IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

Anonymous and Verifiiable Decentralized Voting Using Zero-Knowledge Proofis: A Groth16 and Merkle-Membership Implementation on Ethereum

Om Gajanan Wakchaure, Akash Ravindra Dongare, Vishal Dipak Bhalekar, Ruday Kailas Gadekar,
Guide: M.S. Dighe, Co-Guide: A. R. Kardile
Department of Computer Engineering
Adsul's Technical Campus, Chas, Ahilyanagar, India
Savitribai Phule Pune University (SPPU), Maharashtra, India

Abstract: *Electronic voting systems must reconcile two conflicting goals: ballots must be anonymous, yet the tally must be publicly verifiable. Conventional systems resolve this by trusting a central authority, which becomes a single point of failure and censorship. This paper presents a decentralized voting system that uses zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) to provide both properties. Eligible voters are registered as Poseidon commitments in a depth-20 Merkle tree whose root is published on an Ethereum-compatible blockchain. To vote, a participant generates a Groth16 proof that they know a secret whose commitment is a leaf of the tree, without revealing which leaf. The proof also binds a per-poll nullifier, which a smart contract records to prevent double voting. We describe the circuit design, the contract logic, and the registration pipeline, and we derive circuit size, proof size, and on-chain verification cost analytically. We analyze the system's security properties and discuss its limitations, including trusted-setup assumptions and the absence of receipt-freeness. The design is documented in enough detail for independent reproduction.*

Keywords: zero-knowledge proofs, zk-SNARK, Groth16, blockchain voting, Merkle tree, Poseidon hash, nullifier, Ethereum, anonymity

I. INTRODUCTION

Voting is a canonical example of a protocol in which privacy and integrity pull in opposite directions. Voters need assurance that their ballot is secret and cannot be linked to them. Observers need assurance that only eligible voters voted, that each voted at most once, and that the published tally matches the cast ballots. Paper-based elections achieve this through procedural controls and trusted officials. Electronic systems often move the trust to a server operator, whose database can be altered or inspected.

Blockchains offer a public, append-only ledger that removes the need to trust a single operator for storage and tallying. However, a naive on-chain vote is linked to the voter's account address, which defeats ballot secrecy. Zero-knowledge proofs address this gap: a voter can prove a statement about their eligibility and uniqueness without revealing their identity.

This paper makes the following contributions:

1. A complete design for anonymous, one-person-one-vote ballots using Merkle-membership proofs and per-poll nullifiers, implemented in Circom with Groth16 proofs verified on-chain.
2. An open implementation comprising the circuit, Solidity contracts, a registration tool, and a web client.
3. An analytical evaluation of circuit size, proof size, and on-chain gas cost, derived from the constraint structure and the EVM pricing rules.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

4. A security analysis that states the properties achieved and the ones that are not, notably receipt-freeness and coercion resistance.

We do not claim a new cryptographic primitive. The construction follows the identity-commitment and nullifier pattern used by existing anonymous-signaling protocols such as Semaphore [3]. Our contribution is an end-to-end implementation and evaluation applied to a voting scenario. Specific to our design are poll-scoped nullifiers, which let one voter registry serve many elections, and a range constraint that binds every ballot to a valid candidate index.

II. BACKGROUND

2.1 zk-SNARKs and Groth16

A zk-SNARK lets a prover convince a verifier that they know a witness w satisfying a public relation, with a proof that is short and fast to verify and reveals nothing about w. Groth16 [1] produces constant-size proofs (three group elements) and verification cost that is nearly independent of circuit size, which makes it attractive for on-chain use where verification gas is the dominant cost. Its drawback is a circuit-specific trusted setup: if the setup randomness ("toxic waste") is retained by anyone, that party can forge proofs.

2.2 Poseidon hash

Hash functions designed for CPUs, such as SHA-256, require many constraints inside an arithmetic circuit. Poseidon [2] is an algebraic hash designed for proof systems and uses far fewer constraints per hash, which keeps the Merkle-path verification small.

2.3 Circom and snarkjs

Circom [4] is a domain-specific language for describing arithmetic circuits, compiled to an R1CS constraint system. snarkjs generates proving and verification keys, produces proofs in JavaScript or WebAssembly, and exports a Solidity verifier contract.

2.4 Nullifiiers

A nullifier is a deterministic value derived from a voter's secret and a public context (here, the poll identifier). It is revealed along with each proof. The same voter and poll always produce the same nullifier, so a contract can reject a repeat. Different polls produce unlinkable nullifiers, and the nullifier does not reveal which leaf it came from.

III. RELATED WORK

Helios [5] is a web-based, open-audit voting system with end-to-end verifiability, but it relies on trustees and a central server. Open Vote Network [6] implements a self-tallying boardroom vote on Ethereum with maximum voter privacy, but its cost grows with the number of voters and it requires interaction among all voters, which limits scale. Zerocash [7] introduced zk-SNARK-based anonymous payments, whose commitment and nullifier structure underlies many later designs. Semaphore [3] provides a general protocol for anonymous group membership and signaling and is the closest existing primitive to our construction. MACI [8] adds coordinator-based collusion resistance to on-chain voting at the cost of trusting the coordinator for privacy. Our system sits between these approaches: it is non-interactive and constant-cost to verify, and it requires no tallying trustee, but it does not provide the coercion resistance that MACI targets.

IV. SYSTEM DESIGN

4.1 Threat model and goals

We consider an adversary who can observe all on-chain data and network traffic, can submit arbitrary transactions, and can attempt to vote without being registered or to vote more than once. We assume the registration authority correctly vets eligibility, and that the trusted setup was performed honestly by at least one participant.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

Goals:

- Eligibility: only registered voters can cast a counted ballot.
- Uniqueness: each registered voter can cast at most one ballot per poll.
- Ballot secrecy: a ballot cannot be linked to the voter's identity, the leaf they registered, or their transaction sender.
- Public verifiability: anyone can check each proof and recompute the tally from on-chain events.
- Individual verifiability: a voter can confirm their nullifier is recorded.

4.2 Actors and filow

The system has three roles: an administrator who opens polls and publishes Merkle roots, voters, and an off-chain registration service that stores the voter list and serves each voter's Merkle path.

1. Registration. Each voter generates two secrets locally, voterSecret and trapdoor, and submits only the commitment C = Poseidon(voterSecret, trapdoor) to the registrar. The registrar verifies eligibility, inserts C into the Merkle tree, and the administrator publishes the new root to the contract.

2. Voting. The voter retrieves their Merkle path for C, then in their own browser generates a Groth16 proof. The proof and the public signals are submitted to the contract, optionally through a relayer so that the transaction sender is also unlinked from the voter.

3. Tally. The contract verifies the proof, checks that the nullifier is unused, records it, and increments the chosen candidate's count.

4.3 Circuit

The circuit has depth d = 20, supporting up to 2^20 (about 1.05 million) voters.

Private inputs: voterSecret, trapdoor, and the Merkle path (sibling hashes and left or right indicators).

Public inputs: merkleRoot, nullifierHash, pollId, and candidateId. The circuit enforces four constraints:

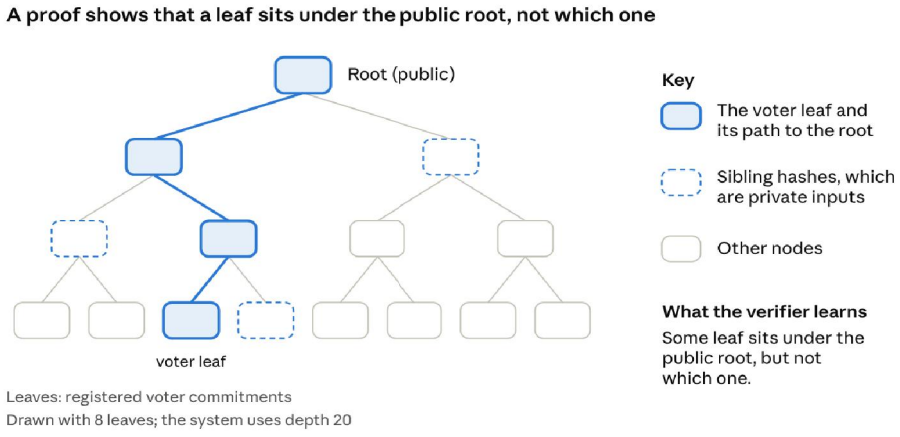
1. leaf = Poseidon(voterSecret, trapdoor).

2. The Merkle path from leaf hashes up to merkleRoot. At each level the path indicator selects the left and right child order using an arithmetic selection, because circuits cannot branch.

3. nullifierHash = Poseidon(voterSecret, pollId).

4. candidateId < numCandidates, enforced with a comparator.

Binding pollId and candidateId as public inputs prevents a proof generated for one poll or candidate from being replayed for another. Fig. 1 shows how the Merkle path links a voter's leaf to the public root.



Merkle membership proof · 8 leaves, 1 path

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

Fig. 1. Merkle membership: the voter's leaf and its path to the root (solid) with the sibling hashes (dashed) that are supplied as private inputs.

4.4 Smart contracts

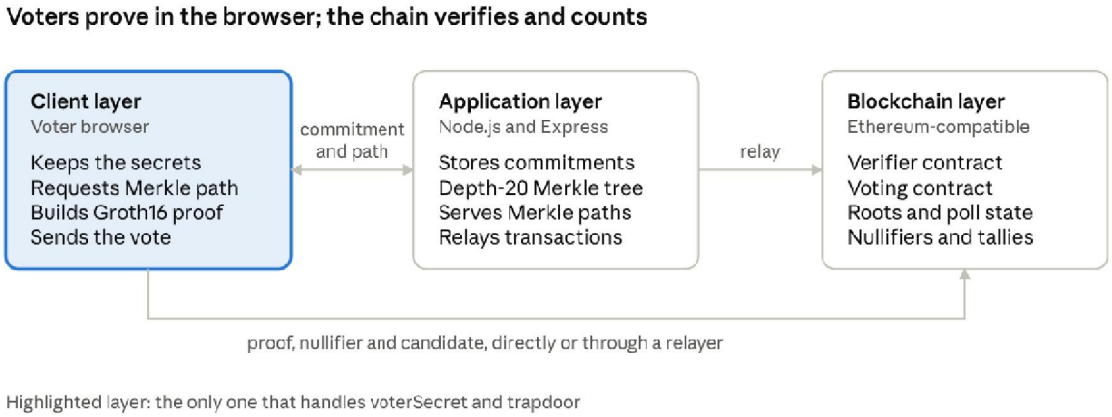
The verifier contract is generated by snarkjs from the final proving key. The voting contract stores the set of valid Merkle roots, the nullifier set per poll, and per-candidate tallies. The function rejects the call if the poll is closed, the candidate is out of range, the root is not recognized, or the nullifier has been spent. Otherwise it calls, then marks the nullifier spent and increments the tally. Because authorization comes from the proof and not from relayer, can submit it. any account, including a

4.5 Trusted setup

Circuit-specific parameters were produced with a two-phase Powers of Tau ceremony followed by a Groth16 circuit-specific contribution, both run locally for this prototype. A local single-party ceremony gives no meaningful trust guarantee, so a production deployment would reuse a public multi-party Powers of Tau transcript and run its own multi-party second phase.

4.6 System architecture and data filow

The system is organized in three layers that communicate only through narrow interfaces. Fig. 2 summarizes the three layers and the interfaces between them.



system architecture · 3 layers

Fig. 2. System architecture: the client layer holds the secrets and builds the proof, the application layer serves Merkle paths and relays transactions, and the contracts verify proofs and count ballots.

Client layer. The voter's browser holds the voter's secrets, requests a Merkle path, builds the Groth16 proof using the circuit's WebAssembly witness generator and proving key, and sends the vote transaction. In the intended design the secrets never leave this layer.

Application layer. A Node.js and Express backend stores the registry of voter commitments, maintains the depth-20 Poseidon Merkle tree, returns the Merkle path for a given commitment, and relays transactions. It works only with commitments, so it has no need for a voter's secret.

Blockchain layer. Two contracts run on an Ethereum-compatible network. The verifier contract is generated from the circuit's verification key and checks proofs with the pairing precompiles. The voting contract stores the accepted Merkle roots, the poll state, the set of spent nullifiers per poll, and the tally for each candidate.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

Data placement follows from the threat model. The commitment list, the Merkle tree and the Merkle paths are kept off-chain, because only the root is needed for verification.

Roots, nullifiers, tallies and events are kept on-chain, because they must be publicly auditable. The voter trusts only the client layer for secrecy. The backend is trusted for availability and for eligibility decisions, but not for secrecy or for the tally. The blockchain is trusted for integrity.

The end-to-end flow has seven steps:

1. Setup. The circuit is compiled, the trusted setup produces the proving and verification keys, and the verifier contract is exported and deployed.
2. Registration. Each voter generates voterSecret and trapdoor locally and submits only the commitment. The backend inserts it into the Merkle tree.
3. Poll creation. The administrator publishes the new Merkle root to the voting contract and opens a poll with its list of candidates.
4. Proof generation. The voter retrieves the Merkle path for their commitment and generates a proof for the chosen candidate and poll.
5. Submission. The proof, the nullifier and the candidate are sent to the voting contract, directly or through a relayer.
6. Verification and tally. The contract checks the poll state, the candidate range, the root and the nullifier, verifies the proof, records the nullifier, and increments the tally.
7. Audit. Anyone can read the events and the tallies, and a voter can confirm that their nullifier is recorded without revealing their identity.

Fig. 3 shows the same flow as a sequence of messages.

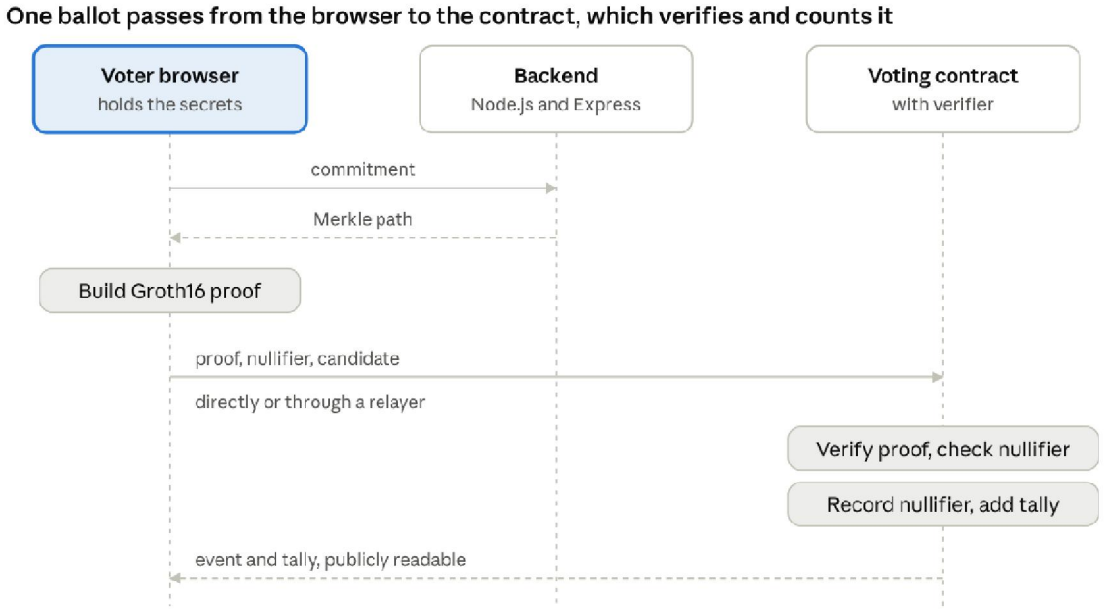


Fig. 3. Message sequence for one voter: registration, Merkle path retrieval, proof generation, submission, and on-chain verification and tally.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

V. IMPLEMENTATION

The circuit is written in Circom 2 with circomlib's Poseidon and comparator templates. Proofs are generated and verified with snarkjs using WebAssembly witness generation. The contracts are written in Solidity and tested with Hardhat on a local EVM network. The web client uses ethers.js and runs proof generation in the browser, so voter secrets never leave the voter's device. A Node.js backend serves Merkle paths and relays transactions.

VI. EVALUATION

The evaluation is analytical. It derives the size and cost of each component from the circuit structure and from the pricing rules of the Ethereum Virtual Machine, so every figure below can be reproduced by inspection.

6.1 Circuit size

The circuit evaluates one two-input Poseidon hash per tree level, one for the leaf commitment, and one for the nullifier, which makes 22 hash evaluations. The circomlib implementation of a two-input Poseidon hash uses about 240 R1CS constraints, so the hashes account for roughly 5,300 constraints. The ordering logic adds two multiplications per level, the boolean checks on the path indicators add one constraint per level, and the candidate range check adds a few more. The total is therefore roughly 5,400 constraints and grows linearly with tree depth. The compiler reports the exact count for the compiled circuit. A SHA-256 hash costs tens of thousands of constraints per compression, which is why Poseidon is used.

6.2 Proofi size and proving cost

A Groth16 proof holds two G1 points and one G2 point on the BN254 curve, which is 256 bytes uncompressed whatever the circuit size. Proving time grows roughly linearly with the number of constraints. A circuit of a few thousand constraints is small by current standards and is expected to be proved in a few seconds at most in a browser using WebAssembly. The proving key that the browser downloads scales with the same constraint count.

6.3 On-chain cost

Verification performs one pairing check over four pairs. Under EIP-1108 this precompile costs 34,000 gas per pair plus 45,000 gas, or 181,000 gas. The verifier also combines the four public inputs with the verification key using elliptic-curve multiplications and additions at about 6,150 gas per input, around 24,600 gas in total. Verification therefore costs approximately 205,000 gas before contract overhead. Recording the ballot adds a storage write for the nullifier (about 22,100 gas for a new slot), an update to the candidate tally, the event log, and the 21,000 gas base transaction cost. A valid ballot thus costs on the order of 300,000 gas. None of this depends on the number of registered voters, because verification touches a constant number of storage slots. In our contract the nullifier check precedes proof verification, so a replayed ballot is rejected before any pairing is computed and costs far less than a valid one.

6.4 Functional validation

The test plan covers six cases, each run against a local EVM network with real proofs. A valid ballot is accepted and increments the tally. A second ballot with the same nullifier reverts. A proof whose candidate value was altered after generation fails verification. A proof against an unrecognized Merkle root is rejected. A ballot submitted after the poll closes reverts. A proof generated for a different poll fails, because the poll identifier is a public input.

VII. SECURITY ANALYSIS

7.1 Properties achieved

Eligibility. Soundness of Groth16 means a prover cannot produce a valid proof without knowing a secret whose commitment is a tree leaf, except by breaking the proof system or the hash's collision resistance.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

Uniqueness. The nullifier is a deterministic function of the secret and poll, so a second ballot from the same voter produces the same nullifier and is rejected by the contract.

Ballot secrecy. The proof is zero-knowledge, so it reveals nothing about the leaf index or secrets beyond the public signals. This holds only if secrets are generated and used on the voter's device. A system that sends secrets to a server for proof generation lets that server link voters to ballots, so client-side proving is a requirement of the design, not an optimization.

Public verifiability. All proofs, nullifiers, and tallies are on-chain, so anyone can re-verify the tally.

7.2 Limitations and attacks

Trusted setup. Retained toxic waste would allow forged proofs and therefore ballot stuffing, though not deanonymization.

No receipt-freeness or coercion resistance. The nullifier and candidate are published together, so a voter who discloses their secret can prove how they voted. This enables vote buying and coercion. Systems such as MACI [8] address this with additional trust assumptions. Receipt-freeness [9] is a known open requirement for blockchain voting.

Network-level linkage. Submitting from a personal wallet links the transaction sender to the ballot. A relayer or fresh address is needed, and timing analysis may still reveal information.

Registrar trust. The registrar decides who is eligible, and could register fake voters. Auditable registration or identity-linked credentials could mitigate this.

Small anonymity sets. Anonymity is bounded by the number of registered voters, and timing or turnout patterns can shrink the effective set.

Secret loss or theft. A lost secret means a lost vote, and a stolen secret allows another party to cast the vote first.

Public tally during voting. Intermediate tallies are visible, which may influence later voters. A commit-and-reveal or end-of-poll tally could address this.

VIII. DISCUSSION AND FUTURE WORK

Potential extensions include replacing the Groth16 setup with a universal-setup or transparent proof system such as PLONK or STARK-based schemes, adding coercion resistance through re-voting or a MACI-style coordinator, moving to a layer-2 network to reduce cost, supporting ranked-choice or weighted ballots, and integrating verifiable credentials for registration. A formal verification of the circuit constraints, for example with automated under-constrained-circuit analysis tools, would also raise assurance.

IX. CONCLUSION

We presented a zk-SNARK-based voting system that provides eligibility, uniqueness, ballot secrecy, and public verifiability on a public blockchain without a tallying trustee. Each ballot carries a constant-size 256-byte proof and costs on the order of 300,000 gas to verify and record, independent of the number of voters, while proving cost grows linearly with tree depth. The main open problems are receipt-freeness and removing the trusted setup, and we outline directions for both.

REFERENCES

[1] J. Groth, "On the size of pairing-based non-interactive arguments," in EUROCRYPT 2016, 2016.

[2] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in USENIX Security Symposium, 2021.

[3] Privacy and Scaling Explorations, "Semaphore protocol," documentation and source code. Available online at semaphore.pse.dev.

[4] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, and J. Baylina, "Circom: A circuit description language for building zero-knowledge applications," IEEE Transactions on Dependable and Secure Computing, 2023.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

[5] B. Adida, "Helios: Web-based open-audit voting," in USENIX Security Symposium, 2008.
[6] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in Financial Cryptography and Data Security, 2017.
[7] E. Ben-Sasson et al., "Zerocash: Decentralized anonymous payments from Bitcoin," in IEEE Symposium on Security and Privacy, 2014.
[8] V. Buterin, "Minimal anti-collusion infrastructure," Ethereum Research forum post, 2019. Available online at ethresear.ch.
[9] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections," in ACM STOC, 1994.