AnonEvote
A Project on Blockchain Evoting
Functional Specification

Michael Wall
13522003

Monday 21st November, 2016

Version	**Comments**
0.1	Initial Draft.

1 Introduction

This document describes an implementation of an electronic voting system which utilises a blockchain[1] database. The system is used to allow a user to electronically cast a vote in an anonymous, verifiable and tamper-proof manner. Votes are ecnrypted and cast over a peer-to-peer network. The final votes can be counted and verified by any user, but no users' votes can be traced back to a voter.

The need for this system arises from the lack of a secure, trustless, tamper-proof and anonymous electronic voting system. Paper ballots are slower, and more expensive to conduct (work hours to organise polling stations, tallying votes, re-counts of votes, multiple voting options, errors filling out a ballot, etc). The AnonEvote system aims to tackle some of these issues.

Blockchain The blockchain is the database which maintains the *ledger* of *votes*. It is distributed amongst all participants in the system via peer-to-peer networking. Each client will maintain an up-to-date version of the *ledger* to the best of their knowledge. The blockchain is described further in section §3.2 on page 7.

Ledger The *ledger* is a growing record of each *vote* which has been cast. It is publicly viewable.

Vote Each block in the *ledger* contains a single *vote*, which is encrypted to provide anonymity and to prevent tampering. The structure of a secure *vote* is described further in §3.1 on page 5.

2 General Description

2.1 Product/System Functions

The AnonEvote system is designed to enable electronic voting to be carried out securely, tamper-proof and anonymously. A user of the system will cast a vote electronically through some user interface. The user's vote is then encrypted to form a *secured vote*. The user can verify that their vote is what they intended by decrypting it. This will spoil the ballot, but after repeated casting and decrypting the user can be sure that their vote is accurate. When the user is happy with their ballot, it is then broadcast to all of the client's peers to be verified. The verification of the transaction requires a proof of work to be a valid block. This proof of work entails some computationally expensive task. When the proof of work is complete, the block is then broadcast to a client's peers. If there is disagreement between a new block, consensus is used to select the correct version which is then broadcast to all peers.

To verify their vote, a user can look at the blockchain and verify that their receipt exists on the chain. Homomorphic encryption of votes allows them to be tallied without decrypting individual votes. Any user will be able to perform the computation to tally the votes.

2.2 User Characteristics

A user will be any eligible voter. The users are not expected to require any prior knowledge of blockchains, cryptography or other technical concepts described in this document. Because different users may have different user needs or certain levels of ability, the system must be accessible. As a graphical user interface is not a major priority for the system, this challenge will not be addressed to any major extent. The system will be used by text inputs, similar to a command line program, on top of which a UI can be built at a later date.

A user should be able to input a vote, and verify their selection. Once the user has made their selections, they should be able to validate the encryption of their vote as many times as they see sufficient. Once the user is happy that the system is registering their intended vote, the user should then be able to cast their vote to the network to have the transaction verified and added to the ledger. The user should also receive a digital receipt of their transaction so that they can verify their vote at a later date. The user should also be able to perform the required computation to tally the votes should they wish to do so.

2.3 Operational Scenarios

2.3.1 Main

The main scenario in which the system is intended to be used is any large scale elections, referendums or other votes in which a large body of people will participate in. These votes are required to provide anonymity to voters, tamper-proof security, easy tallying and re-counting of ballots, and a trustless[2] environment.

2.4 User stories

2.4.1 User casts a vote which is valid

A user selects candidate A on their electronic ballot. They verify that their selection was correctly registered and cast the vote. Their vote is then verified by peers on the network and added to the ledger.

2.4.2 User ranks 3 out of 5 options on their ballot

A user selects candidate options A, D and E on their ballot. They choose not to rank the last two options B or C. Their vote is verified, and added to the ledger.

2.4.3 User selects no options on their ballot

A user chooses not to select an option on their ballot. This is represented as selecting a ”None of the above” option for the purposes of the vote.

2.4.4 User spoils their ballot by selecting wrong options

The user selects both candidate A and B on their electronic ballot. The system will not allow an incorrect selection such as this, and alerts the user to the error. The user then starts the process from the start.

2.5 Constraints

A voting system in general needs a few things to make it fair. A prime factor of a successful voting system is that a voter should not be able to sell their vote. For this requirement to be met, a user should have no way of proving what way they filled out their ballot to the outside world. This also means that the ballot itself should have no way of identifying who filled it out.

Ballots should also not be tamperable. This is why some voting systems use pencil instead of pens, to ensure that no erasing ink could be used to invalidate a voter’s ballot.

Different voting systems[3] also have different requirements.

In the simplest case a voting system in which a voter selects a single option from two or more choices, and the option with the most votes wins. Other systems are more complicated and require more complicated comutation to achieve a result.

Such systems include weighted voting systems, proportional, semi-proportional, rated and multiple winner systems. Such systems pose challenges as the votes may not be very easily calculated as with a simple tally.

Some voting systems also require the polling statistics to be kept a secret until the end of the vote. This could prevent early predictions from causing a swing in the vote due to people believing that a particular candidate has already won.

Each voting system would require its own rules to specified for the vote to be successful. For this reason the AnonEvote system will focus on two systems as a proof of concept, with the ability to add more systems. The two systems will be *Proportional Reprisentation with a Single Transferable Vote*[4] and *Party List Proportional Representation*[5].

3 Functional Requirements

3.1 Secure Votes

3.1.1 User can access the system

Description A user should be able to securely access the system and be verified to cast their vote. The no user should be able to masquerade as another user to cast their vote.

Criticality Low

Technical issues It is a challenge to authenticate a single user, as the method of registering for the system must ensure that each citizen can only have one account, and that no other citizen can use another's account. It is expected that this authentication of users will be managed by a third party who wishes to implement this system.

Dependencies on other requirements None.

3.1.2 User can make selections on their ballot

Description A user should be able to input the appropriate options on their ballot, including multiple selections, scorings, abstaining from selections etc.

Criticality High

Technical issues A certain type of vote may require text input, in which case the user could enter an error or mispelling into the system.

Dependencies on other requirements This is dependent on 3.3.1.

3.1.3 User can verify their ballot is as intended

Description A user should be able to verify that their input ballot is encrypted as intended. This may invlove decrypting their transaction before it is broadcast to the network in order to verify that it represents their intended ballot. This will invalidate this particular encryption of the ballot. The user should be able to perform this a number of times until they are sufficiently sure that the system is correctly representing their vote.

Criticality High

Technical issues It may be an issue that once a transaction is decrypted, the network should know that it is invalid and so not accept it into the ledger.

Dependencies on other requirements This is dependent on 3.1.5.

3.1.4 A user cannot reveal the contents of their ballot to the outside world

Description In order to satisfy the constraints described in 2.5, a user should not be able to reveal what way they filled out their ballot to the outside world.

Criticality High

Technical issues The votes must be countable without a user being able to identify the vote as their own.

Dependencies on other requirements This is dependent on 3.1.5.

3.1.5 The ballot can be encrypted and formed into a transaction

Description A ballot should be represented in some agreed format, and encrypted so as the contents of the ballot are protected, and that no user can have their vote identified.

Criticality High

Technical issues It is a huge issue to ensure that a user cannot link their vote to their identity, as this enables the sale of votes. The result of the vote should be calculatable, while also not revealing information about the voter.

Dependencies on other requirements This is dependent on 3.3.2 and 3.3.3.

3.2 Blockchain

3.2.1 A client node can broadcast to its peers

Description A peer-to-peer network will be required to enable a node to broadcast transactions, blocks and other necessary information to other nodes involved in maintaining the system.

Criticality High

Technical issues Node discovery on the peer-to-peer network may be problematic without the use of a peer discovery server or installed lists of peers to begin seeding the network.

Dependencies on other requirements None.

3.2.2 A node can verify a transaction and create a block

Description A node should perform some cryptographic operation which is computationally expensive, like a proof of work, in order to verify a transaction and create a block for the network.

Criticality High

Technical issues Selecting a proof of work of sufficient difficulty without being too expensive. If a transaction can be verified instantaneously, it becomes easier to hack the system. If it takes too long to verify, then the user could have to wait an inconvenient amount of time before their vote is successfuly added to the system.

Dependencies on other requirements None.

3.2.3 Nodes can add new blocks to the chain using a consensus

Description Nodes should be able to agree on the correct version of the voting history using consensus. This would mean to hack the system an attacker would need to be in control of 51% of the nodes in the network. This is not feasible in a large scaling system.

Criticality High

Technical issues This may be a generally difficult task to complete across a network of peers where fully operational communications are not guaranteed, and not all peers are guaranteed to be connected at all times.

Dependencies on other requirements This is dependent on 3.2.1.

3.3 Voting System

3.3.1 A particular voting system can be implemented

Description Ideally, the style of voting sytem used should be independent from the blockchain implementation, meaning the system can be used in different regions which require different rules and regulations. However, the system will be required to work on at least one system as a proof of concept.

Criticality High

Technical issues If homomorphic encryption is to be used to tally votes, this may not work in some voting systems which are not a simple count, but have weighting towards votes and different criteria for an outcome to be determined.

Dependencies on other requirements This is dependent on 3.3.2 and 3.1.5.

3.3.2 The ledger can be processed to obtain a vote's result

Description The sytem should be implemented in such a way as to allow the result of the vote to be calculated without compromising voter identities or linking votes to voters.

Criticality High

Technical issues This faces very similar issues as mentioned in 3.3.1.

Dependencies on other requirements This is dependent on 3.3.1 and 3.1.5.

3.3.3 A user can see their vote in the ledger

Description A user should be able to see that their ballot has been added to the ledger correctly. They should be able to do this without identifying it as their vote and without revealing the contents of the vote.

Criticality Medium

Technical issues A user should not be able to identify the contents of their ballot so as to avoid the selling of votes.

Dependencies on other requirements None.

4 System Architecture

Please see Figure 1 on page 12 for the diagram. The system architecture consists of a network of nodes, all of which run the client software. Nodes are connected to form a peer-to-peer network.

5 High Level Design

Please see Figure 2 on page 13 for the diagram. A user will interact with the system to perform completion of their ballot. The client node will independently carry out peer discovery. It is not required that a node communicate with specific peers on the network. A node will perform encryption of a ballot and broadcasting of transactions for its user. It will also perform transaction verification, consensus agreements, and chain updating in collaboration with other peers on the network. Any node can perform these functions if they are running the client software.

6 Development schedule

My project will not consist of a traditional schedule which has a large block of time for research, then development and then testing or polishing of the project. I will instead be using an agile approach for development.

I will be working with a sprint duration of two weeks, enabling sufficient time for research and development of selected tasks, while also ensuring that any necessary feedback can be provided between sprints. Any issues that arise during a sprint can be dealt with before progressing further with the project.

Tasks for any given sprint will be selected from the project backlog at the beginning of the sprint. Any new issues will be added to the backlog as the project progresses. Any tasks which require grooming will be expanded into more specific sub tasks which are suitable for development in a sprint.

I will be using a private repository on GitHub to manage my code, and will push changes to my GitLab repository frequently. I will use *Waffle.io* to manage the project backlog and to track issues. This system will be integrated with the GitHub repository. I will use a Travis server to perform continuous integration testing where applicable.

Notes

[1]A blockchain is a distributed database which maintains a growing ledger of records called blocks. Each block is timestamped and linked to the previous block to create the chain. The data in a block cannot be altered without breaking the chain

[2]A trustless environment is one in which no one individual or group requires to be trusted to ensure the integrity of the system. For example, a bank is not trustless, as you must trust the bank to keep its record of your account accurate and free from tampering.

[3]A voting system consists of the set of rules which must be followed for a vote to be valid, and defines how otes are cast, counted and aggregated to yield the final result. Examples include a plurality, majority representation and many other variations.

[4]The PR-STV system is used in Ireland.

[5]The Party List PR system is used in 85 countries

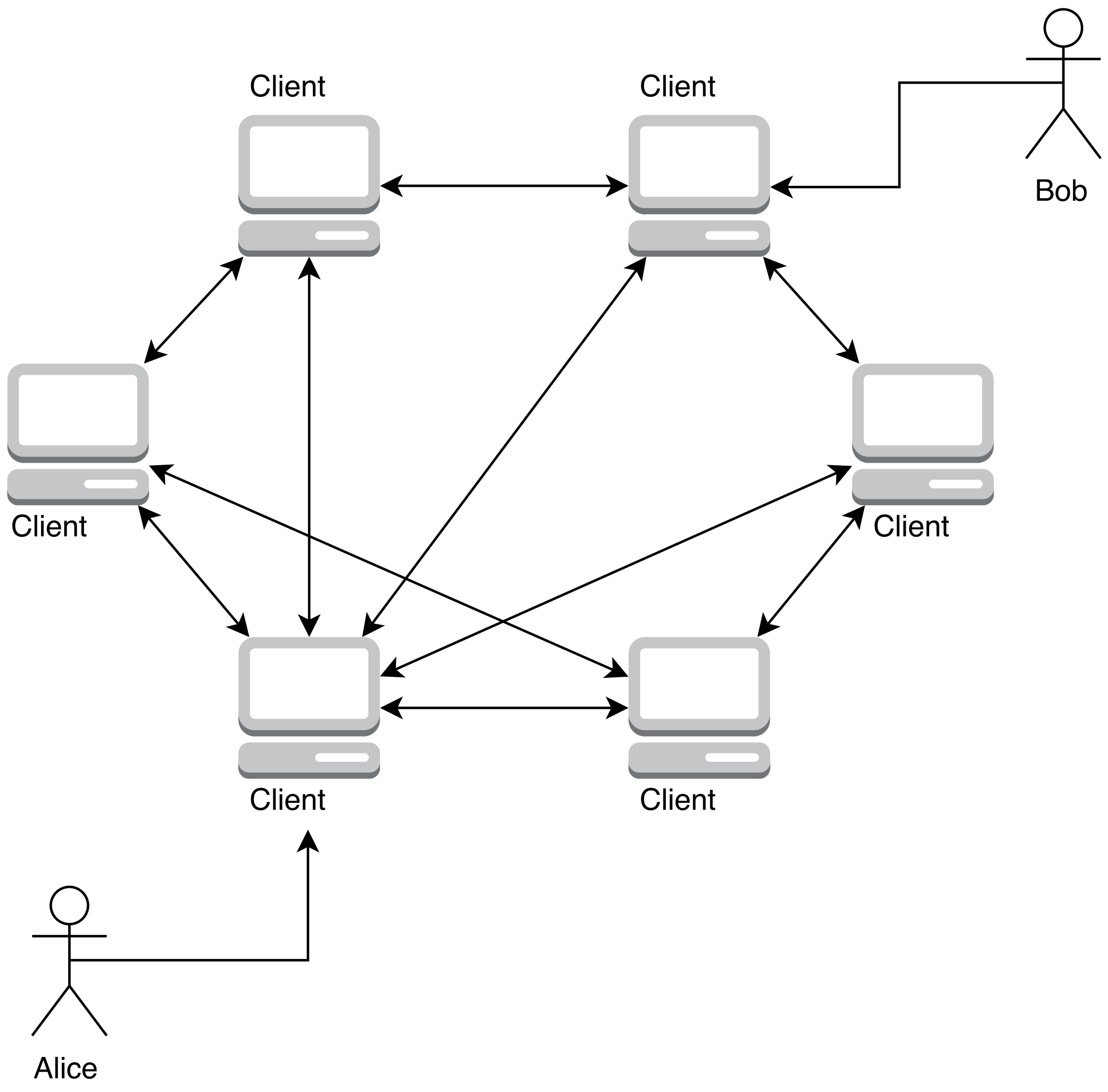

Figure 1: The system consists of a peer to peer network of nodes, all of which are running the client software. The user interacts with the system through a node running the client software.

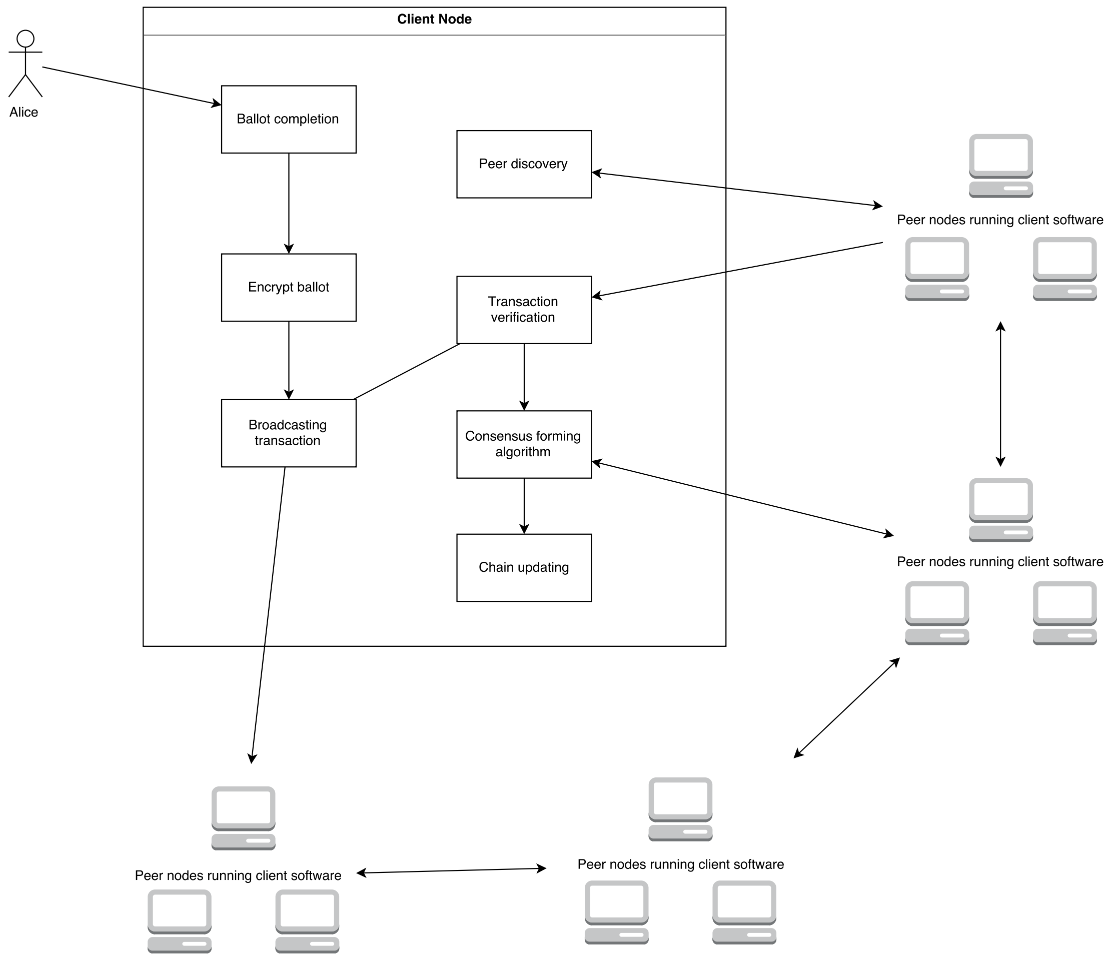
Client Node
Alice
Ballot completion
Encrypt ballot
Broadcasting transaction
Peer discovery
Transaction verification
Consensus forming algorithm
Chain updating
Peer nodes running client software
Peer nodes running client software
Peer nodes running client software
Peer nodes running client software

Figure 2: This diagram shows the high level design of the system. The image depicts the functions which a client node must perform. A user of the system will interact with ballot completion. A node can communicate with any of its peers for successful operation.