☑ VoteSecure

📦 **Version: Draft v0.9**

📅 **Date: 19 October 2025**

👤 **Author: Patrick Sawyer (psawyer@gmx.net)**

📄 **License: MIT**

🔗 **Repository: github.com/psawyerberlin/votesecure**

🌐 **Live Site: https://votesecure.net**

🎯 **Intended Audience:** Anyone running, auditing, or participating in secure and verifiable elections or polls.

Executive Summary

VoteSecure is a decentralized application (dApp) for running transparent, tamper-evident, and privacy-preserving elections and polls. It leverages the Nervos CKB (Common Knowledge Base) blockchain for public verifiability while keeping voter choices confidential. The application is open-source and designed so results can be independently audited without trusting a central authority.

Key properties

- **Integrity & verifiability:** Every ballot and tally is on-chain and auditable.

- **Voter privacy:** Ballots are encrypted; individual choices are never exposed.

- **Configurable access & anonymity:** Public, invite-key, or curated voter lists; per-group or aggregate result disclosure.

- **Organizer-sponsored gas:** Organizers fund transaction fees so voting remains frictionless for voters.

Terminology: We use **Organizer** for the party setting up an election and **Voter** for participants using a JoyID wallet. "Event" refers to a poll or election instance.

Problem & Goals

Traditional online polling lacks credible auditability and often compromises privacy. Blockchains offer immutability, but naïve designs can leak voter identities or enable coercion. VoteSecure aims to:

1. Provide end-to-end verifiability without exposing who voted for what.

2. Keep the user flow simple for voters (web + JoyID) while allowing flexible organizer configuration.

3. Prevent tampering, double counting, and post-hoc manipulation with on-chain proofs.

Non-goals

- Identity verification beyond wallet possession (optional integrations can strengthen this).

- Censorship resistance against powerful network-level adversaries (out of scope for MVP).

System Overview

VoteSecure consists of a web front end, an event configuration and publishing workflow for organizers, and a set of on-chain CKB cells that store event metadata, ballots, and results. JoyID provides key management and transaction signing for voters and organizers.

Roles

Organizer

Creates and funds an event, defines questions and answer options, sets eligibility rules, controls visibility of intermediate statistics, and manages result disclosure.

Voter

Authenticates with JoyID, casts a ballot, optionally updates it within organizer-defined limits, and later audits inclusion of their ballot and final tallies.

Organizer Workflow

1. **Create Event**
   - Define **title** and **description** (supports Markdown; limited, sanitized HTML only— scripts and event handlers are stripped to prevent injection).
   - Configure **questions** (single- or multi-select), **answer options**, and **allowed updates per voter**.
   - Set **schedule**: start, end, and (optional) results release time.
   - Set **eligibility**: public, invite-key, per-voter keys, or vetted list (emails used only for distribution; addresses never stored on-chain).
   - Choose **anonymity level** and **reporting granularity** (totals only; by group; or fully public per-ballot *anonymized* view).
   - Define **grouping fields** (e.g., ZIP, city, class, cohort); set **minimum group size** for k-anonymity.

2. **Publish Event**
   - Connect **JoyID** as the sponsoring wallet.
   - Fund and broadcast the event. The app creates:
     - **Lockscript Cell** (sponsor & fee escrow)
     - **Metadata Cell** (event parameters + code hash)
     - **Result Cell** (initialized, locked until release)

o   Receive the **event URL/QR** and **invite material** (invite-key or voter-specific keys).

3. **Distribute Invites**

   o Send URL/QR + invite-keys or voter keys via secure channels (email, SMS, in-person, etc.).

4. **Result Release**

   o After the configured release time, results are unlocked. (See *Result Disclosure Policies*.)

Voter Workflow

1. Open the **event URL** (or scan QR) in a browser.

2. Connect **JoyID** to authenticate and obtain a unique voter session.

3. Complete the ballot form and submit. If allowed, you may **revote** (the final on-chain ballot before the deadline counts) up to the organizer-set limit.

4. Upon submission, the app displays a **human-readable ballot receipt**—a commitment hash plus a link to the inclusion proof—so voters can verify on-chain inclusion without revealing their choices. The receipt can also be sent via email; email addresses are used solely for delivery and are never recorded on-chain.

5. After results are released, return to **audit**: confirm your ballot's inclusion (proof of inclusion) and review final tallies.

Features

- **Organizer-sponsored fees:** Each ballot's CKB fee is paid from the event's Lockscript Cell.

- Show **cost estimation** based on expected voters and data before creating event.

- **Revoting with limits:** Organizers set a max number of updates per voter. Only the **last valid** ballot before the deadline is tallied.

- **Flexible eligibility:** Public, single invite-key, per-voter keys, or curated lists. Email is **delivery only**; keys are never stored on-chain.

- **Voter privacy:** Ballots are encrypted with the voter's JoyID key and/or event public key; only aggregate results are revealed per policy.

- **Automatic tallying:** Smart-contract logic (via scripts) computes results on unlock; no central counter required.

- **Anonymity controls:** Totals only; per-group results with **k-anonymity** (groups under threshold auto-merged); or fully public totals.

- **Live statistics (configurable):** Show real-time totals or hide until release; during hidden mode, only participation counts and group counts are shown.

- **health dashboard** during voting (turnout, anomaly flags, k-anonymity status) without leaking result.

Clarifications & fixes

- Double voting in public or invite-key modes is a **Sybil** risk. Mitigations include **rate limits, CAPTCHA, email verification, and device detection**; these reduce—but do not eliminate—Sybil attempts. Elections using **per-voter keys** largely avoid this issue. Post-election **group-level audits** can flag inconsistencies (e.g., votes exceeding the eligible population for a group). If detected, the organizer should **invalidate the affected group's results** or **rerun the vote** for that group under stricter eligibility.

Result Disclosure Policies

During voting: The organizer selects either **Real-time** (show aggregate participation and, if enabled, option-level totals) or **Closed until result date** (show participation counts only; no option-level totals).

At release: Results are protected by a **timelock** and a **3-of-N multisig**. After the configured release time, when the **third distinct eligible party** (organizer or voter) confirms the release, the tally decrypts. Before that time, decryption is impossible—preventing premature disclosure.

Publication modes (post-release):

- **Public:** Anyone can view final tallies.

- **Invite-Key:** Viewing results requires the same invite-key used to vote.

- **Private:** Only organizer(s) and authenticated voters can view final tallies.

Note: Group-level reporting enforces the configured **k-anonymity** threshold; sub-threshold groups are automatically merged on the blockchain by the unlocking process.

Security & Privacy Model

- **Data confidentiality:** Ballots are encrypted client-side; only aggregated outputs are decrypted at release under policy.

- **Integrity:** Every state change is recorded as immutable CKB cells; code hash of the front-end logic is committed in the Metadata Cell to prevent organizer-side script substitution.

- **Anonymity:** Group-level reporting enforces minimum cohort size; sub-threshold groups are merged automatically at unlock.

- **Availability & liveness:** Anyone (or a threshold set) can trigger tallying after the release time to avoid organizer censorship.

- **Auditability:** Voters can verify inclusion of their (encrypted) ballot and everyone can independently recompute tallies from public cells.

Known limitations

- **Coercion resistance:** Receipts can help audit inclusion but may also enable proof-of-vote. We recommend receipts that **do not reveal choices** and include optional **plausible deniability** mechanisms.

- **Sybil attacks:** Public modes remain vulnerable; use stronger eligibility for binding elections.

Threats & Mitigations (Summary)

- **Automated voting (bots):** CAPTCHA + rate-limits; optional email verification; organizer review for anomalies.

- **Small-group deanonymization:** Enforce k (e.g., k ≥ 5) and auto-merge groups below k before displaying.

- **Phishing of invite links:** Signed invitations and short-lived magic links; encourage out-of-band verification.

- **Front-end tampering:** Commit the **front-end code hash** in Metadata Cell; app validates at load.

Governance, Compliance & Operations

- **Open-source:** Code and build reproducibility docs published on GitHub.

- **Data protection:** No PII written on-chain; email/phone used only for delivery and stored off-chain with consent.

- **Audit logs:** On-chain provenance plus optional off-chain event logs for UX analytics (opt-in).

Technical Architecture

On-chain CKB Cells

- **Lockscript Cell:** Holds organizer funds for sponsoring transactions. After the event, unspent CKB is withdrawable to the organizer.

- **Metadata Cell:** Stores event parameters (schedule, questions, anonymity policy, grouping config), a **hash of the deployed front-end code**, and public keys required for ballot encryption and later decryption. Created at publish time.

- **Voter Cells:** One per voter submission. Contains the encrypted ballot payload, voter public key (or commitment), and sequence to support revoting (the latest valid cell supersedes earlier ones).

- **Result Cell:** Stores aggregated counts per option and per allowed group, plus references (hashes) to included Voter Cells for later pruning.

Cryptographic Building Blocks

- **Ballot confidentiality:** Client-side encryption with the event public key; optional hybrid with JoyID keys for personal audit.

- **Time-locked tally:** Results are encrypted and **only decryptable** after the configured time via a time-lock mechanism combined with **threshold key shares** across organizer(s)/auditors.

- **Code hash attestation:** Metadata Cell includes a hash of the front-end bundle; the app checks this at runtime to ensure the organizer didn't publish a modified script.

Revoting Semantics

- Each ballot includes a **monotonic counter** or timestamp; the tally script includes only the final ballot per voter (subject to the configured update limit).

Storage & Pruning

- Organizers pay for occupied **live cells**. When online results are no longer required, the Result and Voter Cells can be **consumed** and CKB freed; only historical proofs remain.

Future Work

- **Automated Release Integration:** Improve GitHub version control by aligning official **VoteSecure** releases with blockchain-based event results. Automate tagging and deployment based on finalized, verifiable on-chain data.

- **Customizable Voter Interfaces:** Develop a public API to allow third-party applications or websites to embed fully customized voter interfaces, while maintaining the underlying blockchain voting logic and integrity guarantees.

- **Credential-Based Eligibility:** Explore integrations with modern identity frameworks such as OIDC (OpenID Connect), Verifiable Credentials (VCs), and proof-of-personhood protocols to strengthen voter eligibility — without compromising anonymity or requiring centralized identity disclosure.

Solution distribution and maintenance

The complete **VoteSecure** source code is publicly hosted and maintained on GitHub:

🔗 **Repository:** https://github.com/psawyerberlin/votesecure

The latest stable release of the application is deployed at:

🌐 **Live Site:** https://votesecure.net

All source code is distributed under the **MIT License**, allowing anyone to freely use, modify, and distribute the software, including for commercial purposes, provided that proper attribution is given to the original authors.

✅ **Version Control and Transparency**

Each release is linked to a specific Git commit hash, providing full traceability between the published source code and the deployed version. This ensures that independent parties can verify the integrity and authenticity of the application as it is publicly available.

🔄 **Maintenance and Contributions**

The project is actively maintained by the authors and is open to community contributions via pull requests. Bug reports, feature suggestions, and community involvement are welcome through GitHub's issue tracking system.

Glossary

- **CKB:** Nervos Common Knowledge Base, a Layer-1 blockchain focused on state verification.

- **Cell:** CKB's primary data structure for storing state.

- **JoyID:** Passwordless Web3 wallet used for key management and signing.

- **k-anonymity:** Releasing statistics only if at least k individuals share the same group, to protect privacy.