An introduction to Internet voting and it's different approaches from a security standpoint

Diogo Cepa
up201403367

Gonçalo Ribeiro
up201403877

Nuno Martins
up201405079

May 19, 2018

Abstract

Our aim for this report is to offer a general understanding about the current state of the art around online voting from a security perspective. Noways, there's a wide variety of voting software products being employed all over the world by different companies and countries (Estonia's 2005 general elections). Our research shows that, at it's core, the vast majority of these products will employ one or more of the following security primitives: homomorphic cryptography, mix networks and blockchain databases.

In this report, these primitives are analyzed from an abstracted, simplified point of view to better illustrate their implications and usefulness for electronic voting protocols. Based on what we gathered from our research, we then developed our own functional voting demo based on the Ethereum Blockchain.

1 Introduction

Electronic voting, also known as e-voting, is a collection of methods that allow voting with the use of electronic devices. There is a wide range of voting systems used worldwide nowadays, and some are going to be explored in this project, as well as particular use cases, possibles vulnerabilities and a practical component with the implementation of a simple voting demo based on *Ethereum Smart Contracts*.

There can be identified two main types of e-voting:

- **Offline:** the election is done on special machines supervised by the electoral authorities;
- **Online:** the election is done on it's entirety over the Internet, from any location.

In this project, we are going to explore the security features of the latter as they better fit the context of security that is being studied.

1.1 Motivation

E-voting has many benefits over normal paper voting, but also has bigger security risks. E-voting is cheaper once implemented, allows much faster counting of votes, and is better to the environment, as no resources are wasted. On the other hand, usage of e-voting has greater security problems, as it is done over the Internet, an insecure channel, and is much harder to supervise [14].

Although voting on special supervised machines has greater security, the use of online voting allows the reach of a wider audience and is cheaper.

1.2 History

1.2.1 Estonia

The first country to officially use an i-voting system was Estonia in 2005, where 1.9% of the votes were cast online. Since then, it was used in 2007, 2009, 2011, 2013, 2014 and 2015. In the last parliamentary elections in 2015, 64.2% (577,910 voters) of the eligible voters participated actively in the election. 30.5% (176,491 voters) of these voters used e-voting to place their vote. This underlines the acceptance of I-voting in the Estonian population [12].

In this system, the votes are cast over the Internet in any machine, an validation of a voter is done through insertion of an ID card that every citizen processes on a compatible machine.

1.2.2 Norway

Norway used a remote electronic voting system for the county council elections in 2011. The general design is similar to the Estonian system. It is presented as the first governmental voting system fulfilling the requirements for coercion-resistance and voter verifiability [12].

The system is developed with the help of *Scytl3*, a Spanish company which is awarded for the electronic voting systems they are developing. However, in June, 2014 the Ministry of Local Government and Modernization decided to discontinue the Internet voting pilot project due to security concerns; many parts of the eligible population feared that their votes could become public, which might undermine democratic processes [6].

2 Security features

According to a 2004 paper by C. Boyd and E. Dawson [1], an ideal electronic voting solution must capable of guaranteeing the following requirements:

- **Privacy:** Voter-vote relationship must be kept private, to ensure that voters express their true opinion without fear of being intimidated.
- **Eligibility:** Only authorized voters are allowed to vote, preventing fraudulent votes from being counted in the tally stage.

- **Prevention of double voting:** This ensures that all voters are allowed to vote only once, such that each voter has equal power in deciding the outcome of the voting.
- **Fairness:** No partial tally is revealed before the end of the voting period, to enforce privacy and ensure that all candidates are given a fair chance during the voting period.
- **Robustness:** The system must be able to tolerate certain faulty conditions by managing some disruptions.
- **Verifiability:** Correct voting processes must be verifiable to prevent incorrect voting result.

2.1 Homomorphic Cryptography

Homomorphic cryptography provides a third party with the ability to perform simple computations on encrypted data without revealing any information about the data itself [8]. Typically, a third party can calculate the encrypted sum or the encrypted product of two encrypted messages. This kind of cryptography plays a major role in the tally phase of an e-voting system since it allows public signed encrypted votes to be counted without revealing their contents [9].

Blind signing was first introduced by D. Chaum on his 1983 paper about utilizing homomorphic cryptography to create anonymous payment systems [4]. This technique uses homomorphic encryption properties to encrypt the contents of a message before the message is signed [3].

The concept of a blind signature can be illustrated by an example taken from the familiar world of paper documents. The paper equivalent of a blind signature can be implemented with carbon paper lined envelopes. Writing a signature on the outside of such an envelope leaves a carbon copy of the signature on a slip of paper within the envelope [4]. This is mainly useful to protect the content of the votes, while assuring that they come from a reliable source.

2.1.1 Advantages

Encryption schemes with homomorphic properties have several benefits for electronic voting systems, which depend on the algorithms. Being able to aggregate the encrypted ballots simplifies the tallying process, since only one decryption is needed after all ballots were aggregated.

Different schemes of homomorphic encryption also enable a number of mechanisms, like secret-sharing or the re-encryption of the ballots, which is heavily used in voting systems which use mix-nets for anonymization.

2.1.2 Disadvantages

Some schemes are only suitable for elections where yes or no are possible answers. Another big drawback is the computing time needed to aggregate homomorphic

encrypted ballots. This is very complex and might not be applicable on big amounts of encrypted ballots.

With that being said, K. Gjøsteen proposed a mathematical method to massively reduce the size of the encrypted ballots and consequently decrease the computational time of tallying [7].

2.2 Mix Networks

A mix network is a routing protocol that guarantees anonymity by hiding the correspondence between the network's input and output of messages. The term was first introduced by David Chaum on his work about anonymous email systems [5]. Mix networks have a wide range of practical applications. They can be found in use today for anonymous browsing purposes [15] and secure payment systems [11].

The basic mix network can be represented as a chain of network nodes, each with its own public and private key, that stand between the source and destination of a message [10]. The source client must encrypt its message with each of the nodes' public keys, in an order chosen by him. Afterwards, the message is sent down the chain, where the nodes are in the reverse order of the previous encryption. Each node then decrypts the message using its private key. The message leaves the end node unencrypted and arrives at its intended destination. This layered encryption scheme guarantees total confidentiality within the mix network [15].

One of the fundamental properties of mix networks is that each of the nodes in a specific chain only know the address of the previous and next nodes [5]. This means that a man-in-the-middle attack between two of the nodes wouldn't be able to reveal the origin of the message, thus guaranteeing the message's anonymity inside the chain.

2.2.1 Advantages

Mix-net servers provide anonymity with a simple, well-known procedure and are robust against attacks on the voter's identity. These servers can easily be distributed among multiple and independent authorities. As long as one of these authorities is honest, the mix is successful and the connection between the voter and her ballot is removed [15]. As a result all ballots are anonymized.

2.2.2 Disadvantages

Ideally, many dedicated servers are needed for a mix-net to perform the mixes [12]. The setup and maintenance of this dedicated network can become more expensive than other anonymization schemes, such as Blind Signing (explained previously).

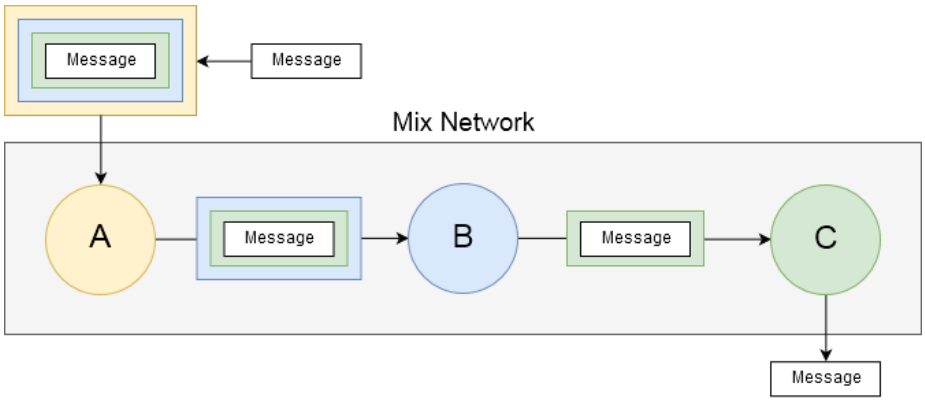

Figure 1: Illustration of a message being sent through a mix network. Each of the colored boxes represents one layer of encryption with the public key of the node with the same color. Every time the message reaches a node, it loses one of its layers of encryption.

2.3 Blockchain

The term blockchain was first coined in 2008 by Satoshi Nakamoto [13] on the whitepapper for what was later to become the first cryptocurrency, Bitcoin. At its core, a blockchain is a distributed database that maintains a secure and ever-growing ledger of records (usually transactions) known as blocks.

This database is managed by a network of nodes that all have their own copy of the blockchain. The nodes are simply computers connected to the network that have agreed to process the validity of transactions on the blockchain based on a set of rules the network has agreed to. Once a node validates a transaction, it adds it to a chronological group known as a block that is then added to the blockchain. Valid transactions are therefore grouped together and added to the database in a block, one after the other, hence the name blockchain.

The process of discovering valid blocks is referred to as mining. To find a valid block, a miner has to find a hash that contains most of the pending transactions (not part of a block yet) and a reference to the previous block.

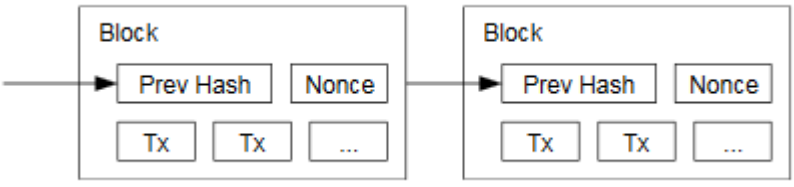

Figure 2: Bitcoin blocks pocess a reference to the previous block's hash, as well as unmined transaction

The validity of an hash is verified with a required number of zero's at the start of the hash, being the difficulty of discovering it proportional to the number of zero's required. The most typically used hashing process is SHA-256, which is to day considered safe, and therefore the process of finding a valid hash is NP-hard, and can only be solved using brute-force.

The usage of this solution is beneficial because, for an attacker to change data on an old block, he also has to find valid hashes equal to the number of blocks between the target and the most recent one, as each block's hash contains a reference to the previous one, and this task is practically impossible, as the resources required are too high for a single party to achieve. To promote users to mine, each time a miner finds a valid hash, he is rewarded a set amount of currency. Also, to assure authenticity of every transaction, public key encryption is used to sign every transaction that is made.

2.3.1 Blockchain in e-voting

For most blockchains, each transaction contains information about the sender, the receiver and the amount transfered, along with other metadata.

For some existing blockchains (mainly the Ethereum blockchain) it is also possible to encode objects called *Smart Contracts* as transactions, making the blockchain an effective state machine. These contracts can contain executable code that is stored in the blockchain. Each call to a function that modifies the internal state of the contract is considered a transaction, and thus the execution of the code is validated when it is mined. Operations that don't change the internal state of the contract are not a transaction, and thus have no fees associated with them. [2]

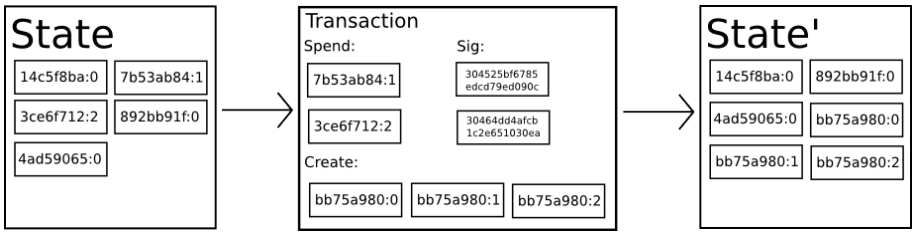

Figure 3: Ethereum simplified as a state machine. Each transaction affects the state of the blockchain.

This implementation allows the developer to build custom voting protocols on top of the blockchain, with transactions being operations such as voting, adding users to the white list to vote, creating new polls, etc.

As such, the main advantages of using Ethereum for e-voting are the following:

- Authenticity: The autheticity of the voter is assured by the signature of every transaction with the voter's private key;

- Privacy: although every transaction(vote) is public, encryption methods can be used to assure privacy of the voter, such as asymmetric cryptography on the vote(although only the creator of the vote will know the result) or homomorphic tallying (which assures total privacy, but requires that every user casts a vote);
- Eligibility: eligibility can be assured by creating a list of white-listed addresses that are allowed to vote in the blockchain;
- Prevention of double voting: prevention of double voting can be implemented in the logic of the contract, and mining assures validity of the execution of the operations;
- Fairness: can be implemented through the methods mentioned in privacy;
- Robustness: the system is error proof if implemented correctly, as every function call is safe due to mining;
- Verifiability: the voting result will be correct as long as every vote(transaction) is mined.

Although using a blockchain for e-voting grants total integrity of every operation, as well as authenticity of every voter, there is a big disadvantage which relies in the costs associated with it. The process of mining verifies that every transaction is valid, but the computational costs associated with discovering valid hashes for the blocks are very high for large scale events, and therefore the system is not very scalable.

3 Practical Component

For this mini project we developed a demonstration of an e-voting app on top of the Ethereum blockchain, along with a user friendly interface and proper safety precautions.

The idea was to build a safe e-voting application that assures anonymity, confidentiality, integrity and security.

The app consist of a web application where users can vote on polls created by other users, or create polls themselves and invite other users to vote on them.

3.1 Use cases

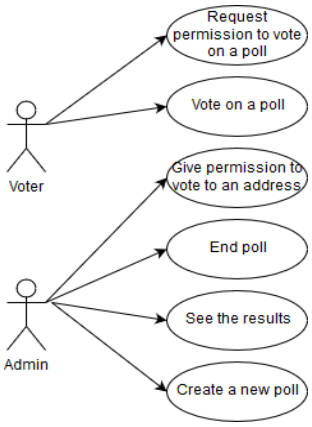

Figure 4: Use cases diagram. Upon the creation of a poll, a user becomes an admin of that poll

3.2 Usage

To use this application, the user must install a browser extension called Metamask, which allows the browser to interact with the blockchain (make transactions and fetch data). Metamask should be configured to run on the Ropsten Test Network, where ether has no value, making the testing easier. The user should create a wallet was request ether using the instructions displayed on the extension's interface.

After that, the user should load up the app on localhost and the following screen will be shown:

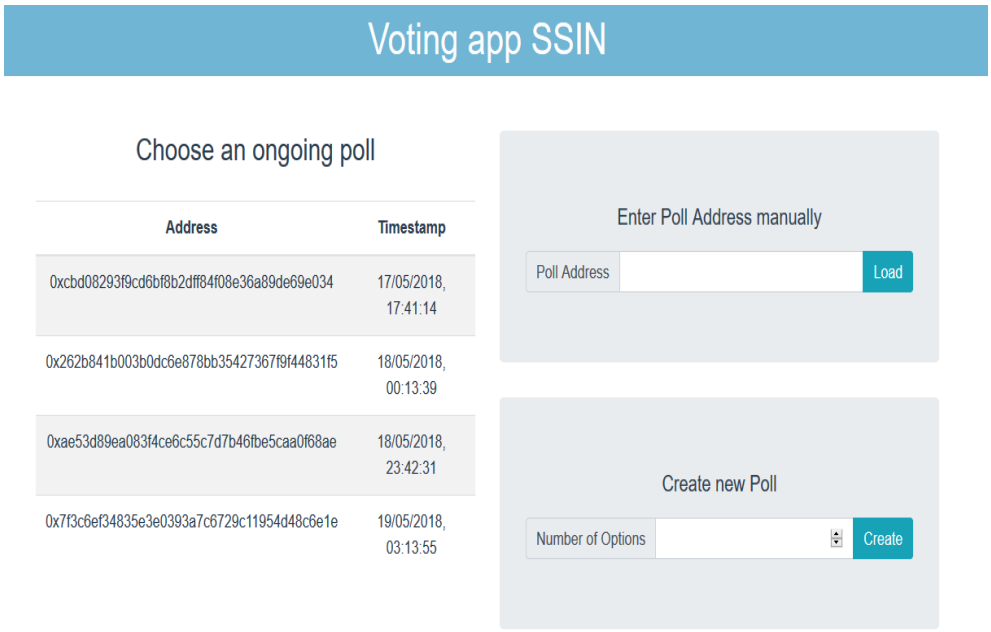

Figure 5: App main menu. Contains list of polls on the left, option to open poll manually, or creation of a poll with a set number of options to vote in

Here, the user can see on the left a list of ongoing polls, and click them to open the voting interface. He can also create his own poll, choosing the number of voting options available. Upon creation, a private key will be supplied, which the creator should save to later decrypt the votes.

Since this is a demonstration only, polls and votes have no names associated with them, only addresses and indexes respectively.

Upon opening the voting interface, the user can cast a vote, and a success message will be displayed if the vote was successful. If the user has no right to vote, a message will be shown informing him of such.

When the owner of a poll opens his own poll, he can see an admin interface, where he can give permissions to addresses to vote, see the results of the poll by supplying the private key, or end the poll.

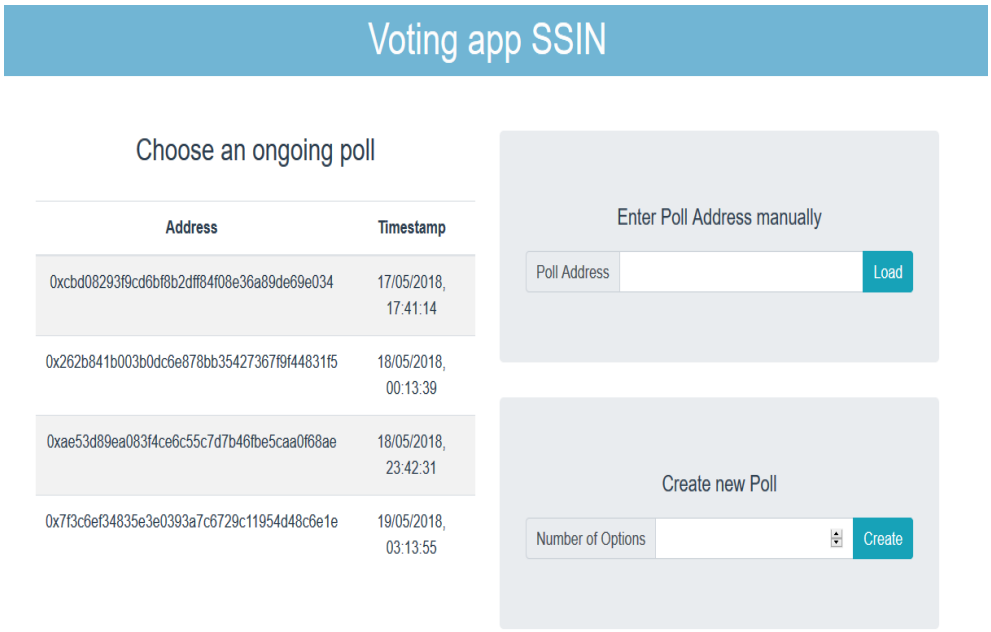

Figure 6: Interface containing the admin's interface, aswell as the common voter's interface.

3.3 Implementation

As mentioned before, this project was implemented on top of the **Ethereum** blockchain.

The frontend was developed in **VueJS** and the extension **Metamask** was used to manage Ethereum wallets(accounts).

To be able to use the application, the user must create a wallet in Metamask, which offers an interface in Javascript for interacting with the Ethereum blockchain when logged in.

The contracts were coded on the programming language **Solidity**, which is a high level object-oriented like language specifically made for designing Ethereum contracts.

The development and deployment of the contracts was done on the **Remix Ethereum IDE**(remix.ethereum.org).

Our implementation consists of 2 contracts, one being **PollCreator** and the other **Poll**. The first is a single contract whose methods are usable by anyone, and allow any user to deploy a **Poll** contract. Upon doing so, the owner of the poll is defined upon the creation of the poll, being the sender of the transaction that leads to it's creation. The contract **PollCreator** also includes a function

that retrieves the list of polls created using this contract. To create a poll, the user must include the number of options that the users can vote on, as well as supplying a **Public Key** that the voters must use to encrypt their votes to maintain confidentiality(only encrypted votes are counted).

When the voting ends, the owner of the poll can use his **Private Key** to decrypt the cast votes.

3.4 Analysis

This e-voting solution assures all of security features mentioned in the blockchain section, which are: authenticity, privacy, eligibility, prevention of double voting, fairness, robustness and verifiability.

To assure authenticity and eligibility, the creator of a poll can choose the addresses of the users who can vote. Since the transactions(votes) are signed with the voter's private key, it's impossible to forge messages from a certain address.

To assure privacy, all the votes are encrypted using an RSA public key that is available in the contract and supplied by the creator of the poll upon creation. The usage of RSA encryption can be unnecessary in circumstances where the wallet address cannot be linked to the user's identity, which in some cases can not be the case.

As to prevent double voting, the implementation assures that if two vote transactions arrive from the same address, one will overwrite the other.

Robustness and verifiability is assured by the implementation of the blockchain itself. Since every operation is executed exactly has it is written to be executed and is impossible to forge other requests, a good implementation of the contract assures that the system is fault-proof.

The biggest weakness of this system is verifiability, as only the poll creator can verify the results of the poll. This weakness can be solved by giving the private key to decrypt the results to every party competing in the poll, will mutual confirmation assuring the validity of the poll. Alternatively, homomorphic tallying could be employed, but at the cost of reduced scalability on polls with a lot of options. Non usage of RSA encryption can also assure verifiability, as every vote would be public, but there is the chance in some scenarios that a wallet address can be linked to a voter's identity.

Another less glaring weakness is that when two votes are cast by the same address in the same block, there is no way to predict which will be the final one, as there is no way assure that one transaction is mined before the other.

References

[1] Riza Aditya et al. "An efficient mixnet-based voting scheme providing receipt-freeness". In: *International Conference on Trust, Privacy and Security in Digital Business*. Springer. 2004, pp. 152–161.

[2] Vitalik Buterin. "Ethereum: a next generation smart contract and decentralized application platform (2013)". In: *URL {http://ethereum. org/ethereum. html}* (2017).

[3] David Chaum. "Blind signature system". In: *Advances in cryptology*. Springer. 1984, pp. 153–153.

[4] David Chaum. "Blind signatures for untraceable payments". In: *Advances in cryptology*. Springer. 1983, pp. 199–203.

[5] David L Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". In: *Communications of the ACM* 24.2 (1981), pp. 84–90.

[6] *E-voting experiments end in Norway amid security fears*. `http://www.bbc.com/news/technology-28055678`. Accessed: 2018-05-19.

[7] Kristian Gjøsteen. "The Norwegian internet voting protocol". In: *International Conference on E-Voting and Identity*. Springer. 2011, pp. 1–18.

[8] Kevin Henry. "The theory and applications of homomorphic cryptography". MA thesis. University of Waterloo, 2008.

[9] Martin Hirt and Kazue Sako. "Efficient receipt-free voting based on homomorphic encryption". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2000, pp. 539–556.

[10] JOAKIM UDDHOLM HJALMARSSON. "Implementing a mix-net for use in electronic voting systems". In: (2009).

[11] Markus Jacobson and David M'Raïhi. "Mix-based electronic payments". In: *International Workshop on Selected Areas in Cryptography*. Springer. 1998, pp. 157–173.

[12] Christian Meter. "Design of Distributed Voting Systems". In: *arXiv preprint arXiv:1702.02566* (2017).

[13] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: (2008).

[14] Alexander Prosser, Karl Schiessl, and Martin Fleischhacker. "E-voting: Usability and acceptance of two-stage voting procedures". In: *International Conference on Electronic Government*. Springer. 2007, pp. 378–387.

[15] Michael G Reed, Paul F Syverson, and David M Goldschlag. "Anonymous connections and onion routing". In: *IEEE Journal on Selected areas in Communications* 16.4 (1998), pp. 482–494.