INTERNATIONAL JOURNAL OF ADVANCED RESEARCH
ISSN: 3106-7980: 3106-7972
Volume 3, Issue 3, 2026

A TRUST-MINIMIZED PRIVACY-PRESERVING BLOCKCHAIN VOTING SYSTEM ON ETHEREUM USING ZK-SNARKS WITH CLIENT-SIDE PROVING AND RELAYER-BASED UNLINKABILITY

Arafat Ali Khan[*1], Khalid Hamid[2], Muhammad Husnain Shahid[3], Malik Waqar Ali[4], Waqar Ali[5], Muhammad Zain Amir[6]

[*1,2,4]*Department of CS & IT, Superior University, Lahore, 54000, Pakistan*
[3,5]*Department of Criminology, Lahore Garrison University, Lahore, 54000, Pakistan*
[6]*Department of Smart Computing & Cyber Resilience, Sunway University, Kuala Lumpur, 47500, Malaysia*

[*1]arafatalikhan3@gmail.com, [2]khalid6140@gmail.com, [3]husnainshahix@gmail.com, [4]waqaralibcs@gmail.com, [5]waqaralifarzand@gmail.com, [6]zainamir1@live.com

DOI: https://doi.org/10.5281/zenodo.21735625

Keywords
account abstraction, blockchain, e-voting, Ethereum, Merkle tree, nullifier, Poseidon hash, privacy, relayer, unlinkability, zero-knowledge proof, zk-SNARK

Article History
Received: 01 May 2026
Accepted: 28 May 2026
Published: 31 July 2026

Corresponding Author: *
Arafat Ali Khan

Abstract

Public blockchains give elections a tamper-evident bulletin board, yet the transparency that makes results auditable also threatens ballot secrecy and voter anonymity. Prototype systems that pair Ethereum with zero-knowledge proofs commonly reintroduce trust through the back door: a server that holds voter secrets and generates proofs, transactions that link a voter wallet to a ballot, plaintext candidate choices in public calldata, and nullifiers that are accepted without being cryptographically bound to the verified proof. This paper proposes a trust-minimized voting architecture for small and medium electorates that removes all four weaknesses by construction. Eligibility is proven inside a Groth16 circuit against an on-chain Poseidon Merkle registry; proofs are generated entirely in the voter's browser through a WebAssembly prover so the voting secret never leaves the device; ballots are cast as hiding commitments and submitted through ERC-4337 relayers so no voter address ever appears on chain; and the double-vote nullifier is recomputed inside the circuit and checked against the proof's public signals. We give the full protocol, a property-by-property security analysis against a concrete threat model, and an analytical evaluation grounded in recent published benchmarks, which projects per-ballot costs of roughly 352,000 gas and sub-six-second casting latency on a rollup. The design is intended as the specification for a subsequent thesis implementation.

I. INTRODUCTION

Electronic voting promises accessibility and fast, auditable tallying, and public blockchains supply the property that paper elections obtain only through procedure: an append-only, replicated record that no single authority can rewrite. A broad body of recent work therefore treats Ethereum and comparable ledgers as the bulletin board for digital elections [1]. The same transparency, however, cuts against two requirements that democratic elections treat as absolute. Ballot secrecy demands that the content of a vote stay hidden, and anonymity demands that no observer can connect a recorded ballot to the person who cast it [31]. A systematic evaluation of blockchain voting systems published between 2022 and 2025 finds that reconciling public verifiability

with these two privacy requirements remains the central unresolved tension in the field [2].

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) are the standard cryptographic answer to that tension. A voter can prove, in a few hundred bytes, that a ballot satisfies every rule of the election without revealing who they are or what they chose, and a smart contract can check that proof in constant time [3]. In principle the combination yields elections that are simultaneously private and universally auditable [32, 33, 34, 35]. In practice, published prototypes and student-built systems repeatedly compromise the guarantee at the systems layer even when the cryptography is sound [36-38]. Our own earlier capstone prototype, which this paper extends, exhibited four failures that we have since found to be representative of the prototype class as a whole. First, a backend server generated voter secrets and computed proofs, so the operator of that server could deanonymize or impersonate every voter. Second, voters submitted their own transactions, so the wallet address in every ballot transaction linked the vote to a fundable, traceable identity. Third, the candidate choice travelled as a plaintext public input, so ballot content was readable in calldata by anyone. Fourth, the nullifier that was supposed to stop double voting was passed to the contract as a separate argument and never compared against the public signals of the verified proof, so a single valid proof could be replayed with fresh nullifiers without limit.

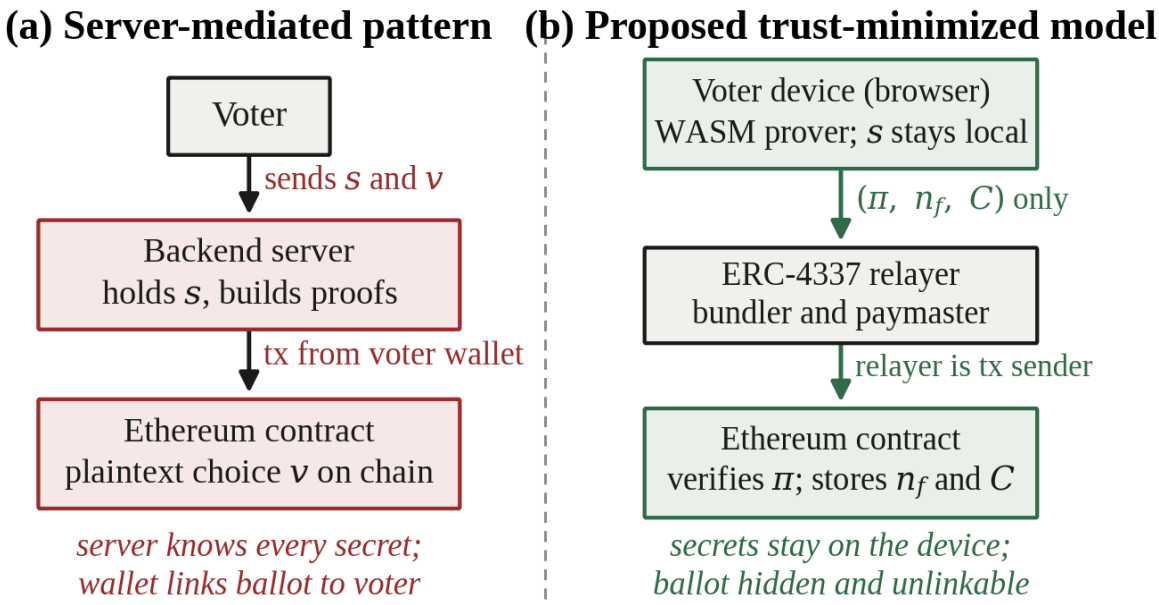

Fig. 1. Trust relationships in (a) the server-mediated prototype pattern, where a backend holds voter secrets, ballots appear in plaintext, and the sending wallet links each vote to a voter, and (b) the proposed model, where proving happens on the voter's device, the ballot is a hiding commitment, and a relayer breaks the wallet-to-ballot link.

Fig. 1 contrasts the two trust models. The failures on the left are architectural rather than cryptographic; that is, they can be fixed without resorting to a more robust proof system. They must be removed by redesigning where secrets live, who submits transactions, and what the contract actually checks. That redesign is the subject of this paper. We deliberately present a design and analysis rather than an implementation: the protocol given here is the specification against which a full implementation and measurement study will be carried out in a subsequent thesis, and the analytical projections in Sections IV and V define the acceptance thresholds that the implementation must meet.

The proposed system rests on four mechanisms that, to the best of our knowledge based on the 2023 to 2026 literature surveyed in Section II, have not previously been combined in a single voting design. (C1) Proof generation runs entirely in the voter's browser through a WebAssembly build of the Circom and snarkjs toolchain, so the voting secret is created on the device and never transmitted, removing the trusted proving server. (C2) Ballots are submitted as ERC-4337 user operations through independent relayers, so the transaction sender recorded on chain is a relayer address and no voter wallet ever appears in the election's transaction graph. (C3) The double-vote nullifier is computed inside the circuit as a Poseidon hash of the voting secret and the election identifier, is exposed as a public signal, and is the only nullifier value the contract will consult, which binds replay protection to the proof itself. (C4) Voting is performed without the candidate's choice even showing on the ballot and never can be correlated to an identity after the vote. In addition to mechanisms, the paper provides a property-by-property security argument that maps each guarantee to the opponent that it beats, and an analytical cost and latency model populated from recent published benchmarks that allows the design to be compared to existing designs and sets measurable targets for the thesis implementation (C5) and (C6).

The rest of the paper is structured in the following way. Section II summarizes the existing research on voting systems, proving infrastructure, client side proving, transaction-level privacy, coercion resistance, fee economics, and post-quantum perspective. The methodology is then described in Section III: system and threat model, layered architecture [39, 40, 41, 42], and the entire protocol and its circuit parameterisation. The results are presented in Section IV, a property-by-property security analysis is followed by analysis of security performance. Section V covers the design against previous systems, corrects the evaluation plan, and lists limitations, while the thesis roadmap is presented in Section VI.

II. Literature Review

A. Blockchain E-Voting Systems

Vladucu et al. provide a survey of e-voting and blockchain and list the requirements that the system must meet to be credible: Eligibility, Uniqueness, Privacy and Endtoend Verifiability, noting that deployed systems rarely meet all of these requirements [1]. The systematic review by Revelo Sánchez et al. addresses the designs and reports from the 2022-2025 generation and how they compromise privacy first to satisfy engineering requirements [2]. In the realm of Ethereum in particular, Lupu and Aciobăniţei present a zkSNARK-based Solidity election and show how to verify the proofs within a contract at a classroom level [4], while Rabia et al. suggest a decentralized anonymous system where zkSNARKs are equipped with both eligibility and validity, but where voters remain in control of their accounts [5]. ElSheikh and Youssef go the self-tallying route: their open, non-voter verification voting network collapses each round into zk-SNARK proofs, and reports the cost of the contract to be around 12.5 United States dollars on mainnet, and that the cost of their on-chain voting grows only logarithmically with electorate size, indicating that careful circuit design keeps the public-chain voting economically viable [6]. None of these systems however, solves the problem of connecting the submitting wallet to the ballot, and our relayer layer does just that.

B. Zero-Knowledge Voting Protocols and Nullifier Schemes

The nullifier pattern that underlies our double-vote protection descends from anonymous signaling schemes in which a member of a Merkle-committed group emits a one-time tag that is deterministic in the member's secret yet unlinkable to it. zkVoting by Park et al. is the strongest recent instantiation in the voting domain: it combines a nullifiable commitment scheme with Groth16 proofs to deliver coercion resistance and end-to-end verifiability, and it reports 2.3 seconds to cast a ballot and 3.9 milliseconds per ballot to tally [7]. Its trust model, however, includes a registrar that issues credentials and therefore can, if corrupted, undermine anonymity, and proof generation is not constrained to the voter's device. SmartphoneDemocracy moves

proving onto the voter's phone and bootstraps eligibility from the European Digital Identity wallet over a peer-to-peer bulletin board, which validates the feasibility of consumer-device proving for elections, though it targets its own TrustChain infrastructure rather than an EVM chain [8]. Burgos and Alchieri give a reusable recipe for zk-SNARK-based privacy inside permissioned smart contract platforms, including delegated submission flows in which a third party relays a user's proof, an idea we adapt to the permissionless setting through ERC-4337 [9]. The evidence collected in the surveys confirms that the main application area for zero-knowledge tooling is voting [3].

C. Proving Systems, Hash Functions, and Circuit Tooling

Our choice of Groth16 over universal-setup alternatives follows the benchmarking literature. Guo et al. benchmark ZK-friendly hash functions and proving systems for EVM-compatible chains and report that Poseidon and Poseidon2 dominate circuit-internal hashing, at roughly 240 rank-1 constraints per hash, with Poseidon2 reducing on-chain verifier cost by 73 percent relative to their baseline on EVM chains; they also find that a depth-seven Merkle circuit completes setup in about 70 seconds where a comparable GMiMC circuit needs 3,476 seconds [10]. Kuznetsov et al. systematically benchmark Groth16 through the circom-snarkjs pipeline and confirm the properties that make it attractive on chain: proof size constant at roughly 800 bytes in JSON serialization, near 256 bytes in binary form, and constant-time verification of about 0.66 seconds regardless of circuit size across circuits from one to more than 1.5 million constraints [11]. Correctness of hand-written circuits is a recognized risk: Chaliasos et al. demonstrate that fuzzing uncovers real bugs in widely used zero-knowledge libraries [12], and zkFuzz formalizes under- and over-constrained circuit vulnerabilities in the trace-constraint consistency framework and finds them at scale in Circom code [13]. The gap between zero-knowledge research and deployed practice is systematized by Liang et al. [14], the surrounding framework ecosystem by Sheybani et al. [15], and the range-proof constraints we reuse for candidate encoding by Christ et al. [16]. Together these results justify our stack, Groth16 with Poseidon over the BN254 curve compiled by Circom, and motivate the circuit-fuzzing step in our evaluation plan.

D. Client-Side Proof Generation

It's a fact that can now be proven by voters on their own equipment. Palahusynets et al. compare zkSNARK protocols end to end, and measure the proving-time cost that the client device must bear, in particular, proving-time for Groth16 grows sublinearly with the number of constraints and is fine for commodity hardware [17]. FibRace provides a massive field benchmark of client-side proving on real mobile devices and browsers that reveals both that circuits of the size used in membership-and-nullifier voting statements can be proved in seconds on consumer phones and that browser proving is measurably slower than native proving, a loss we explicitly model in our latency model in Section IV [18]. Ismayilov's zQR framework provides a full mobile verification pipeline where proofs are created and verified on mobile devices with blockchain anchoring, another piece of evidence that the heavy end of zero-knowledge cryptography can be done without the need for a server and its associated hardware. With these results we can now consider the browser WASM prover as an engineering task instead of a research experiment, with known constants.

E. Relayers, Account Abstraction, and Transaction Unlinkability

It's impossible to provide anonymity when the transaction envelope contains the voter's name, that's why we have account abstraction through ERC-4337 with every ballot. Lin et al. record the number of ERC-4337 contracts deployed on Ethereum and estimate the gas above baseline that user operations incur compared to the gas of regular transactions that they are based on [20]. The 167,830 gas of pure layer-2 execution for a sponsored token transfer reported by Jiao and Udomlertsakul's SuperPaymaster shows that decentralized gas sponsorship is viable compared to two commercial baselines reported by 205,951 and 328,937 gas in each case.The 167,830 gas of pure layer-2 execution reported by Jiao and Udomlertsakul's SuperPaymaster indicates that

decentralized gas sponsorship is competitive compared to two commercial baselines, reported on 205,951 gas and 328,937 gas, respectively. R. Baldimtsi et al. have systematically classified the design space of privacy-preserving transactions, and identified the relayer-based submission approach as part of the mixing and shielded-pool design space [22]. Of particular relevance is Minaei et al.'s data tumbling layer, which provides composable unlinkability for arbitrary contract state, shows an unlinkable weighted voting application, and reports operation latencies of less than 1.5 seconds (even though this is for data that has been committed, not data that has been checked for eligibility) [23]. Empirical anonymity literature provides similar warnings of user behavior degrading protocol anonymity: Wu et al. empirically demonstrate that deposit/withdrawal patterns deanonymize Tornado Cash users [24] and Huseynov et al. quantify a comparable anonymity loss in Railgun from timing and amount correlations [25]. We try to counter this by ensuring that the shape of the payload on ballots is fixed, and proposing to introduce a random delay in submitting ballots, thus rendering any behavioral side channels as in those studies, voter-specific.

F. Coercion Resistance and Receipt-Freeness

The Coercion resistance is the most powerful privacy property in the literature on voting and the one we explicitly consider as partial. Giustolisi et al. present a scheme resistant even to the last minute coercer, who attacks just before the deadline, and they explain the mechanism behind full coercion resistance, such as re-voting and indistinguishable fake credentials [26]. zkVoting does so by employing nullifiable commitments with a trusted registrar [7] for coercion resistance. Nothing a voter is able to prove in this public chain of transactions is a connection to a recorded ballot, but a coercer who attends the voter during voting proves the weaker; precisely this boundary is stated in Section IV and assigned to the future in Section VI.

G. Transaction Costs and Layer-2 Scaling

Whether a per-ballot on-chain footprint is affordable depends on fee dynamics. Roughgarden's economic analysis of EIP-1559 establishes the fee mechanism under which our per-vote gas figures translate into currency cost and shows why base-fee variability argues for rollup deployment of bursty workloads such as election-day traffic [27]. Dyade quantifies the cost reduction that EIP-4844 blob space delivers to layer-2 rollups and supports the deployment recommendation we adopt, an optimistic or zero-knowledge rollup as the default venue with layer-1 reserved for the registry root and final tally commitment [28]. ElSheikh and Youssef's logarithmic on-chain growth result [6] composes with this: circuit design bounds gas per ballot, and rollup economics bound the price of that gas.

H. Post-Quantum Considerations

Groth16 rests on pairing assumptions that quantum adversaries break, so long-horizon ballot confidentiality requires a migration path. Farzaliyev et al. bring lattice-based zero-knowledge proofs to electronic voting with concrete performance figures, demonstrating that quantum-resistant proof systems for this application class are already practical at moderate scale [29], and Steinfeld et al. survey the post-quantum zero-knowledge landscape and its application constraints [30]. Our architecture separates the proof system behind an interface that is accessed through a verifier contract, allowing for a change from Groth16 to a lattice-based or hash-based proof system to be a contained change; this is recorded as thesis-stage work and not claimed for now.

III. METHODOLOGY

This section defines the full design – actors participating in the design, the adversary model, the security requirements and scope, the layered architecture, the deployment topology, and the end-to-end protocol and its circuit parameterization.

A. Actors

A total of six roles is involved in an election. The voter has a secret s, which is generated on their own device, and only connects via browser. The election authority sets up and defines the electorate, publishes the eligibility registry, opens and closes the elections, is trusted for liveness of the registry but never sees s and never handles ballots. Relayers are standalone ERC-4337 bundlers, all of which will relay a well-formed user operation; a paymaster pays

gas for voters that do not need to have ether. The tally is computed, nullifiers and ballot commitments are recorded, and the contract verifies the proofs, this is all in the contract layer of Ethereum or a rollup. Lastly, anyone can participate in the program and be a verifier by replaying the on-chain record. No tallier can have private state: tallying is purely a function of public data.

B. Adversary Capabilities

We assume that there are five adversaries that can collude. Each transaction, event, and state cell are read by an observer in a chain of observers forever. A network observer also observes encrypted metadata of the voters and relayers. A malicious relayer can drop, delay, shuffle or attempt to alter the user operations, and can log the network from where operations were submitted. A wrongdoer casts a vote when he or she is not entitled to vote, casts two or more votes, or fills in ballots in a way that causes the ballots to be counted incorrectly. The remote coercer checks only what the voter can present to him, the public chain, the supervising coercer watches the voter while he casts his votes. The election authority is a good person but curious for privacy reasons: It follows the protocol, but if the record were not anonymous it would deanonymize voters. We assume the standard blockchain model where the chain itself does not equivocate and the Groth16 setup ceremony is known to have at least one honest contributor, and discrete-log and pairing assumptions hold against current adversaries and a direct-submission fallback if all misbehave for censorship purposes.

C. Security Requirements

Table I lists six requirements the design must satisfy, and assigns each requirement to an adversary it is supposed to defeat and the mechanism that enforces the requirement. Section IV is the table, with each row explicitly being given an argument.

TABLE I SECURITY REQUIREMENTS, ADVERSARIES, AND ENFORCING MECHANISMS

Requirement	Adversary countered	Enforcing mechanism
Eligibility	Ineligible voter; ballot-stuffing registrar	Merkle membership proof inside circuit against on-chain root R
Ballot secrecy	Chain observer; authority; relayers	Hiding commitment $C=H_P(v,r)$; reveal only after close
Unlinkability	Chain and network observer	Relayer is on-chain sender; single-use session keys; sponsored gas
One vote per voter	Double-voting voter	Nullifier $n_f=H_P(s,e_{id})$ recomputed in circuit, read from public signals only
E2E verifiability	Corrupt result announcer	Tally is a pure function of public proofs, nullifiers, and openings
Censorship resistance	Malicious relayer(s)	Any one honest relayer suffices; direct-submission fallback

D. Design Goals and Non-Goals

The design aims at the institutional elections (of approximately 100,000 voters), student unions, professional bodies, cooperatives, shareholder ballots, where the result needs to be public and withstand hostile scrutiny, and the electorate is small enough to avoid an impractical per ballot on-chain cost. In that context the design aims at three objectives: Firstly, no one except the voter should ever possess material that allows him to de-anonymize or impersonate that voter. Secondly, every property which has been promised must fail only if a stated assumption is proven incorrect, and, thirdly, the whole system should be auditable based

solely on published artifacts. The claim was limited by three limitations. This design cannot cover national-level elections where there are questions of turnout, identity and legality. A coercer who is physically present and monitoring the voter is not protected, because it is this type of coercer that Section IV argues the design excludes the need for trusted machinery. Unlike the transport itself, network-layer anonymity is not re-engineered here; rather existing anonymizing transports are used. A voting protocol does not add anything to the anonymity of the transport.

E. Architecture Overview

In Fig. 2, the architecture is decomposed into three layers, with strictly narrowing knowledge. The relayer layer is only aware of the ciphertext equivalent payloads [43, 44] and network metadata, the contract layer is only aware of what the whole world may know, and the client layer knows everything. Privacy failures in the prototype class arise exactly where this monotonicity is violated, a server that knows secrets, a transaction that knows the voter, and the layering is our structural remedy.

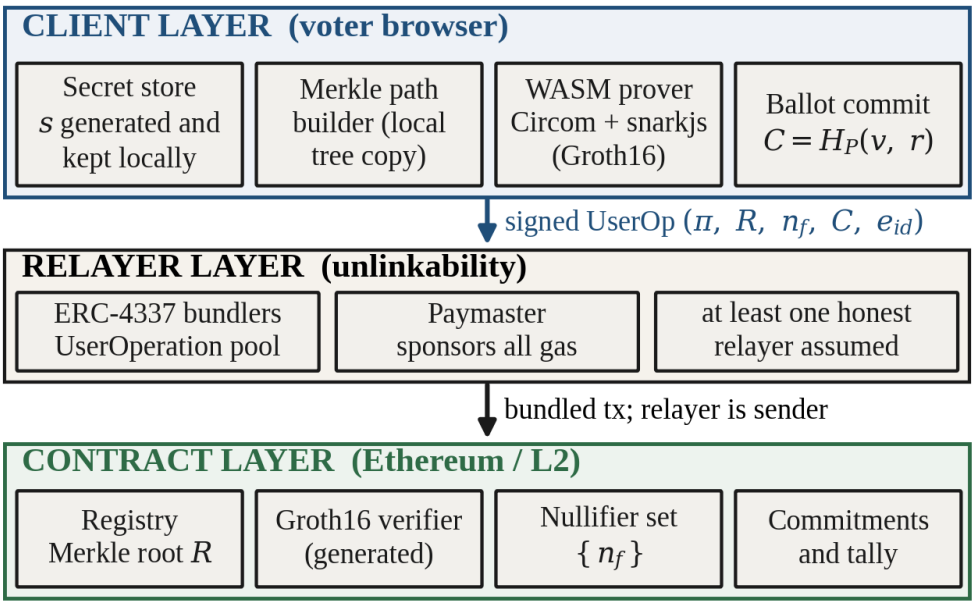

Fig. 2. Layered architecture. Secrets exist only in the client layer; the relayer layer forwards opaque user operations and appears as the on-chain sender; the contract layer verifies and records public values only.

F. Client Layer

The client is a static web application with no backend. On first use it generates the voting secret s with the browser's cryptographically secure random source and stores it locally under user control. The client embeds the compiled circuit as a WebAssembly module together with the proving key, both integrity-checked against hashes published by the election authority, and produces witnesses and Groth16 proofs entirely on the device, following the deployment pattern whose device-side cost the client-proving literature has now quantified [17], [18]. The client also maintains a local copy of the eligibility tree so that Merkle paths are computed without querying any party at voting time, which removes a lookup that would otherwise leak which leaf, and therefore which voter, is about to vote.

G. Relayer Layer

The relayer layer is designed to be commodity infrastructure: a paymaster, funded by the elections authority, and unmodified ERC-4337 bundlers. User operation contains the proof and its public signals as calldata sent to the voting contract, and its signature is generated by a new, one-use session key, which the client generates for each ballot, and which

is not connected to any wallet that the voter may own. The transaction graph of the election will only contain addresses of the relayers, since the voters are not listed in the graph but rather the senders (bundlers). There are multiple independent bundlers, any one of which is sufficient for delivering a vote, and a voter who is censored by all of them can also vote from an externally owned throwaway account at the expense of gas but no more, but not be worse off for the proof, which shows nothing about which leaf the voter voted.

H. Contract Layer

The design is finished with the addition of three contracts or three modules for one contract. The Merkle root R of the eligibility leaves and the parameters of the election itself (election identifier eid, and the phase schedule) are stored in the registry. The verifier is automatically generated from the circuit by the toolchain, and results in exposing a single pure function that verifies a Groth16 proof against the public signals. The ballot module maintains a mapping of spent nullifiers, an append-only list of ballot commitments, and the tally counters, and implements the phase machine: register, vote, reveal, closed. Contract logic is intentionally kept minimal at just the verifier call, two storage write instructions and the phase checks, as seen in Section IV, so that the gas used to audit the contract is small and the gas used to audit the per-ballot execution is small.

I. Deployment Topology and Data Availability

The default deployment runs the verifier, nullifier set, commitment list, and tally on a general-purpose rollup, and two anchors on the Ethereum layer 1: the registry root R at the end of registration and a hash of the end-of-reveal tally record. Rollup placement is tied to fee analysis of Section IV, the traffic on layer 1 is bursty and burst gas is expensive on layer 1 [27] while EIP4844 blob pricing makes rollup data cheap [28]. The layer-1 anchors are resistant to misbehaving rollup operators because the electorate is fixed by the registry root anchor before the occurrence of any ballot, and the tally hash anchor is fixed by the tally hash before the ballot is finished, meaning a layer-1 sequencer cannot affect either end of the window through censorship or reordering, and major layer-1 rollups have mechanisms to bound the cost of forced-inclusion during the window. The commitment list, one element of ballot per electorate, also can be expanded into blob space if electorate size is large enough, and is treated as a pre-committed backup by the evaluation plan of Section V.

J. Election Setup and Parameter Ceremony

Both the authority sets the size m of the electorate, the timing, and eid, then constructs a circuit and performs a circuit-specific phase-two contribution ceremony with many independent contributors, publishing all the details so that anyone can check that a contribution they trust was part of it. The proving key, verification key, circuit source, and WASM artifact hashes are published; the verification key is embedded in the verifier contract at deployment. A fresh phase-two ceremony per election is inexpensive at our circuit size, on the order of a minute of setup by benchmark [10], and confines any hypothetical ceremony compromise to a single election.

K. Voter Registration and Eligibility Registry

During registration each eligible voter's client derives the leaf

$$L = H_P(s, 1) \quad (1)$$

where H_P is the Poseidon permutation over the BN254 scalar field, and submits L through the authority's authenticated enrollment channel, for example the institution's single sign-on. The authority verifies eligibility of the person, inserts L into the tree, and after the registration deadline publishes the full leaf list and commits the root R on chain. Publishing all leaves costs nothing in privacy, leaves are hiding commitments to secrets, and it is what allows clients to build Merkle paths locally. The authority learns which persons registered, which any registrar must, but learns no secret and therefore can neither vote for a registrant nor recognize their ballot later. Registration is the one phase in which identity and cryptography touch, and the design confines that contact to a value from which nothing further can be derived.

L. Client-Side Witness and Proof Generation

To vote for candidate $v \in \{0, \ldots, m\text{-}1\}$, the client samples a blinding scalar r, reads the current R and its own path, and evaluates the circuit of Fig. 3. The circuit accepts private inputs s, the Merkle path, v, and r, and outputs public results R, eid, as well as the two public outputs that it computes and exposes: the nullifier and the ballot commitment. One constraint group recalculates L from s and checks the path to R to establish the membership, but not the identity of the leaf. Constraint group two computes

$$n_f = H_P(s, e_{id}) \quad (2)$$

so that the nullifier is deterministic per voter per election, unlinkable to L under the hiding of Poseidon, and impossible to grind because the voter controls no free variable in it. Constraint group three computes the commitment

$$C = H_P(v, r) \quad (3)$$

and range-checks v against m using standard comparator gadgets [16]. Proving runs in the WASM module in the browser; the published benchmarks for circuits of this size, tens of thousands of constraints dominated by two Poseidon evaluations and a depth-d path check, place proving at low single-digit seconds on laptops and modern phones [11], [17], [18]. The proof π is roughly 800 bytes serialized [11].

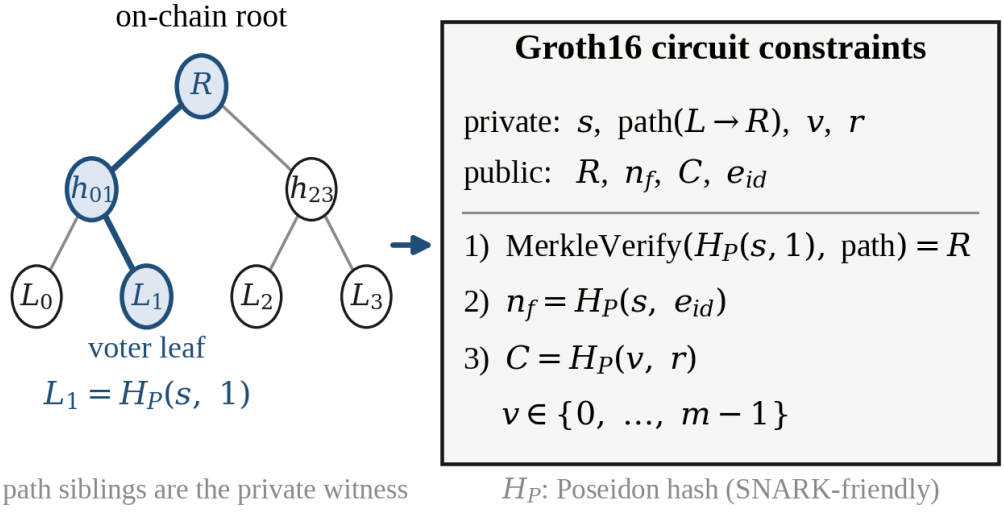

Fig. 3. Eligibility and nullifier derivation. The voter proves that a leaf recomputed from the private secret lies under the on-chain root, while the circuit recomputes the nullifier and ballot commitment that appear as public signals. Path siblings never leave the device.

M. Relayed Submission

The client packages (π, R, n_f, C, e_{id}) as calldata in an ERC-4337 user operation signed by a fresh session key, attaches the election paymaster, and hands the operation to one of the advertised bundlers over an anonymity-respecting channel; the client applies a small randomized delay before submission so that network timing does not partition voters, the countermeasure the mixer deanonymization studies motivate [24], [25]. The bundler wraps the operation into a bundle transaction and pays its gas through the paymaster, following the measured ERC-4337 flow [20] and the decentralized sponsorship pattern of [21]. The chain records the bundler as sender. Nothing in the operation, signature, calldata, or gas payment, is derived from any voter-owned account.

N. On-Chain Verification and Nullifier Binding

The voting contract's cast function receives the proof and the public signal vector and performs, in order: phase check, that the voting window is open; proof check, a single call to the generated verifier over exactly the received public signals; nullifier binding, reading n_f from position two of the verified public signal vector, never from a separate argument, and requiring that the nullifier mapping does not already

contain it; and recording, setting the nullifier consumed and appending C to the ballot list. The binding step is the direct repair of prototype failure four: because the only nullifier the contract ever consults is one the verifier has just attested was computed as (2) from the secret inside the proof, replaying a proof reproduces the same n_f and is rejected, and supplying a different n_f requires a different valid proof, which requires a different secret, which fails membership. Fig. 4 gives the end-to-end sequence.

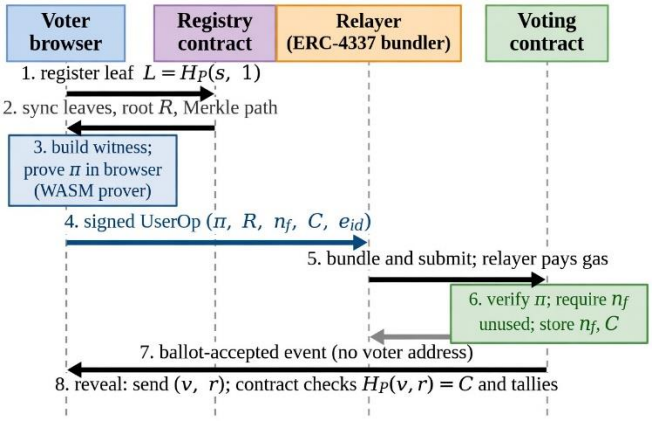

Fig. 4. End-to-end protocol sequence. Steps 1 and 2 occur during registration; steps 3 to 7 during the voting window; step 8 during the reveal window. The event log never contains a voter address.

O. Reveal, Tallying, and Result Verification

When the voting period ends, the contract is in reveal phase. The client who created the ballot sends the opening (v, r), also via a relayer and under a new session key, to the client indicated by the ballot number they are using. If HP(v, r) = C, then open it up and add one to v's counter; otherwise do nothing. Delayed opening provides running-tally secrecy, there is no information to an observer about any intermediate results during the vote, and the opened values can't be connected to the voters because both the commitment and the opening were sent through relayers. Ballats that are not opened on the reveal day are counted as abstentions—there is no cost to the voter to open the ballot, there is sponsored gas, and the client automates it. The set of accepted proofs bounds ballots by eligible secrets, and the nullifier set ensures that each ballot is unique; so individual voters can verify that their commitment appears in the set of accepted proofs, and any observer can verify the announced count, the two halves of end to end verifiability.

P. Circuit Size and Parameterization

Today, the statement is small, making browser proving realistic. A depth-d registry tree adds d path hashes and the leaf/candidate range check adds a comparator of a few hundred [16] and wiring overhead adds to bring the circuit to well under ten thousand constraints, with a Poseidon arity-two permutation costing around 240 rank-1 constraints per evaluation [10]. They are at the high end of all published benchmarks for this size of circuit: the setup takes about a minute [10], the proving key remains small enough, tens of megabytes, to ship along with the WASM prover, and the proving itself takes only low single-digit seconds [11], [17], [18]. The depth parameter is an inverse linear trade-off between the number of registries that can be used and the length of time needed to prove, meaning a campus deployment can reduce to d = 14 and path hashing can be reduced by almost a third. The constraint counts listed here are estimates that will be used to drive the acceptance tests in Section V - the actual count for the implemented circuit will be published with the artifact hashes.

IV. Results

The results are presented in two sections: a property-by-property security argument for every row of Table I against the opponents of Section III; and analytical performance projections. These security arguments

have been reduced, i.e., each of the properties holds only if the security assumptions made by the problem are true or a specified trust assumption holds.

A. Eligibility

A ballot is only recorded when the verifier accepts a proof with the statement containing recomputation of a leaf from the witness secret and a path check to the on-chain root R. If a party is ineligible, then it requires a witness for an ineligible leaf, which corresponds either to a preimage in some leaf published by Poseidon, ruled out by the one-wayness of the memory of the hash benchmarks that it builds on (as in the random-oracle style analysis), or to a Groth16 forgery, ruled out by the knowledge soundness of the proof system under its pairing assumptions [11]. The registrar can't put ballots in either, since he needs a secret to cast with them, and he gets only the letter L. The soundness depends instead on the correctness of the circuit: for this reason, the evaluation plan requires the deployment of the fuzzing and constraint-consistency tooling of [12] and [13] before the circuit is deployed for evaluation.

B. Ballot Secrecy

A proof and a nullifier along with a commitment C = HP(v, r) are stored per ballot within the chain during the voting window. The proof is zero-knowledge – it tells nothing other than the truth of the statement, and the commitment is hiding under the blinding scalar r – it tells nothing about one of the m candidates for v. Consequently, no observer, including the authority and the relayers, learns any ballot's content, and no running tally exists to influence late voters. After reveal, the multiset of votes is public by design, individual openings are public but arrive unlinkably, so secrecy after reveal reduces to the unlinkability property below rather than to content hiding.

C. Unlinkability and Anonymity

The linkage surfaces available to a chain observer are the transaction sender, the signature, the gas payment, the payload, and timing. The sender is a bundler for every ballot; the signature is by a single-use session key generated independently of any voter account; gas flows from the election paymaster [21]; the payload has identical shape for every ballot, a proof of fixed size [11] and three field elements; and the client's randomized submission delay removes the timing partition that behavioral studies exploited against Tornado Cash and Railgun [24], [25]. The nullifier is the one voter-deterministic value on chain, and linking it to a voter requires inverting $H_P(s, e_{id})$ to the same secret behind a published leaf $H_P(s, 1)$, which is excluded by the hiding of Poseidon under distinct domain tags. A network observer or a logging relayer additionally sees origin metadata; the design therefore recommends submission over an anonymizing transport, and notes that even a fully logging relayer learns at worst that some network endpoint voted, never which leaf or which candidate. Anonymity is thus information-theoretic on chain up to the anonymity set of the electorate, degrading only through off-chain network metadata, which is the correct place for the residual risk to live because voters can independently mitigate it.

D. Double-Vote Prevention

Equation (2) makes the nullifier a deterministic function of (s, e_{id}) with no voter-controlled freedom, and the binding step of Section III-N ensures the contract consults only the in-proof value. A second ballot from the same secret reproduces n_f and is rejected; a different n_f requires a different secret and fails eligibility. Cross-election linkage is also excluded: nullifiers for different e_{id} are unlinkable outputs of independent Poseidon evaluations, so participation across elections cannot be correlated. This ends prototype failure four in a non-contract-based fashion, with a proofcarried invariant.

E. End-to-End Verifiability

Individual verifiability: a voter's client keeps C and the index, and can verify the presence of the commitment and the opening of it in public state. Deterministic: It is a function of public data, accepted proofs, nullifier set, commitments and openings; any party can recompute the tally, and if there is a discrepancy between the announced tally and the recomputed tally then it can be publicly demonstrated. There is no trustworthy tallier to mess with, and the best attack on the count is

censorship or refusing to include ballots – which is simply a chain-level censorship resistance provided by the multi-relayer assumption combined with the direct-submission fall-back of Section III-G.

F. Coercion Resistance Boundary

The scheme is receipt-free in the practical sense with respect to the remote coercer: There is no evidence that the voter could present to the coercer that is transferable and binding as to a recorded ballot; the coercer can tell no difference between a real opening claim and a coerced one without the chain linking the ballot with the voter, which Section IV-C excludes. Against the supervising coercer, who watches the device while it is being cast, there is no scheme that doesn't fail, because any scheme that didn't have re-voting or fake credentials adds exactly the elements that the design is supposed to eliminate. We therefore make the claim of coerce resistance a scoped claim and give it to the thesis phase, and advertise a re-voting extension over the same nullifier algebra to be compatible.

G. Replay, Malleability, and Front-Running

There are three attacks at the transaction level that merit special consideration. Replay across elections fails because eid is a public input to the circuit, which gets bound into the nullifier, and replay across chains fails for the same reason as eid also includes the chain identifier as part of the circuit parameter, which is required by the parameter ceremony. There is proof malleability here, but it doesn't matter – the contract interacts with the public signals, the signals are determined by the statement, and a malleated proof is a proof with the same nullifier, so it is a duplicate and can be rejected. By definition, front-running is safe. In the cast phase, a copied cast operation emits the same cast, because the payload contains no extractable value, and no extra priority to be stolen, from the cast operation, and in the reveal phase, a copy of the reveal operation is a reveal of the same cast to the same candidate, and all other reveals are rejected. The only other grieving path, burning a captured operation's gas without including it, falls back to the censorship path of Section IVE and remains bounded by the honest-relayer assumption and the direct submission fallback.

H. Residual Trust and Failure Modes

Three assumptions remain. One honest contributor is required in the phase two ceremony, which is tempered by public multi-party ceremonies and keys per election. The circuit has to be properly constrained, and protected by the required fuzzing pass [12,13] and by publishing the circuit for independent auditing. The registration authenticity relies on the channel of the institutional identity, which is not within the cryptographic perimeter and is common to all designs based on registrars [2]. This means trust minimization in practice: nothing of privacy or correctness shall depend on the good behavior of any of the parties before or after the election.

I. Analytical Projections

Because this paper specifies rather than implements, the performance results are analytical: every constant is taken from a published 2023 to 2026 measurement, cited at point of use, and composed into projections that double as acceptance thresholds for the thesis implementation. Fig. 5 presents the two projections, and Section V positions them against the systems reviewed in Section II.

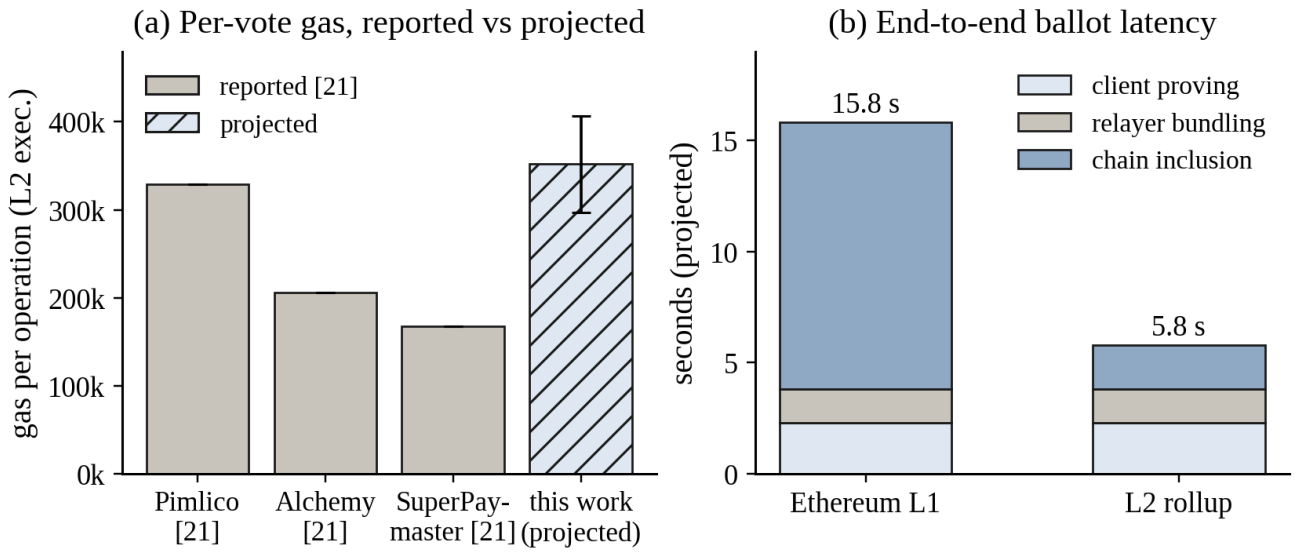

Fig. 5. Analytical projections, derived from published benchmarks rather than new experiments. (a) Projected per-vote gas for the cast user operation against reported ERC-4337 operation costs from [21]; the projection composes verifier, storage, and account-abstraction overhead, with the whisker spanning verifier-cost variants [10], [20]. (b) Projected end-to-end casting latency on Ethereum L1 versus a rollup, composing the 2.3 s device-proving figure reported by zkVoting [7] with bundling and inclusion delays [20], [28].

J. Cost Model

The cast operation pays for one Groth16 verification, whose on-chain cost is constant in circuit size [11] and falls substantially under Poseidon2-oriented verifier optimization, reported at 73 percent verifier-cost reduction on EVM chains [10]; two first-time storage writes, the nullifier flag and the commitment append; and the ERC-4337 envelope, whose overhead relative to a plain transaction Lin et al. measure directly [20] and whose sponsored-execution cost Super Paymaster demonstrates at 167,830 gas of pure layer-2 execution against 205,951 and 328,937 for commercial paymaster baselines [21]. Composing these yields the projected 352,000 gas per ballot shown in Fig. 5(a), with an uncertainty band of roughly 55,000 gas spanning verifier variants; the reveal operation, one hash check and one counter increment inside the same envelope, projects well below it. The deployment recommendation comes straight after, as the thousand-voter election needs around 0.7 billion gas in total at that footprint, both in Layer 1 and in Blob Economics at small-change levels on rollups, which is outpriced by the EIP-1559 fee-dynamics in congested Layer 1 [27] and EIP-4844 Blob Economics at small-change levels on rollups [28]. The on-chain cost per ballot is constant for an electorate, and the electorate registry growth is logarithmic in function of the tree, similarly to the scaling behavior ElSheikh and Youssef cite of circuit-compressed voting [6].

K. Latency Model

The latency of casting is broken down into three parts: device proving, bundling, and inclusion. We assume the 2.3 second balloting figure reported by zkVoting for a similar statement of commodity hardware [7] (as with the constraint-count scaling in [11] and [17] and with the browser penalty for WASM execution versus native execution reported in FibRace [18]): about five seconds on a mid-range phone, and failure to meet this time threshold triggers the "fallback plan" discussed in Section V, which results in the 15.8 and 5.8 second stacks of Fig. 5(b). Tallying is a linear pass over openings – in this case, zkVoting's 3.9 milliseconds per ballot [7]

means it's less than a minute per ballot for the electorate we're targeting here.

V. Discussion

This section provides an interpretation of the projections made in Section II based on the systems under review, outlines the definition of the measurement plan the implementation should meet and specifies the limits of the analytical method.

A. Comparison With Previous Systems

TABLE II
POSITIONING AGAINST RECENT SYSTEMS (REPORTED FIGURES FROM THE CITED SOURCES; LAST ROW IS THIS PAPER'S ANALYTICAL PROJECTION)

System	Proof system	Ballot privacy	Wallet unlink-ability	Client-side proving	Coercion resistance	Reported performance
zkVoting [7]	Groth16 + nullifiable commitment	Yes	Not addressed	Not required	Yes (trusted registrar)	2.3 s cast; 3.9 ms per ballot tally
Dispute-free OVN [6]	Groth16, self-tallying	Yes (round-compressed)	No	Off-chain, unconstrained	No	Deployment approx. USD 12.5; logarithmic on-chain growth
SmartphoneDemocracy [8]	ZKP on EUDI identity	Yes	P2P bulletin board (non-EVM)	Yes (smartphone)	Partial	Smartphone-native; TrustChain infrastructure
DTL voting application [23]	zk commitments (tumbling)	Yes	Yes (composable layer)	Yes	No	Operation under 1.5 s (payments application)
Capstone prototype (baseline)	ZoKrates Groth16	No (plaintext choice)	No (voter wallet sends)	No (server proves)	No	Local Hardhat network only; unbound nullifier
This work (proposed)	**Groth16 + Poseidon, BN254**	**Yes (commit, delayed reveal)**	**Yes (ERC-4337 relayers)**	**Yes (browser WASM)**	**Receipt-free vs. remote coercer**	**Approximate: 352 k gas per ballot, less than 6s on rollup (Fig. 5)**

The positioning is made concrete in -Table II. zkVoting delivers the strongest coercion story at the price of a trusted registrar and unconstrained proving location [7]; the dispute-free open vote network delivers self-tallying economy without addressing wallet linkage [6];

SmartphoneDemocracy proves on the device but off the EVM ecosystem [8]; the tumbling layer delivers composable unlinkability without eligibility-checked balloting [23]; and the capstone prototype, included as the baseline this work repairs, fails all four privacy columns. The proposed design is the only row combining device-confined secrets, relayer unlinkability, in-proof nullifier binding, and hidden ballots on a public EVM chain, which is precisely the integration claim C1 to C4 of Section I, and the comparison also shows what it does not claim, supervision-proof coercion resistance, keeping the contribution honestly bounded.

B. Evaluation Plan and Thresholds

The thesis implementation will measure, against this paper's projections: browser proving time across three device tiers with a five second mid-range threshold [18]; cast and reveal gas on a public testnet and one rollup against the 352,000 gas projection with a 20 percent tolerance [20], [21]; verifier-variant savings against the 73 percent Poseidon2 figure [10]; end-to-end latency against Fig. 5(b); and a circuit assurance pass with the fuzzing tools of [12], [13] reporting zero unresolved constraint findings. Two threshold breaches carry pre-committed responses: if mid-range proving exceeds five seconds, the plan moves to a universal-setup proving system with WASM-friendlier arithmetic at the cost of larger proofs [14], [15]; if rollup per-ballot cost exceeds the tolerance, the plan batches reveals and moves the commitment list into blob space [28].

C. Limitations of the Analytical Method

Composing constants from separately conducted studies carries known hazards, and three deserve disclosure. The gas figures in [20] and [21] were measured under network conditions and entry-point versions that will differ from any future deployment, so the 352,000 gas projection should be read with its stated tolerance rather than as a point estimate. A different hardware or a different witness-generation path was used to reach the 2.3 second proving time in [7] and the actual browser penalty measured in FibRace is indicated, hence the acceptance threshold of five seconds. Conclusions about the fee level are inherited from the EIP-1559 and blob-market conditions discussed in [27] and [28] [45] and both markets are subject to changes; in this sense, the deployment recommendation is structural, that is, it is based on the fact that cheap marginal gas should be used for bursty workloads, not on any given price [29]. The implementation phase is the kind of phase that is created just to replace these composed constants with measurements taken in one coherent environment.

VI. Conclusion

The voting system outlined in this paper overcomes four types of trust failures that plague prototype-class ZK elections: voting moves into the voter's browser, so that no secret is ever sent to the chain, moves are made behind relayers of ERC-4337, so that no voter address reaches the chain, the double-vote nullifier is embedded in the verified proof, and so that the voter is not responsible for replay protection, ballots are sent as hiding commitments, opened after the polls close. The security analysis breaks each promised property down into standard assumptions plus three properties that can be inspected before the election: the analysis projects about 352,000 gas and less than six seconds per ballot on a rollup, which are also thresholds that the implementation must satisfy.

The thesis phase will feature the full stack, execution of the measurement plan of Section V and three extensions: a re-voting mechanism over the same nullifier algebra towards a system that offers supervision-resistant coercion protection in the spirit of [7, 26]; batching and blob-space engineering for cost at larger electorates [28]; and a study of migrating from Groth16, behind the fixed verifier interface, to a lattice-based system, whose electronic-voting instantiations are provided by [29] and the post-quantum landscape by [45]. The desired outcome is that there are no residual trust assumptions that can be checked by anyone except the voter, and that the election will occur at the campus level and take place before any ballots are cast.

References

M.-V. Vladucu, Z. Dong, J. Medina, and R. Rojas-Cessa, "E-voting meets blockchain: A survey," IEEE Access, vol. 11, pp. 23293–23308, 2023.

O. Revelo Sánchez, A. Barón Salazar, and M. Bolaños González, "Democratic innovation: Systematic evaluation of blockchain-based electronic voting (2022–2025)," Technologies, vol. 14, no. 2, Art. no. 95, 2026.

R. Lavin, X. Liu, H. Mohanty, L. Norman, G. Zaarour, and B. Krishnamachari, "A survey on the applications of zero-knowledge proofs," arXiv:2408.00243, 2024.

M. Lupu and I. Aciobăniţei, "Enhanced blockchain-based e-voting system using zero-knowledge proofs," in Proc. 23rd Int. Conf. Informatics in Economy (IE 2024), Smart Innovation, Systems and Technologies, vol. 426, Springer, 2025, pp. 237–246.

F. Rabia, A. Sara, and G. Taoufiq, "Toward a decentralized and anonymous e-voting system using zk-SNARKs," in Advances in Intelligent Systems and Digital Applications (ISDA 2025), Lecture Notes in Networks and Systems, vol. 1486, Springer, 2025.

M. ElSheikh and A. M. Youssef, "Dispute-free scalable open vote network using zk-SNARKs," in Financial Cryptography and Data Security: FC 2022 International Workshops, LNCS, Springer, 2023, pp. 499–515.

S. Park, J. Choi, J. Kim, and H. Oh, "zkVoting: Zero-knowledge proof based coercion-resistant and E2E verifiable e-voting system," Cryptology ePrint Archive, Paper 2024/1003, 2024.

M. Jóźwik and J. Pouwelse, "SmartphoneDemocracy: Privacy-preserving e-voting on decentralized infrastructure using novel European identity," arXiv:2507.09453, 2025.

A. Burgos and E. Alchieri, "Privacy-preserving smart contracts for permissioned blockchains: A zk-SNARK-based recipe (Part 1)," arXiv:2501.03391, 2025.

H. Guo, Y. Feng, C. Wu, Z. Li, and J. Xu, "Benchmarking ZK-friendly hash functions and SNARK proving systems for EVM-compatible blockchains," arXiv:2409.01976, 2024.

O. Kuznetsov et al., "Performance analysis of Groth16 zkSNARK: Systematic benchmarking with circom-snarkjs," International Journal of Computing, vol. 24, no. 4, pp. 645–660, 2025.

S. Chaliasos, I. Al-Fath, and A. Donaldson, "Towards fuzzing zero-knowledge proof circuits (short paper)," arXiv:2504.14881, 2025.

H. Takahashi, J. Kim, S. Jana, and J. Yang, "zkFuzz: Foundation and framework for effective fuzzing of zero-knowledge circuits," arXiv:2504.11961, 2025.

J. Liang, D. Hu, P. Wu, Y. Yang, Q. Shen, and Z. Wu, "SoK: Understanding zk-SNARKs: The gap between research and practice," in Proc. 34th USENIX Security Symposium, 2025.

N. Sheybani, A. Ahmed, M. A. Kinsy, and F. Koushanfar, "Zero-knowledge proof frameworks: A systematic survey," arXiv:2502.07063, 2025.

M. Christ, F. Baldimtsi, K. K. Chalkias, D. Maram, A. Roy, and J. Wang, "SoK: Zero-knowledge range proofs," in Proc. 6th Conference on Advances in Financial Technologies (AFT 2024), LIPIcs, 2024.

R. Palahusynets et al., "Performance evaluation of zk-SNARK protocols for privacy-preserving sensor data verification: A systematic benchmarking study," Sensors, vol. 26, no. 8, Art. no. 2486, 2026.

KKRT Labs and Hyli, "FibRace: A large-scale benchmark of client-side proving on mobile devices," arXiv:2510.14693, 2025.

G. C. Ismayilov, "zQR: A verifiable QR-driven zkSNARK proof verification framework for mobile platforms," arXiv:2606.27092, 2026.

Z. Lin, T. Wang, C. Zhao, S. Zhang, Q. Yang, and L. Shi, "A measurement investigation of ERC-4337 smart contracts on Ethereum blockchain," in Proc. International Conference on Computing, Networking and Communications (ICNC), 2024, pp. 1164–1170.

H. Jiao and N. Udomlertsakul, "SuperPaymaster: Eliminating centralized signer authority via asset-oriented abstraction to reconcile usability and decentralization in account abstraction," arXiv:2605.05774, 2026.

F. Baldimtsi et al., "SoK: Privacy-preserving transactions in blockchains," Cryptology ePrint Archive, Paper 2024/1959, 2024.

M. Minaei, P. Moreno-Sanchez, Z. Fang, S. Raghuraman, N. Alamati, P. Chatzigiannis, R. Kumaresan, and D. V. Le, "DTL: Data tumbling layer: A composable unlinkability for smart contracts," in Proc. 20th ACM Asia Conference on Computer and Communications Security (AsiaCCS), 2025, doi: 10.1145/3708821.3736221.

Y. Wu et al., "Investigating Tornado Cash: Empirical insights into mixing service anonymity," Springer, 2025, doi: 10.1007/978-981-95-3480-7_9.

K. Huseynov, A. Shahzaib, I. A. Seres, and J. Tapolcai, "A tattered cloak of invisibility: Measuring anonymity loss in Railgun on Ethereum," arXiv:2606.25926, 2026.

R. Giustolisi, M. S. Garjan, and C. Schuermann, "Thwarting last-minute voter coercion," in Proc. IEEE Symposium on Security and Privacy (SP), 2024, pp. 3423–3439.

T. Roughgarden, "Transaction fee mechanism design for the Ethereum blockchain: An economic analysis of EIP-1559," Journal of the ACM, vol. 71, no. 4, pp. 1–63, 2024.

N. Dyade, "Cost optimization in Layer 2 rollups via EIP-4844: A gas efficiency and economic analysis," IET Blockchain, 2025, doi: 10.1049/blc2.70014.

V. Farzaliyev, C. Pärn, H. Saarse, and J. Willemson, "Lattice-based zero-knowledge proofs in action: Applications to electronic voting," Journal of Cryptology, vol. 38, no. 1, Art. no. 6, 2025.

M. D. Rasheed, K. Hamid, M. W. Iqbal, M. W. Iqbal, M. Ibrar, and M. A. Khan, "Secure evolutionary model empowered smart homes in AI environment: An efficient way of life," Proc. Korean Next Generation Computing Society Conf., pp. 254–257, Dec. 2025.

K. Hamid, M. W. Iqbal, M. Aqeel, X. Liu, and M. Arif, "Analysis of Techniques for Detection and Removal of Zero-Day Attacks (ZDA)," in Ubiquitous Security, G. Wang, K.-K. R. Choo, J. Wu, and E. Damiani, Eds., Singapore: Springer Nature, 2023, pp. 248–262. doi: 10.1007/978-981-99-0272-9_17..

K. Hamid, F. Naeem, M. Ibrar, A. M. Delshadi, F. Ahmed, M. Aamir, and G. Ahmad, "Behavior and characteristics of ransomware – A survey," Lahore, Pakistan, [Publication details to be verified].

K. Hamid, M. W. Iqbal, M. Aqeel, T. A. Rana, and M. Arif, "Cyber security: Analysis for detection and removal of zero-day attacks (ZDA)," in Artificial Intelligence & Blockchain in Cyber Physical Systems. Boca Raton, FL, USA: CRC Press, 2023.

S. Riaz et al., "Software development empowered and secured by integrating a DevSecOps design," Journal of Computing & Biomedical Informatics, p. 02, Mar. 2025, doi: 10.56979/802/2025.

M. Ibrar et al., "Econnoitering data protection and recovery strategies in the cyber environment: A thematic analysis," Int. J. Electronic Crime Investigation, vol. 8, Dec. 2024, doi: 10.54692/ijeci.2024.0804216.

K. Khaliq, N. Rahim, K. Hamid, M. Ibrar, U. Ahmad, and M. Ullah, "Ransomware attacks: Tools and techniques for detection," 2024, doi: 10.1109/ICCR61006.2024.10532926.

Z. Zafar, K. Hamid, M. Kafayat, M. W. Iqbal, Z. Nazir, and A. Ghani, "AI-based cryptographical framework empowered network security," Journal of Jilin University (Engineering and Technology Edition), vol. 42, pp. 497–510, Apr. 2023, doi: 10.17605/OSF.IO/W69VT.

K. Hamid et al., "Empowering robust security measures in Node.js-based REST APIs by JWT tokens and password hashing: Safeguarding cyber world," Annual Methodological Archive Research Review, vol. 3, May 2025, doi: 10.63075/w2nam443.

M. Delshadi et al., "Empowerment of artificial intelligence (AI) in preventing and detecting ransomware: An analytical review," Spectrum of Engineering Sciences, vol. 3, no. 9, pp. 36–48, Sep. 2025.

M. Delshadi, M. Zubair, K. Hamid, and M. W. Iqbal, "Preventing and detecting malicious activities during phishing attacks using AI-based sandbox bypass," Annual Methodological Archive Research Review, vol. 3, no. 8, pp. 249–261, Aug. 2025, doi: 10.63075/fahtkz35.

M. N. Ahmad, M. N. Amin, K. Hamid, S. M. Rizwan, and S. A. A. Naqvi, "Enhanced IoT network security for network intrusion detection model based on machine learning technique," Annual Methodological Archive Research Review, vol. 3, no. 8, Aug. 2025, doi: 10.63075/0vcg0093.

M. H. Akhtar, T. Jabeen, R. Aziz, M. N. Amin, S. M. Rizwan, and K. Hamid, "Intelligence-based self-healing network design: An automated incident response system for troubleshooting of IoT security breaches," Annual Methodological Archive Research Review, vol. 3, no. 8, Aug. 2025, doi: 10.63075/6dhhj119.

P. Tahir, P. K. Hamid, P. D. A. Kahloon, and P. S. Zubair, "Cyber sovereignty challenges: A strategic framework for national data protection using blockchain authentication," Contemporary Journal of Social Science Review, vol. 3, no. 3, pp. 1314–1328, Aug. 2025, doi: 10.63878/cjssr.v3i3.1100.

W. Aslam, W. Tariq, T. Waheed, K. Hamid, S. M. Rizwan, and A. Ahmed, "Enhancing privacy and security in multi-party computation for data mining in cryptographically protected environments," Annual Methodological Archive Research Review, vol. 3, no. 8, pp. 510–531, Aug. 2025, doi: 10.63075/fxe5gg65.

R. Steinfeld, M. F. Esgin et al., "Post-quantum zero-knowledge proofs and applications," in Proc. 10th ACM Asia Public-Key Cryptography Workshop (APKC), 2023.