Decentralized voting system using Blockchain

Thomas Holvoet, Anel Muhamedagic, Lukas De Loose

November 2019

1 Introduction

In most countries, people still go to vote in person, resulting in low participation rates. Furthermore, a problem in many elections, is that the electoral commission itself is corrupt or untrustworthy. A web-based voting approach might seem like an easy solution to prevent the masses from heading to the Electoral Offices, but is it secure enough? Moreover, is it possible to build a decentralized voting system using a Blockchain? In this project, we will implement a proof-of-concept decentralized voting system, using our own Blockchain and homomorphic encryption.

2 Topic Changes

The topic did not change. We will however add a feature that prevents DoS attacks.

3 Related Work

E-voting is clearly a well researched topic, and there exist various approaches. One of the main examples for an e-voting system, is the one developed by Swiss Post [1]. This was issued to actually be used in the elections of 19 May. However due to errors in the source code, the project was suspended, but the system should be in use by 2020. The architecture of this e-voting platform is however completely different from what we have in mind, as the system commissioned by the Swiss government, has a centralised architecture. This has several disadvantages compared to a decentralised solution. One of the main ones, is that there is single point of failure. Attacking this central node, could compromise the entire elections. Secondly, this also means that the agency operating the central node, i.e. the government, has to be trusted to handle the results of the election correctly. One of the points we would be improving upon, is that for example for sham democracies, where the results of the election are often manipulated, we have a system in which the users can verify the results themselves. This means it cannot be tampered with by a central party.

We do not have to look far to find a decentralised e-voting solution. DEDIS lab at EPFL developed an e-voting system, that has already been used in the internal elections. One of the differences with their approach, is that our system will not have administrator users, that oversee the election. Every node will be able to launch a poll, of which it is the administrator. The EPFL system is also able to leverage the existing - centralized - authentication system. User authentication is however not in the scope of this project. One of the differences with this system, is that the counting of the votes is also distributed over the different nodes. In our implementation the results are decrypted by the pollmaster, however because of the use of Paillier encryption, the votes are already counted, simply when being stored. This way there is no need to anonymise the votes, the identity of the user is not stored. It does however mean that the user cannot verify his vote after issuing it. A second advantage of this approach, is that there is no need to shuffle and distribute the votes over the different nodes to count them.

[1] https://www.post.ch/-/media/post/evoting/dokumente/evoting-system-dokumentation.pdf?la=envs=2

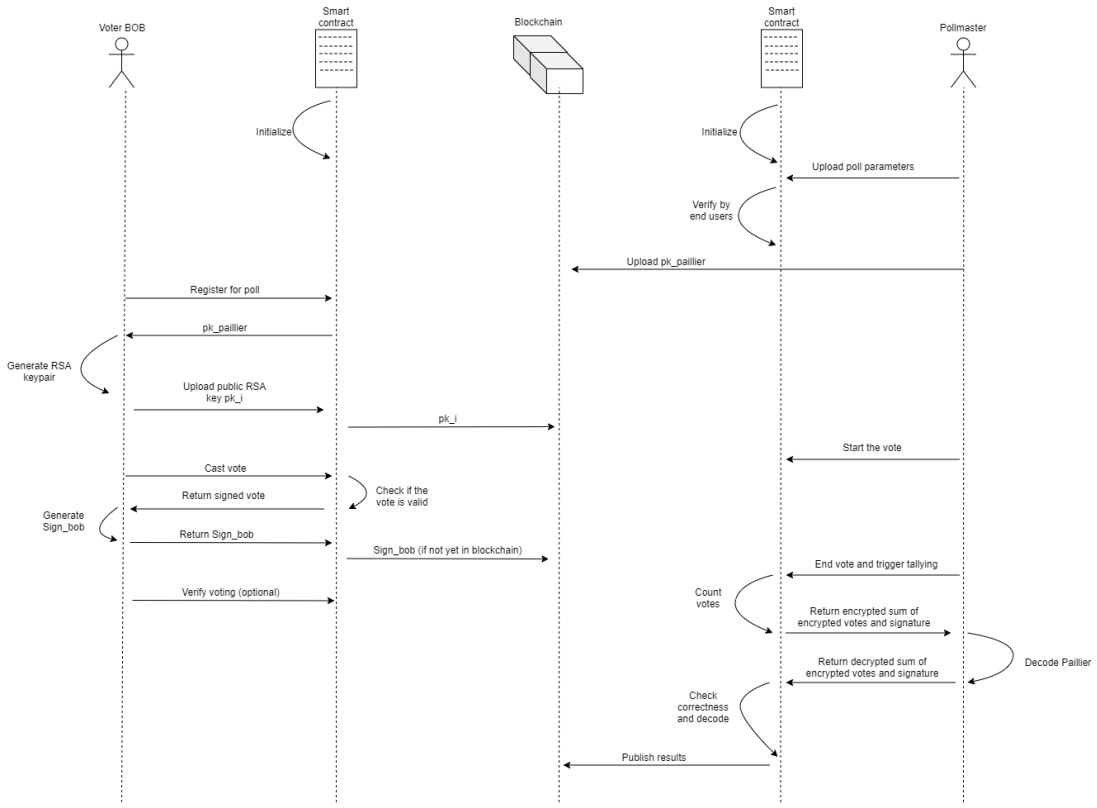

Figure 1: Diagram of a normal voting procedure in our decentralized voting system.

An interesting paper is: "A Smart Contract For Boardroom Voting with Maximum Voter Privacy" [2]. It proposed the first implementation of a decentralized and self-tallying internet voting protocol with maximum voter privacy using the Blockchain, called The Open Vote Network (OVN). The OVN is written as a smart contract for the Ethereum blockchain. In its general idea the OVN is an implementation of the Anonymous voting by two-round public discussion. The creators of the OVN came to the conclusion after implementing the system, that the cost of running such system on the Ethereum blockchain was 0.73$ per voter. The safe upper limit of voters was 50 voters, but the cost could be considered reasonable as it provided maximum voter privacy and is publicly verifiable (we will of course run our implementation on an Ethereum test-blockchain). The limitation of number of voters was recommended because of the gas limit on the public Ethereum blockchain. However, the OVN is vulnerable to denial of service attack because it is implemented on the Ethereum blockchain, which has had numerous DOS attacks through its lifespan. The implementation could therefore be optimal for small boardroom voting, with the downside of having each individual voter downloading the full Ethereum blockchain to confirm the voting protocol is being executed correctly (these drawbacks will most probable also be the case for our implementation).

4 System Goals and Functionalities

in figure 1, the voting process is visualized.

[2]https://eprint.iacr.org/2017/110.pdf

4.1 Goals

1. Send Peerster Message to other voters
2. Spread polls in the network
3. Create a poll
4. Register for a poll
5. Vote for a given poll
6. Signing the vote reply
7. Identify Participants
8. Compute Results of a poll
9. Encrypt Messages
10. Store results
11. Ensure Anonymity of vote replies
12. Hinder nodes to exploit the system (DoS)

4.2 Functionalities

In order to realize our goals, we fill implement certain functionalities. These functionalities will be split into subcategories and then further elaborated in this chapter. We can split the given goals into [1] - [2] communication, [3] - [8] voting and [9]-[12] security.

4.2.1 Communication

Sending messages should be possible within seconds. All the messages send by peersters should be at any other node within seconds. The nodes should be able to communicate privately as well as broadcasting to the network. Broadcasting is important since it allows the network to spread news about the vote and give important information to other nodes. It also enhances free speech since information is not being filtered. These functionalities are already implemented from the homework.

Spreading polls in the network needs to be possible in multicast. Polls should reach a subgroup of participants within seconds. The polls need to be signed by the Pollmaster (the person initiating the poll) to enable authenticity and integrity. The recipient needs to acknowledge the reception to the pollmaster of a poll signed with its own private key in order to be sure everyone received the respective poll.

4.2.2 Voting

Everyone should be able to create a poll. From that moment, he/she becomes the **Pollmaster**. The Pollmaster must be able to create a poll and keep track of it. He must be able to receive and send messages. He needs to keep track of the reception of the poll by waiting for an ack of every participant.
Creating a poll needs to be easy and doable for every node. Creating a poll includes:

1. Setting up a poll question/request
2. Choosing participants who participate in the vote

3. Setting up a deadline

Nodes **registering for a vote** need to have a unique identifier and a private/public key in place. This ensures authenticity and integrity of replies as well as identification. Nodes will need to authenticate themselves at the pollmaster by sending the acknowledgement and the poll reply messages given their sciper number, **signed with their private key**. It is crucial to not enclose the private key.

Voting for a poll needs to be doable by a node within seconds. He should be able to pick one of the possibilities of the poll request.

The pollmaster and the smart contract need to **identify participants** by their poll reply. They will check if the reply for a pole is valid. If not the reply must be discarded. in order to ensure this every node needs to have a public/private key pair.

Computing the result of a poll needs to be done within seconds. It should work using homomorphic encryption so the anonymity of each participant can be guaranteed.

4.2.3 Security

In order to create our Public Voting System, we will need to implement different parties.
We need first a Pollmaster who decides who is able to vote and on what. He decides about the ballet, the deadline and triggers the Smart Contract so people are able to vote. For simplicity reasons, we will restrict the votes to being no (0) or yes (1) questions.
We have the voters who are nodes in our Blockchain Network. The voters mine in order to be able to participate in the vote. All of them need to have a public and private key so they can be determined by the administrator.
Our third and last party is the Blockchain, which uses Smart Contracts to check for the results.

4.3 Initialisation

4.3.1 Administration

Before starting the vote, the Pollmaster should be able to register the candidates... After this, a Smart Contract will be generated, with all the candidates in it.

4.3.2 Authentication

In real-life situations, we could use two-factor authorisation to provide authentication (e.g. code send by post and social security number). In our project, we will use students sciper number. These numbers are stored in the Blockchain. Upon registering, one enters his/her sciper number and publishes it to the Blockchain. A smart contract is executed to check whether the sciper number is valid (stored on the Blockchain), and send back the public Pallier key. Next, the person can register its RSA public key, so that people can sign their votes with their private key.

4.4 Cryptography

4.4.1 Paillier

We will use Paillier as our Crypto Scheme[3]. Paillier allows us to use homomorphic encryption. This means that we are able to sum up the encrypted votes (0 no /1 yes) because and publish a result without who voted for what. We realise this by letting a Smart Contract decrypt the result and announce it after the vote is finished (Deadline approach).

[3]https://godoc.org/bitbucket.org/ustraca/crypto/paillier

4.5 Blockchain

We will implement our own Blockchain structure using the rumormongering process of the first assignment. Every block is a structure, and the Blockchain is a slice of blocks. Using the proof-of-work algorithm, nodes will 'mine' to find the next block. When it is found, the block is broadcast to the other nodes, that validate it and add it to the Blockchain. For simplicity, we will not implement a rewarding system.

5 Performance

One of the main performance bottlenecks comes from using a proof-of-work Blockchain. As all votes have to be stored in the Blockchain, the number of votes that can be processed and stored at any time is determined by the number of blocks that can be mined. Because it should not be too easy to mine blocks to ensure the validity of the Blockchain, there is a trade-off between security and performance.

Homomorphic encryption also has some performance issues. It is known that the execution time for Paillier encryption explodes for en-/decrypting large strings[4]. However, this is not an issue as long as the votes are limited in size. Scaling up to be able to handle votes on hundreds of options, might be a problem.

6 Task division

We will split our task mainly into three-part according to our building blocks.
The first part is the initialisation, this means implementing a public key system to be able to distinguish a valid from a non-valid user. Therefore we need to be able to have an RSA in place. This person will also implement the support for the pollmaster's functionalities, and mechanisms to prevent DoS-attacks.
The second building block is setting up the Blockchain, proof-of-work algorithm, and creating the Smart Contracts.
Finally, the third building block is the setup of the crypto scheme Paillier, together with the smart contracts that add up the votes and the publication of the results.

[4]http://www.wseas.us/e-library/conferences/2012/Paris/ECCS/ECCS-19.pdf