Eon: Blockchain Voting for University Clubs

Myron Ladyjenko
University of Guelph, ID: 1172255

Eric Buys
University of Guelph, ID: 1176672

ABSTRACT

In elections, votes often seem to vanish like needles in a haystack among thousands of others, leaving one to wonder if individual participation truly makes a difference. With a few Blockchain-based E-voting solutions offered for high-level elections, universities are left in the dark. In a World, where transparency and trust are paramount, a visionary solution emerges – a Blockchain-Based E-Voting System tailored to the needs of university clubs.

In this paper, we propose a solution that offers security and transparency to university club elections. We aim to build a Bitcoin-based blockchain voting system that utilizes the Blind Signature Protocol to ensure unlinkability between a voter and their ballot. We offer a mechanism to verify the identity of a voter, cast a vote and transparently tally the votes. In addition, we utilize Merkle trees, SHA-256 hashing and RSA encryption to ensure the integrity of the data exchanged during the election. With this system, we hope to foster trust among student communities.

ACM Reference Format:
Myron Ladyjenko and Eric Buys. 2024. Eon: Blockchain Voting for University Clubs. *, CIS*4520 Introduction to Cryptography, University of Guelph (CIS*4520 Introduction to Cryptography, University of Guelph),* 8 pages.

1 INTRODUCTION

Voting today relies on the credibility and trustworthiness of centralized authorities to accurately and faithfully manage electoral processes. While this system has been in place for many years, it still suffers from the inherent weaknesses of a trust-based model. The vulnerabilities of this system can be seen in a variety of examples, one of which being the 2020 Belarusian Presidential Election[10]. In this election, the results were falsified which caused a wave of protests across the country. Such problems occur because voters are required to place their implicit trust in a central authority to uphold the integrity of the electoral system and provide transparent and fair outcomes. These problems are especially present in university club elections where the voting system in place involves casting a vote in an online form and being presented the results by the clubs themselves. Such systems not only lack transparency but fail to enable voters to independently verify election results. Yet, the emergence of blockchain technology offers a promising alternative by decentralizing the voting process, thereby removing the necessity for blind trust in central authorities.

Our research project explores the potential of blockchain technologies in enhancing electronic voting systems for university clubs. We aim to address traditional voting systems challenges such as susceptibility to fraud, corruption, lack of transparency, inaccuracies in vote counting, and time-consuming tallying processes by leveraging the decentralized, transparent, and immutable nature of blockchain technology. We propose a solution which offers a secure and transparent voting system for university clubs that ensures both the privacy of voters and the integrity of election outcomes through the utilization of cryptographic techniques within a blockchain framework.

As part of our research, we've designed and implemented a Bitcoin-based blockchain voting system tailored for university club elections. Our system can be used in any university by utilizing the following structure: the university acts as the Central Authority (CA) and the Blockchain Network (BN), which consists of the university club that hosts the election, as well as other clubs that act as Miners (Nodes). This system ensures privacy by utilizing the Blind

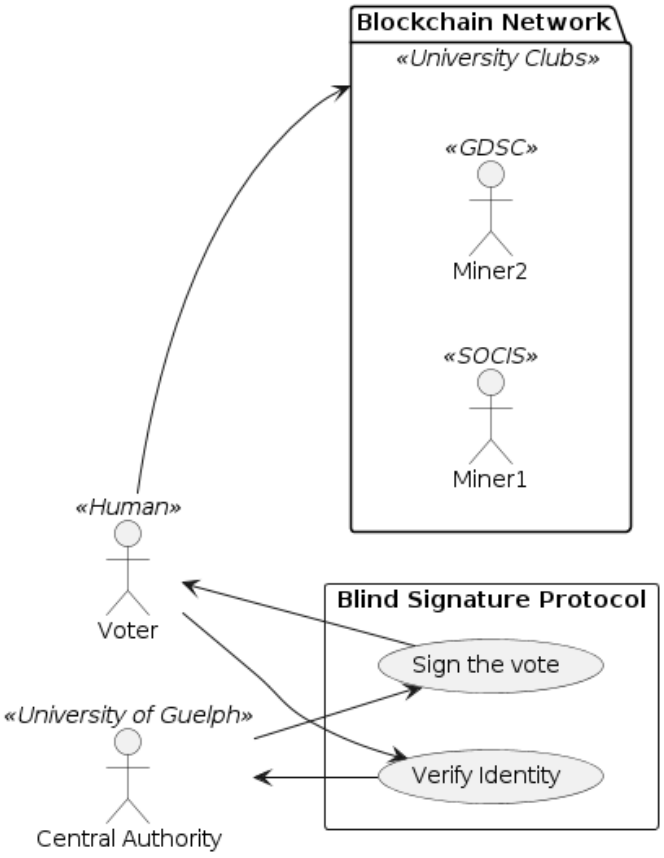

Figure 1: Use Case Diagram: Bitcoin-based E-voting system

Signature Protocol, which allows voters to verify their identity and sign their vote without revealing their identity. The security of the system, data integrity and authentication are ensured through

*CIS*4520 Introduction to Cryptography, University of Guelph, 2024*

ACM 123-4567-24-567/08/06.

cryptographic algorithms such as Proof-of-Work, SHA-256 hashing, RSA encryption and Merkle's tree. This system provides a secure and transparent platform for conducting elections within university clubs.

The remainder of this paper is organized as follows. In section two, we present our main goals and describe and high-level overview of the system, along with threat model and security considerations. Further, in section 3 we discuss research papers related to our works and analyze current approaches as well as giving a background into some of concepts we use. In section 4 we dive into the implementation details, thoroughly describing the structure of our project as well as the outcome of our implementation. Lastly, in sections 5 and 6 we summarize our findings and accomplishments and describe further improvements to our system.

2 STATEMENT OF THE RESEARCH PROBLEM

2.1 Research Questions and Objectives

The primary objectives of our research project revolve around analyzing, modelling and implementing a Blockchain-Based E-Voting System for University Clubs. The questions that our research project seeks to answer are:

1. What are the building blocks that are required to implement a blockchain-based e-voting system?
2. How can the privacy of voters be ensured?
3. How can a voter be prevented from voting with multiple accounts or multiple times within the same account?
4. How can the vote which was cast become public information to allow the transparency of the election?

The primary objectives of our research project are:

1. Implement the key features of the Bitcoin-based blockchain (tailored to be a consortium blockchain to more precisely follow the use case of University clubs).
   a. Miners and mining algorithms.
   a. Block Structure with chaining.
   a. Transaction Structures for Votes.
   b. Private/public keys, hashing.
   c. Set up a peer-to-peer (P2P) network.
2. Implement a front-end application to allow users to log in to the system and participate in the election.
3. Implement a system where voters can cast their votes while maintaining their identity private at any point in time.
4. Implement a system where a voter is only able to cast a single vote per election.
5. Implement a system where all votes are visible and each person can verify their vote is counted.

2.2 System Model

Our system will be modelled after a Bitcoin-based blockchain architecture. This allows our system to utilize the benefits that come with combining the various components that comprise a Bitcoin Blockchain Network. These major components are transactions, a Timestamp Server, the Proof-of-Work system, a Peer-to-Peer network structure and Merkle Trees [7]. Combining these components together, we can ensure the immutability of an election's results due to the nature of the Bitcoin blockchain.

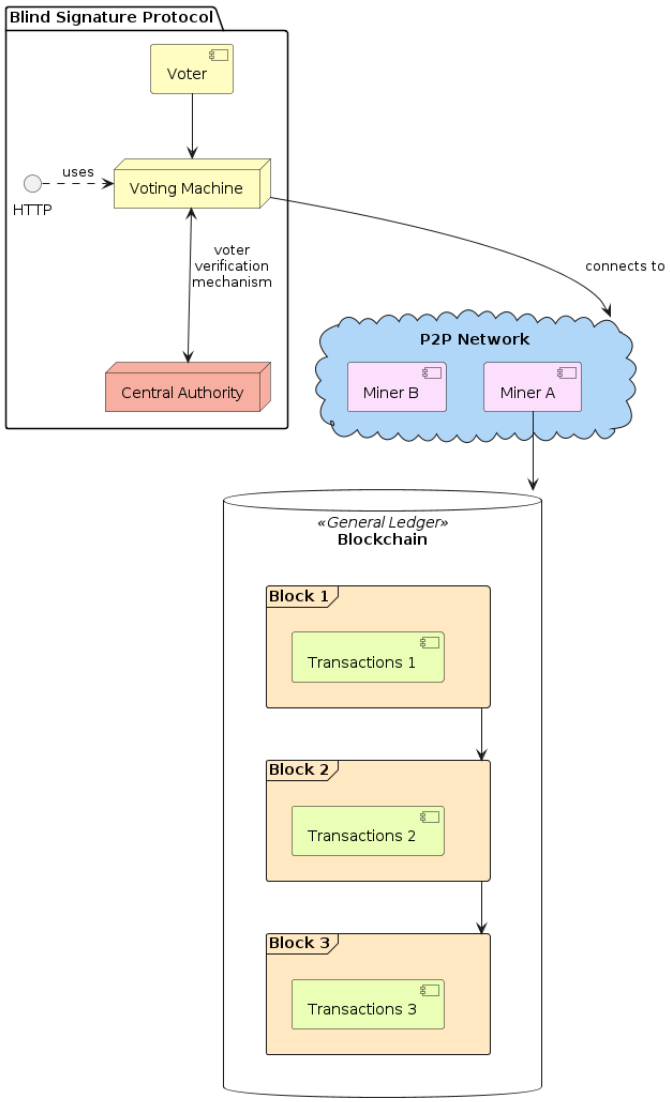

Figure 2: High-level System Architecture

The Blind Signature Protocol will be incorporated to allow voters to verify their ballot, without revealing their ballot information to the central authority. Allowing for the unlinkability between a voter and their ballot helps to protect a voter against a malicious central authority.

The peer-to-peer network will consist of voting machines (VM), the central authority (CA), the blockchain miners (BM) and the blockchain network (BN). A voting machine is a non-mining node which communicates with the central authority to verify votes and will subsequently broadcast these vote transactions to the network of miners. The central authority is the governing body of an election and also the node that authenticates voters. The blockchain miners are nodes who uphold the structure of the blockchain and store the validated ballots. Therefore, it allows the blockchain to act as a transparent ballot box [2].

The last component of our system is for starting and ending an election. The central authority and blockchain network will need to generate a pair of public/private keys to start an election. They will both need to publish their public keys and the central authority will need to publish the list of candidates as well. To end an election, the private keys will be published, and any future vote transactions will be ignored. This enables the results of the election to be verified. See Figure 2 for a diagram of the system.

2.3 Security Requirements

A decentralized e-voting system has the following requirements: [2], [1], [5], [4].

- Election Verifiability: All parties involved in the election can independently verify the results of an election.
- Individual Verifiability: Every voter can verify that their vote has been accurately recorded and considered in the result of an election.
- Eligibility and Authentication: A guarantee that only eligible voters cast votes and no individual was able to vote more than once.
- Fairness: Results of an election are only available once the election has concluded. Impossible for voters to change their decision based on partial results.
- Anonymity and privacy: No party has access to a voter's identity at any time. There is no way to link a voter to a specific ballot (unlinkability).
- Immutability: There is no way to change the value of a ballot after it has been cast.

2.4 Threat Models

There are two main threat models that can affect the described system, dishonest nodes and a dishonest central authority.

Dishonest Nodes: A dishonest node may try to modify a past block in the chain to change a vote they cast. To do this, they would need to redo the proof-of-work for the block and all the blocks after it, surpassing the work of the honest nodes. It has been shown that, if a dishonest node attempts to surpass the work done by honest nodes while not having a majority of the compute power in a network, the probability they succeed becomes exponentially lower as the number of blocks the dishonest node needs to catch up with increases [7]. Thus making it infeasible for a dishonest node if the majority of the network is controlled by honest nodes.

Dishonest Central Authority: A dishonest central authority may try to manipulate the results of an election by signing invalid votes they cast themselves. Preventing a central authority from doing so would be impossible, however, voters can limit the effectiveness of this attack simply by voting. If a majority of voters vote, a central authority can only manufacture the votes for eligible voters who have not yet cast a vote. Additionally, if the central authority casts more fake votes than the total voter population itself, then it will be obvious that tampering has occurred by the central authority. Due to this, tampering from a dishonest central authority can be minimized by voting and can be exposed if the total signed votes exceed the voter population.

2.5 Final Notes

One of the major technical challenges with incorporating a blockchain voting system is the scalability of the system. Typically for elections, a large number of votes will be cast over a short period of time. For Bitcoin systems, a major limiting factor is the number of transactions that can be processed per second due to the proof-of-work algorithm [4].

In summary, our research project aims to investigate, model, and implement a Bitcoin-based E-Voting system which is tailored for University Clubs. Our study will include all the necessary components of building the system which is modelled in Figure 2. By doing so, a blockchain-based e-voting system that is applicable to university clubs can be built.

3 LITERATURE REVIEW

3.1 Blockchain

A Blockchain is a system that peers can use to agree on a shared ledger over a network of peers without any individual peer having more say than another peer. A ledger is made up of transactions that are grouped into a series of blocks. These blocks are chained together to then form a blockchain. To ensure that past transactions cannot be modified by another peer in the network, the system links these blocks together by including the hash value of the previous block in the current block. This hash value consists of the block's metadata which also includes the hash of each transaction in the block. Note that it is not possible to modify past transactions using this system, since changing a transaction will modify the block hash, which will prevent that block from being slotted back into the blockchain in its original position [3], [8], [5], [2].

3.2 Bitcoin

Bitcoin introduced a system for a decentralized, blockchain-based banking system. Bitcoin consists of a few major components such as transactions, a Timestamp Server, a Proof-of-Work algorithm, Merkle Trees and a peer-to-peer network. Each transaction contains information about where money will be transferred from and where it will be transferred to, signed by the sender's private key. The private key is like a bank password, if you lose it, you lose access to your funds, and if someone steals it, they can steal your funds. The timestamp server is the method of chaining blocks containing transactions together using the method described in the Blockchain section above, while also ensuring that a timestamp for the block is included in its hash. The proof of work algorithm helps reinforce this timestamp server by making it sufficiently difficult to generate a new block for the chain by requiring the hash of a block to begin with a certain number of 0 digits. A nonce value is used to generate a new hash by incrementing it by one which results in an entirely new hash. The average work required to compute a valid hash is exponential with the number of zero digits required. This ensures that a block cannot be changed without having to redo work for the block, which is further exemplified if more blocks are added to the chain [7]. Merkle Trees are utilized in each block to reclaim disk space. Merkle trees allow a more concise method of storing transaction data by combining multiple transaction hashes into a singular hash. This Merkle tree hash will be stored in place of

each transaction in older blocks. This allows old transactions to be discarded since their information is still being recorded. Additionally, this hash can be utilized to verify the existence of a transaction in a block [6]. Lastly, there is the peer-to-peer network. In this network, every new transaction is broadcast to all nodes on the network, where each node puts these transactions into a block and starts mining for a block until it finds a proof-of-work. Once a node succeeds, it will broadcast this block to all other nodes. The other nodes on the network will only accept this node if all transactions are valid. Then the process continues by adding this new block to the chain and beginning to mine for the next block. Each node will presume the longest chain to be the correct one and will continue to work on extending it. If two blocks are broadcast simultaneously, some nodes may have received a different block first. Each node will continue to work on the block they received first but will keep the other branch in case it becomes longer. This tie will be broken once the next node is found and one of the chains becomes longer [7].

3.3 Merkle Tree

A Merkle tree is a method for storing the existence of data in a more concise format. A Merkle tree works by taking a list of transactions and hashing each one. Then, each of those hashes are split off into pairs, concatenated into one string, and then rehashed. This process is continued until there is only one hash value left. If there is an odd number of transactions, the last one is duplicated for symmetry. This results in a tree-like structure with the final hash being the Merkle root hash, see Figure 3 for an example [7], [6].

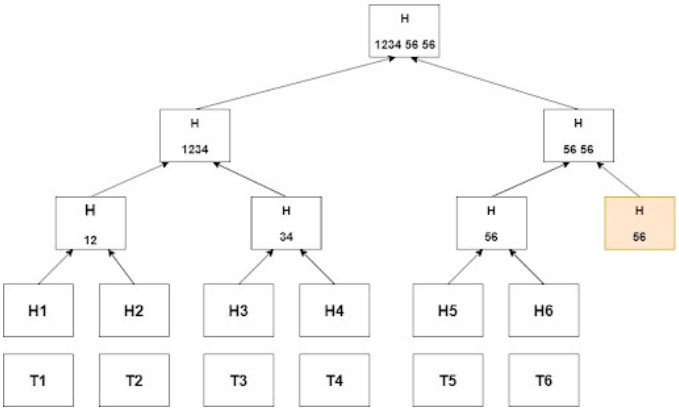

Figure 3: Merkle Tree Structure

3.4 P2P Network

Traditionally, most networks operate in a client/server relationship, where a central authority will have some nodes on a network that are servers. These servers provide information to the nodes on the network (clients). In a peer-to-peer network, each node acts as both a client and a server. This can be seen in a Bitcoin blockchain where a miner will broadcast a block it has mined to other nodes on the network, and will similarly accept new blocks mined from other nodes on the network [7], [11].

3.5 Blind Signature Protocol

The Blind Signature Protocol is a combination of the method of blinding combined with the concept of signatures. A signature is a mathematical scheme for verifying the authenticity of a message. Blinding is the process of masking and unmasking a message, while also allowing operations on it to remain. This allows a message to be signed without revealing the content of the message. This message can then subsequently be unblinded to retrieve the original message that is now signed. The usage of this in an e-voting system proves very helpful. A voter will be able to mask their ballot, preventing a central authority from seeing the contents of their ballot through blinding, and request a signature from them on their ballot. Once signed, they can then unblind their ballot and submit a verified and authenticated ballot. This then prevents non-authenticated voters from voting since they will need to go through the central authority. Additionally, the privacy of a voter's decision is preserved [2]. See Figure 4 for the full process flow of the Blind Signature Protocol.

3.6 Related Works

A similar study has been conducted called BlockVOTE. BlockVOTE is an architecture for a blockchain-based e-voting system, tailored towards the Ethereum and HyperLedger Fabric blockchains. Due to this, the usage of smart contracts are used to simulate an election, without the need for a custom blockchain architecture as seen when modelling after Bitcoin. This is due to the fact that these blockchain architectures support Turing-complete programming, which allows any program to be able to run on top of the blockchain [12]. The BlockVote architecture contains many of the major components as seen in our proposed model such as voting machines and a blockchain structure for storing votes. A key difference is the nature in which an election is started/ended. Instead of using key transactions, a smart contract is used to programmatically start and end an election [1]. Another thing to note is the absence of the usage of a Blind Signature protocol. A key disadvantage to this model is the unlinkability property. Since a voter's identity is tied to a ballot when added to the blockchain, it violates the property of unlinkability and can help reveal information about who an individual voted for.

4 IMPLEMENTATION METHOD AND RESULTS

The implementation of the Bitcoin-based blockchain for university clubs represents a significant advancement in the realm of decentralized voting systems. Our approach involves analyzing target users and drawing architectural diagrams all culminating in a secure, robust and scalable solution.

For our implementation, we used a variety of languages and technologies such as Python, JavaScript, HTML and CSS. To streamline the project setup and execution, and ensure consistency between development stations, a conda environment was configured with all the required dependencies. Leveraging modular code structures, we have organized our project into distinct components, facilitating code maintenance and extensibility. Our adherence to Object-Oriented Programming Principles and the SOLID design principles, ensured flexibility and modifiability of our code base, which allowed us to accommodate our evolving requirements.

Our implementation is comprised of four major components: a Front-End application, enabling users to seamlessly submit votes; a Flask web server, orchestrating communication between the Front-End, central authority, and blockchain network; the Blind-signature protocol algorithm, safeguarding voter anonymity by establishing a separation between voters and their ballots. The last component is the Bitcoin-based blockchain architecture to support an election.

4.1 Bitcoin Blockchain Implementation

To implement the blockchain for the e-voting system, a class diagram was constructed to specify all the components required to implement a functioning blockchain. This resulted in the following required classes: BlockChain, Block, BlockHeader, MerkleTree, Transaction, KeyTransaction and VoteTransaction. Each of these classes are required to create a functioning blockchain in an object-oriented programming manner. The key components of the Block class are the block header, the list of transactions on the block, and the functions for adding transactions and mining a block. The BlockHeader class includes all the important metadata for a block that is required for chaining blocks together. This includes the previous block's hash value, a timestamp, a Merkle root hash, the number of difficulty bits for the proof-of-work algorithm, and a nonce value. In addition to this, the functionality for setting and adjusting these values as well as computing a block hash is included. The MerkleTree class represents the functionality for implementing a functioning Merkle tree and was implemented through a 3rd party library called Merkly [9]. Moving on to the Transaction abstract class, it has two possible variations, a key transaction and a vote transaction. The KeyTransaction class is used for publishing the public and private keys of the central authority and blockchain network to start and end an election. The VoteTransaction class is used for submitting a ballot. Both of these classes include the functionality for storing this information and the functionality for creating a hash of the transaction, including a timestamp as well. Lastly, the Block class and Transaction class are both serializable to ensure that the data for a given object can be transmitted over the peer-to-peer network and be reconstructed perfectly after being broadcasted. This is all finally tied together with the BlockChain class which holds the chain of blocks and functions used for adding a new block to the chain. In Figure 4, the class diagram for this blockchain architecture is shown, revealing how each class is related to one another as well as the required high-level implementation details to create a system like this.

4.2 Peer-to-Peer Network Implementation

A class diagram was also constructed to specify all the components required for the peer-to-peer network. Figure 5 shows the relationships between all the classes and their corresponding attributes and methods. This resulted in the following classes: Peer, CentralAuthority, BlockchainNetwork, VotingMachine, BlindSignatureProtocol, Miner and MinerPool. Each of these classes are required to have a functioning peer-to-peer network for the e-voting system. The VotingMachine class represents the Front-end interface a voter would use to cast a vote. By casting a vote, it utilizes the BlindSignatureProtocol class to create a ballot and blind/unblind it accordingly. The BlockchainNetwork and CentralAuthority classes both contain the necessary components to store their public/private keys and start/end an election. Additionally, the CentralAuthority class has a function to verify the ballot of a voter. The last peer-to-peer network participant is represented by the Miner class. This class represents the nodes that hold the infrastructure of the blockchain and validate all transactions and blocks going on the blockchain. These four classes are all sub-classes of the Peer class. The Peer class contains the necessary functionality to join the peer-to-peer network and create and broadcast messages to the network. Lastly, the MinerPool class represents a list of other peers on the network that an individual broadcasts to and receives messages from. Utilizing all of these classes in conjunction, a peer-to-peer network for our e-voting system can be constructed.

4.3 Blind Signature Protocol Implementation

In our system, the Blind Signature protocol was used to ensure voter anonymity and privacy. The protocol has two main properties which are blindness and forgery resistance [2]. By utilizing this protocol, our system is able to ensure the unlinkability of the voter and their ballot. To implement it, a sequence diagram was created that depicts all aspects of the interactions of each participating party. The parties that interact to perform the Blind Signature protocol are a Voter, Core Module (Flask Webserver), Central Authority (CA) and Peer-to-Peer Network which can be seen in Figure 6.

Initially, a Voter initiates a vote request which consists of them sending some unique identifier (token) which can be a student ID, and the candidate they voted for. Next, the Flask Webserver (i.e. Core Module) processes a request and begins to perform the Blind Signature protocol. First, the $ballot\_confirmation\_algorithm$ gets performed, which is used to mask and randomize the candidate ID thus making brute-force attack on the protocol significantly harder. It produces a point (P) and slope (λ), which uniquely identifies the voters' ballot. Next, our system utilizes the RSA encryption algorithm along with SHA-256 hashing which gives us an encrypted message to send to the CA (using the public keys of the blockchain network and central authority). The message also gets 'blinded' by utilizing modular exponentiation to hide the contents of the message from the CA. Once the CA receives an encrypted token (i.e. encrypted ID of the voter), along with the message, it first verifies that the voter is a 'real' person who is eligible to vote. In the case of a failure, it returns a Failure and indicates that the voter is unable to vote. In the case of success, it produces a digital signature on the given message (note that the CA doesn't know the contents of the message). The signature is then returned back and the Core Module unblinds the message by using modular exponentiation by finding the modular inverse with the extended Euclidean algorithm. Lastly, the (λ) gets encrypted, which will be used by miners to obtain the candidate that the voter voted for. Finally, all three components are sent to the Peer-To-Peer Network where miners are able to process the transactions and verify them.

To conclude, in the Blind Signature protocol, the voter first blinds the message using a random factor before sending it to the CA for signature. The CA then signs the blinded message, unaware of its content. After receiving the signed blinded message, the voter unblinds the message to obtain a valid signature on the original

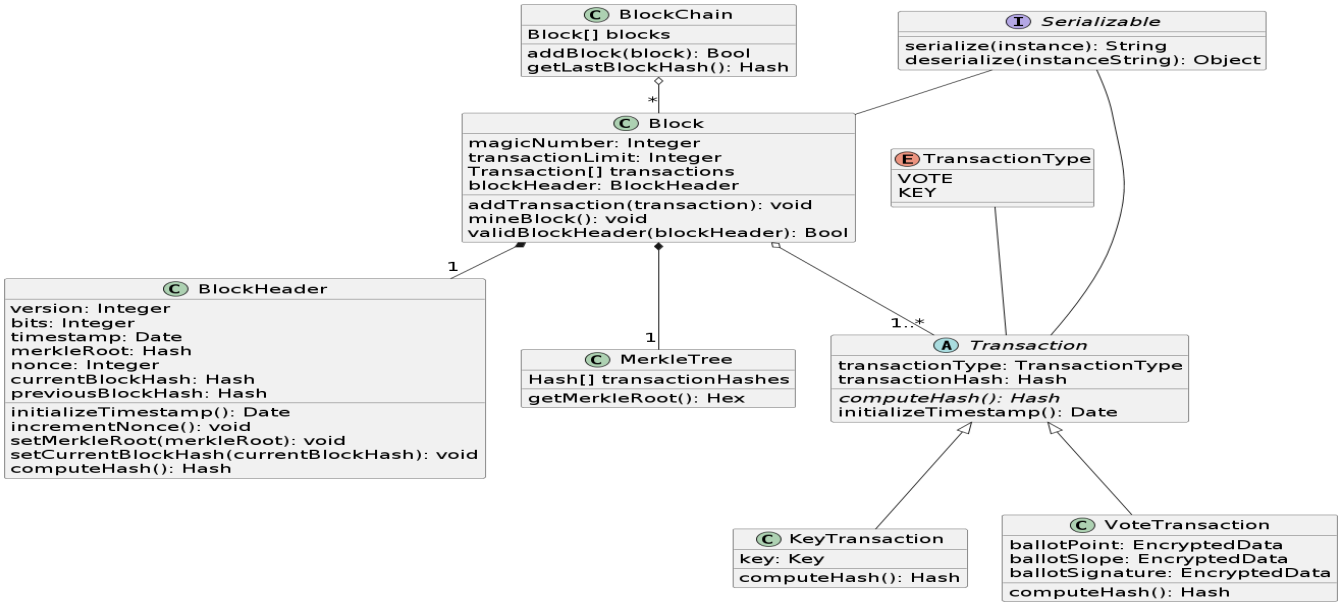

Figure 4: Blockchain Class Diagram

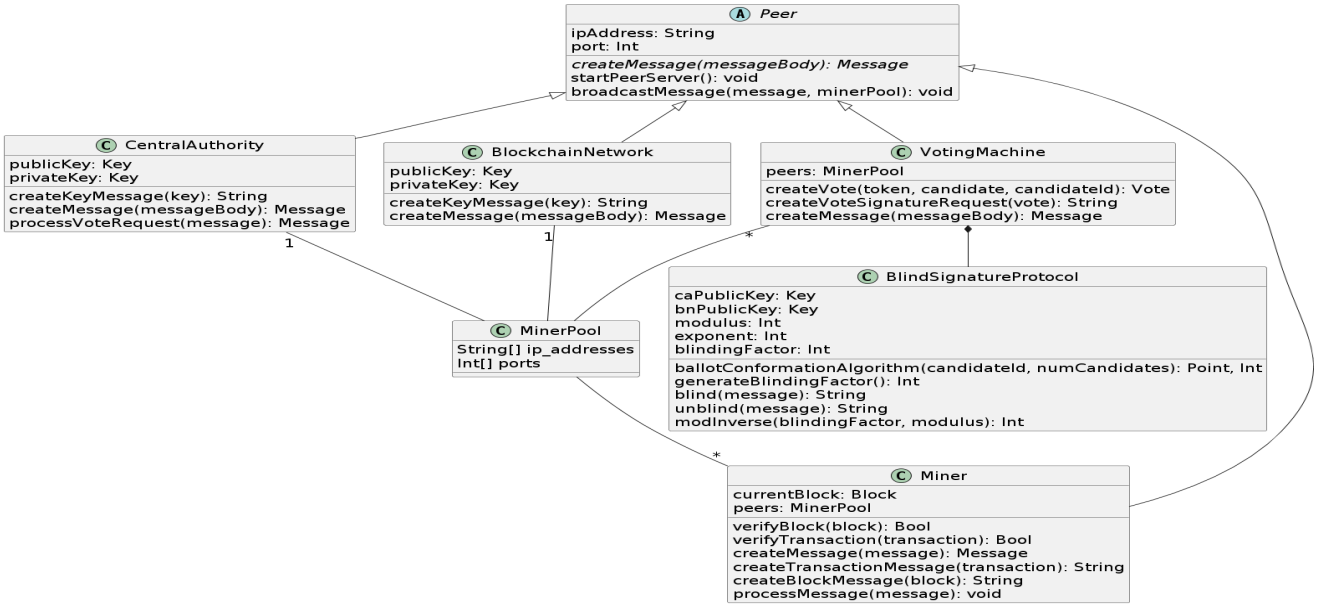

Figure 5: P2P Class Diagram

message. This process ensures that the CA cannot link the signature to the specific message, providing anonymity and confidentiality to the voter's communication.

4.4 Voting Implementation

To implement the Voting Transaction Flow, a sequence diagram was created. Refer to Figure 7, to understand all the parties involved (Voter, Core Module, Peer-to-Peer Network, Miner and General Ledger) as well as their interactions. Once a voter submits a vote, the request gets forwarded to the Core Module, where, as described in section 4.4, the Blind Signature protocol gets performed. After this, the transaction vote gets broadcasted to the peer-to-peer network and thus propagates to each Node (Miner) in the network. This is done through WebSockets as they allow Miners to be both clients and servers. Next, each miner verifies the transaction (i.e.

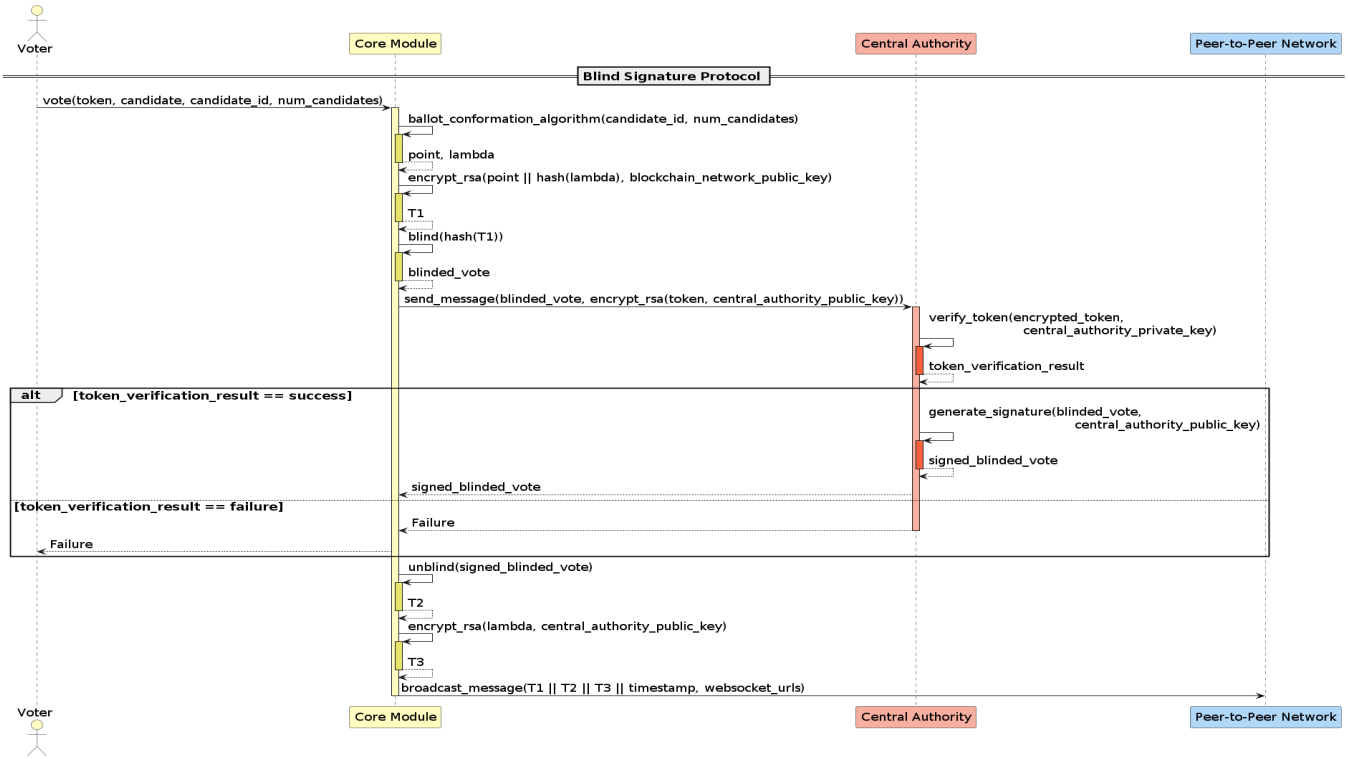

Figure 6: Blind Signature Protocol

transaction format, digital signatures). Based on the result, the transaction can be rejected, or, alternatively, the Miner starts mining for a block with the new transactions. Once a block has been mined, (i.e. Miner found the proper hash), they broadcast the Block to the P2P Network where all other miners are able to verify the block and include it in their copy of the general ledger.

4.5 Final Notes

In conclusion, our implementation presents a paradigm shift in decentralized voting systems for university clubs. Some of the challenges that were encountered were understanding the parties needed to successfully use our system in the university setting and communication between them, as well as implementing specific cryptographic protocols to allow voters' identities to remain anonymous (i.e. students and professors). By integrating innovative technologies and adhering to best practices in software engineering, we have realized a fully functional blockchain. For further insights into our project, including code organization, platform compatibility, and deployment instructions, we invite interested parties to explore our repository at *Blockchain Voting University Clubs*

5 CONCLUSION

The research project addresses a critical need for a transparent e-voting system, tailored to the specific needs of university clubs utilizing blockchain technology. Through an exploration of existing blockchain-based voting solutions and the development of a transparent electoral system, we aim to mitigate the lack of trust prevalent in club elections. Our proposed solution not only establishes a secure blockchain infrastructure for vote storage and validation, but also prioritizes voter privacy and the prevention of duplicate votes. Ultimately, the project's overarching objective is to contribute to the advancement of democratic processes within university communities through the application of blockchain technology.

The key outcomes of our project encompass the successful implementation of a Bitcoin-based blockchain, the integration of the Blind-signature protocol to enhance security and privacy, and the provision of transparent and accessible election results. Our project bears significant implications for students and staff of the universities by fostering integrity and transparency. Furthermore, the project underscores educational value by offering software that promotes transparency in club elections and incentivizes students' participation. Overall, this project advances the use of blockchain technology by promoting democratic principles within academic institutions.

6 FUTURE WORK

Our system involves the integration of cutting-edge technology within academic environments and promises transformative possibilities for future work paradigms. In previous sections, the significance of the project, implementation and the setup needed to use the system were underscored. 'Eon: Blockchain Voting for University Clubs' projects require collaboration with universities

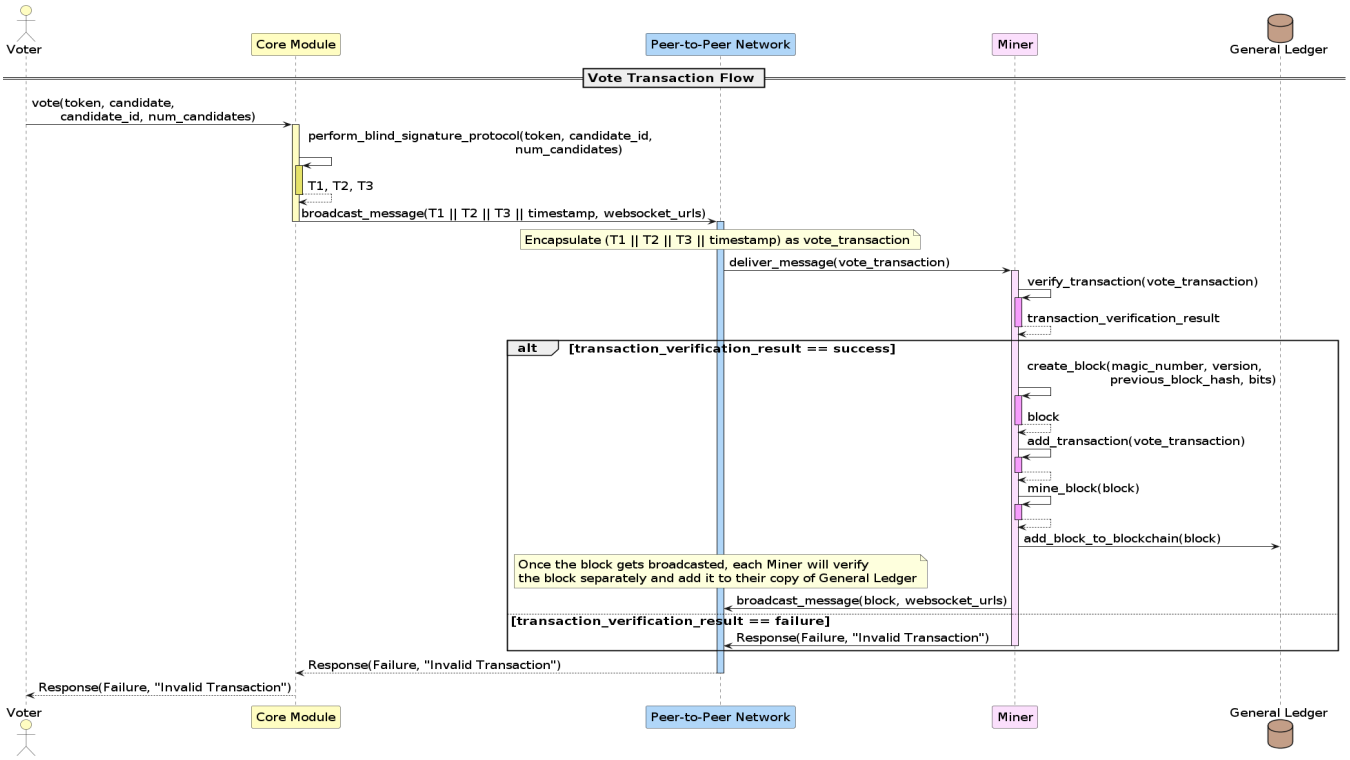

Figure 7: Vote Transaction Flow

and affiliated clubs to streamline processes such as elections and effectively use our solution.

At the forefront of discussions, the scalability and security of a major system of our solution emerges, considering the infrastructure required for their implementation and proper execution. The system utilizes the Blind Signature Protocol which is a novelty and requires thorough testing in a real university-setting environment. These are pivotal topics that were partially addressed in our current research. In the future, to ensure the robustness of our infrastructure, the following steps will be accomplished: implementing transaction and block validation upon broadcast to miners, enhancing asynchronous communication and implementing a scalable solution for the P2P network, and establishing connectivity with a central authority (i.e. University of Guelph).

In summary, advancing blockchain-based solutions, particularly for university clubs, signifies a major stride toward the potential use of decentralized technologies in universities. In the future, the focus remains on refining scalability and fostering collaborative partnerships to enhance the efficacy of blockchain applications. Key areas of emphasis include further optimizing transaction and block validation mechanisms, expanding asynchronous communication capabilities and ensuring the transparency and security of our application and the elections themselves.

REFERENCES

[1] Chinnapong Angsuchotmetee and Pisal Setthawong. 2020. BlockVOTE : An Architecture of a Blockchain-based Electronic Voting System. 14 (11 2020), 174–189. https://doi.org/10.37936/ecti-cit.2020142.227455

[2] Julio César Perez Carcia, Abderrahim Benslimane, and Samia Boutalbi. 2021. Blockchain-based system for e-voting using Blind Signature Protocol. In *2021 IEEE Global Communications Conference (GLOBECOM)*. 01–06.

[3] W. Dai. 1998. b-money.

[4] Jun Huang, Debiao He, Mohammad S. Obaidat, Pandi Vijayakumar, Min Luo, and Kim-Kwang Raymond Choo. 2021. The Application of the Blockchain Technology in Voting Systems: A Review. 54, 3 (2021).

[5] Vivek S K, Yashank R S, Yashas Prashanth, Yashas N, and Namratha M. 2020. E-Voting Systems using Blockchain: An Exploratory Literature Survey. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)* (2020), 890–895. https://api.semanticscholar.org/CorpusID:221473891

[6] Ralph C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology — CRYPTO '87*, Carl Pomerance (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 369–378.

[7] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com* (03 2009).

[8] Mario Oettler. 2021, 2023. How to Calculate and Verify a Hash of a Block." Blockchain Academy. blockchain-academy.hs-mittweida.de/courses/blockchain-introduction-technical-beginner-to-intermediate/lessons/lesson-13-bitcoin-block-hash-verification/topic/how-to-calculate-and-verify-a-hash-of-a-block/

[9] olivmath. 2024. Merkly (2024) [Source code]. https://github.com/olivmath/merkly

[10] Olga Onuch and Gwendolyn Sasse. 2022. The Belarus crisis: people, protest, and political dispositions. *Post-Soviet Affairs* 38, 1-2 (2022), 1–8.

[11] R. Schollmeier. 2001. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*. 101–102. https://doi.org/10.1109/P2P.2001.990434

[12] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.