Scalability of Decentralized E-Voting on Blockchain

Jaroslav Podmajerský, xpodmaj00@stud.fit.vutbr.cz

Supervisor: doc. Ing. Ivan Homoliak, Ph.D.

Excel@FIT 2026

Goals:

- Design a secure protocol capable of handling nationwide elections
- Build a system derived from the protocol
- Test & analyse the cost, speed and security of the system

Design & System details:

- Protocol was inspired and built upon existing BBB-voting scheme with Z-Cash ispired transactions thus earning the name **ZBBB-voting**

Protocol overview:

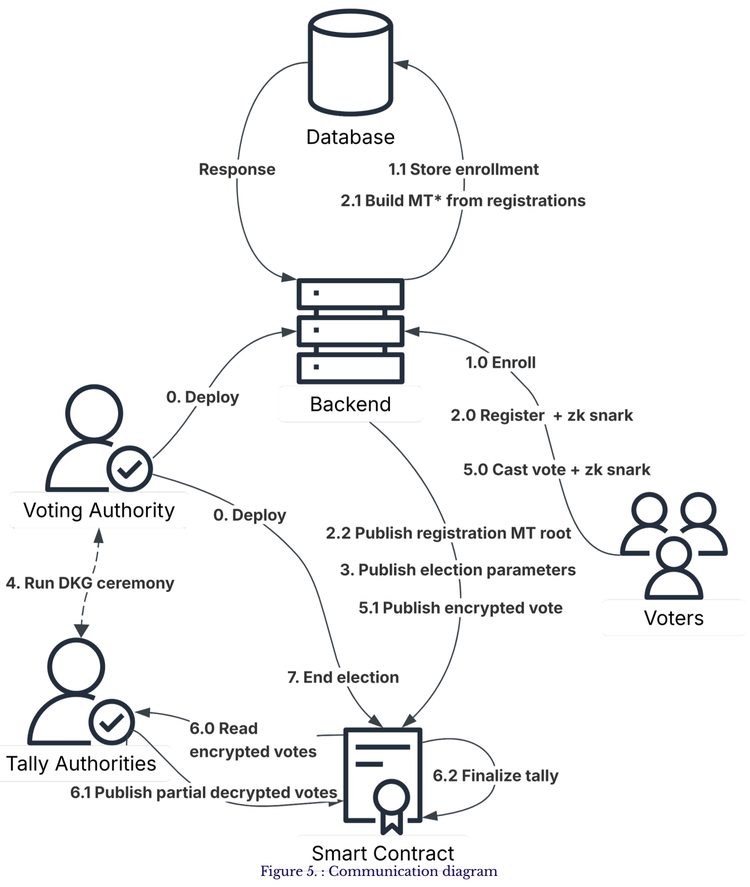

Figure 5. : Communication diagram

Phase	Goal	Entities
0. Deployment	Deploy the off-chain backend and the on-chain smart contract so the election infrastructure is ready to accept participants.	Voting Authority, Backend, Smart Contract
1. Enrollment & Registration	Enroll eligible identities into the backend and let voters register anonymously on-chain with a zk-SNARK; the Merkle tree of registrations is then frozen and published.	Voter, Voting Authority, Backend, Smart Contract
2. Setup & Pre-voting	Prepare the election parameters and run the Distributed Key Generation (DKG) ceremony so tally authorities jointly hold the decryption key without any single party knowing it.	Voting Authority, Tally Authority, Smart Contract
3. Voting	Voters submit ElGamal-encrypted ballots together with a zk-proof of eligibility; the contract stores the vote and its nullifier to prevent double-voting.	Voter, Smart Contract
4. Tally	Tally authorities submit their decryption shares; the secret key is reconstructed, ballots are decoded and the final result is published on-chain.	Tally Authority, Smart Contract

Figure 4. : Phase Table

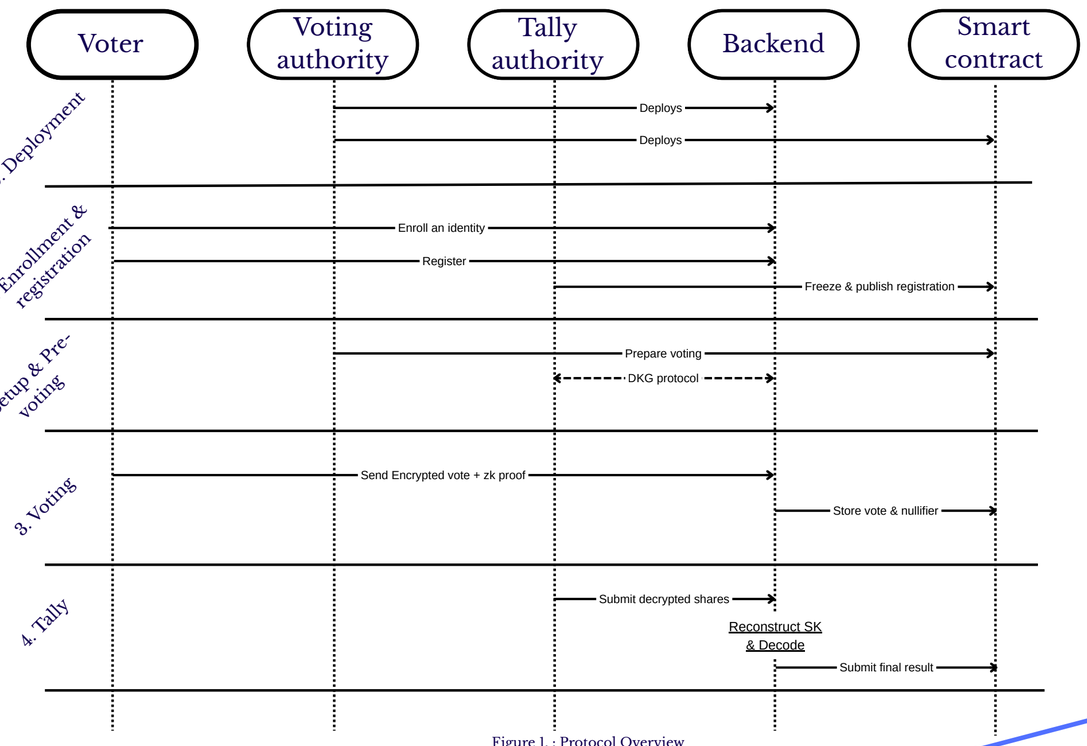

Figure 1. : Protocol Overview

Tech Stack:

- **Backend:**
  - **Python REST API**
  - **Redis**
  - **PostgreSQL**
- **Blockchain**
  - **Solidity + Hardhat**
  - **Python WEB3**
- **Infrastructure**
  - **Docker + Alembic**

Cryptographic primitives:

- **Threshold ElGamal**: Ballots encrypted under a joint public key; decryption requires **t-of-n** tally authorities, tolerating up to **t–1** corruptions.
- **Pedersen DKG:** distributed generation of the election key among tally authorities. No single party learns the private key.
- **Nullifiers:** Deterministic one-time tokens derived via **Poseidon hash** that prevent double-voting while preserving voter anonymity.
- **Groth16 zk-SNARKs: P**rove voter eligibility and ballot validity without revealing identity.
- **Poseidon HASH:** zk-SNARK friendly hash function (alternative to SHA-256).
- **Incremental Merkle Tree:** Efficient membership proofs for enrollment & registration.
- **EIP-712 typed signature:** Wallet-based authentication for initial enrollment

Security properties:

- Votes are anonymous and unlinkable to voter identity.
- The system verifies voting eligibility via **zero-knowledge proofs** without revealing voter identity.
- No single party can compromise the tally. Decryption requires a majority of tally authorities.
- All election data and results are publicly verifiable on the blockchain.

Comparison to real-world elections:

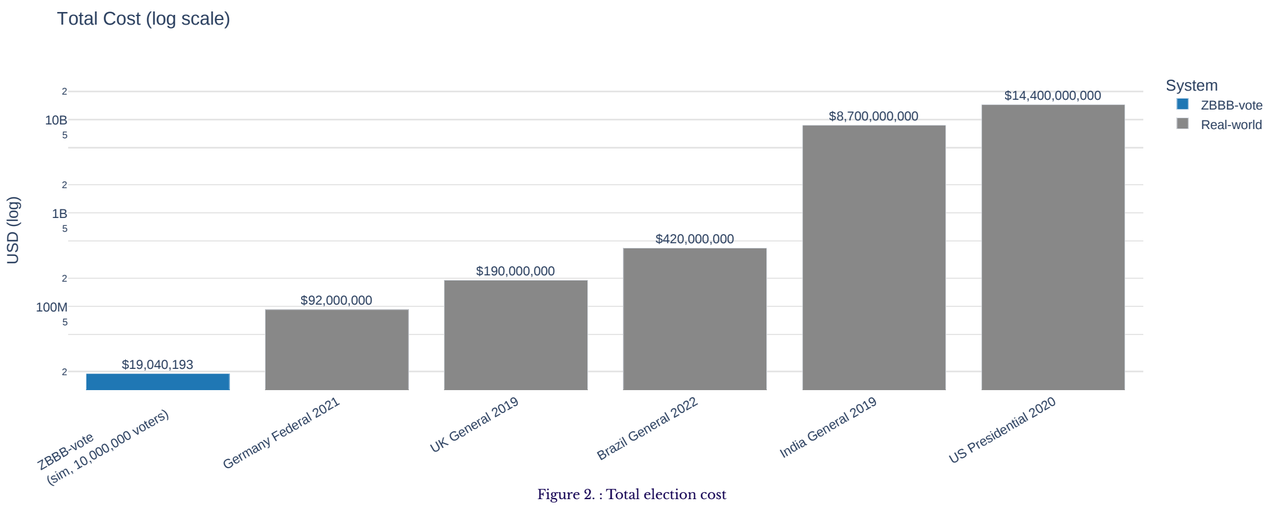

Figure 2. : Total election cost

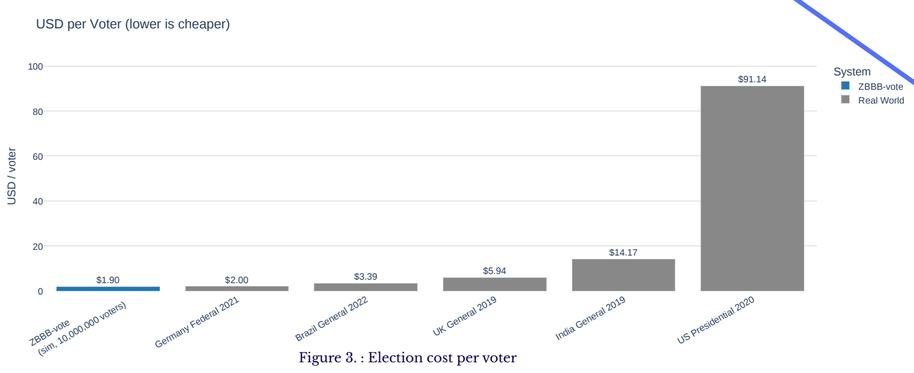

Figure 3. : Election cost per voter

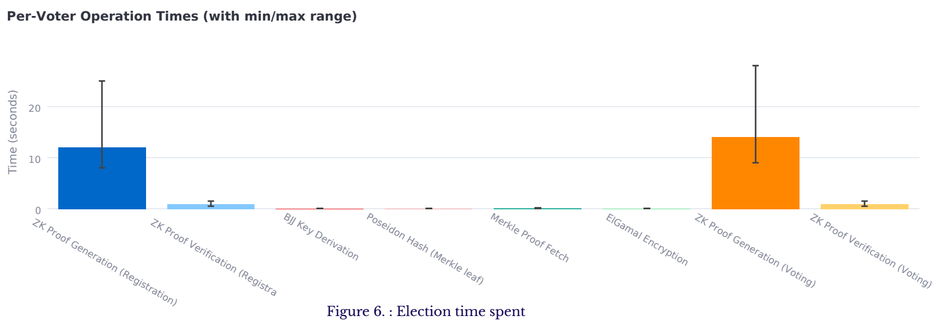

Figure 6. : Election time spent