Reinventing Voting with Digital Identity, Trust and Anonymity

Open Milestones Details

Project ID	1200194
Link	Open full project
Challenge	F12: Cardano Open: Developers
Budget	ADA 176,500.00
Funds distributed	ADA 25,599.99
Start date	Aug 12, 2024

Milestone Title	**M1 - Finalise all specifications**
Milestone Outputs	We will use the first milestone to finalise the specifications. The outputs are: •Assessing the Identus/Agent node system for managing DID and VCs •Reviewing choices, validating capabilities, and defining minimal requirements (functional and nonfunctional) for a DID wallet as a browser extension •Defining all requirements (functional and nonfunctional) for a minimal Voting App, including for those roles: Admin, Designer, Voter, Viewer •Reviewing choices for a voting App (App vs DApp) and establishing final choice •Defining high level architecture for the chosen Voting App choice (App vs DApp), including what will be delivered as smart contract, and what will be delivered as more classic "Web2" Each of those outputs will be documented (with documentation and/or video) in the GitHub repo (https://github.com/incubiq/vote_with_did).

System Architecture (overview)

Components

- Identus for identity management
- DID Wallet for authentication & issuance
- Cloud Backend for core operations
- A Cardano dApp for transparency
- WebApp for user interface

Roles

- Admin: ballot approval and registration mgt
- Designer: ballot creation and configuration
- Voter: voting operations
- Viewer: results and proofs access

System Architecture (modules)

1. Identus for identity management *planned for M2*

Functional Requirements

- as per the current specs of Identus

Non-functional Requirements

- Hosted solution
- Testnet

Assessment

A full 30min “Proof of Capability” review of Identus

https://youtu.be/4DyPuZr_3PA

Use case

- At pre-registration time, a voter will be issued a Verifiable Credential to allow voting for a particular ballot (including voting right and voting power).

- At voting time, the VC is presented to the Verifier, as Proof to the voting app, and the vote is recorded

System Architecture (modules)

2. Digital Identity Wallet (DIDW) *planned for M2*

Functional Requirements

- Browser extension implementation
- Sign message authentication for wallet access
- Secure storage of DIDs (onto Cardano)
- Storage and management of Verifiable Credentials (VCs) via Identus
- Integration with Identus SDK and/or our
- Credential presentation capability

Non-functional Requirements

- Cold wallet backup/restore functionality
- Maximum wallet unlock time < 3 seconds
- Encrypted local storage
- Chrome / Brave compatibility at first

Assessment

Reviewed open-source Identity wallets, from which we will fast track our design, and code.

- https://hyperledger.github.io/identus-docs/identus-edge-agent-sdk-ts/sdk/ (the Identus Edge Agent SDK)
- https://github.com/cardano-foundation/cf-identity-wallet (identity wallet provided by the Cardano Foundation, although NOT based on Identus, but using KERI)
- (continued next slide...)

System Architecture (modules)

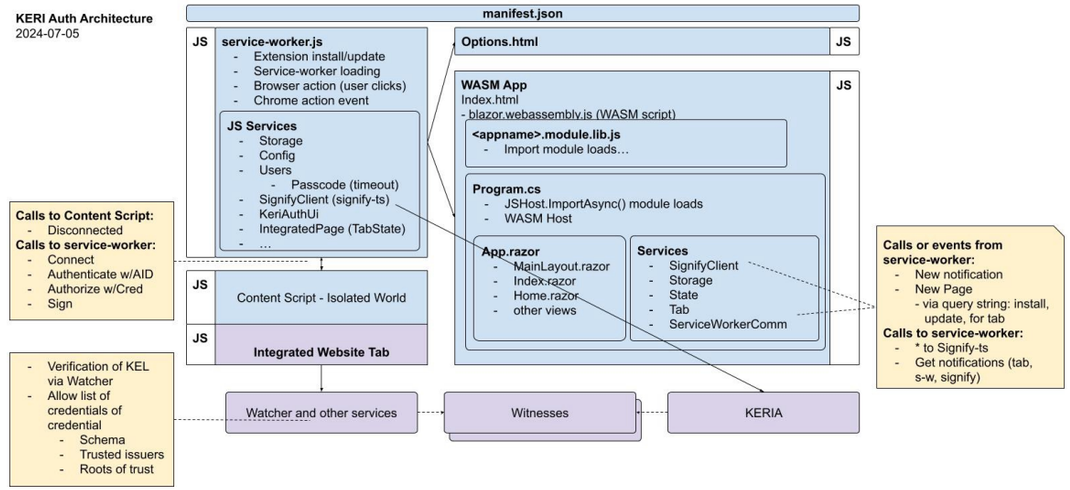
2. Digital Identity Wallet (DIDW) *planned for M2*

Assessment (continued)

- https://github.com/KERIAuth/keriauth-blazor-wasm/tree/main (the design of the KERI Browser extension wallet will help us design ours)

- https://github.com/bsandmann/blocktrust-identity-wallet (a Browser Ext wallet built by a community member of Identus, although not open source, and built in .NET)

- https://github.com/roots-id/rootswallet (by RootsId, a ReactNative wallet integrating with Identus, open source, however not updated last 2 years)



- https://github.com/socious-io/socious-wallet : by Socious - this project is currently active, was a catalyst F11 funded project (https://milestones.projectcatalyst.io/projects/1100211) ; the developer is an active member of the Identus community. This is the best match for speeding up our Digital Identity Wallet.

System Architecture (modules)

3. Cloud Backend for core operations

3.1. Admin System *planned for M3*

Functional Requirements

- Admin authenticate with a (cardano) wallet (sign message)
- Configure voting system parameters
- Manage DID integration settings
- Monitor system health and security
- Manage user roles and permissions (onboard designers, minimal validation workflow)
- Handle voter registration and verification
- Approve ballots to enable voting
- (v2) Generate system reports and audit logs

Non-functional Requirements

- (v2) Audit trail of admin actions
- (v2) 2FA authentication
- Response time < 2 seconds for admin operations

System Architecture (modules)

3. Cloud Backend for core operations

3.2. Identity System *planned for M2*

Functional Requirements

- Creation of a user's Identitus Entity and wallet
- Integration with Identus for DID and VC issuance
- (v2) Revocation capability for invalid VCs

Non-functional Requirements

- VC issuance time < 1 minute
- Audit trail of all VC issuances
- (v2) Rate limiting for VC requests

System Architecture (modules)

3. Cloud Backend for core operations

3.3. Pre-registration System *planned for M4 (but started in M3)*

Functional Requirements

- Verification of voter eligibility
- VC template creation with following attributes:
  * Voter identification (DID)
  * Voting power allocation
  * Validity period
  * Eligible ballot identifiers
  * Custom attributes for special voting rights
- Automated VC issuance workflow

Non-functional Requirements

- (v2) Support for batch pre-registration

System Architecture (modules)

3. Cloud Backend for core operations

3.4. Ballot management system *planned for M4*

Functional Requirements

- Create and configure ballot templates
- Set voting period parameters
- Define voting rules and validation criteria
- Configure result display options
- Set-up voter eligibility criteria

Non-functional Requirements

- Intuitive ballot design interface
- Preview functionality for ballot layouts
- Maximum ballot creation time < 5 minutes
- (v2) Version control for ballot designs

System Architecture (modules)

3. Cloud Backend for core operations

3.5. Voting System *planned for M4*

Functional Requirements

- Register/authenticate using wallet
- View available ballots
- Cast votes securely
- Verify vote submission / Receive voting confirmation
- (v2) View personal voting history

Authentication / voting rights Flow

- User unlocks DID Wallet via sign message
- DID Wallet connects to voting dApp
- User presents relevant VC for ballot access
- System verifies VC validity and voting rights
- User granted access to eligible ballots

Non-functional Requirements

- Maximum authentication time < 30 seconds
- Voting process completion < 2 minutes
- Mobile-responsive interface
- (nice to have) Offline vote preparation capability
- (nice to have) Support for accessibility standards (WCAG 2.1)

Vote Privacy Implementation

- Implementation of Zero-Knowledge Merkle Tree (similar to Tornado Cash)
  * Generation of zero-knowledge proofs for votes
  * Merkle tree construction for vote commitments
  * Vote nullifier generation to prevent double voting
  * Anonymous vote verification system

- Alternative (not fully researched yet)
  * Ring signature implementation for group anonymity
  * Vote mixing protocol

System Architecture (modules)

3. Cloud Backend for core operations

3.6. Result verification System *planned for M4 (but started in M3)*

Functional Requirements

- View election results in real-time (if ballot was set for real-time view ; otherwise view at end of voting deadline)
- Access historical voting data
- View voting statistics
- (v2) Generate basic reports
- (v2) Export results in standard formats

Non-functional Requirements

- Real-time result updates (< 5 sec delay)
- Support for concurrent viewers (min 100)
- (v2) Data export in multiple formats (CSV, PDF, JSON)

System Architecture (modules)

4. Cardano dApp for transparency *planned for M4*

Functional Requirements

- Nullifier tracking to prevent double voting
- Publishing vote commitments for public verification
- Merkle root publication
- Public vote verification
- Transparency proofs

Transparency Layer

[Cloud Backend] <-> [Transparency Service] <-> [Voting dApp]
- Merkle root publication
- Vote commitment verification
- Public audit trail

Result Publication Flow

[Cloud Backend] -> [Result Aggregator] -> [Voting dApp] -> [Cardano Blockchain]
- Anonymous result publication
- Public verification
- Immutable record

Verification Components

[WebApp] <-> [Verification Service] <-> [Voting dApp]
- Proof verification
- Result validation
- Public accessibility

System Architecture (modules)

5. WebApp for user interface *planned for M4*

Functional Requirements

User Interface
- Provide intuitive and user-friendly UI for all user roles (Voter, Admin, Designer, Viewer)
- Allow users to authenticate with the DID Wallet
- Display available ballots and voting options for eligible voters
- Enable voters to cast their votes securely
- Allow administrators to manage ballot creation and configurations
- Present election results and verification proofs to viewers

Ballot Management
- Integrate with the Backend to fetch available ballots
- Display ballot details, incl title, description, voting period
- Provide interfaces for administrators to create, update, and publish ballots
- Ensure ballot configurations are appropriately applied

Voting Operations
- Facilitate the voting process for eligible voters
- Integrate with the DID Wallet for authentication and credential verification
- Securely transmit voter selections to the Backend
- Provide real-time feedback on vote submission status

Results Visualization
- Fetch and display election results from the Voting dApp
- Present the results in an easy-to-understand format
- Allow viewers to verify results using provided proofs
- Enable drilling down into detailed voting statistics and analytics

Transparency Integration
- Integrate with a Transparency Service to fetch and display Merkle roots
- Allow viewers to verify vote commitments and audit trails
- Provide clear explanations and guidance on the verification process

Other requirements

1. Critical paths

1.1. Vote Submission Path

Voter -> WebApp -> Cloud Backend -> Voting dApp -> Cardano
- Identity verification (DID Wallet)
- Vote encryption (Cloud Backend)
- Commitment publication (Voting dApp)

1.2. Result Verification Path

Viewer -> WebApp -> Cloud Backend -> Voting dApp -> Cardano
- Proof verification
- Result validation
- Public accessibility

1.3. Audit Trail Path

Admin/Viewer -> WebApp -> Cloud Backend -> Voting dApp -> Cardano
- Operation logging
- Proof generation
- Public verification

Other requirements

2. Database

2.1. Ballot metadata
- Configuration
- Status
- Results cache

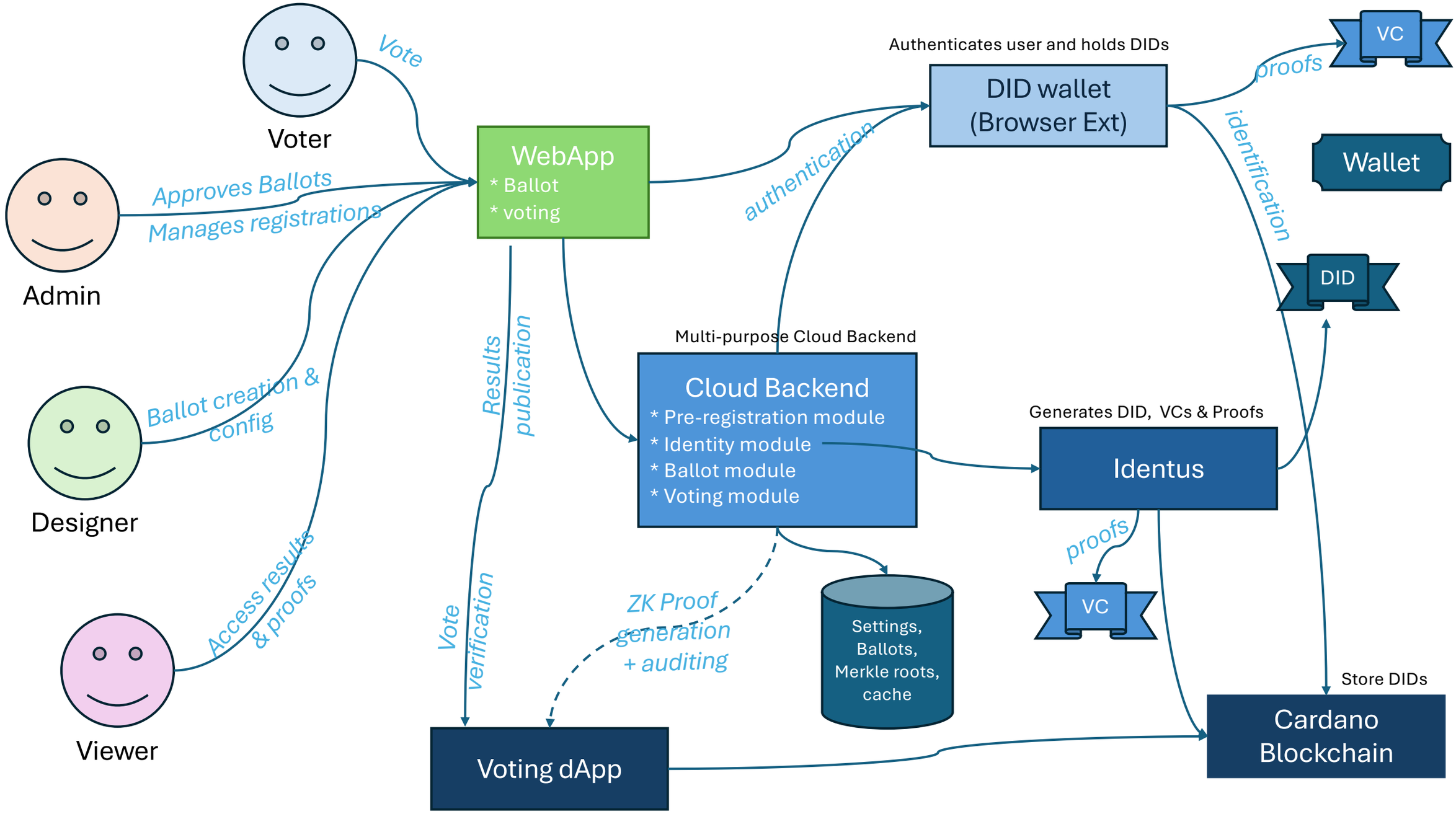
2.2. Vote data
- Encrypted votes
- ZK proofs
- Commitment mappings

2.3. Audit data
- Operation logs
- Merkle tree states
- Verification proofs

Other requirements

3. Privacy Measures

- Zero-knowledge proofs for vote privacy
- Merkle tree implementation for vote anonymity
- Separation of identity from voting records
- Encrypted communication channels
- Private metadata management


Voter
Vote
Admin
Approves Ballots
Manages registrations
Designer
Ballot creation & config
Viewer
Access results & proofs
WebApp
* Ballot
* voting
Results publication
Vote verification
Authenticates user and holds DIDs
DID wallet (Browser Ext)
authentication
proofs
VC
identification
Wallet
DID
Multi-purpose Cloud Backend
Cloud Backend
* Pre-registration module
* Identity module
* Ballot module
* Voting module
Generates DID, VCs & Proofs
Identus
proofs
VC
ZK Proof generation + auditing
Settings, Ballots, Merkle roots, cache
Store DIDs
Voting dApp
Cardano Blockchain