups and assumption. Let $(\mathbb{G}_\lambda, q_\lambda, g_\lambda)$ be a family of odd-prime-order cyclic groups, each with a fixed public non-identity generator. Assume fixed-base DDH,

$$(g, [\alpha]g, [\beta]g, [\alpha\beta]g) \approx_c (g, [\alpha]g, [\beta]g, [\gamma]g)$$

for independent uniform $\alpha, \beta, \gamma \in \mathbb{Z}_q$, against the same adversary class used for knowledge soundness.
Protocol. Let Π^g_{CSZ} be CGY's conditional-gate protocol [4, Alg. 7, 9, 10, 63, 65, 67] with its original auxiliary relation, proofs, rerandomization, decryption, and communication resources, modified in two ways: 1. The generator query and check of [4, Alg. 67, ll. 3–5] are replaced by a check that the input generator equals the fixed g. 2. The auxiliary element is $\tilde{h} = H_{\mathbb{G}}(\mathsf{DST}_{\mathrm{aux}} \| \mathsf{sid} \| \mathrm{enc}(g) \| \mathrm{enc}(h))$ for a programmable group-valued oracle $H_{\mathbb{G}}$. The scalar Fiat–Shamir oracles are retained.
Conclusion. [4, Lemmas I.1, I.2, I.3] and [4, Thm. I.4] hold for Π^g_{CSZ} under the same hypotheses (restricted environments, at least one honest participant, the $(\mathcal{F}_{\mathrm{RO}}, \mathcal{F}_{\mathrm{B}})$-hybrid model), with DDH replaced by fixed-base DDH for g.

Proof. The proof proceeds in five steps.

Step 1: algebra, Lemma I.1, and Lemma I.2. For a fixed non-identity g, the maps $r \mapsto [r]g$ and $r \mapsto \mathsf{Enc}_h(m; r)$ have exactly the properties CGY use: translating a uniform exponent yields a uniform exponent. Neither of the following simulators ever chooses a base:

- the rerandomization simulator of [4, Lemma I.1] adjusts one honest contribution and programs its proof;
- the decryption simulator of [4, Lemma I.2] chooses r, forms $([r]g, [y]g + [r]h)$, and computes honest shares $[r]h_j$.

Both lemmas therefore apply verbatim.

Step 2: gate simulator. Use the simulator of [4, Lemma I.3], but keep g fixed. At the first query $H_{\mathbb{G}}(\mathsf{DST}_{\mathrm{aux}} \| \mathsf{sid} \| \mathrm{enc}(g) \| \mathrm{enc}(h))$, sample $\kappa \leftarrow \mathbb{Z}_q^*$, answer $\tilde{h} = [\kappa^{-1}]g$, and store κ. Programming happens when a query is first answered, never by overwriting a returned value.

Consider a valid corrupt transition with first components U_{j-1} and U_j. Then $U_j = [\sigma_j]U_{j-1} + [r_{X,j}]g$ and $e_j = [r_{X,j}]\tilde{h}$, hence

$$U_j - [\kappa]e_j = [\sigma_j]U_{j-1}.$$

Extraction is needed only for transitions after the last honest trustee, and their predecessors have passed the non-identity check of [4, Alg. 10]. The two candidates $\pm U_{j-1}$ are therefore distinct, so $\sigma_j \in \{\pm 1\}$ is extracted without any plaintext, randomness, or committee secret. All other simulator operations are CGY's.

Step 3: the DDH hybrid. CGY compare their real-distribution simulator with the efficient simulator at the last honest transition, using a DDH reduction that sets the generator to the challenge's first component. Instead, take a fixed-base challenge $(g, A = [\alpha]g, B = [\beta]g, T)$ and proceed as follows:

- on the relevant auxiliary query, answer $\tilde{h} = [u]A$ for $u \leftarrow \mathbb{Z}_q^*$;
- for the incoming X-wire (U, V) and a chosen sign σ, set $U' = [\sigma]U + B$, $V' = [\sigma]V + [sk]B$, and $e = [u]T$;
- handle the Y-wire and the proofs as in CGY.

The reduction knows sk exactly as CGY's does: it runs the environment, which writes consistent shares on all input tapes [4, proof of Lemma I.3]. Later corrupt signs are recovered by decrypting Y-wires with sk.

If $T = [\alpha\beta]g$, then $(U', V') = [\sigma](U, V) + \mathsf{Enc}_h(0; \beta)$ and $e = [\beta]\tilde{h}$, which is the correct auxiliary relation. If T is random, e is uniform and independent of the other challenge-dependent values, which is CGY's random-auxiliary distribution. CGY's indistinguishability argument then goes through unchanged, and no ciphertext or proof statement changes its generator.

Step 4: many sessions. Distinct sessions use distinct oracle prefixes, as CGY require. A standard hybrid replaces one session's last-honest transition at a time, with polynomial loss. Every challenge shares the first component g, so there is no need to randomize the generator across sessions.

Step 5: composition. With the fixed-base versions of I.1–I.3, [4, Thm. H.3] yields the fixed-base Theorem I.4. The implementation rejects an identity auxiliary point, that is, it samples $\tilde{h}$ from $\mathbb{G} \setminus \{\mathcal{O}\}$. Coupling this with CGY's full-group sampling costs at most (number of auxiliary samples)$/q$. □

Remark B.1 (Scope). Lemma FG assumes DDH for the specific generator Base8. It does not derive that assumption from DDH averaged over random

generators, and it does not condition CGY's random generator on the event $g = \mathsf{Base8}$.

Remark B.2 (Why $\tilde{h}$ is a hash-to-subgroup output). Suppose $\tilde{h} = [k]g$ for a publicly computable k, for example $k = H(\text{context})$. Then any observer can compute $U_j - [k^{-1}]e_j = [\sigma_j]U_{j-1}$ and recover every mask sign on a non-identity chain. Combined with the opening $y' = (\prod_j \sigma_j)(2y - 1)$, this reveals the selector y. Public multiples of h fail in the same way. A hash-to-subgroup map must therefore not reveal these discrete logarithms [33].

B.3 Reference instantiation of $H_{\mathbb{G}}$. For successive counters, the routine does the following:

1. Expand the public context with SHAKE256 into a candidate canonical coordinate y and a sign bit; reject an out-of-range y.
2. Solve $x^2 = (1 - y^2)/(168700 - 168696\, y^2) \bmod p_{\mathbb{F}}$; reject a zero denominator or a non-square.
3. Choose the root indicated by the sign bit, rejecting a non-canonical sign for $x = 0$.
4. Accept only a non-identity point P with $[q]P = \mathcal{O}$, and return the first accepted point.

The proof models $H_{\mathbb{G}}$ as a programmable group-valued random oracle. The reference implementation instantiates this deterministic interface using SHAKE256 and the stated rejection/decode procedure. Because the routine decodes points and never multiplies $\mathsf{Base8}$ or h by a public scalar, it exposes no discrete logarithm of $\tilde{h}$ relative to $\mathsf{Base8}$ or the committee key. With ideal independent outputs and injective encodings, the first accepted point is uniform on $\mathbb{G} \setminus \{\mathcal{O}\}$. The routine processes only public setup and session data, so variable-time rejection is acceptable, and its encodings are domain-separated from proof challenges.

C Lemma BA and Registry Attribution

Assumption KS (ordinary auxiliary-input computational knowledge soundness). For the fixed relation R_{ballot} and every PPT prover program P^*, there is a PPT extractor $\mathcal{X}_{P^*}$ with the following property. For every benign auxiliary-input distribution $\mathcal{Z}$ sampled independently of the reference string,

$$\Pr\left[\begin{array}{l} crs \leftarrow \mathsf{Setup};\ z \leftarrow \mathcal{Z}; \\ (s, \pi) \leftarrow P^*(crs, z; \varrho); \\ \omega \leftarrow \mathcal{X}_{P^*}(crs, z, \varrho) : \\ \mathsf{Vfy}(crs, s, \pi) = 1 \ \wedge \ R_{\text{ballot}}(s, \omega) = 0 \end{array}\right] \leq \varepsilon_{\text{KS}}(\lambda).$$

Statements may be chosen after the prover sees the reference string. This is the knowledge-soundness notion of [11, Sect. 2.2], not simulation extractability. The auxiliary input consists of the independently generated committee setup and ordinary application inputs. It excludes any Groth16 trapdoor, simulated-proof oracle, or trapdoor-correlated advice.

Lemma C.1 (Lemma BA). Let P be an admissible real-prefix process (Definition A.4) that yields at most $N(\lambda)$ accepted ballots, with N polynomial. Under Assumption KS, there is a PPT procedure $\mathcal{E}_P$ with the following behavior. Given the saved state of the prefix execution, it outputs witnesses ω_j with $R_{\text{ballot}}(s_j, \omega_j) = 1$ for every accepted statement s_j of the sealed board, except with probability at most $N\varepsilon_{\text{KS}}$. $\mathcal{E}_P$ works on copies: the original board and continuation state are unchanged, and no witness reaches a runtime party or a simulator.

Proof. Define one ordinary prover program A_P. On input (crs, z), it runs P until sealing, and only then samples $J \leftarrow \{1, \ldots, N\}$. It outputs the accepted statement and proof at position J, or $\perp$ if that position is unused.

Every honest proof inside this execution was produced by the real prover routine on its real witness, and that routine is part of A_P. Interleaving introduces no proof oracle. Knowledge soundness therefore applies to A_P as a whole.

Let $\mathcal{X} = \mathcal{X}_{A_P}$ and save the prefix state. For each position j, run $\mathcal{X}$ on a copy with the same (crs, z) and prefix coins, and with the selector fixed to $J = j$. Because J is sampled after the prefix, every copy refers to the *same* board.

Let p_j be the probability that position j is accepted and extraction at j fails. Knowledge soundness for A_P gives $\frac{1}{N}\sum_j p_j \le \varepsilon_{\text{KS}}$, so a union bound gives

$$\Pr[\exists \text{ accepted position without witness}] \le \sum_j p_j \le N\varepsilon_{\text{KS}}.$$

The reduction continues the saved original execution, so no cryptographic non-malleability is needed to preserve state. The committee key is part of z, so the extractor never receives its generation coins. The lemma uses the strict polynomial-time form of knowledge soundness; an expected-time extractor would need its own running-time convention. □

Lemma BA does not assert three things: that each malicious voter individually knew its witness, that extraction works from an opaque proof-supplying service, or that any runtime party learns plaintexts.

Corollary C.2 (Registry attribution and exact contributions). Assume (A4) and condition on the success of Lemma BA. Then, except with probability $\varepsilon_{\text{reg}} = \varepsilon_{\text{leaf}} + \varepsilon_{\text{node}} + \varepsilon_{\text{cred}} + \varepsilon_{\text{unused}}$, there is an injective map ι from accepted positions to active registry credentials such that for all accepted i and all k, $CT_{i,k}$ encrypts $x_{i,k}$ with

$$x_i = \sum_k 2^k x_{i,k} = v_i w_{\iota(i)}, \qquad v_i \in \{0, 1\}.$$

In particular, $x_i \in \{0, w_{\iota(i)}\}$.

Proof. The extracted witness authenticates the leaf

$$\text{Poseidon}_2(\text{Poseidon}_1(sec), w)$$

to rt at some index. Compare this path with the actual registry path

$$m_0, \ldots, m_D = rt$$

at the same index. Boolean directions, the canonical index, and the shared untagged node function make the two paths comparable. Two cases arise:

- If the leaves differ, we obtain a credential-identity or leaf-binding violation, or, if the extracted leaf occupies an unused slot, a target preimage.
- Otherwise the paths differ higher up, and the first differing child pair converges to the same parent, giving an internal-node collision. The first converging parent is unique, so the bound carries no depth factor.

Outside these events, the leaf is that of an active credential j with weight w_j, and we set $\iota(i) = j$. Conditions (4)–(5) of Definition A.1 then fix the encrypted bits $x_{i,k} = v_i w_{j,k}$.

For injectivity, suppose $\iota(i) = \iota(i')$ for distinct accepted positions. Credential binding forces the same *sec*. Since the poll is fixed and the nullifier has no free input, $\nu_i = \nu_{i'}$, and the contract would have rejected the later ballot. A collision between different credentials can deny a ballot, but it cannot produce two accepted uses of one credential. □

D CGY Theorem and Algorithm Mapping

Table 7. Correspondence between CGY [4] (full version) and our instantiation.

Component	In CGY [4]	Our instantiation	Premise
Group and encryption	App. A.1	BabyJubjub $\langle$Base8$\rangle$, exponential ElGamal, point checks	fixed-base DDH (Lemma FG)
Threshold keys and decryption	App. A.2, Alg. 4	degree-t Shamir sharing, $h_j = [s_j]g$; reference $(5, 4, 5)$; test-only dealer	consistent sharing (App. I.1 condition)
Conditional gate	Alg. 7 (protocol), Alg. 67 (formal)	masking by all a trustees; $\tilde{h}$ from $H_{\mathbb{G}}$; $Y_0 = [2]Y - E_1$	Lemma FG
Gate proof	Alg. 9, 10	same-branch six-equation disjunction; identity-X rejection	programmable RO (A2)
Rerandomization	Alg. 8, 63; $\mathcal{F}_{\text{rerand}}$ Alg. 64; Lemma I.1	zero-encryption contributions via synchronous broadcast on the canonical board	$\mathcal{F}_{\text{SB}}$, ≥ 1 honest

Component	In CGY [4]	Our instantiation	Premise
Verified decryption	Alg. 65; $\mathcal{F}_{\mathrm{Dec}}$ Alg. 66; Lemma I.2	Chaum–Pedersen shares; $q_{\mathrm{dec}} = t + 1$ distinct shares; Lagrange at zero; rerandomize-then-decrypt	$\mathcal{F}_{\mathrm{rerand}}$, $\mathcal{F}_{\mathrm{RO}}$
Gate security	Lemma I.3, Thm. I.4; $\mathcal{F}_{\mathrm{CSZ}}$ Alg. 62	fixed-base versions	≥ 1 honest, restricted I/O
Framework	App. H.1–H.3; Def. H.1–H.2; Thm. H.3	SUC [28], static malicious, authenticated router; at most $t < a$ corruptions	—
Restricted I/O	App. H.4; [29, Thm. 3.5]	common inputs; selectors in $\mathcal{L}_{\mathrm{bits}}$; consistent sharing	Lemma WT(b); Groth16 soundness
Arithmetic	Alg. 21 (carry retained, as for Alg. 40), Alg. 23	balanced AddBits tree; SubLT against trivial τ bits	Lemma WT
Release	Thm. J.1	only C_b decrypted; terminal adapter	Lemma WT(d)
Trace removal	App. J.2, Game 2	fixed-base DDH embedding with Lemma BA extraction	Corollary 1
Voting privacy	Thm. 7.2; App. J.2, Game 3	not used	—

E Weighted-Threshold Compilation

Building blocks. We write $\mathsf{CSZ}(X, Y)$ for a conditional-gate invocation.

AddBits on two m-bit inputs (X_i) and (Y_i) [4, Alg. 21, final carry retained] computes

$$R_0 = \mathsf{CSZ}(X_0, Y_0), \qquad Z_0 = X_0 + Y_0 - [2]R_0,$$

and, for $i = 1, \ldots, m-1$,

$$\begin{aligned} A_i &= X_i + Y_i - [2]\,\mathsf{CSZ}(X_i, Y_i), \\ Z_i &= A_i + R_{i-1} - [2]\,\mathsf{CSZ}(A_i, R_{i-1}), \\ R_i &= [2^{-1}](X_i + Y_i + R_{i-1} - Z_i). \end{aligned}$$

It sets $Z_m = R_{m-1}$ and outputs $(Z_0, \ldots, Z_m)$.

SubLT [4, Alg. 23] computes

$$P_0 = \mathsf{CSZ}(X_0, Y_0), \quad Z_0 = X_0 + Y_0 - [2]P_0, \quad R_0 = Y_0 - P_0,$$

and, for $k = 1, \ldots, m-1$,

$$\begin{aligned} A_k &= \mathsf{CSZ}(Y_k, R_{k-1}), \\ B_k &= Y_k + R_{k-1} - [2]A_k, \\ C_k &= \mathsf{CSZ}(X_k, B_k), \\ Z_k &= X_k + B_k - [2]C_k, \\ R_k &= Y_k + R_{k-1} - A_k - C_k. \end{aligned}$$

It outputs R_{m-1}. The difference bits Z_k are computed but never opened.

The circuit. The circuit $\Gamma_{n,L_{\text{vote}},\tau}$ is built in four steps:

1. Level 0 holds the operands $\mathbf{X}_i^{(0)} = (CT_{i,0}, \ldots, CT_{i,L_{\text{vote}}-1})$.
2. For $j = 1, \ldots, k$, level j computes $\mathbf{X}_i^{(j)} = \mathsf{AddBits}(\mathbf{X}_{2i-1}^{(j-1)}, \mathbf{X}_{2i}^{(j-1)})$.
3. The comparison computes $Q = \mathsf{SubLT}(\mathbf{X}_1^{(k)}, (E_{\tau_0}, \ldots, E_{\tau_{L_{\text{agg}}-1}}))$.
4. The decision ciphertext is $C_b = E_1 - Q$.

Lemma E.1 (Lemma WT). For qualifying parameters, with $L = L_{\text{vote}}$ and $L_{\text{agg}} = L + k$:

(a) *Structure.* Γ consists of exactly $C_{\text{total}} = (2L+1)n - 2$ invocations of CSZ interleaved with public affine maps over $\mathbb{Z}_q$. Every wire is assigned once, there is a single decision wire C_b, and every gate input is a deterministic public function of the input ciphertexts and earlier gate outputs.

(b) *Typing.* Suppose all inputs $CT_{i,k} \in \mathcal{L}_{\text{bits}}(h)$ and every CSZ output encrypts the product of its inputs' plaintexts. Then every wire encrypts a value in $\{0,1\}$; in particular, every selector input encrypts a bit.

(c) *Correctness.* Under the hypothesis of (b), C_b encrypts $b = [\sum_i x_i \geq \tau]$, where $x_i = \sum_k 2^k x_{i,k}$.

(d) *Terminal form.* $C_b = E_1 - D + C_*$, where C_* is the output of the last invocation and $D = E_{\tau_{L_{\text{agg}}-1}} + R_{L_{\text{agg}}-2} - A_{L_{\text{agg}}-1}$ is fixed before that invocation.

(e) *Schedule.* Evaluating each adder in bit order, and starting it once both operands are complete, gives depth $\Delta_{\text{total}} = (k+1)(2L+k-1)$ and maximum ready width $n/2$.

Proof.

Addition. For bits, $u \oplus v = u + v - 2uv$. By induction, $z_i = x_i \oplus y_i \oplus r_{i-1}$. Moreover, $x_i + y_i + r_{i-1} - z_i \in \{0, 2\}$, so $r_i = \lfloor (x_i + y_i + r_{i-1})/2 \rfloor$ is the carry, and $[2^{-1}]$ is exact on $\{0, 2\}$. Retaining $z_m = r_{m-1}$ makes the output the full $(m+1)$-bit sum. The adder uses $1 + 2(m-1) = 2m - 1$ gates.

Aggregation. Level j adds $(L+j-1)$-bit operands into $(L+j)$-bit exact sums. After k levels the result is $\sum_i x_i$ in L_{agg} bits, and $\sum_i x_i \leq n(2^L - 1) < 2^{L_{\text{agg}}}$. The aggregation gate count is

$$\sum_{j=0}^{k-1} \frac{n}{2^{j+1}}(2L + 2j - 1) = (2L-1)(n-1) + 2(n-k-1).$$

Comparison. Let $x^{(k)} = x \bmod 2^{k+1}$; we claim $r_k = [x^{(k)} < y^{(k)}]$. For the base case, $r_0 = y_0 - x_0 y_0 = [x_0 < y_0]$. For the step, write $r = r_{k-1}$, so that $r_k = y_k + r - y_k r - x_k(y_k \oplus r)$. The four cases are:

- $(x_k, y_k) = (0, 1)$ gives $r_k = 1$;
- $(1, 0)$ gives $r_k = 0$;
- $(0, 0)$ and $(1, 1)$ give $r_k = r$.

This is exactly the recursion for $[x^{(k)} < y^{(k)}]$. With $y = \tau$, the final borrow encrypts $[W < \tau] = 1 - b$, so C_b encrypts b. The comparison uses $2L_{\mathrm{agg}} - 1$ gates, and the total is $(2L-1)(n-1) + 2(n-k-1) + 2(L+k) - 1 = (2L+1)n - 2$.

Typing. By the invariants, every plaintext is a bit. The selectors are Y_0, Y_i, R_{i-1} in the adders and E_{τ_0}, R_{k-1}, B_k in the comparison. Trivial encryptions lie in $\mathcal{L}_{\mathrm{bits}}(h)$.

Terminal. The last comparison step reads $Q = E_{\tau_{L_{\mathrm{agg}}-1}} + R_{L_{\mathrm{agg}}-2} - A_{L_{\mathrm{agg}}-1} - C_* = D - C_*$. Here $A_{L_{\mathrm{agg}}-1}$ is the penultimate invocation and C_* the last. Hence $C_b = E_1 - Q = E_1 - D + C_*$.

Common inputs. All maps are public and deterministic, so every trustee derives identical inputs from BB and the verified outputs.

Schedule. An adder of width m contributes $2m - 1$ layers. The critical path passes one adder per level ($j = 0, \ldots, k-1$, widths $L+j$) and then the comparison (width $L + k$). This gives $\sum_{j=0}^{k}(2(L+j)-1) = (k+1)(2L+k-1)$ layers. At level 0, all $n/2$ adders are ready concurrently. □

F Exact Chain/Backend Binding

The seven conjuncts of Eq. (3) are defined as follows.

- Chain. The chain identifier, contract addresses, runtime bytecode hashes, verifying key, poll parameters (rt, p, τ, h), and setup hash all match the trusted deployment configuration, and the sealing transaction is final.
- Snapshot. Acceptance events, ordered by block, transaction index, and log index, carry consecutive positions $1..n$. The recomputed domain-separated accumulator over (position, nullifier, ciphertext vector) equals the sealed $commit_{\mathrm{set}}$.
- Backend. The public setup check holds, and every gate record verifies:
  - its masking proofs, including identity-X rejection;
  - both rerandomizations;
  - q_{dec} distinct share proofs and their Lagrange combination;
  - decoding to $\pm g$;
  - the output, recomputed by Eq. (5).

  In addition, every affine map is recomputed, Q and C_b are recomputed by Eq. (6), and the final release verifies and decodes to $\mathcal{O}$ or g. Any attempt to release C_* is rejected.

- Binding. For every $i \le n$ and $k < L_{\text{vote}}$, compare the tally input at i, k with the snapshot ciphertext at the same position. The four coordinates must agree exactly:
$$(R.x, R.y, S.x, S.y).$$
Each point passes the point checks, with $R \neq \mathcal{O}$.
- Hash. $H(\mathcal{T})$ is the hash of exactly the canonical bytes that were verified.
- Bit. The decoded bit equals the claimed bit.
- Claim. The claim's $commit_{\text{set}}$ equals the snapshot accumulator.

Lemma F.1 (Lemma EB). Suppose Accept $= 1$ and the chain view is authentic. Then, except with probability $\varepsilon_{\text{hash}}$: 1. the sealed accepted set is exactly the event-reconstructed snapshot; 2. the tally's inputs are exactly those ciphertexts in canonical order; 3. the verified transcript is the one whose digest is claimed; 4. $\hat{b}$ is the bit decoded from that transcript's final release; 5. the claim refers to this sealed set.

Proof.

1. A different set with the same accumulator value yields a collision in the domain-separated accumulator.
2. Coordinate equality is exact, so no rerandomized or re-encrypted ciphertext passes.
3. Verified bytes are hashed bytes, so a different transcript with the same digest is a collision.
4. This is Bit.
5. This follows from Claim together with the contract's comparison of $commit_{\text{set}}$ against its sealed value. □

G Proof of Theorem 1

Bad events. Let Bad be the union of four events:

- E_1: extraction fails at some accepted position (Lemma BA); $\Pr[E_1] \le n\varepsilon_{\text{KS}}$.
- E_2: registry binding fails; $\Pr[E_2] \le \varepsilon_{\text{reg}}$.
- E_3: some verified Fiat–Shamir proof has a false statement.
- E_4: a hash collision occurs; $\Pr[E_4] \le \varepsilon_{\text{hash}}$.

Bounding E_3. Every checked proof is a Fiat–Shamir transform of a special-sound Σ-protocol with challenge space $\mathbb{Z}_q$. For the disjunctive gate proof, the challenge is $d = d_1 + d_{-1}$, and two accepting transcripts with distinct d share a branch with distinct branch challenges. Hence, for a false statement and fixed commitments, at most one challenge value $d \in \mathbb{Z}_q$ is accepting.

The implementation derives d by reading a SHA-256 output as a big-endian integer and reducing it modulo q, without rejection sampling. Since $2^{256} = 42q + r$ with $0 < r < q$, every residue has at most 43 preimages. A random-oracle output therefore hits any fixed challenge value with probability at most

$p_{\mathrm{FS}} = 43/2^{256}$. Each of the adversary's q_H queries, and each of the verifier's N_{FS} checks, thus succeeds on a false statement with probability at most p_{FS}, so $\Pr[E_3] \le (q_H + N_{\mathrm{FS}})\, p_{\mathrm{FS}}$.

The count is $N_{\mathrm{FS}} = (3a + q_{\mathrm{dec}})C_{\mathrm{total}} + a + q_{\mathrm{dec}}$. Each gate contributes a masking proofs, a zero-encryption proofs for each of its two rerandomized wires, and q_{dec} share proofs. The final release contributes $a + q_{\mathrm{dec}}$.

Argument outside Bad**.**

Step 1 (legal admission). By Corollary C.2, ι is injective and $CT_{i,k}$ encrypts $x_{i,k}$ with $x_i = v_i w_{\iota(i)}$.

Step 2 (exact board). By Lemma F.1(2), the circuit leaves are exactly these ciphertexts.

Step 3 (gate correctness). Consider a gate whose inputs (X, Y) encrypt (x, y) with $y \in \{0, 1\}$; by induction and Lemma WT(b), every gate is of this form. We follow the gate through its phases:

- The public setup check places $(h, h_1, \ldots, h_a)$ on one degree-t polynomial f in the exponent. For $t = a - 1$ this is the single check $h = \sum_j \ell_{[a],j} h_j$.
- True masking statements give signs σ_j applied to both wires. X_a therefore encrypts σx and Y_a encrypts $\sigma(2y - 1)$, where $\sigma = \prod_j \sigma_j$.
- True zero-encryption statements preserve both plaintexts through rerandomization.
- Every accepted share satisfies $w_j = [f(j)]u$. Any q_{dec} distinct shares combine to $[f(0)]u = [sk]u$, so the decoded value is the true $\sigma(2y - 1) \in \{\pm 1\}$.
- Eq. (5) then yields an encryption of $(x + \sigma(2y - 1)\sigma x)/2 = xy$.

By Lemma WT(c), the publicly recomputed C_b encrypts $[\sum_i x_i \ge \tau]$.

Step 4 (terminal). The verifier itself computes $Q = D - C_*$ and $C_b = E_1 - Q$ (Lemma WT(d)). The physical C_* is never accepted as a release input.

Step 5 (release). True rerandomization and share statements decode $M = [b]g$.

Step 6 (binding). By Lemma F.1(3)–(5), $\hat{b}$ is this decoded bit and the claim refers to the sealed set.

Hence $\hat{b} = [\sum_{i \in \mathrm{Acc}} v_i w_{\iota(i)} \ge \tau]$, and a union bound over Bad gives

$$\varepsilon_{\mathrm{corr}} \le n\, \varepsilon_{\mathrm{KS}} + \varepsilon_{\mathrm{reg}} + (q_H + N_{\mathrm{FS}})\, p_{\mathrm{FS}} + \varepsilon_{\mathrm{hash}}.$$

The proof uses no DDH, no honest trustee, and no setup secrecy. The non-identity check serves the privacy simulator, not correctness. □

H Proof of Theorem 2

H.1 Framework. We work in CGY's instantiation of SUC [4, App. H], [28]. It has three ingredients:

- a participants and a static malicious adversary that corrupts a set C with $|C| \le t < a$ before execution;
- a router that delivers ideally authenticated messages;

– an environment that chooses inputs and reads outputs.

Restricted environments [4, App. H.4] give all participants the same public inputs, supply selector inputs in $\mathcal{L}_{\text{bits}}(h)$, and supply a consistent degree-t sharing [4, App. I.1]. Nielsen's theorem [29, Thm. 3.5] preserves composition for environments whose inputs are restricted to a decidable language.

H.2 Gate layer. By Lemma FG together with [4, Lemmas I.1–I.3, Thm. I.4], $\mathit{\Pi}^{g}_{\text{CSZ}}$ SUC-securely computes $\mathcal{F}_{\text{CSZ}}$ in the $(\mathcal{F}_{\text{RO}}, \mathcal{F}_{\text{B}})$-hybrid model for restricted environments with at least one honest participant. The hypothesis $f < q_{\text{dec}} \leq a$ guarantees that participant.

H.3 Circuit layer. Consider the protocol that evaluates $\mathit{\Gamma}$ with each gate replaced by an ideal $\mathcal{F}_{\text{CSZ}}$ call. By Lemma WT(a,b), every invocation receives common public inputs with a selector in $\mathcal{L}_{\text{bits}}(h)$ whenever the admitted inputs are in $\mathcal{L}_{\text{bits}}(h)$, which is exactly the restricted environment of Theorem 2. There are polynomially many invocations, with distinct session identifiers. [4, Thm. H.3], which is valid for restricted environments, therefore replaces each real gate by $\mathcal{F}_{\text{CSZ}}$ with negligible loss.

H.4 Release layer. We show that the $\mathcal{F}_{\text{CSZ}}$-hybrid protocol, followed by the release, SUC-securely computes $\mathcal{F}^{\text{trace}}_{\text{WT}}$. The simulator proceeds in four steps:

1. It forwards inputs and obtains from $\mathcal{F}^{\text{trace}}_{\text{WT}}$ fresh encryptions (Z_η) of all gate outputs, together with b.
2. It answers each non-terminal call with the corresponding Z_η.
3. For the terminal call, it computes D from the answers already given, samples ρ, and returns $C^*_* = D - \mathsf{Enc}_h(1-b;\rho)$.
4. It simulates the release as in [4, Lemma I.2] and the proof of [4, Thm. J.1]. It chooses the rerandomized ciphertext $([r]g, [b]g+[r]h)$ with known r, computes honest shares $[r]h_j$, programs their Chaum–Pedersen proofs through $\mathcal{F}_{\text{RO}}$, and checks corrupted shares as the protocol does.

In the hybrid, $\mathcal{F}_{\text{CSZ}}$ returns a fresh encryption of the true plaintext $z_* = \text{pt}(D) - (1-b)$, with uniform randomness independent of everything else. C^*_* has plaintext $\text{pt}(D) - (1-b)$ and randomness $\text{rand}(D) - \rho$, which is uniform and independent, so the two distributions are identical. Moreover, $C^*_b = E_1 - D + C^*_* = \mathsf{Enc}_h(b;-\rho)$ has known randomness.

Whenever the simulator programs a proof, it samples the 256-bit oracle output uniformly and derives the challenge exactly as the implementation does. Simulated proofs therefore follow the implemented challenge distribution. The simulation is perfect except for negligible oracle-programming collisions, and transitivity with H.2 and H.3 proves Theorem 2.

H.5 Parameters. Two conditions enter the proof:

– CGY's gate lemmas require at least one honest participant.
– The framework bounds corruptions by the threshold $t = q_{\text{dec}} - 1$ [4, App. H.2]. The trace-hiding step of Appendix I needs the same bound to simulate corrupted shares without the key.

The hypothesis $f < q_{\mathrm{dec}} \le a$ implies both. When $q_{\mathrm{dec}} = a$, as in the reference instance $(5, 4, 5)$, the two conditions coincide: $f \le 4$, and at least one trustee is honest. □

I Proof of Corollary 1

Hybrids. The prefix is generated identically in every hybrid.

- H_0**.** The real experiment.
- H_1**.** The suffix is replaced by the ideal process for $\mathcal{F}_{\mathrm{WT}}^{\mathrm{trace}}$ with Theorem 2's simulator $\mathcal{S}_2$. The prefix, together with the distinguisher, forms a restricted environment: the sealed board is common to all trustees, and its ciphertexts lie in $\mathcal{L}_{\mathrm{bits}}(h)$ unless extraction fails. Hence $|\mathsf{H}_0 - \mathsf{H}_1| \le \varepsilon_{\mathrm{SUC}} + n\varepsilon_{\mathrm{KS}}$.
- H_2**.** The trace is produced by the reduction $\mathcal{B}$ below on a DH tuple. This equals H_1 whenever extraction succeeds; that failure event is already charged in the previous step.
- H_3**.** $\mathcal{B}$ runs on a random tuple. Then $|\mathsf{H}_2 - \mathsf{H}_3| \le \mathrm{Adv}^{\text{fb-ddh}}_{\mathbb{G},g} + 1/q$.
- H_4**.** The ideal experiment. $\mathcal{S}$ runs $\mathcal{S}_2$ on a self-sampled uniform trace and b. This is identical to H_3.

The reduction $\mathcal{B}(g, A, B, T)$.

1. *Setup.* Set $h := A$. Choose a set $I \supseteq C_T$ of t positions, which is possible because $f \le t$.
   - For $j \in I$, sample $s_j \leftarrow \mathbb{Z}_q$ and set $h_j = [s_j]g$.
   - For $j \notin I$, interpolate h_j in the exponent from $(0, h)$ and $(j', h_{j'})_{j' \in I}$.

   This is exactly the joint distribution of the public values and corrupted shares of a uniform degree-t sharing of a uniform secret, and $\mathcal{B}$ never learns α.
2. *Prefix.* Generate the reference string honestly and run the real prefix with genuine proofs. An honest voter encrypts under h with her own randomness and proves R_{ballot} without knowing sk. The run yields $(\mathsf{pp}, \mathsf{BB}, \mathsf{st}_{\mathcal{A}})$.
3. *Extraction.* Apply Lemma BA to obtain every $x_{i,k}$, aborting on failure. From these, compute every gate plaintext m_η of Γ and the bit b.
4. *Trace.* For each non-terminal gate η, sample $u_\eta, v_\eta \leftarrow \mathbb{Z}_q$ and set
$$Z_\eta = \big([u_\eta]g + [v_\eta]B,\ [m_\eta]g + [u_\eta]h + [v_\eta]T\big).$$
   For the terminal gate, compute D from these values and set $C_* = D - \mathsf{Enc}_h(1 - b; \rho)$.
5. *Suffix.* Run $\mathcal{S}_2$ on $((Z_\eta), C_*, b)$ against $\mathcal{A}$'s continuation, and output the joint view.

Analysis.

- *DH case.* If $T = [\alpha\beta]g$, then $Z_\eta = \mathsf{Enc}_h(m_\eta; u_\eta + \beta v_\eta)$ with independent uniform randomness. These are fresh encryptions of the correct plaintexts, so the view is H_1.

- *Random case.* If $T = [\gamma]g$ with $\gamma \neq \alpha\beta$ (probability $1 - 1/q$), then $Z_\eta = ([u_\eta + \beta v_\eta]g, [m_\eta + \alpha u_\eta + \gamma v_\eta]g)$. The map $(u_\eta, v_\eta) \mapsto (u_\eta + \beta v_\eta, \alpha u_\eta + \gamma v_\eta)$ has determinant $\gamma - \alpha\beta \neq 0$. The Z_η are therefore independent, uniform in $\mathbb{G}^2$, and independent of m_η, so the view is H_3.
- *Terminal consistency.* In all hybrids, $C_* = D - \mathsf{Enc}_h(1 - b; \rho)$ preserves $Q = D - C_*$ and $C_b = E_1 - Q$, and C_b encrypts b.

Why dummy trace values are admissible. Theorem 2 was applied to the correctly typed computation in H_1. Afterwards, the trace serves only as input to the fixed PPT procedure $\mathcal{S}_2$, extended if necessary to a total procedure on arbitrary encodings. Computational indistinguishability is preserved under such post-processing, and no CGY gate is ever run on uniformly random "selectors".

Oracles and extraction. $\mathcal{S}_2$ programs $H_{\mathbb{G}}$ and the Fiat–Shamir oracle from their first queries. Ballots use Groth16 and query no protocol oracle, so the programming is consistent with the prefix. Extraction appears only in $\mathcal{B}$; the simulator uses no witness.

Conclusion. Summing the hybrid losses gives the bound of Corollary 1. Corollary C.2 identifies b with the registry-attributed decision, except with probability $\varepsilon_{\mathrm{reg}} + n\varepsilon_{\mathrm{KS}}$. Finally, $\mathcal{S}$ receives b even in executions that $\mathcal{A}$ aborts after learning it; this is the explicit leakage upper bound discussed in Section 9. □

J Implementation and Evaluation Details

J.1 Code-to-algorithm map

Table 8. Security-critical components and the CGY algorithms they implement.

Implementation component	CGY algorithm	Role
Group encoding and validation	group premise (Alg. 7–10, 63–67)	canonical encoding; curve, subgroup, and role-specific identity checks
Auxiliary group oracle	$\tilde{h}$ interface (Lemma FG)	domain-separated hash-to-subgroup (App. B.3)
Fiat–Shamir challenge derivation	Alg. 9, 63, 65	Fiat–Shamir challenges over complete context, separate domains; SHA-256 reduced mod q
Threshold setup	App. A.2, Alg. 65	degree-4 setup, verification keys, public setup check, interpolation
Common-view board	$\mathcal{F}_{\mathrm{B}}$, $\mathcal{F}_{\mathrm{SB}}$	reference board service: authenticated immutable slots, barriers, abort on equivocation
Proof verification	Alg. 9, 10, 63/64	masking proofs; zero-encryption proofs
Rerandomization	Alg. 63	all-trustee synchronous rerandomization
Threshold decryption	Alg. 65	Chaum–Pedersen shares; five distinct verified shares; Lagrange at zero
Validation state machine	Alg. 65/67 dependencies	typed states Received → ProofValidated → BroadcastAgreed → RerandomizationValidated → AuthorizedForDecryption → PartialShare; no raw-ciphertext share path
Conditional gate	Alg. 7, 67	complete gate; output by Eq. (5)
Weighted arithmetic	Alg. 21 (carry retained), 40, 23	balanced aggregation; comparison; $Q = D - C_*$, $C_b = E_1 - Q$
Final release	Alg. 63–65	rerandomize-then-decrypt of C_b only
Accepted-board binding	exact admission (App. F)	unchanged sealed-board decoder; coordinate equality; opaque admitted board
Producer and replay pipeline	composition, public verification	producer pipeline; independent replay from public artifacts only

J.2 Adversarial test suite

Mutation matrix. The mutation matrix alters public protocol objects or attempts forbidden transitions. Each of the following mutations is rejected with the error the protocol prescribes:

- an altered auxiliary element;
- an altered masking proof;
- a wrong challenge or session context;
- a changed trustee identity;
- a stale broadcast slot;
- a malformed rerandomization proof, input, or output;
- replacement of an accepted input ciphertext after admission;
- partial decryption before authorization or on raw values;
- a share for the wrong ciphertext or session;
- an altered share proof;
- a duplicate trustee share;
- insufficient shares;
- a reordered or replaced ballot;
- a wrong board hash;
- an attempt to release the physical C_*;
- a changed result bit;
- tampered transcript payloads, signatures, or hashes.

Protocol-ordering violations produce deterministic outcomes instead. A conflicting write to an immutable slot aborts deterministically. A reveal before the commit barrier waits and exposes no reveal vector. A missing commitment yields a deterministic wait.

Threshold boundary tests. Five valid shares succeed, and three or four valid shares are insufficient. The following are rejected:

- four valid shares plus an invalid one;
- a duplicate, foreign, wrong-ciphertext, or wrong-session fifth share;
- an altered public share;
- a final release with four shares.

Scope. These tests check protocol-input handling; they do not substitute for the theorems.

J.3 Board service and benchmark methodology

Board service. The canonical authenticated board realizes the common-view broadcast $\mathcal{F}_{\mathrm{B}}$ and the synchronous broadcast $\mathcal{F}_{\mathrm{SB}}$:

- Every submission is authenticated under the sending trustee's registered verification key.
- Each (setup, execution, gate, phase, trustee) slot accepts one immutable payload, and a conflicting duplicate triggers a deterministic abort.
- Synchronous phases stay hidden until all required slots are filled. They are then sealed and revealed atomically in trustee order.
- A missing slot remains an explicit wait.

The service is trusted for availability, authentication, atomic sealing, and non-equivocation. It is not trusted for secrecy or for tally correctness.

Run setup. Every run uses a fresh setup and fresh proof-valid ballots. The native backend evaluates the ready frontier of the gate DAG in parallel within one process, using projective BabyJubjub arithmetic.

Replay and oracle. Public replay runs as a separate process with a streaming native verifier over bounded batches, and it reads public artifacts only. The cleartext oracle lives in a separate directory, and only the final comparator reads it, after replay. Normal protocol processes never receive W or oracle data.

Measurements. Timings are wall-clock seconds. For every run, the compact codec round-trips losslessly to the same SHA-256.

Host. All local runs used one workstation with an Intel Core i7-14650HX (16 physical/24 logical cores) running Windows, and the native backend is built with rustc 1.98.1. Measurements are host-specific, and these measured configurations do not establish an asymptotic law.

Co-location. The harness co-locates all trustee roles and exposes only protocol messages that pass verification to the board and verifier. This is an evaluation convenience, not a process-isolation claim.

J.4 Arbitrum Sepolia record

Table 9. Identifiers for the public execution. Full values are in the artifact.

Field	Value
Network / chain ID	Arbitrum Sepolia / 421614
Groth16 verifier / poll contract	`0x94Cb...6d3d` / `0x6a8e...88fC`
Poll ID / τ	910004 / 500
Admission blocks / seal block	311,761,087–311,761,234 / 311,761,234
Setup hash	`67ae...9627`
Sealed-board hash	`0x6487...a172`
Claimed transcript hash $H(\mathcal{T})$	`0c48...d064`
Outcome-claim transaction	`0x9248...4ff7`
Deployment gas: poll / Groth16 verifier	1,436,337 / 1,653,005
Ballot gas: total / median / p95	4,438,147 / 558,892 / 584,778
Outcome-claim gas	76,652

A read-only preflight confirmed the chain identifier and RPC responsiveness before deployment, and no credential is stored with the artifact. The run used the unchanged poll contract, with no batch interface.

J.5 Tally-reconstruction baseline

All mean leakage percentages are per affected proposal. The released implementation of [2] produced byte-identical output in ten executions in our environment. The four-proposal difference in vulnerable counts between the reported and

re-executed results remains unexplained. The released implementation's 19 incorrect determinations occur in the whale stage, where floating-point subtraction residues at equality boundaries can flip the strict elimination test. Our exact-integer variant emits no incorrect determinations, and on the six proposals where the two variants differ it is conservative in every case. This baseline measures the disclosure policy it motivates, not a security or performance property of our construction.

Table 10. Tally-reconstruction baseline: reported values, released re-execution, and exact-integer analysis.

Quantity	Reported [2]	Released re-execution	Exact-integer
Proposals	3,844	3,844	3,844
DAOs	31	31	31
Vulnerable proposals	3,118	3,114	3,112
Full recovery	1,122	1,122	1,118
Mean ballots identified (vulnerable)	41.6%	41.558%	41.53%
Mean weight identified (vulnerable)	85.1%	84.992%	85.08%
Determinations emitted	–	79,854	79,735
Incorrect determinations	–	19	0

J.6 Full evaluation tables

The following tables preserve the complete local-scale and public-chain measurements from the final manuscript; the main body shows only the values needed for the evaluation argument.

Table 11. Local scale with five trustees, degree 4, and five-share openings. Times are wall-clock seconds; "total" is the complete pipeline including replay. "JSONL" is the canonical gate journal plus the final-release record; "compact" is the lossless compressed journal. Every run passes independent replay and matches the oracle.

n	gates	depth	width	proving	admission	backend
8	134	72	4	10.043941	0.153007	4.054594
64	1,086	147	32	146.507326	1.282232	15.980992
256	4,350	207	128	368.448950	5.215520	47.128688
1,024	17,406	275	512	1,414.928296	19.163596	156.715285

n	release	replay	total	JSONL (B)	compact (B)	result / oracle
8	6.926112	0.919379	22.122671	27,315,984	7,360,585	0/0
64	8.926647	11.684111	184.408306	221,173,348	59,657,477	0/0
256	8.402803	32.910089	462.133677	886,045,853	239,114,329	1/1
1,024	4.568317	114.539565	1,709.946249	3,546,634,443	956,889,369	1/1

Table 12. Fresh Arbitrum Sepolia execution with the reference instance.

Item	Value
Network / chain ID	Arbitrum Sepolia / 421614
Ballot proving	10.360267 s
Poll ID / τ	910004 / 500
Chain admission	41.261974 s (deployment + 8 ballots)
Voters / accepted / rejected	8 / 8 / 0
CGY backend	3.797074 s
Threshold / share IDs	5-of-5 / 1–5
Final release	4.226528 s
Gates / depth / width	134 / 72 / 4
Producer total (incl. claim)	63.223557 s
Transcript JSONL / compact	27,315,148 B / 7,357,605 B
Independent public replay	62.143068 s
Ballot gas (median / p95)	558,892 / 584,778
Outcome-claim gas	76,652
Result / oracle	1 / 1
Binding, replay, claim	all verified