Trust Votes 블록체인 기반 선거 시스템

Version 1

Date: January 3, 2025

목차 **(**한국어**)**

---

1. 소개

1.1 배경 및 문제점

한국은 기술적으로 매우 발전했지만, 선거 과정에서의 부패와 정부 개입 문제로 국민들의 신뢰가 흔들리는 경우가 적지 않습니다. 기존의 종이 투표와 중앙집중식 개표 방식은 조작의 여지를 남기며, 권력 기관이 변화를 막을 수도 있습니다. **TrustVotes**는 블록체인 기반 투표에 다층 보안(생체 인식, 투표용지 사진 검증, AI 기반 모니터링)을 결합함으로써, 광범위한 조작을 거의 불가능하게 만들고자 합니다.

1.2 왜 블록체인 투표인가**?**

1. 불변 원장**(Immutable Ledger)**: 투표 데이터를 변조하거나 삭제하기 어렵습니다.
2. 탈중앙 검증: NGO, 학계, 국제 감시 기관 등 다양한 주체가 검증 노드로 참여해, 특정 세력의 독점을 방지합니다.
3. 투명성 **&** 감사 용이성: 실시간 대시보드, 공개 API, 참관 노드를 통해 누구나 개표 상황을 확인하고 감사할 수 있습니다.

1.3 문서 목적

이 백서(버전 1)는 **TrustVotes**가 고도의 부패 및 정부 저항 상황에서도 작동할 수 있도록, 기술 구조, 코드 예시, **AI/**바이오메트릭 연동, 사회·정치 전략을 통합하여 종합적인 청사진을 제시합니다.

---

2. 시스템 개요 및 구조

2.1 핵심 목표

1. 보안 **&** 무결성: 선거 조작이 극도로 어렵도록 설계.
2. 투명성 **&** 검증 가능성: 실시간 개표 확인 및 암호학적 검증.
3. 포용성: 스마트폰, 생체 스캔, 종이 투표 등 다양한 방식 지원.
4. 저항성: 정부가 시스템 중단을 시도해도 대체 노드, 국제 감시 등으로 생존 가능.

2.2 Avalanche Subnet 선택 이유

Avalanche는 EVM 호환, 빠른 처리 속도, 낮은 지연 시간을 갖춘 플랫폼입니다.
Permissioned Subnet을 통해, 국제 NGO나 학계 기관만 검증 노드가 되도록 설정함으로써 부정세력이 네트워크를 장악하는 것을 방지할 수 있습니다.

2.3 다층 보안**:** 바이오메트릭**,** 사진**, AI**

- 얼굴 스캔**(**생체 인식**)**: 사진 원본 대신 암호화 해시만 사용해 개인정보 노출을 최소화.
- 투표용지 사진 비교: 종이 투표가 끝난 후 사용자가 투표 용지를 찍어 업로드하면, 실제 결과와 대조 가능.
- **AI** 이상 징후 감지: 비정상적인 지역별 투표 급증, 이미지 변조 등을 실시간으로 파악.

---

3. 기술 구현

3.1 신원 및 영지식증명**(ZKP)**

- 영지식증명을 통해 유권자 자격을 증명하되, 민감정보를 온체인에 노출하지 않음.
- 대안 신원체계: 정부 DB 연동이 어렵거나 의심스러울 경우, NGO/커뮤니티 기반 신원확인 방식을 고려.

3.2 스마트 컨트랙트 및 예시 코드

```
// 예시 전용, 실제 사용 전 보안 감사 필수
contract TrustVotesVoting {
   // 후보, 투표 중 여부, 관리자(관리자: NGO/DAO 등)
   // ...
}
```

3.3 프론트엔드 **& UX** 고려사항

- 모바일**/**브라우저 지원: 투표 과정에서 사진 촬영**(**얼굴 스캔, 투표용지**)** 가능.
- 접근성: 시니어층, 농어촌 등 데이터 인프라가 취약한 지역을 위한 안내 및 오프라인 시스템 고려.

3.4 바이오메트릭 **&** 사진 검증 연동

1. 생체 데이터 해시화: 사용자 얼굴 스캔 → 해시/토큰 생성 → 온체인에 민감정보 비저장.
2. 투표 시 재확인: 투표할 때 다시 얼굴 확인 → 동일인 여부 인증.
3. 종이 투표 사진: 종이에 인쇄된 고유 ID(바코드/QR)와 함께 사진을 찍어 업로드.

---

4. AI 기반 부정행위 감지

4.1 실시간 이상 탐지

- 머신러닝 모델: 지역별 투표 추이, 시간대별 집계 등을 추적. 이례적인 패턴 발생 시 경고.
- 소셜 그래프 분석: 여러 지갑이 특정 그룹에게 집중되는 등 이상 징후 파악.

4.2 투표용지 사진 포렌식

- **OCR** 적용: 사진 속 문구, 체크 표시 등을 인식해 공식 기록과 비교.
- 위조 탐지: AI가 이미지 합성, 딥페이크 흔적 포착.

4.3 인적 감수 및 감사 기록

- 공개 로그: AI가 감지한 이상 사례는 NGO, 언론 참관 노드에도 자동 공개.
- 수작업 심의: AI가 의심한 케이스를 독립 감사관이 최종 확인.

---

5. 사회·정치적 및 조직 전략

5.1 부패 및 정부 저항 대응

1. 국제 **NGO** 연합 노드: 국내 정부가 네트워크 정지를 시도해도 해외 노드가 계속 운영.
2. 병행 선거**(Shadow Election)**: 공직선거 결과와 별도 블록체인 결과를 공개 비교해 조작 의혹을 파악.
3. 법률·외교적 대책: 프로젝트를 보호하기 위한 글로벌 여론 환기, 국제 협력.

5.2 커뮤니티 연합 **&** 병행 선거

- 재외동포**(**해외 거주 한국인**)**: 국내 통제가 어려운 노드 운영.
- 언론 협업: 실시간 집계 장면을 생방송하거나 뉴스 플랫폼에 연동.

5.3 법률적 및 규제 고려사항

- 공직선거법 준수 혹은 예외 조항 협의.
- 개인정보보호법**(PIPA)**: 생체정보, 투표정보 보호.
- 정부 탄압 가능성: 법적 소송, 규제 불허에 대한 사전 대비.

---

6. 구현 로드맵

6.1 1단계: MVP & 소규모 파일럿 (6~12개월)

- 핵심 컨트랙트 개발: Voting, ZKP Verifier, Governance 등.
- 대학/NGO 선거 파일럿: 실제 투표 상황에 적용해 피드백 확보.

6.2 2단계: 병행 선거 & 확장 (12~24개월)

- **Shadow Election**: 국회의원·지방선거 시기에 병행 투표 시스템 도입.
- **AI** 모니터링: 실시간 데이터 분석과 얼굴 스캔 기능 부분 도입.

6.3 3단계: 전국적 또는 하이브리드 통합 (24~36개월)

- **NEC**와 협력(가능하다면): 부분적 혹은 전체 도입 시도.
- 노드 네트워크 확장: 더 많은 NGO, 해상 교민 단체, 학술기관 참여.

---

7. 확장 기능: 얼굴 스캔 & 투표용지 사진 비교

7.1 얼굴 스캔 등록 & 프라이버시

- 해시 보관: 얼굴 원본 이미지는 블록체인에 저장하지 않음.
- 현장 **vs.** 원격: 투표소 키오스크 또는 스마트폰 앱을 통한 스캔. 안면 인식 위조 방지를 위한 라이브니스(**liveness**) 체크 필요.

7.2 종이 투표용지 사진 비교

- 고유 **ID**: 종이 투표용지에 QR/바코드로 고유 식별코드 부여.
- 업로드 & 해시화: 촬영된 이미지는 해시 처리되어 체인에 기록. 공식 개표 결과와 대조해 불일치 시 경고.

7.3 한계 및 실무적 문제

- 법적 제약: 투표소에서 사진 찍는 행위를 금지하는 나라나 지역이 있을 수 있음.
- 기술 장벽: 신뢰도 높은 얼굴 인식 기술 도입 비용, 기기 보급률 문제 등.

---

8. 거의 불가능한 조작을 위한 방안

8.1 계층화된 검증 및 교차 확인

- 블록체인, 생체 스캔, 사진 비교, AI 모니터링 등 복수 레이어를 통해 한 영역만 뚫어도 나머지 단계가 방어 역할 수행.

8.2 분산 및 국제 노드

- 국내 정부가 물리적으로 서버를 압수해도 해외 노드가 계속 가동, 원장 무결성 유지.

8.3 사전 경보 시스템으로서의 **AI**

- 실시간 이상감지: 대규모 비정상 패턴이 발견되면 즉시 언론, NGO에 경고 발령.
- 거버넌스 투표: AI가 포착한 문제에 대해 DAO나 커뮤니티가 행동 결정.

---

9. 결론 **&** 향후 방향

TrustVotes는 한국처럼 부패 위험이 있는 환경에서도 안전하고 투명한 투표를 가능케 하려는 종합 솔루션입니다. 블록체인과 AI, 바이오메트릭, 사진 검증, 그리고 국제 협력을 결합함으로써, 어떠한 권력도 선거 결과를 마음대로 조작하기가 극도로 어려워집니다.

앞으로의 발전:

- 완전동형암호**(FHE)** 등 고급 암호기술로 투표 프라이버시와 집계 정확성을 향상.
- **AI** 알고리즘 고도화로 더 정교한 위·변조 및 딥페이크 탐지.
- 국제 선거 연합: 해외 감시기관과의 협업 확대, 글로벌 표준 추진.

개발자, NGO, 연구기관, 해외동포 및 다양한 시민 단체의 참여로, 한국은 물론 세계 여러 나라의 공정한 선거에 기여할 수 있는 혁신적 모델이 될 것입니다.

---

10. 참고자료

1. Avalanche Documentation
2. 공직선거법 (Korean Law)
3. 개인정보보호법 (PIPA)
4. ISO/IEC 27001 - 정보보호
5. Zcash 기술 문서 (ZK Proof 참고)
6. OpenZeppelin 컨트랙트

---

11. 부록

11.1 추가 스마트 컨트랙트 코드

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.17;

/**
 * @title Governance
 * @dev 제안, 온체인 투표 기능을 제공하는 간단한 거버넌스 컨트랙트 예시
 */
contract Governance {
    struct Proposal {
        uint256 id;
        address proposer;
        string description;
        uint256 votesFor;
        uint256 votesAgainst;
        bool open;
    }

    uint256 public proposalCount;
    mapping(uint256 => Proposal) public proposals;
    mapping(uint256 => mapping(address => bool)) public voted;

    event ProposalCreated(uint256 id, address proposer, string desc);
    event Voted(uint256 proposalId, address voter, bool support);
    event ProposalClosed(uint256 proposalId, bool passed);

    function createProposal(string memory _desc) external {
        proposalCount++;
        proposals[proposalCount] = Proposal({
            id: proposalCount,
            proposer: msg.sender,
            description: _desc,
            votesFor: 0,
            votesAgainst: 0,
            open: true
        });
        emit ProposalCreated(proposalCount, msg.sender, _desc);
    }

    function voteOnProposal(uint256 _proposalId, bool _support) external {
        require(_proposalId > 0 && _proposalId <= proposalCount, "잘못된 제안 ID");
        require(proposals[_proposalId].open, "이미 종료된 제안");
        require(!voted[_proposalId][msg.sender], "이미 투표함");

        voted[_proposalId][msg.sender] = true;
        if (_support) {
            proposals[_proposalId].votesFor++;
        } else {
            proposals[_proposalId].votesAgainst++;
        }
```

```
        emit Voted(_proposalId, msg.sender, _support);
    }

    function closeProposal(uint256 _proposalId) external {
        require(_proposalId > 0 && _proposalId <= proposalCount, "잘못된 제안 ID");
        require(proposals[_proposalId].open, "이미 종료됨");

        proposals[_proposalId].open = false;
        bool passed = proposals[_proposalId].votesFor > proposals[_proposalId].votesAgainst;
        emit ProposalClosed(_proposalId, passed);
    }
}
```

11.2 ZKP 의사 코드 & 예시

```
function generateZKProof(userInfo, privateKey) returns (proof) {
    // 1. 사용자 정보로부터 고유 커밋 생성
    // 2. ZK 라이브러리(SnarkJS, Semaphore 등)로 영지식증명 생성
    // 3. proof 객체 반환
}

function verifyZKProofOnChain(proof) returns (bool) {
    // 1. proof 형식·유효성 검증
    // 2. 이미 사용된 proof 여부 확인 (이중 투표 방지)
    // 3. true 반환 시 검증 완료
}
```

11.3 프론트엔드 보일러플레이트 샘플 (React/Next.js)

```
// frontend/pages/index.js
import React, { useState, useEffect } from 'react';
import { ethers } from 'ethers';
import VotingArtifact from '../artifacts/contracts/TrustVotesVoting.sol/TrustVotesVoting.json';

const CONTRACT_ADDRESS = "0x배포된컨트랙트주소";

export default function Home() {
  const [candidates, setCandidates] = useState([]);
  const [account, setAccount] = useState('');

  useEffect(() => {
    (async () => {
      if (window.ethereum) {
        const [acct] = await window.ethereum.request({ method: 'eth_requestAccounts' });
        setAccount(acct);
        await fetchCandidates();
```

```
    }
  })();
}, []);

async function fetchCandidates() {
  // ...
}

async function castVote(candidateId) {
  // ...
}

return (
  <div>
    <h1>TrustVotes 메인 화면</h1>
    <p>접속 계정: {account}</p>
    {candidates.map((c) => (
      <div key={c.id}>
        <span>{c.name} - 득표수: {c.voteCount}</span>
        <button onClick={() => castVote(c.id)}>투표하기</button>
      </div>
    ))}
  </div>
);
}
```

Version 1 — End of Document